UNIVERSITY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Distributed Systems, Security and Emerging Information Technologies»

ΠΜΣ «Κατανεμημένα Συστήματα, Ασφάλεια και Αναδυόμενες Τεχνολογίες Πληροφορίας»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	***Integrating Hyperledger Aries Self-Sovereign Identities using Hyperledger Fabric as a Verifiable Data Registry*** *Ενσωμάτωση του Hyperledger Aries Self-Sovereign Identities με χρήση του Hyperledger Fabric ως επαληθεύσιμου μητρώου δεδομένων.*
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Misiakoulis Georgios** Μησιακούλης Γεώργιος
Father's name: Πατρώνυμο:	**Konstantinos** Κωνσταντίνος
Student's ID No: Αριθμός Μητρώου:	ΜΠΚΣΑ20002
Supervisor: Επιβλέπων:	**Panagiotis Kotzanikolaou, Associate Professor** Παναγιώτης Κοτζανικολάου, Αναπληρωτής καθηγητής

October 2023 / Οκτώβριος 2023

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Panagiotis Kotzanikolaou **Associate Professor**	**Christos Douligeris** **Professor**	**Michael Psarakis** **Associate Professor**
Παναγιώτης Κοτζανικολάου Αναπληρωτής Καθηγητής	Χρήστος Δουληγέρης Καθηγητής	Μιχαήλ Ψαράκης Αναπληρωτής Καθηγητής

Table of Contents

Abstract

The problem of identity management has been a major concern of the scientific community for several years. In particular, the combination of identity management with the full implementation of the GDPR raised new problems about how to confirm the identity of users without sharing much unnecessary information. In this direction, the use of Self Sovereign Identities has been gaining momentum in recent years. Self Sovereign Identity (SSI) is an application that allows users to keep their identities secure while proving the latter to the verifier without revealing it. One of the most mature SSI implementations is Hyperledger Aries (HLA), which currently is dependent on Indy as a Distributed Ledger and Ursa as a cryptography library. At the same time, the most well-known platform particularly for industrial applications is Hyperledger Fabric. So far, Hyperledger Aries operates as a separate system that "runs" alongside the main application and integrates the functions of SSI, increasing management and operational costs and making interoperability difficult. Identity verification is deemed preferable by enterprises, while constructing solutions with Hyperledger Fabric within their platforms. In this thesis we properly modify Aries Verifiable Data Registry (VDR) in order to be able to interact with Hyperledger Fabric as a ledger, allowing SSI applications to be built on top of Hyperledger Fabric.

Περίληψη

Το πρόβλημα της διαχείρισης της ταυτότητας απασχολεί την επιστημονική κοινότητα εδώ και αρκετά χρόνια. Ειδικότερα, ο συνδυασμός της διαχείρισης ταυτότητας με την πλήρη εφαρμογή του GDPR δημιούργησε νέα προβλήματα σχετικά με τον τρόπο επιβεβαίωσης της ταυτότητας των χρηστών χωρίς να μοιράζονται πολλές περιττές πληροφορίες. Προς αυτή την κατεύθυνση, η χρήση τεχνολογιών αυτοκυριαρχικής ταυτοποίησης (Self Sovereign Identities - SSI) έχει κερδίσει έδαφος τα τελευταία χρόνια. Η τεχνολογία SSI επιτρέπει στους χρήστες να διατηρούν την ασφάλεια των ταυτοτήτων τους, επιτρέποντας την αυθεντικοποίηση τους σε έναν ελεγκτή χωρίς να αποκαλύπτουν οποιαδήποτε μυστική και προσωπική πληροφορία. Μία από τις πλέον ώριμες τεχνολογίες SSI είναι το Hyperledger Aries. Το Hyperledger Aries εξαρτάται από το Hyperledger Indy ως υποκείμενη πλατφόρμα κατανεμημένου - καθολικού (blockchain) και το Hyperledger Ursa ως βιβλιοθήκη κρυπτογραφίας. Παράλληλα, η πιο γνωστή πλατφόρμα ειδικά για βιομηχανικές εφαρμογές είναι το Hyperledger Fabric. Μέχρι στιγμής, το Hyperledger Aries λειτουργεί ως ξεχωριστό σύστημα που λειτουργεί παράλληλα με την κύρια εφαρμογή και ενσωματώνει τις λειτουργίες του SSI, αυξάνοντας το κόστος διαχείρισης και λειτουργίας και δυσχεραίνοντας τη διαλειτουργικότητα. Η επαλήθευση της ταυτότητας κρίνεται προτιμότερη από τις επιχειρήσεις, ενώ κατασκευάζουν λύσεις με το Hyperledger Fabric εντός των πλατφορμών τους. Στην παρούσα εργασία τροποποιούμε κατάλληλα το Aries Verifiable Data Registry (VDR), ώστε να μπορεί να αλληλεπιδράσει με το Hyperledger Fabric ως καθολικό βιβλίο, επιτρέποντας την κατασκευή εφαρμογών SSI πάνω στο Hyperledger Fabric.

Acknowledgements

This work was conducted within the framework of my master's thesis at the **"Distributed Systems, Security and Emerging Information Technologies"** program at the **University of Piraeus** and was finalized in conjunction with my full-time engagement at **INLECOM INNOVATION Non-Profit Organization** within the **ERATOSTHENES** research project (www.eratosthenes-project.eu). The code and mechanisms associated with the company's research efforts in the **ERATOSTHENES** research project are outside the scope of this thesis and pertain to the intellectual property of **INLECOM INNOVATION Non-Profit Organization**.

The **ERATOSTHENES** project aims to address critical challenges in relation to the "Security of Things," recognizing its fundamental role in the future success of the Internet of Things (IoT). The project's primary objective is to develop a decentralized and contextually-driven Trust and Identity Management Framework tailored for IoT environments with limited resources, based on a self-sovereign approach.

Furthermore, the project seeks to enable automated lifecycle monitoring of devices, thereby enhancing trust, identities, and overall resilience within the IoT ecosystem. This initiative aligns with the enforcement of key regulations such as the NIS (Network and Information Security) directive, GDPR (General Data Protection Regulation), and the Cybersecurity Act.

Abbreviations

GDPR	**G**eneral **D**ata **P**rotection **R**egulation
NIS	**N**etwork and **I**nformation **S**ecurity
IoT	**I**nternet **o**f **T**hings
WWW	**W**orld **W**ide **W**eb
SSI	**S**elf-**S**overeign **I**dentity
VDR	**V**erifiable **D**ata **R**egistry
DID	**D**ecentralized **Id**entities
VC	**V**erifiable **C**redentials
gRPC	**g**[1] **R**emote **P**rocedure **C**all
DLT	**D**istributed **L**edger **T**echnology
URI	**U**niform **R**esource **I**dentifier
PROTOC	**PROTO**buf **C**ompiler
K8S	Kubernetes
KUBE	**Kube**rnetes
PAAS	**P**latform **A**s **A** **S**ervice
OS	**O**perating **S**ystem
VM	**V**irtual **M**achine
AWS	**A**mazon **W**eb **S**ervices
HLF	**H**yper**l**edger **F**abric
HF	**H**yperledger **F**abric
HSM	**H**ardware **S**ecurity **M**odule
HLA	**H**yperledger **A**ries
NFS	**N**etwork **F**ile **S**ystem
IPFS	**I**nter**p**lanetary **F**ile **S**ystem
CRUD	**C**reate-**R**ead-**U**pdate-**D**elete

[1] Google changes the meaning of the "**g**" for each version to the point where they even made a README to list all the meanings.

Table of Figures

Table of Tables

Terminology

This specification employs the following terminology to elucidate its concepts **[1]**.

Claim An assertion made about a subject / holder

Entity A person, organization, or device with distinct and independent existence that is assigned to one or more roles in the ecosystem.

Holder An entity can assume a role by possessing one or more verifiable credentials and generating presentations derived from these credentials. Typically, a holder is the individual associated with the verifiable credentials they possess, although exceptions exist. These credentials are typically stored by holders in credential repositories.

Credential A compilation of assertions established by an issuer constitutes a verifiable credential. Such a credential is designed to be tamper-resistant and can affirm its authorship through cryptographic mechanisms. Verifiable credentials are instrumental in constructing verifiable presentations, which too can undergo cryptographic verification. These assertions within a credential may pertain to various subjects.

Decentralized Identifiers A **DID**, which stands for Decentralized Identifier, is a transportable identifier based on a URL and linked to an entity. DIDs are frequently utilized within a verifiable credential and are linked to subjects in a way that allows the verifiable credential to be effortlessly transferred from one repository to another without necessitating the reissuance of the credential. An example of a DID is:

Figure 1: Decentralized Identifier Example

Decentralized Identifier Document It is usually referred to as a **DID Document**, this is a document that can be accessed via a verifiable data registry (VDR). It comprises details pertaining to a particular decentralized identifier (DID), including information about the associated repository, public key data, or methods for carrying out cryptographic authentication of a DID controller.
An example of a DID Document, related to the **did:example: 123456789abcdefghi** DID could be the following:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Figure 2: Decentralized Identifier Document Example

Issuer	An entity can assume a role by making claims about one or more subjects, crafting a verifiable credential based on these claims, and subsequently transmitting this verifiable credential to a holder.
Derived Predicate	A verifiable boolean statement regarding the value of another attribute within a verifiable credential. These are valuable in zero-knowledge-proof-style verifiable presentations as they have the capability to restrict the disclosure of information.
Identity Provider	An identity provider, often abbreviated as **IdP**, is a system responsible for generating, upkeeping, and overseeing identity data for holders. It also furnishes authentication services to relying party applications within a federation or distributed network. In this context, the holder invariably serves as the subject. Even when dealing with verifiable credentials as bearer credentials, it is presupposed that these verifiable credentials are retained by the subject, and any deviation from this implies that they have been illicitly obtained by an attacker.
Presentation	Data derived from one or more verifiable credentials, issued by one or more issuers, is disclosed to a specific verifier. A verifiable presentation is a secure representation that has been encoded to establish trust in the data's authorship through cryptographic validation. Some varieties of verifiable presentations may encompass information generated from, but not necessarily containing, the original verifiable credentials – examples of this include **zero-knowledge proofs**.
Repository	A software application, such as a storage vault or a personal verifiable credential wallet, designed to securely store and safeguard access to the verifiable credentials of holders.
Subject	An entity or object for which assertions are formulated or stated.
Validation	The confidence or guarantee that a verifiable credential or a verifiable presentation aligns with the requirements and expectations of a verifier and other parties reliant on it.
Verifiable Data Registry	A role, often abbreviated as **VDR**, could involve facilitating the generation and verification of identifiers, keys, and other pertinent information. This includes elements like verifiable credential schemas, revocation registries, issuer public keys, and more, which may be necessary for the utilization of verifiable credentials.
Verification	The assessment of whether a verifiable credential or verifiable presentation accurately represents the issuer or presenter, considering factors such as: This includes checking that: 1) Conformance to the specification of the credential (or presentation). 2) Satisfaction of the proof method. 3) Success of the status check (if applicable). **Verification of a credential does not imply evaluation of the truth of claims encoded in the credential.**

Verifier	A role undertaken by an entity involves receiving one or more verifiable credentials, possibly enclosed within a verifiable presentation, for subsequent processing.
URI	A Uniform Resource Identifier, as defined by [RFC3986]**[2]**.

[Page Intentionally Left Blank]

1. Introduction

The problem of managing electronic identities is a structural issue that stems from the initial conception of the World Wide Web (WWW). Internet's addressing system is based on the identification of physical network endpoints (computer machines) and not people. Therefore, the Internet cannot uniquely identify users and the identification process is handled at the application layer, by websites and applications. The absence of a secure, portable, and user-controlled identity has dire repercussions. It signifies that a person's identity and personal information exist only within the context of each website or application that he or she uses **[3]**. This has a significant impact on both the security issue and the increased costs required to set up and maintain multiple identity management mechanisms.

The evolution of identity management mechanisms tries to meet three (3) main requirements: (i) security, by ensuring that the relevant information must be protected from unintended disclosure, (ii) control, by ensuring that only the owner is the one who controls who can access it and for what purpose, and (iii) portability, in the sense that the user is not tied to a single vendor. According to **[4]**, there are four stages in the development path of digital identities. Centralized identity is the first stage where identities are owned and controlled by a single entity and therefore users don't own their identity record. At a more complex level, it can allow different services to securely share user details without prior knowledge. Federated identity is the second stage of evolvement that answers some of the problems of centralization. Even at the simplest implementation it gives a certain degree of portability compared to the previous approach. For example, a user is able to login into one service using credentials of another. At a more complex level it can allow different services to share details about the user. The User-Centric identity is most frequently manifested in the form of independent personal data stores at one end of the spectrum and large social networks at the other end **[5]**. However, the entire spectrum still relies on the user selecting an individual identity provider and agreeing to their often one-sided adhesion contracts.

The latest advancement on identities is **Self - Sovereign Identity (SSI)** which gives users the ability to exercise control over the information that is associated with their digital identities and the capability to demonstrate to websites, services, and applications across the internet that they are who they say they are **[6]**. Up until relatively recently, individuals were required to maintain multiple persistent identities across the internet. As a result, individuals were forced to use large identity providers like Google and Facebook, which controlled the information that was associated with their identities. The SSI system offers a method to conceal your identity while still demonstrating it to the verifier in a way that does not cause your identity to be exposed and also in compliance with GDPR requirements **[7]**. As mentioned by the authors in **[8]**, blockchain technology is not explicitly required for a Self-Sovereign Identity solution but it is a good foundation to build upon, due to various technical advantages that the blockchain has to offer. A solution towards this direction is Hyperledger Aries which was utilized throughout the development of these applications. In the context of decentralized identity, a **Verifiable Data Registry (VDR)** is a system that mediates the generation and authentication of identifiers, keys, and other pertinent information. It is also a place where Decentralized Identifiers (DIDs) can be anchored to. Blockchain technology is typically utilized in the construction of VDRs. This is due to the fact that blockchains, with the help of smart contracts, are regarded as an excellent location to store data in an unchangeable form **[9]**.

Unfortunately, the current status of SSI applications that enable Hyperledger Aries to create identities is restricted to make use of Hyperledger Indy as a VDR. This limitation prevents Hyperledger Aries from using other blockchains as a VDR. Currently, the Blockchain platform

with the most adoption at the moment especially for industrial applications, is Hyperledger Fabric. Businesses developing solutions with Fabric would like their platforms to support identity verification services. Until now, SSI is used in Hyperledger Aries with a specific VDR package which limits its potential implementation.

1.1. Thesis Scope

The scope of this dissertation is to solve the above-mentioned problem by modifying Hyperledger Aries Go while adding a new VDR package, so that it can support Hyperledger Fabric or any other underlying blockchain platform as a ledger. This could eventually lead to a wider adoption of SSI technology and also help overcome interoperability issues.

1.2. Thesis Structure

This thesis is organized in the following manner: In Section 2, we delve into the background of the study, exploring the emerging relevance of Self-Sovereign Identity (SSI) within blockchain ecosystems, by examining the core elements of the SSI ecosystem, the Hyperledger Fabric blockchain platform, the Kubernetes container orchestration platform, the gRPC protocol, and the GOB binary data format. Continuing with Section 3, a comprehensive literature review is presented. Section 4 unveils the intricacies of the proposed architecture, detailing the interaction of these components. The design, implementation, and testing of smart contracts are meticulously covered, followed by a step-by-step guide to deployment, including the setup of environmental variables, organizations, orderers, and chaincode. This section also outlines the Verifiable Data Registry (VDR) gRPC server's design and implementation, as well as the integration of the Hyperledger Aries Framework Go. In Section 5, we substantiate the proposed architecture's functionality and effectiveness through real-world scenarios. Finally, Section 6 engages in a critical analysis of the research findings and their implications, summarizing the contributions made to the field and suggesting avenues for future research.

2. Background

2.1. The Self-Sovereign Identity (SSI) Ecosystem

Self-Sovereign Identity (SSI) [10] is a term mainly used to describe the digital movement that enables individuals or organizations to have sole ownership of their identity and full control on how their personal data is shared and used. SSI enables individuals to engage in the digital realm with the same level of freedom and trust as they experience in face-to-face interactions in the physical world. SSI allows entities to prove control of one or multiple identifiers without any intermediary and provides the flexibility for each entity to control which information or claims are eventually revealed to third parties. The importance of providing citizens full control of their data and the decision of which data to disclose is a main objective of SSI and targets vulnerabilities of existing procedures that fail to handle personal and sensitive data due to unauthorized accesses or even worse data breaches. At the same time, companies find it extremely challenging to ensure secure and trusted digital interactions with customers and manage employee authorizations, while the need for inter-domain communications as well as for cross-organization authorization is increasing day by day. For this purpose, the SSI technology including **Decentralized Identifiers (DIDs)**, **Verifiable Credentials (VCs) and Distributed Ledger Technology (DLT)**, can address the above issues in order to unleash tremendous economic potential improving the security, trust, and efficiency while also supporting the privacy of user data.

Credentials [11] are a significant part of people's lives. In the physical world, credentials provide a lot of benefits, because it might held information related to identifying the subject

(a photo or a name), information related to the issuing authority (a city government or national agency), information related to the type of the credentials (a passport or a license), information related to specific attributes or properties being asserted by the issuing authority about the subject (date of birth or nationality), evidence related to how the credential was derived and information related to constraints on the credential (expiration date, or terms of use). Despite the significance that the credentials provide in the physical world their use on the Web continues to be elusive.

A verifiable credential has the capability to convey the same information as a physical credential. The incorporation of technologies like digital signatures enhances the tamper-evident nature and trustworthiness of verifiable credentials beyond what physical credentials offer. Verifiable credential holders can create verifiable presentations and share them with verifiers to demonstrate their possession of verifiable credentials with specific attributes. Both verifiable credentials and presentations can be quickly transmitted, making them more convenient compared to their physical counterparts when establishing trust over long distances.

The word "**verifiable**" in the terms verifiable credential and verifiable presentation refers to the characteristic of a credential or presentation as being able to be verified by a verifier. **Verifiability of a credential does not imply that the truth of claims encoded therein can be evaluated**. However, the issuer can include values in the evidence property to help the verifier apply their business logic to determine whether the claims have sufficient veracity for their needs.

Within the SSI ecosystem **[12]**, specific roles of the core actors exist and the relationships between them in a form where verifiable credentials are expected to be useful are thoroughly presented. There are three main entities in a VC - the **Issuer** (the entity that issues a credential), the **Holder** (the entity that owns the credential), and the **Verifier** (the entity that verifies the credential). Since verifiable credentials use public key infrastructure and digital signatures to prove a claim, there is also one significant component, necessary for the operation of the ecosystem that can be found within the ecosystem and is named **Verifiable Data Registry**, or sometimes abbreviated as **VDR**. VDRs are responsible for managing identifiers, keys, and essential data like credential schemas, revocation records, and issuer keys, all crucial for verifiable credentials. These VDRs can encompass trusted databases, decentralized databases, and distributed ledgers like blockchains. Often, SSI ecosystems make use of multiple VDR types. It's common to employ multiple types of VDRs within an SSI ecosystem. In the following image, the main participants as well as the flow of the information that forms a complete ecosystem, known as “the trust triangle” is clearly depicted:

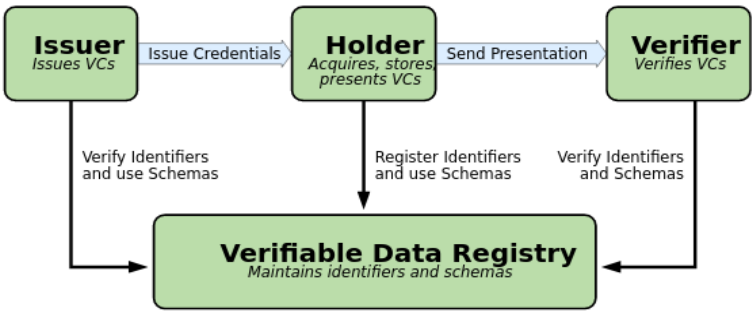

Figure 3: Self-Sovereign Identity Triangle

According to the image above, there are a lot of characteristics that can be extracted and identified in a SSI ecosystem **[13]**:

- Credentials denote statements originated by an issuer.
- Verifiable credentials signify statements from an issuer presented in a secure and privacy-preserving manner.
- Holders compile collections of credentials and/or verifiable credentials from diverse issuers into a unified entity called a presentation.
- To render presentations tamper-evident, holders transform them into verifiable presentations.
- Issuers can issue verifiable credentials concerning any subject.
- Participation as an issuer, holder, or verifier necessitates no registration or approval from an authority, as trust is mutual between parties.
- Verifiable presentations enable any verifier to authenticate verifiable credentials from any issuer.
- Holders can receive verifiable credentials from any source.
- Holders can engage with any issuer or verifier using any user agent.
- Sharing verifiable presentations is feasible for holders, allowing verification without disclosing the verifier's identity to the issuer.
- Holders can store verifiable credentials anywhere without impacting their verifiability, and issuers remain unaware of their storage location or access times.
- Presenting verifiable presentations to verifiers doesn't compromise the authenticity of claims or disclose this action to issuers.
- Verifiers can validate verifiable presentations from any holder, containing claim proofs from any issuer.
- Verification should not rely on direct interactions between issuers and verifiers.
- Verification should not unveil the verifier's identity to any issuer.
- Issuers must possess the capability to issue verifiable credentials supporting selective disclosure, without mandating support for this feature in all conformant software.
- Issuers can issue verifiable credentials that facilitate selective disclosure.
- If a single verifiable credential supports selective disclosure, holders can present claim proofs without revealing the entire credential.
- Verifiable presentations can either reveal credential attributes or fulfill requested derived predicates, which are Boolean conditions like greater than, less than, equal to, or in a set.
- Issuers can issue revocable verifiable credentials.
- The processes of cryptographically safeguarding credentials and presentations, as well as verifying verifiable credentials and verifiable presentations, must be deterministic, bidirectional, and lossless. Verification results must transform into a generic data model in a deterministic manner, ensuring semantic and syntactic equivalence with the original construct for interoperable processing.
- Verifiable credentials and verifiable presentations must be serializable in one or more machine-readable data formats. Serialization and deserialization processes must be deterministic, bidirectional, and lossless. Serialization into a generic data model must be feasible without data or content loss.
- The data model and serialization must allow for extensions with minimal coordination.
- Revocation by issuers should not divulge identifying information about subjects, holders, specific verifiable credentials, or verifiers.
- Issuers can provide the reason for revocation.
- Issuers should differentiate between revocation due to cryptographic integrity concerns (e.g., compromised signing key) and status changes (e.g., suspended license).

- Issuers can offer a service for verifiable credential refreshing.

In the following sections the core data model, which forms the foundation of the SSI ecosystem, such as **claims, credentials** and **presentations** is thoroughly outlined.

A **claim** **[14]** represents a statement concerning a subject. A subject is an entity for which claims can be formulated. Claims are conveyed through **subject-property-value** relationships. The data model for claims, depicted in Figure 4 below, offers versatility and can accommodate a wide range of statements. Individual claims can be amalgamated (in multiple claims) to depict a network of information related to a subject. This introduces the concepts of a claim and a network of information. For claims to be deemed trustworthy, additional information must be appended to the network.

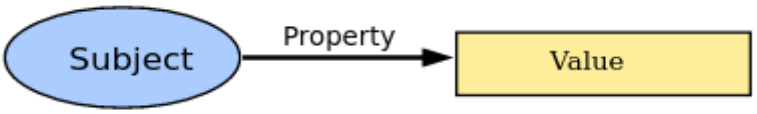

Figure 4: The Basic Structure of a Claim

For example, to express a claim, stating that someone has graduated from a particular university and extending this claim by adding new claims stating that this particular person knows another person that happens to be a professor at this university, can be illustrated below:

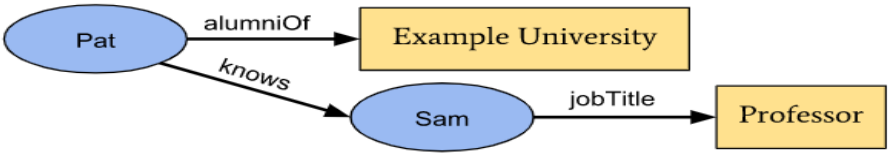

Figure 5: Multiple Claims can be Configured to Express a Graph of Information

A **credential** **[15]** constitutes a collection of one or more claims formulated by the same entity. These credentials may also encompass an identifier and metadata designed to elucidate credential attributes, including details like the issuer's identity, expiration date and time, a representative image, a public key for verification, the revocation mechanism, and more. The metadata could be signed by the issuer. A verifiable credential encompasses tamper-resistant claims and metadata that furnish cryptographic evidence of the issuer's identity.

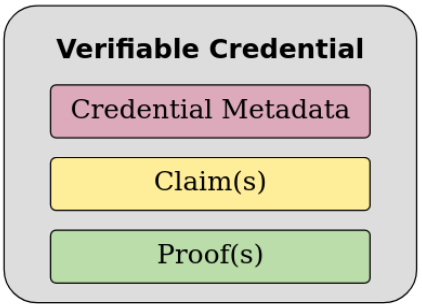

Figure 6: Basic Components of a Verifiable Credential

The figure 6 above shows the basic components of a verifiable credential, but abstracts the details about how claims are organized into information graphs, which are then organized into verifiable credentials. Figure 7, displayed below, offers a more comprehensive illustration of a verifiable credential, typically consisting of a minimum of two information graphs. The initial graph represents the verifiable credential, encompassing **metadata** and **claims**. The second graph represents the **digital proof**, often embodied as a **digital signature**.

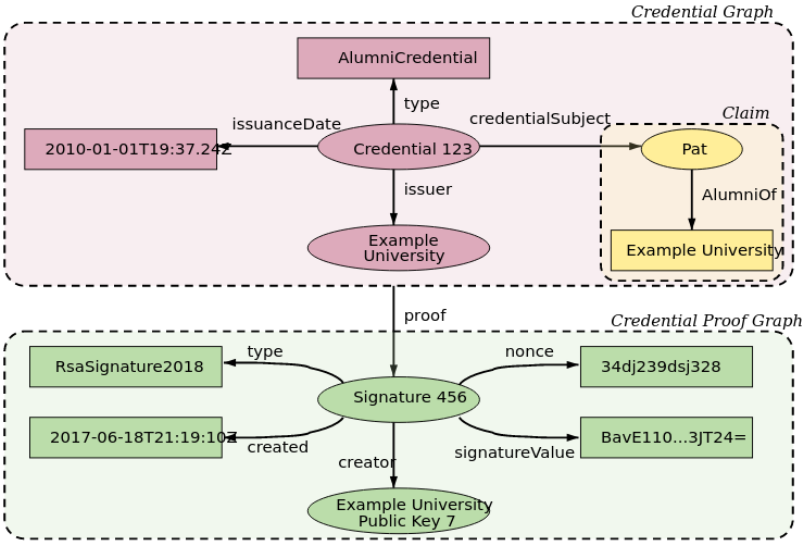

Figure 7: Information Graphs associated with a Basic Verifiable Credential

A verifiable **presentation** **[16]** communicates information extracted from one or multiple verifiable credentials in a manner that preserves the ability to verify the authenticity of the data's source. When verifiable credentials are presented directly, they assume the form of verifiable presentations. Additionally, data formats originating from verifiable credentials that are cryptographically verifiable, yet do not inherently contain verifiable credentials, can also qualify as verifiable presentations. The data within a presentation typically pertains to the same subject, although it may originate from multiple issuers. The amalgamation of this information typically portrays an aspect of an individual, organization, or entity, and it is often employed to enhance the subject's privacy.

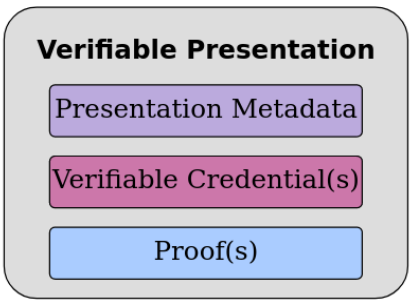

Figure 8: Basic Components of a Verifiable Presentation

The figure 8 above presents the components of a verifiable presentation, but abstracts the details about how verifiable credentials are organized into information graphs, which are then organized into verifiable presentations. Figure 9, depicted below, provides a more comprehensive representation of a verifiable presentation, typically consisting of at least four information graphs. The initial graph, the **Presentation Graph**, represents the verifiable presentation itself and includes presentation **metadata**. In the Presentation Graph, the **verifiableCredential** property points to one or more verifiable credentials, each represented by a self-contained Credential Graph, which serves as the second information graph. These Credential Graphs contain **credential metadata** and **claims**. The third information graph is the **Credential Proof Graph**, which articulates the **proof for the Credential Graph**, typically through a digital signature. Lastly, the fourth information graph is the Presentation Proof Graph, which conveys the **proof for the entire Presentation Graph**, generally in the form of a digital signature.

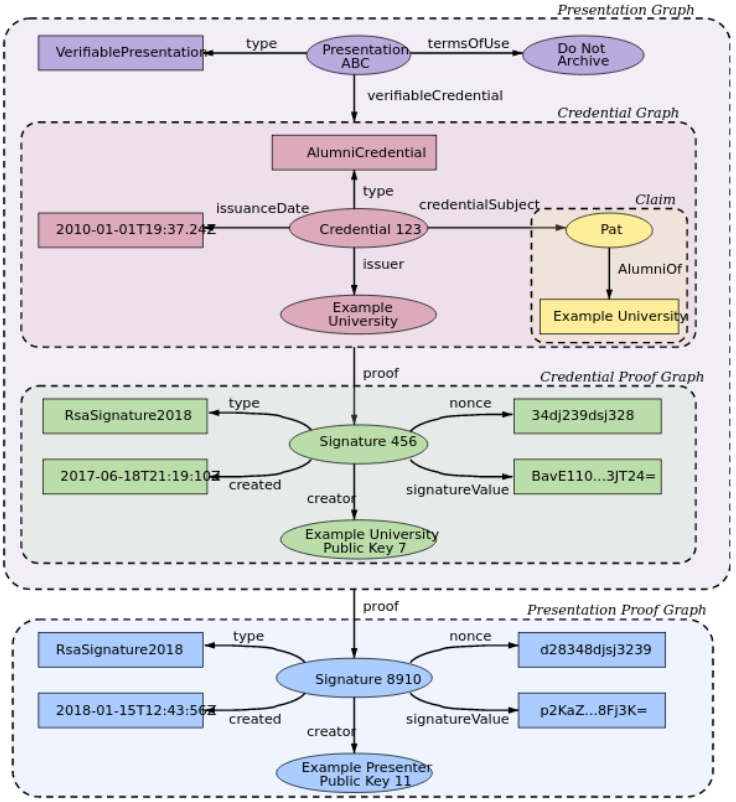

Figure 9: Information Graphs associated with a Basic Verifiable Presentation

The lifecycle of credentials and presentations in the Verifiable Credentials Ecosystem frequently follows a shared trajectory:

1. **Issuance of one or more verifiable credentials by one or multiple issuers.**
2. **Storage of these verifiable credentials in a credential repository (e.g., a digital wallet).**
3. **Compilation of these verifiable credentials into a coherent verifiable presentation for the benefit of verifiers.**
4. **Verification of the verifiable presentation's authenticity and claims by the verifier.**

To illustrate this lifecycle **[17]**, the example presented above, will be properly extended to match a much more realistic scenario of redeeming an alumni discount from a university. In the example below, Pat receives an alumni verifiable credential from the university and stores it in a digital wallet.

```
{
 // set the context, which establishes the special terms we will be using things such
 // as 'issuer' and 'alumniOf'.
 "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
 // specify the identifier for the credential
 "id": "http://example.edu/credentials/1872",
 // the credential type, which declare what data to expect in the credential
 "type": ["VerifiableCredential", "AlumniCredential"],
 // the entity that issues the credential
```

```
"issuer": "https://example.edu/issuers/565049",
// when the credential was issued by the issuer
"issuanceDate": "2010-01-01T19:23:24Z",
// claims about the subjects of the credentials
"credentialSubject": {
  // identifier for the only subject of the credential
  "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  // assertion about the only subject of the credential
  "alumniOf": {
     "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
     "name": [{
       "value": "Example University",
       "lang": "en"
     }, {
       "value": "Exemple d'Université",
       "lang": "fr"
     }]
  }
},
// digital proof that makes the credentials tamper-evident
"proof": {
  // the cryptographic signature suite that was used to generate the signature
  "type": "RsaSignature2018",
  // the date that the signature was created
  "created": "2017-06-18T21:19:10Z",
  // the purpose of this proof
  "proofPurpose": "assertionMethod",
  // the identifier of the public key that can verify the signature
  "verificationMethod": "https://example.edu/issuers/565049#key-1",
  // the digital signature value
  "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
     sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUc
     X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtj
     PAYuNzVBAh4vGHSrQyHUdBBPM"
}
}
```


Table 1: An example of a Verifiable Credential

Next, Pat endeavors to avail the alumni discount. The verifier, in this case, a ticket sales system, specifies that any alumni of **"Example University"** qualify for a discount on season tickets for sporting events. Pat initiates the season ticket purchase process using a mobile device. During one of the steps in this process, there's a requirement for an alumni verifiable credential, and this request is directed to Pat's digital wallet. The digital wallet prompts Pat to decide whether to provide a previously issued verifiable credential. Pat opts to furnish the alumni verifiable credential, which is then consolidated into a verifiable presentation. Subsequently, this verifiable presentation is transmitted to the verifier. The verifier receives the verifiable presentation and verifies that the claims that are presented and sealed in the presentation are valid and issued by the "**Example University**".


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "type": "VerifiablePresentation",
  // the verifiable credentials issued in the example above
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "id": "http://example.edu/credentials/1872",
    "type": ["VerifiableCredential", "AlumniCredential"],
    "issuer": "https://example.edu/issuers/565049",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
```

```
    "type": ["VerifiableCredential", "AlumniCredential"],
    "issuer": "https://example.edu/issuers/565049",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "alumniOf": {
        "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
        "name": [{
          "value": "Example University",
          "lang": "en"
        }, {
          "value": "Exemple d'Université",
          "lang": "fr"
        }]
      }
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2017-06-18T21:19:10Z",
      "proofPurpose": "assertionMethod",
      "verificationMethod": "https://example.edu/issuers/565049#key-1",
      "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
        sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUc
        X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtj
        PAYuNzVBAh4vGHSrQyHUdBBPM"
    }
  }],
  // digital signature by Pat on the presentation protects against replay attacks
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-09-14T21:19:10Z",
    "proofPurpose": "authentication",
    "verificationMethod": "did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",
    // 'challenge' and 'domain' protect against replay attacks
    "challenge": "1f44d55f-f161-4938-a659-f8026467f126",
    "domain": "4jt78h47fh47",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..kTCYt5
      XsITJX1CxPCT8yAV-TVIw5WEuts01mq-pQy7UJiN5mgREEMGlv50aqzpqh4Qq_PbChOMqs
      LfRoPsnsgxD-WUcX16dUOqV0G_zS245-kronKb78cPktb3rk-BuQy72IFLN25DYuNzVBAh
      4vGHSrQyHUGlcTwLtjPAnKb78"
  }
}
```

Table 2: An example of a Verifiable Presentation

2.2. The Hyperledger Fabric Blockchain Platform

Hyperledger Fabric [18] - also known as "**HLF**" or "**HF**" - is an open-source, permissioned blockchain framework initiated in 2015 by The Linux Foundation. This modular and versatile framework incorporates distinctive features for identity management and access control, making it applicable across various industry use cases, including supply chain tracking, trade finance, loyalty programs, and financial asset clearing and settlement.

Blockchain technology is a revolutionary system that facilitates the development of applications where multiple parties can directly record transactions, eliminating the need for a central authority to oversee and validate these transactions. This is achieved through a decentralized network where each participant gains access to a shared ledger for transaction recording. These transactions, by design, are immutable and cryptographically verifiable. In essence, blockchain technology comprises three fundamental components:

1. **Distributed Ledger:** A ledger is essentially a transactional log that meticulously records the entire history of data changes. These ledgers are intentionally designed to be immutable and append-only. Moreover, each member within the network can independently verify committed transactions. In the realm of blockchain, every network participant maintains a copy of this ledger.

2. **Consensus Algorithm:** Consensus algorithms are integral to ensuring that network members agree on the methodology for permitting transactions and data to be added to the ledger. They are crucial for the execution of smart contract code. If the predefined consensus requirements are not met, a transaction or operation is deemed invalid.
3. **Smart Contracts:** Smart contracts are pieces of code executed within the blockchain network. They serve to define the rules governing a business contract and are programmatically executed when the contract's predefined conditions are met. Smart contracts automate and enforce the terms of agreements, adding a layer of trust and security to transactions.

Some of the benefits of using the HLF are the following:

	Benefits	Description
1	**Open Source**	The HLF platform is an open-source blockchain framework that is hosted by The Linux Foundation. It boasts a vibrant and expanding community of developers who actively contribute to its development and growth.
2	**Permissioned**	HLF networks are permissioned, meaning that they are designed to ensure that the identities of all participants are known and authenticated. This attribute proves to be particularly beneficial in industries such as healthcare, supply chain management, banking, and insurance, where the protection of data from unknown or unauthorized entities is of paramount importance. For instance, consider an insurance company operating on a Hyperledger Fabric blockchain network. With this permissioned setup, the company can selectively share customer claim data with authorized parties while maintaining stringent control over access. This approach helps uphold customer privacy and data security, demonstrating the versatility and utility of the Hyperledger Fabric framework in various sectors.
3	**Governance and Access Control**	HLF networks are organized into channels, which function as private communication subnets among specific network members. These channels enable network members to engage in confidential transactions. Every blockchain transaction within the network occurs on a channel, with each participating party undergoing authentication and authorization to engage in transactions on that specific channel. This approach adds an extra layer of access control, proving valuable when members seek to restrict data exposure, particularly in situations where competitors share the same network. Additionally, Fabric provides a feature set known as Private Data Collection, allowing access to specific transactions on a channel to be limited to a subset of participants. This enhances data privacy and control within the network.
4	**Performance**	HLF is purpose-built to cater to enterprise-level use cases, offering robust support for such scenarios. It excels in ensuring rapid transaction throughput, thanks to its consensus mechanism. Notably, because Fabric operates as a permissioned blockchain framework, it

		sidesteps the need to address Byzantine Fault Tolerance (BFT), a concern that can lead to slower performance when validating transactions across a network. This design choice contributes to the efficiency and scalability of Hyperledger Fabric, making it well-suited for business-critical applications where speed and reliability are paramount.

Table 3: Benefits of using Hyperledger Fabric

A Hyperledger Fabric network is composed of distinct organizations or members that collaborate within the network. These organizations could represent entities such as banks in a financial network or shipping partners in a supply chain network. In the context of a Fabric network, each organization possesses a **Fabric Certificate Authority (CA)** and one or more peer nodes. Additionally, a Fabric network features an ordering service that is shared by all organizations involved, and this component plays a crucial role in processing transactions across the network.

In a network, an organization is characterized by a **Root Certificate Authority (Root CA)** dedicated to that specific organization. Users and other components, such as peer nodes, within that organization are likewise identified by certificates, and these certificates are generated from the root certificate. This process ensures that other organizations within the network can establish the affiliation of a user with their respective organization. These certificates also define the permissions for each entity on the network, specifying whether they have read-only or full access rights on a particular channel.

Within an organization, the root certificate is securely stored within the Fabric Certificate Authority (Fabric CA). This central component not only manages the root certificate but also takes charge of issuing certificates for individual users within the organization and performs various related functions. For enterprise-level deployments, a Fabric CA is a sophisticated system that comprises various components and offers flexibility in deployment options. Additionally, it can be set up to enhance security through the use of a **Hardware Security Module (HSM)**, which plays a crucial role in safeguarding the root certificate, adding an extra layer of protection and trustworthiness to the overall system.

An organization also establishes one or more peer nodes as integral components responsible for performing actions on behalf of that organization. These peer nodes play several key roles, including endorsing transactions submitted to the network, storing and executing smart contract code (referred to as chaincode in Fabric), and maintaining a local copy of the ledger for access. In the Fabric ecosystem, clients primarily interact with peer nodes to read the ledger, introduce new chaincode to the network, or propose new transactions. Typically, a peer node operates on its own computing environment, such as within Kubernetes Pods.

In a Fabric network, a fundamental component is the ordering service, which is shared by all members of the network. This ordering service plays a critical role in ensuring the orderly processing of new transactions within the network. It is responsible for arranging these transactions into new blocks and validating that they possess the necessary endorsements. Subsequently, the ordering service disseminates these newly formed blocks of transactions to the peer nodes located within each organization. The peer nodes, upon receiving these blocks, update their local copies of the ledger with the included transactions, thereby maintaining the consistency and integrity of the shared ledger across the entire network.

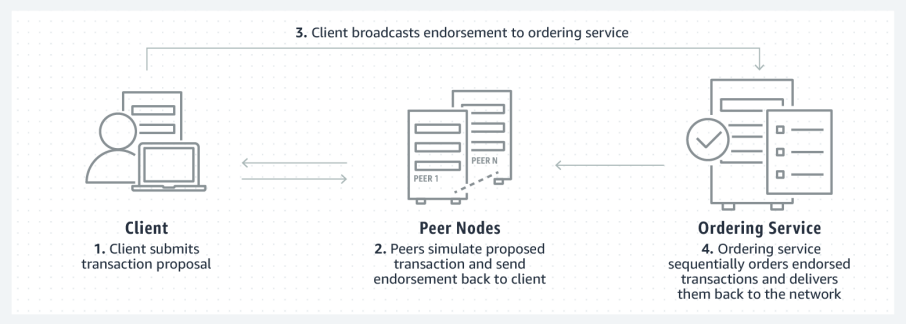

Figure 10: Hyperledger Fabric Transaction Flow

2.3. The Kubernetes Container Orchestration Platform

Kubernetes [19][20] - often abbreviated as "**k8s**" or simply "**kube**," is a container orchestration platform designed for efficiently scheduling, automating, deploying, managing, and scaling containerized applications.

Kubernetes originated from the efforts of Google engineers and was subsequently open-sourced in 2014. It can be traced back to its predecessor, Borg, an internal container orchestration platform utilized within Google. The name "Kubernetes" is of Greek origin, signifying **"helmsman" or "pilot"** which is why you'll find the helm prominently featured in the Kubernetes logo.

Figure 11: Kubernetes Official Logo

Today, Kubernetes and the broader container ecosystem are evolving into a versatile computing platform and ecosystem that competes with, and in many cases surpasses, virtual machines (VMs) as the foundational components of modern cloud infrastructure and applications. This ecosystem empowers organizations to deliver a high-productivity Platform-as-a-Service (PaaS) that tackles various infrastructure and operations tasks and challenges associated with cloud-native development. This allows development teams to concentrate solely on coding and innovation.

Containers, at their core, are lightweight and executable application components that bundle together application source code with all the necessary operating system (OS) libraries and dependencies required to run the code in any environment. They leverage a form of OS virtualization that enables multiple applications to share a single OS instance by isolating processes and regulating access to CPU, memory, and disk resources. Due to their smaller size, resource efficiency, and portability, containers have become the standard computing units for modern cloud-native applications.

In traditional infrastructure, applications run on physical servers, often competing for resources or wasting them if each application requires a dedicated server. VMs emerged as a solution, abstracting servers from the underlying hardware and allowing multiple VMs to run on one physical server. Each VM runs its own OS instance, providing isolation and resource

management. VMs improve resource utilization and scalability, and they can be easily scaled and disposed of when no longer needed.

Containers take this abstraction to a higher level by sharing not only the underlying virtualized hardware but also the OS kernel. They offer the same isolation, scalability, and disposability benefits of VMs but are lighter weight and more resource-efficient since they don't carry their own OS instances. Containers can be easily moved across different environments, including desktops, data centers, and cloud platforms.

As the use of containers grew, with organizations managing hundreds or thousands of them, the need for container orchestration arose. This led to the birth of the container orchestration market. While options like Docker Swarm and Apache Mesos gained some early traction, Kubernetes quickly became the most widely adopted. Developers were drawn to Kubernetes for its extensive functionality, expanding ecosystem of open-source tools, and its support and compatibility across various cloud service providers. All major public cloud providers, including AWS, Google Cloud, IBM Cloud, and Microsoft Azure, now offer fully managed Kubernetes services.

Kubernetes schedules and automates container-related tasks throughout the application lifecycle, including:

- **Deployment:** Kubernetes enables you to deploy a specified number of containers on a designated host and ensures they are maintained in the desired operational state.
- **Rollouts:** Kubernetes facilitates the management of rollouts, which represent changes to a deployment. It allows you to initiate, pause, resume, or even roll back rollouts as needed.
- **Service Discovery:** Kubernetes provides automatic service discovery, making it possible to expose containers to the internet or to other containers through DNS names or IP addresses.
- **Storage Provisioning:** Kubernetes allows you to configure the mounting of persistent local or cloud storage for containers based on your specific requirements.
- **Load Balancing:** Kubernetes offers load balancing capabilities that can distribute workloads across the network, either based on CPU utilization or custom metrics. This helps maintain performance and stability.
- **Autoscaling:** When there is a surge in traffic, Kubernetes can automatically scale by spinning up new clusters to handle the increased workload efficiently.
- **Self-Healing for High Availability:** Kubernetes ensures high availability by automatically restarting or replacing failed containers to prevent downtime. It can also terminate containers that do not meet your defined health-check criteria.

The chief components of Kubernetes architecture include the following:

1. **Clusters and nodes (compute)**

Kubernetes architecture revolves around clusters, which are the foundational components. These clusters are composed of nodes, with each node representing an individual computer host, whether it's a virtual machine or a physical one.

Each Kubernetes cluster consists of:

- **Master Node:** This node serves as the control center for the entire cluster. It manages and coordinates various operations within the cluster, such as scheduling,

scaling, and monitoring. The master node runs a scheduler service responsible for automating the deployment of containers based on developers' defined requirements and the available computing resources.

- **Worker Nodes:** These nodes are responsible for deploying, running, and managing containerized applications. Each worker node includes the container management tool, like Docker, and a critical component called the **Kubelet**. The Kubelet acts as a software agent that receives instructions and commands from the master node and ensures the containers are executed and managed as per the cluster's configuration.

In essence, Kubernetes clusters consist of a central master node for control and coordination, as well as multiple worker nodes that execute and oversee containerized applications, ensuring they run efficiently and according to the specified deployment rules. Developers manage cluster operations using ***kubectl***, a command-line interface (cli) that communicates directly with the Kubernetes API.

2. Pods and deployments (software)

In Kubernetes, pods are a fundamental concept. They are groups of containers that share both the same compute resources and network namespace. Pods serve as the smallest deployable unit in Kubernetes, and they have several important characteristics:

- **Resource Sharing:** Containers within a pod share the same resources, such as CPU and memory, making them tightly coupled. This allows them to communicate with each other easily, as they are on the same network.
- **Scalability:** Pods are the unit of scalability in Kubernetes. When a container within a pod experiences increased traffic or resource demand, Kubernetes can replicate the entire pod across other nodes in the cluster to handle the load. This replication helps distribute the workload effectively.
- **Resource Containment:** It's a recommended best practice to keep pods compact. This means placing only containers that need to share resources within the same pod. This way, you can ensure efficient resource utilization and maintain isolation between different components of your application.

In addition to pods, Kubernetes uses a higher-level resource called a "Deployment" to manage containerized applications. A Deployment defines the desired state of the application, including how many replicas of a pod should be running in the cluster. If a pod fails or needs to be updated, the Deployment controller automatically takes care of creating new pods to maintain the desired state. This ensures that your application is both highly available and always running as intended.

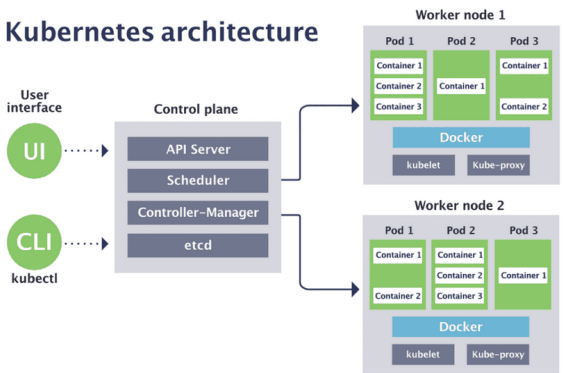

Figure 12: Kubernetes Architecture

2.4. The gRPC Protocol

gRPC [21] [22] is a modern and robust open source high performance **Remote Procedure Call (RPC)** framework that can run in any environment. It was developed by Google in 2015 as an extension of the RPC network in the effort to link many microservices created with different technologies. Initially, it was closely associated with Google's internal infrastructure, but later, it was made open-source and standardized for community use.

It supports both typical request/response interactions and long-running streaming communications. It can efficiently and effortlessly connect services in and across data centers with pluggable support for load balancing, tracing, health checking, custom encoding/decoding, and even authentication. It is also applicable in the last mile of distributed computing to connect devices, mobile applications and browsers to backend services. It allows the client and server applications to communicate transparently and develop connected systems. gRPC is very popular in service to service calls, as often HTTP calls are harder to understand at first glance. Many leading tech firms have adopted gRPC, such as Google, Netflix, Square, IBM, Cisco, & Dropbox. This framework relies on HTTP/2, protocol buffers (a leading technology that performs better than JSON and XML), and other modern technology stacks to ensure maximum API security, performance, and scalability.

Most of the gRPC advantages are the following:

	Advantages	Description
1	**Protocol Buffers (Protobuf)**	Protobuf is Google's serialization/deserialization protocol, designed to simplify the definition of services and enable the automatic generation of client libraries. gRPC, a high-performance remote procedure call (RPC) framework, utilizes Protobuf as its Interface Definition Language (IDL) and serialization toolset. The current version of Protobuf is **proto3**, which offers the latest features and is known for its user-friendliness. In gRPC, services and the message structures exchanged between clients and servers are defined using .proto files. The Protobuf compiler, often referred to as **"protoc"** is responsible for generating client and server code. This generated code loads the **.proto** file into memory at runtime, utilizing the in-memory schema to efficiently serialize and deserialize binary messages. The serialized data is structured using protocol buffer message types, which are defined within **.proto** files. Each protocol buffer message serves as a concise logical record, comprising a series of name-value pairs that organize the information. After the code generation process, the compiler produces data access classes and methods that facilitate connections to gRPC services. Only once this code generation is complete can messages be effectively exchanged between the client and the remote service, enabling efficient and reliable communication in gRPC-based applications.

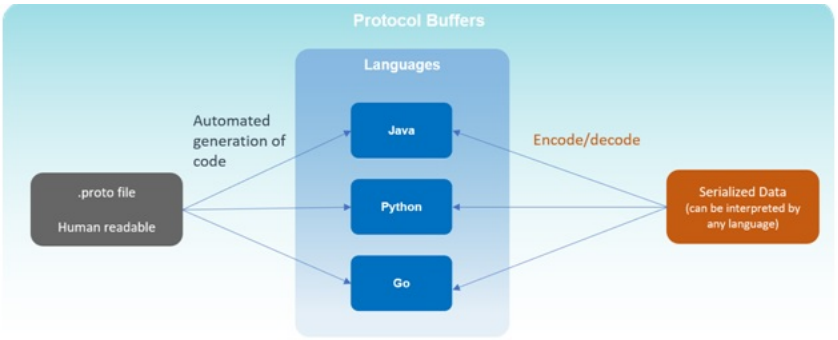

Figure 13: Protocol Buffer Operation

The complete workflow demonstrates that Protobuf provides several notable advantages over JSON and XML. Parsing with Protobuf offers benefits such as reduced CPU resource usage because data is converted into a binary format. Additionally, encoded Protobuf messages are more compact in size. Consequently, this leads to faster message exchange, even on devices with less powerful CPUs, such as mobile devices. Protobuf's efficiency in terms of both CPU utilization and message size makes it a preferred choice for optimizing data transmission and processing in various applications.

It supports multiple languages like **C# / .NET**, **C++**, **Dart**, **Go**, **Java**, **Kotlin**, **Node**, **Objective-C**, **PHP**, **Python**, **Ruby** and more. Since it is technology-agnostic, both server, and clients that are connected to the server, can be developed using different programming languages.

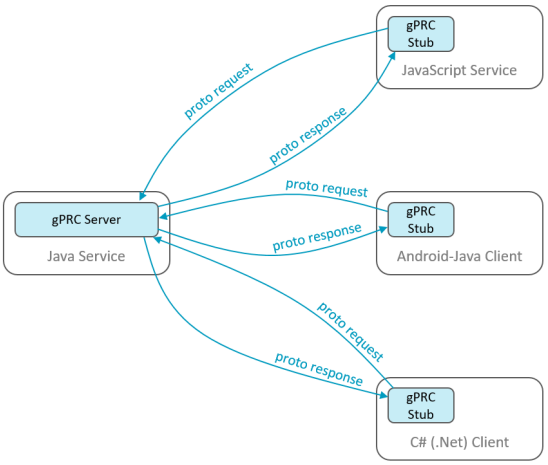

Figure 14: gRPC Clients and Servers can run and talk to each from Different Environments

2	**Streaming**	Streaming is a pivotal concept in gRPC, allowing multiple processes to occur within a single request. This capability is made possible by the multiplexing feature of HTTP/2, which enables the simultaneous transmission of multiple responses or the reception of multiple requests over a single TCP connection. In gRPC, there are several main types of streaming: **1) Unary RPCs (No Streaming) [23]** - The client sends a single request and the server responds with a single message.

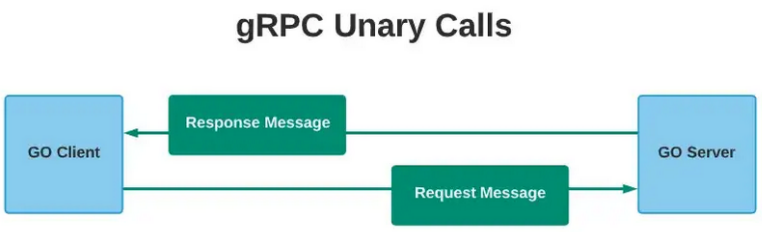

Figure 15: gRPC Unary RPC

2) Server-streaming RPCs [24] - The client sends a single request to the server and receives back a stream of data sequences. The sequence is preserved, and server messages continuously stream until there are no messages left

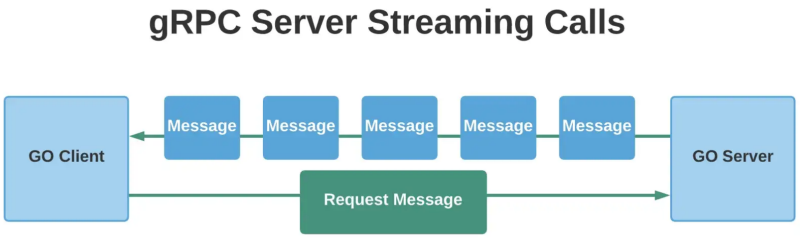

Figure 16: gRPC Server Streaming RPC

3) Client-streaming RPCs [25] - The client sends a stream of data sequences to the server, which then processes and returns a single response to the client. Once again, gRPC guarantees message sequencing within an independent RPC call.

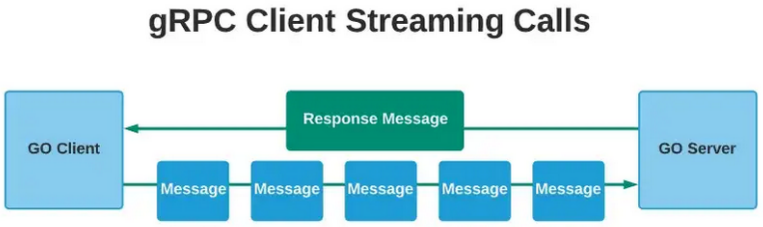

Figure 17: gRPC Client Streaming RPC

4) Bidirectional-streaming RPCs [26] - It is two-way streaming where both client and server sends a sequence of messages to each other. Both streams operate independently; thus, they can transmit messages in any sequence. The sequence of messages in each stream is preserved.

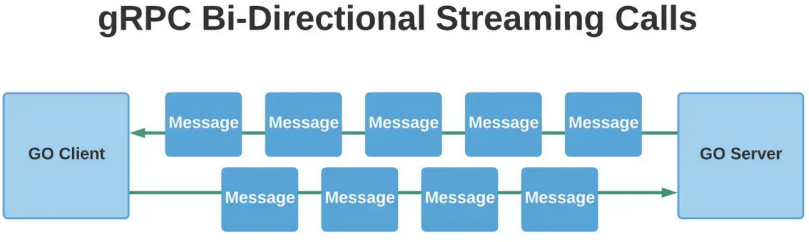

Figure 18: gRPC Bidirectional Streaming RPC

3	**HTTP/2**	gRPC is built on top of HTTP/2, which was introduced in 2015 to address the limitations of HTTP/1.1. While HTTP/2 remains backward-compatible with HTTP/1.1, it introduces several advanced features that significantly enhance web communication: **1) Binary Framing Layer:** HTTP/2 employs a binary framing layer, which breaks down requests and responses into smaller binary messages. This binary framing improves the efficiency of message transmission. By using binary framing, HTTP/2 enables request/response multiplexing without tying up network resources. **2) Streaming:** HTTP/2 supports bidirectional full-duplex streaming, allowing both the client and server to send and receive messages simultaneously. This capability is particularly useful for real-time communication and interactive applications. **3) Flow Control:** HTTP/2 incorporates a flow control mechanism that provides granular control over the allocation of memory used to buffer in-flight messages. This ensures efficient resource management. **4) Header Compression:** In HTTP/2, all aspects of the protocol, including headers, are compressed before transmission. This header compression, implemented using the HPACK method, significantly enhances overall performance by reducing the size of transmitted data. Only differences between successive HTTP header packets need to be shared. **5) Processing Models:** HTTP/2 enables gRPC to support both synchronous and asynchronous processing. This versatility allows gRPC to handle various types of interactions and streaming RPCs, accommodating a wide range of use cases. These features of HTTP/2 provide several benefits, including resource efficiency, reduced response times between applications and services in cloud environments, and extended battery life for clients running on mobile devices. gRPC leverages these capabilities to deliver high-performance and efficient communication between distributed systems, making it a popular choice for modern applications and microservices architectures.
4	**Channels**	Channels are a fundamental concept in gRPC. While HTTP/2 streams enable multiple concurrent streams on a single connection, channels take this concept a step further by supporting multiple streams over multiple concurrent connections. Channels in gRPC serve as a means to establish connections to a gRPC server at a specified address and port. They are instrumental in creating a client stub, which is a client-side representation of a remote gRPC service. These channels help manage the underlying connections, handling tasks such as load balancing, connection pooling, and failover, making it easier for clients to interact with gRPC services efficiently and reliably. In the following gRPC architecture diagram, we have the gRPC client and server sides. The client has a stub that defines remote procedures. When the client wants to make a remote call, it uses the stub, which serializes the data and sends it to the local gRPC library. The server deserializes the data, processes the request, and sends back a response. This architecture enables efficient cross-language communication.

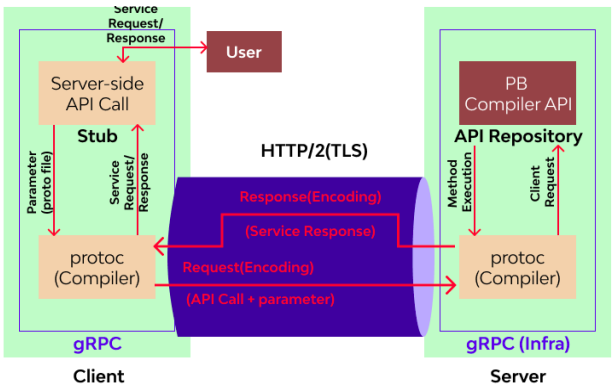

Figure 19: gRPC Architecture

The client OS makes an HTTP/2 call to the server. The server's OS handles the request, invoking the server stub procedure that decodes parameters and executes the corresponding action using Protobuf. The server stub sends the response back to the client. The client stub processes the result and returns it to the caller.

Table 4: Benefits of using gRPC

gRPC introduces a fresh perspective on the traditional Remote Procedure Call (RPC) design method and brings several strengths and advantages that have contributed to its increasing adoption. Some of the key strengths of gRPC include:

	Benefits	Description
1	**Performance**	According to various assessments, gRPC outperforms REST with JSON by delivering up to **10 times faster performance** and bolstered API security. This enhanced performance is achieved through the utilization of Protobuf and the adoption of HTTP/2. Protobuf efficiently serializes messages on both the server and client sides, resulting in concise and streamlined message payloads. Meanwhile, HTTP/2 elevates performance through features like **server push, multiplexing, and header compression**. Server push allows HTTP/2 to proactively transmit content from the server to the client before it's explicitly requested, while multiplexing eliminates the issue of head-of-line blocking. Additionally, HTTP/2 employs a more advanced compression technique, reducing message sizes and thereby accelerating loading times.
2	**Streaming**	gRPC offers support for streaming semantics, which can be applied either on the client side or the server side. These streaming semantics are defined directly in the service definition, simplifying the process of building streaming services or clients. With gRPC, a service can utilize various streaming combinations through the capabilities of HTTP/2, including: **1) Unary (no streaming)** **2) Client-to-server streaming** **3) Server-to-client streaming** **4) Bi-directional streaming**

3	**Code Generation**	The primary advantage of the gRPC approach is its inherent capability to automatically generate code for building client and server applications. gRPC frameworks employ the **protoc** compiler to create code based on the definitions outlined in the .proto file. This code generation process leverages the Protobuf format for specifying message structures and service endpoints. It results in the creation of foundational server-side structures and client-side networking components, streamlining development efforts, especially in applications featuring multiple services.
4	**Interoperability**	gRPC offers a versatile set of tools and libraries that are engineered to function seamlessly across a variety of platforms and programming languages. This includes popular languages like **Java, JavaScript, Ruby, Python, Go, Dart, Objective-C, C#**, and many others. Thanks to the utilization of the efficient Protobuf binary wire format and code generation capabilities that extend to virtually all platforms, developers have the flexibility to create high-performance applications while benefiting from comprehensive cross-platform compatibility and support.
5	**Security**	gRPC prioritizes API security through the utilization of HTTP/2 combined with end-to-end encryption via TLS. This approach ensures robust security for APIs. gRPC strongly advocates for the adoption of SSL/TLS protocols to achieve both authentication and encryption of data transmitted between the client and server. Additionally, gRPC offers support for various security mechanisms, including OAuth2 implementations and JWT tokens. These mechanisms enable authentication of clients and servers connected to the service by passing relevant information in the request headers. This comprehensive approach to security enhances the protection of data and resources within the gRPC-based communication, bolstering overall API security.
6	**Usability and Productivity**	gRPC stands out as a comprehensive RPC solution that seamlessly operates across diverse programming languages and platforms. Furthermore, it offers robust tooling capabilities, automating a significant portion of the necessary boilerplate code. This automation not only results in substantial time savings but also empowers developers to allocate their efforts more effectively toward implementing and refining business logic, rather than getting bogged down in repetitive coding tasks.
7	**Built-in Commodity Features**	gRPC comes equipped with native support for essential features commonly used in software development, including metadata exchange, encryption, authentication, deadline/timeouts and cancellations, interceptors, load balancing, service discovery, and a plethora of other capabilities.

		This comprehensive feature set simplifies the implementation of these critical functionalities, allowing developers to efficiently build robust and secure distributed systems without the need for extensive custom coding.

Table 5: Strengths of using gRPC

As with every other technology, gRPC also has the following weaknesses that you need to be aware of when choosing it for developing applications.

	Drawbacks	Description
1	**Limited Browser Support**	Due to gRPC's strong reliance on HTTP/2, making a direct call to a gRPC service from a web browser is not feasible. Web browsers do not offer the level of control required for native gRPC client functionality. Therefore, an intermediary proxy layer and the use of gRPC-web are necessary to bridge the gap and facilitate the conversion between HTTP/1.1 (browser-compatible) and HTTP/2 (gRPC-compatible) protocols. Even with the proxy and gRPC-web in place, there are inherent limitations when it comes to the types of streaming requests and responses that can be effectively translated between web browsers and gRPC servers. As a result, this solution currently supports only Unary RPCs (single request, single response) and Server-Side RPCs (streaming from server) for web browser interactions, while more complex streaming scenarios may not be fully supported.
2	**Non-human Readable Format**	Protobuf is responsible for compressing gRPC messages into a format that is not human-readable. To deserialize these messages correctly, the compiler requires the message's interface description provided in the file. As a result, developers often find it necessary to utilize supplementary tools like the gRPC command-line tool. This tool serves various purposes, including the analysis of Protobuf payloads during transmission, the manual crafting of requests for testing and debugging purposes, and the facilitation of debugging tasks related to gRPC-based communication. These tools prove invaluable for understanding and interacting with the non-human-readable data format employed by Protobuf and gRPC.
3	**No Edge Caching**	In contrast to HTTP, where edge caching intermediaries are supported, gRPC calls utilize the POST method, which poses a challenge to API security. The nature of POST requests means that responses cannot be cached through intermediaries, limiting caching opportunities. Furthermore, it's worth noting that the gRPC specification doesn't include provisions for caching semantics between the server and client. In fact, the specification doesn't express a desire for cache semantics in this context. This means that caching strategies for

		gRPC-based communication need to be carefully considered and implemented separately, as they are not an inherent part of the gRPC protocol.
4	**Steeper Learning Curve**	Learning and adopting gRPC can indeed pose challenges for many development teams. These challenges often include the need to become familiar with Protocol Buffers (Protobuf), the complexities of dealing with HTTP/2 intricacies, and the search for appropriate tools to facilitate these tasks. Consequently, some users may opt to stick with RESTful APIs for an extended period because they are more comfortable with the established REST paradigm. The preference for REST can stem from its familiarity and the abundance of existing tools and resources available for working with RESTful APIs. However, **it's important to recognize that while gRPC may have a learning curve, it offers significant benefits in terms of performance, efficiency, and features once developers become proficient with it**. Therefore, the decision to transition from REST to gRPC often depends on the specific needs and constraints of a project, as well as the willingness of the development team to invest in learning and mastering the gRPC ecosystem.

Table 6: Weaknesses of using gRPC

2.5. The GOB Binary Data Format

Gob [27] is a Go-specific serialization method tailored for encoding Go data types exclusively. It lacks compatibility with other programming languages and is primarily used for encoding and decoding Go data. Gob packets facilitate the transmission of binary data streams between encoders (transmitters) and decoders (receivers) within Go programs. While Gob is versatile in handling various Go data types, it does not support channels and functions.

Gob operates in a manner similar to JSON. When sending data, the sender employs an Encoder to serialize the data structure. Upon receiving the message, the receiver uses a Decoder to deserialize the serialized data into a local variable.

What sets Gob apart from JSON is its ability to serialize struct methods that JSON does not support. Gob offers a means of serializing and deserializing Go data types without requiring the addition of string tags to structs, dealing with JSON compatibility issues, or waiting for JSON serialization and deserialization processes to complete.

This makes Gob a more versatile choice for handling Go data structures, especially when struct methods need to be included in the serialization process.

The structure of the sender and the structure of the receiving party do not need to be fully consistent or identical. Below, there are some examples originated from the official documents:

```
// Defining a structure
struct { A, B int }

// The following types of data can be sent and received:
struct { A, B int }    // the same
*struct { A, B int }   // extra indirection of the struct
struct { *A, **B int } // extra indirection of the fields
struct { A, B int64 }  // different concrete value type; see below

// The following types can also receive:
struct { A, B int }    // the same
struct { B, A int }    // ordering doesn't matter; matching is by name
struct { A, B, C int } // extra field (C) ignored
struct { B int }       // missing field (A) ignored; data will be dropped
struct { B, C int }    // missing field (A) ignored; extra field (C) ignored.

// The following format is problematic:
struct { A int; B uint }  // change of signed ness for B
struct { A int; B float } // change of type for B
struct { }                // no field names in common
struct { C, D int }       // no field names in common
```

Table 7: GOB Official Data Format Examples

Gob incorporates type information into its serialized data. This type information is included only once for each piece of data. For example, when serializing a struct, Gob includes the name of the struct field. This design eliminates the need to create separate files or additional documentation to explain the structure of the data, a requirement often encountered when exchanging protobuf messages.

The inclusion of type information in Gob's serialization process makes it quite resilient to changes. This property means that, with Gob, it's possible to decode a Gob stream stored in a file, even long after the original context and data type have been forgotten. This robustness to changes simplifies data handling and provides flexibility in decoding data, even in scenarios where the type information has been lost or is no longer readily available.

So when gob is used to encode, an internal struct that describes the type gives it a unique number. For example:

```
type T struct{ X, Y, Z int }
var t = T{X: 7, Y: 0, Z: 8}
```

Thus when the first type T is sent, the gob encoder sends a description of T and tags it with a type number (for example, the number 127). All values, including the first, are then prefixed by that number, so a stream of T values looks like the following:

```
("define type id" 127, definition of type T)(127, T value)(127, T value), ...
```

Gob, despite being a very lightweight and robust package, it is also a simple one. It implements a small number of functions, but in our case the following functions were used:

	Function	Description
1	`func Register(value interface{})`	Register records the type and name of the specific value. This name will be used to identify the specific type at the lower level when sending or receiving interface type values. This function should only be

		called during initialization. If the mapping of type and name is not one-to-one, it will panic.
2	`func NewEncoder(w io.Writer) *Encoder`	NewEncoder returns a *Encoder that writes encoded data to w.
3	`func (enc *Encoder) Encode(e interface{}) error`	The Encode method encodes e before sending, and ensures that all types of information are sent first.
4	`func NewDecoder(r io.Reader) *Decoder`	The function returns a * Decoder that reads data from R, if R does not satisfy io.ByteReader Interface will wrap r as bufio.Reader.
5	`func (dec *Decoder) Decode(e interface{}) error`	Decode reads the next one from the input stream and stores the value in e. If e is nil, the value is discarded, otherwise e must be a pointer to the type that can receive the value. If the input ends, the method returns **io.EOF** and doesn't change e.

Table 8: GOB Official Function Signatures and Descriptions

3. Literature review

Identity management has become increasingly important in the digital era we live in, and Self-Sovereign Identity (SSI) based on blockchain technology has emerged as a promising solution to address privacy, security, and control issues. Casper, a mobile identity wallet application that stores users' identities on their mobile devices, and blockchain-based decentralized storage have been proposed to address privacy concerns and security issues with centralized identity systems **[28]**. The authors in **[29]** propose a self-sovereign identity authentication (MFFSIA) solution using a blockchain-based implementation of a formal protocol with a marketplace for challenge-set creation and decentralized knowledge graphs and oracles for response evaluations. The solution aims to address the limitations of existing identity authentication systems and offer a flexible, decentralized solution for secure data exchange. In **[30]** the authors propose an SSI integrated framework with blockchain technology for unmanned aerial vehicle (UAV) delivery systems, providing efficient user identity management and identity verification processes. The proposed framework can protect edge computing-based UAV delivery systems against security flaws and privacy concerns. Blockchain technology and the SSI model utilizing Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) have been proposed as a decentralized digital identity framework for Industrial Internet-of-Things (IIoT) to avoid the drawbacks of traditional centralized approaches **[31]**. Furthermore, a self-sovereign decentralized identity platform, SSIChain, has been proposed to use blockchain to replace centralized trusted authorities and third-party operators **[32]**. Another paper proposes a blockchain-based SSI approach for privacy-preserving Inter-Organizational Business Processes (IOBP), combining SSI with a registry proof smart contract to provide a privacy-preserving solution **[33]**.

A self-sovereign identity management system called BSelSovID has also been proposed to provide security, privacy, and control to users over their digital identities, with their digital identity stored on the interplanetary file system (IPFS) with content addresses stored on the blockchain **[34]**. Moreover, a digital identity management system utilizing SSI-based models, blockchain, and Inter Planetary File System (IPFS) has been proposed to support food supply chains and ensure eligibility, transparency, and traceability of the certifications along the food supply chain **[35]**. The need for identity management in the digital age has led to the emergence of self-sovereign identity based on distributed ledger technology, such as the framework based on Hyperledger Indy proposed to identify scam bank service calls and enhance the trust of third-party agencies delegated by banks to deliver financial products **[36]**. Additionally, Siddhi, a blockchain-based solution for Cyber Threat Information (CTI) sharing, has been proposed to enable traceability, anonymization, and data provenance in a scalable manner while preserving the privacy and anonymity of the CTI participants **[37]**. In **[38]**, the authors propose Sovereign Identity, which is an unrestricted identity that is completely under the control of the user. They suggest using blockchain technology, with a focus on the Ethereum platform, smart contracts, Ganache server, Truffle framework, Metamask, and Zero Knowledge Proof (ZKP), to develop this type of identity. The paper emphasizes the need for a secure and decentralized solution to identity management, and suggests that blockchain technology can offer a promising solution to this problem.

Blockchain technology has the potential to provide a secure and trusted solution, and this has led to the emergence of Self-Sovereign Identity (SSI)-based systems, which are completely under the control of the user. One such system is the SSIDD access control system, which uses blockchain technology to build trust and ensure user privacy, providing

high access policy flexibility and security for global inter-enterprise collaborations from a diverse industrial environment **[39]**. The Connect platform is a blockchain and SSI-based digital contact tracing platform that encodes user's digital identities and activity trace data on a permissioned blockchain platform, ensuring user privacy while empowering them to share information. The platform enables the identification of suspected patients and the notification of the local community in real-time, thereby reducing the rate at which the infection could spread **[40]**. To improve the security and privacy of self-sovereign identities and increase confidence in the overall system, a study proposes enhancements including a model for attribute sensitivity, a method for mitigating man-in-the-middle attacks, and a quantitative model for determining a credential issuer's reputation **[41]**. Similarly, EverSSDI is a framework that uses smart contracts on an Ethereum-based blockchain to provide a unique identifier to normalize user identities, a fine-grained authorization mechanism, and a reliable information verification mechanism, which enables the user to become the dominant owner of their digital identity **[42]**. The retrospective verification of transactions associated with a previous version of the identity is addressed by the self-sovereign dynamic digital identity scheme that uses blockchain technology to create and manage dynamic digital identities that consist of a combination of biometric traits and other identity attributes **[43]**. The proposed Identity Management System (IMS) based on blockchain technology eliminates the need for centralized authorities and instead generates cryptographic identities, enabling the sharing of user data on a permissioned blockchain with identity-based encryption to maintain access control and data security **[44]**.

The self-sovereign identity management system proposed in **[45]**, enables users to manage their own data and generate their private and public keys, with blockchain and chameleon hash eliminating unauthorized access to the blockchain. Similarly, the blockchain-based system proposed by another study allows citizens to actively participate in the management of their personal data, featuring a granular access control mechanism that ensures the privacy and integrity of personal data **[46]**. MediLinker is a patient-centric decentralized health identity management system that uses blockchain technology to manage verifiable credentials issued by healthcare clinics, banks, and insurance companies, enabling patients to register and share their information securely and in a trusted system with healthcare and other service providers **[47]**. These self-sovereign identity-based systems and blockchain technologies provide users with the security, privacy, and control over their digital identities that are necessary for a trusted and decentralized identity management system.

Overall, the proposed self-sovereign identity systems aim to improve security and interactions over networks, provide secure, decentralized, and verifiable identity, and ensure user privacy while also empowering them to share information in a trusted system. These solutions can be used to address issues of trustworthiness, access policy flexibility, and user privacy preservation in data sharing networks, supply chain management, healthcare, and various other domains. It should be noted, also, that several limitations and challenges exist with respect to the aforementioned proposed systems. For example, some systems may have high overhead and latency or require significant computational resources. Additionally, there may be issues with interoperability, scalability, and standardization across different blockchain platforms and SSI systems.

4. System design

4.1. Proposed Architecture

As mentioned in the Introduction section above, the scope of this dissertation is to solve a major interoperability problem, by mitigating a **Hyperledger Aries Verifiable Data Registry (VDR)**, which until now is depending on Hyperledger Indy, to a more widely adopted Blockchain platform used by many corporations. This limitation prevents Hyperledger Aries from using other blockchains as a VDR. Currently, the Blockchain platform with the most adoption at the moment especially for industrial applications, is without a doubt the **Hyperledger Fabric**. Businesses developing solutions with Fabric would like their platforms to support identity verification services, without forcing them to deploy and maintain two different Blockchain networks and eventually increasing the costs of the infrastructure. Until now, SSI is used in Hyperledger Aries with a specific VDR package which limits its potential implementation. The scope of this dissertation is to solve this problem by modifying **Hyperledger Aries Go** while creating and eventually adding a new VDR package, so that it can now support **Hyperledger Fabric** or any other underlying blockchain platform as a ledger. This could lead to a wider adoption of SSI technology and also help overcome interoperability issues.

In the following image, the high level architecture of the proposed solution is depicted:

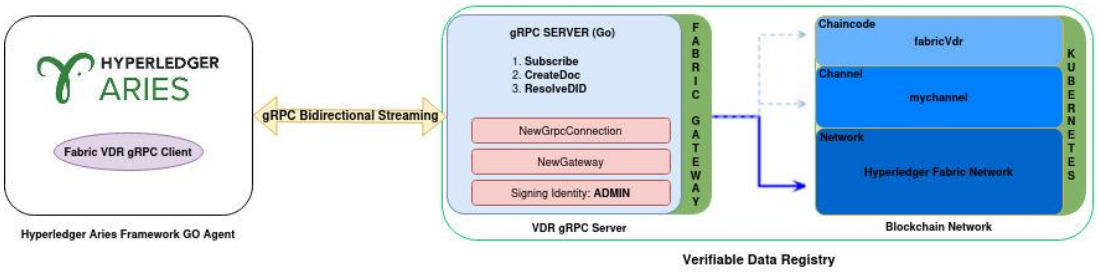

Figure 20: Proposed Solution High Level Architecture

As presented in the architecture above, two major and necessary components were envisioned for this solution:

1. A **VDR** that consists of two components:
    a. A **Kubernetes deployed Hyperledger Fabric Blockchain network**, that hosts the **fabricVdr** smart contract, containing the business logic of the basic **CRUD** operations that are required to be implemented, in order for this component to be eligible to be called VDR. The requirements and the definitions of the operations were defined and eventually extracted after a thorough examination of the **Hyperledger Aries Framework Go**.
    b. A **Bidirectional Long-Lived Streaming gRPC Server**, that facilitates all the communications and handles all the incoming requests originated from the VDR gRPC client located in the **Hyperledger Aries Framework Go**, in the form a **VDR package**. The server is designed to handle the basic operations that are implemented in the smart contract, as well as operations that are required to handle long-lived streams.
2. A **VDR gRPC client**, that connects to the long-lived bidirectional streaming VDR gRPC Server by executing two goroutines simultaneously, one for receiving events from the stream and one for sending requests to the stream. This VDR gRPC client will

later be packaged and imported as a new **VDR** in the **Hyperledger Aries Framework Go**.

All the communications among the components will be protected through the use of **mutual TLS**. In the following image a Detailed version of the High Level Architecture that presents all the interactions and the relationships between the aforementioned components is clearly presented:

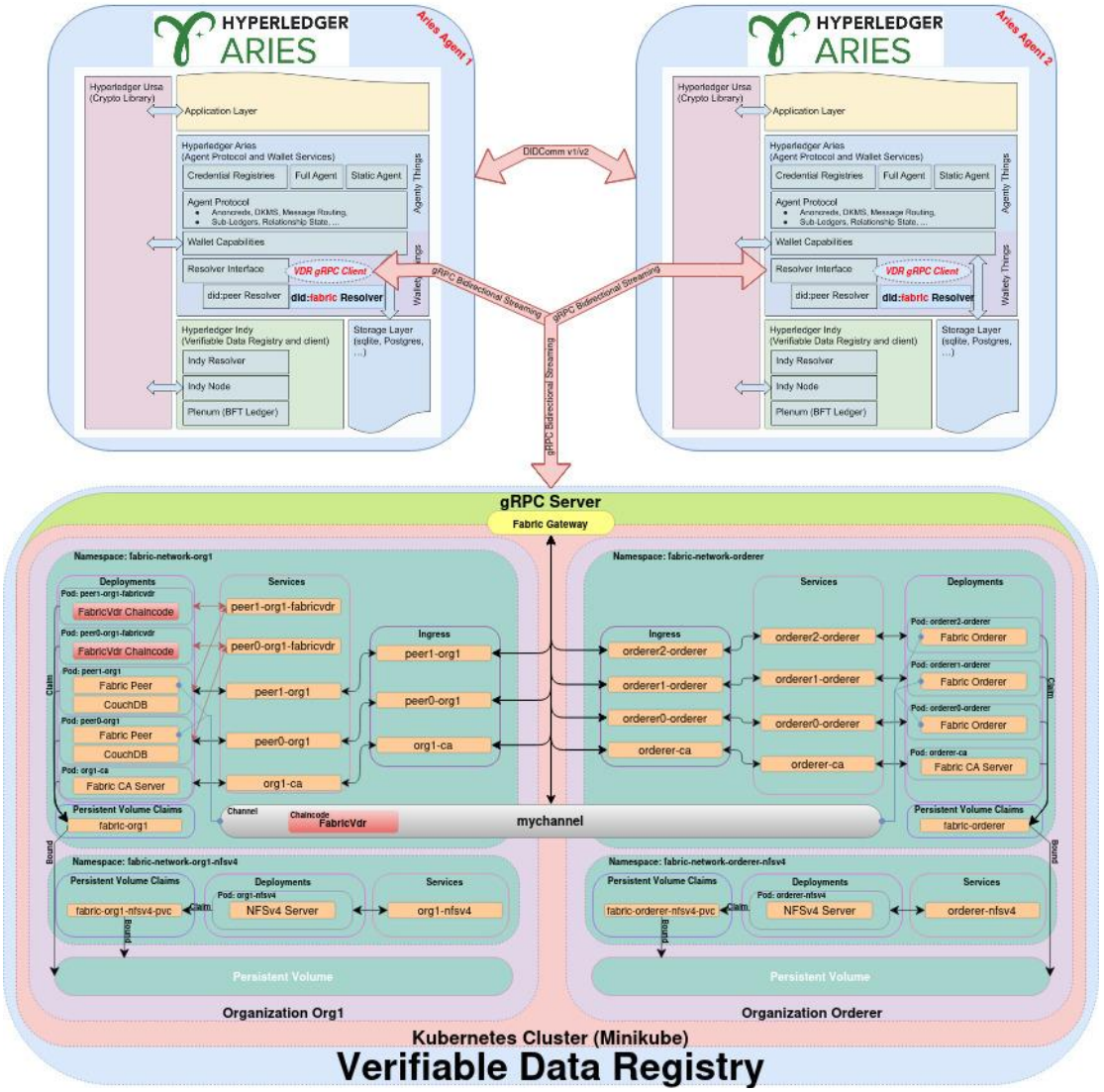

Figure 21: Proposed Solution Detailed Architecture

As depicted in the aforementioned image, **Hyperledger Aries Framework Go** will host the new **VDR** package that implements all the necessary functions that are required to connect with the **long-lived bidirectional streaming VDR gRPC Server** and thus to be able to interact with the **Hyperledger Fabric network hosted in the Kubernetes**.

For the purpose of this thesis **two Organizations (the Org1 and the Orderer Organization)** as well as two **NFSv4 Servers** were deployed through the use of a custom created project called **multihost_k8s**, which minimize the user effort to design, create and maintain a full operational Hyperledger Fabric Blockchain network, which by nature is a quite heavy and intense task.

4.2. Smart Contracts

4.2.1. Design

Since we want to implement a Hyperledger Aries Verifiable Data Registry (VDR) component based on the Hyperledger Fabric Private Blockchain Network, then specific operations and rules must be applied. A VDR must have some basic **Create-Read-Update-Delete** (CRUD) operations and these operations are essentially designed and translated into a smart contract written in Golang and eventually deployed in the HF network in Kubernetes.

In the following table the basic operations of a VDR (inspired by the official Aries Framework GO interface located here), the implemented function signatures as well as the status of the implementation are presented:

Smart Contract Function Signatures				
\	Function	Signature	Description	Done
1	Create **(Register)**	`func CreateDoc(clientId string) error`	**A create operation for a DID Document** ***Input***: 1. ClientId (type string), coming from the agent and pointing out to the agent/client that the request originated from. ***Transient Input:*** 1. The **docResolution** (type ***did.DocResolution** struct as **bytes**) coming from the gRPC server 2. The **did** (type string as **bytes**), coming from the gRPC server and points out to the aforementioned *did.DocResolution. ***Output***: The chaincode as function returns nil, but emits a chaincode event with the following type: **"{DeviceID=<clientId>}, {DID=<did>}"**. The Fabric Gateway (located in the gRPC server) will catch the event and disseminate the DocResolution back to the Hyperledger Aries Framework Go Agent.	✔
2	Read **(Resolve)**	`func ResolveDid(did string) (string, error)`	**A read operation for DID Documents** ***Input***: 1. **Did** (type **string**), the did to be resolved. ***Output***: 1. The ***did.DocResolution** as string.	✔
3	Update **(Replace)**	`func UpdateDoc(doc *did.Doc) error`	**An update operation for a DID Document.** **The complete document gets replaced with a newer version.**	✘
4	Delete **(Deactivate)**	`func DeleteDid(did string) error`	**A delete operation for a DID Document** A DID document cannot be deleted in the sense of being "**erased**", only its contents can be removed as to prevent future changes (**deactivation**). To deactivate a DID Document, all contents must be removed except the id and context. A DID Document **MUST** at least have one non-deactivated controller, otherwise it's to be regarded as deactivated. **Deletion cannot be reverted (undone).**	✘

Table 9: Smart Contract Function Signatures

4.2.2. Implementation

As the design of the VDR is completed and the rules that required to be applied are established, the aforementioned table is extended to withhold information regarding the implementation flow of the smart contracts:

\	Smart Contract Implementation Flow[2]
	smartcontract.go

CreateDoc	`// CreateDoc creates a new DocResolution struct and stores it in the ledger.`
	`func (s *SmartContract) CreateDoc(ctx contractapi.TransactionContextInterface, clientId string) error {`
	`1. Check if clientId is empty`
	`2. Get a new object from the Transient Map`
	`3. Extract DID from the Transient Map`
	`4. Extract DocResolution from the TransientMap`
	`5. Convert DID from Bytes to String`
	`6. Check if DID is empty`
	`7. Check if a DocResolution exists already with this DID`
	`8. Store DocResolution bytes in the ledger under the provided DID`
	`9. Set Event "{DeviceID=<clientId>"}, {DID=<did>}"`
	`}`

ResolveDID	`// ResolveDid returns the DocResolution record stored in the world state under specified did`
	`func (s *SmartContract) ResolveDid(ctx contractapi.TransactionContextInterface, docId string) (string, error) {`
	`1. Search the world state to retrieve the DocResolution if exists`
	`2. Convert DocResolution bytes to string`
	`}`

Table 10: Smart Contract Implementation Flow

4.2.3. Unit Tests

Following the well-known and well-established best practices in the field of software engineering, regarding the smart contracts implementation, unit tests were also created and provided that covers the chaincode logic extensively.

Although the Unit Tests source code as well as the chaincode mock stubs that were used to simulate the chaincode behavior is presented in the **Appendix A**, in the **Smart Contract Unit Test** table, the outcome is visible by executing the following command:

```
cd <path to the smart contract test>
go test smartcontract_test.go -v
```

And the results are depicted in the table below:

[2] The entire content of **smartcontract.go** appears in **Appendix A** in the **Smart Contract** and **Smart Contract Main** table.

\	Smart Contract Unit Tests Results[3]

CreateDoc	`=== RUN TestCreateDoc`
	`=== RUN TestCreateDoc/Empty_Input`
	`=== RUN TestCreateDoc/No_Error`
	`=== RUN TestCreateDoc/Get_State_Error`
	`=== RUN TestCreateDoc/Get_State_Returns_Not_Nil`
	`=== RUN TestCreateDoc/Put_State_Error`
	`=== RUN TestCreateDoc/Set_Event_Error`
	`=== RUN TestCreateDoc/Get_Transient_Error`
	`=== RUN TestCreateDoc/Empty_DID`
	`--- PASS: TestCreateDoc (0.00s)`
	`--- PASS: TestCreateDoc/Empty_Input (0.00s)`
	`--- PASS: TestCreateDoc/No_Error (0.00s)`
	`--- PASS: TestCreateDoc/Get_State_Error (0.00s)`
	`--- PASS: TestCreateDoc/Get_State_Returns_Not_Nil (0.00s)`
	`--- PASS: TestCreateDoc/Put_State_Error (0.00s)`
	`--- PASS: TestCreateDoc/Set_Event_Error (0.00s)`
	`--- PASS: TestCreateDoc/Get_Transient_Error (0.00s)`
	`--- PASS: TestCreateDoc/Empty_DID (0.00s)`
	`PASS`
	`ok command-line-arguments 0.006s`

ResolveDID	`=== RUN TestResolveDid`
	`=== RUN TestResolveDid/Empty_DocID`
	`=== RUN TestResolveDid/No_Error`
	`=== RUN TestResolveDid/DocId_Does_Not_Exist`
	`=== RUN TestResolveDid/Get_State_Error`
	`--- PASS: TestResolveDid (0.00s)`
	`--- PASS: TestResolveDid/Empty_DocID (0.00s)`
	`--- PASS: TestResolveDid/No_Error (0.00s)`
	`--- PASS: TestResolveDid/DocId_Does_Not_Exist (0.00s)`
	`--- PASS: TestResolveDid/Get_State_Error (0.00s)`
	`PASS`
	`ok command-line-arguments 0.006s`

Table 11: Smart Contract Unit Test Results

As depicted in the results above, all the Unit Tests have passed successfully.

4.3. Deployment

4.3.1. Project Description

As mentioned in the Proposed Architecture Section above, **a Kubernetes deployed Hyperledger Fabric Blockchain Network** was utilized to host the fabricVdr smart contracts where the business logic of our VDR was defined (see Smart Contracts section). Thus additional **Bash** scripts had to be defined and eventually created that facilitated the network operations of a Hyperledger Fabric network. This project was created as user-friendly as possible, following production standards that totally eliminates the difficulty of designing, operating and maintaining a Hyperledger Fabric network, in Kubernetes. The project was named **multihost_k8s** and the project structure as well as information regarding each file is clearly visible in the Appendix B - Project Filesystem Structure section.

[3] The entire content of **smartcontract_test.go**, as well as the **Mock Stubs** appears in **Appendix A** in the **Smart Contract Unit Tests , Mock Stubs for Chaincode Unit Tests, Mocks State Query Iterator for Chaincode Test** and **Mock Transactions Context for Chaincode Unit Tests** tables.

Deploying a multi-component system like Hyperledger Fabric to production is quite challenging. In this section it will be demonstrated the way that we used to deploy a Hyperledger Fabric onto Kubernetes through the usage of custom made bash scripts that leverages kubectl to communicate new instructions to the Kubernetes control plane and custom Manifests files to deploy different components. Rather than using a monolithic automation script for the whole deployment, the project provides separate manifests for each Hyperledger Fabric component, namely the Certificate Authority (Fabric-CA), Peer, CouchDB and Orderer. In the following sections the steps that are required to create and operate a full functional blockchain system are going to be thoroughly presented.

Some information regarding the version of the Hyperledger Fabric network that this project is designed to deploy is the following:

\	Hyperledger Fabric Components Version	
	Component	Version
1	Fabric Peer	2.4.6
2	Fabric Orderer	2.4.6
3	Fabric CA	1.5.5
4	CouchDB	3.2.2
5	NFS	4
6	Chaincode Go	1.18.2

Table 12: Hyperledger Fabric Components Version

Additional information regarding the features of the network can be found below:

- There is one **.env** file that holds information regarding all the environment variables that are required to be present in the project directory in order for this project to operate seamlessly.
- One **NFSv4** server is deployed for each organization at a different namespace (**fabric-network-${ORG}-nfsv4**), ensuring a common shared filesystem among all of the different components/pods of the organization. The NFSv4 server upon deployment exposes an IP internally, that at a later time each component/pod attaches to it, thus having access to the same shared filesystem.
- All the components have **mutual TLS enabled** for network communications **(Peer-to-Peer, Peer-to-Orderer, Peer-to-Chaincode, Orderer-to-Orderer, Fabric-CA Client to Fabric-CA Server)**.
- One Fabric-CA server is deployed for each organization, within the organization namespace (**fabric-network-${ORG}**), that holds two different CAs:
    - One **IdentityCA**, for creating Identities, called **ca**.
    - One **TLSCA**, for creating TLS certificates, called **tlsca**.
- For each aforementioned CA one **RootCA** and one **IntermediateCA** is created using OpenSSL:
    - The Identity RootCA **(rca.identity)** creates and signs one Identity IntermediateCA **(ica.identity),** constructing one Identity Certificate Trust Chain **(chain.identity)**.

    - One TLS RootCA **(rca.tls)** creates and signs one TLS IntermediateCA **(ica.tls)**, constructing one TLS Certificate Trust Chain **(chain.tls)**.
- Each Intermediate CA as well as each Certificate Chain of Trust is loaded into the **Cert-Manager** as a custom CA. This will be later used to create and sign client certificates for all the organization components that require mutual TLS.
- **Fabric CA client** binary is utilized to communicate with the **Fabric CA server** deployed in Kubernetes.
- One **Orderer** organization is deployed having minimum 3 Orderer nodes and the number of the Orderer nodes must be always odd. This is to satisfy the **RAFT consensus algorithm** minimum requirements. (The default number is **3**, and it can be configured). Each orderer node has the **Admin Plugin** enabled (available in version 2.4.6 and newer) and the **Admin** of the Orderer Organization can send, create channel requests directly to the orderer node using the **osnadmin** binary.
- Each **Org{N}** Organization **BESIDES** the **Orderer** Organization (where specific rules apply) can have multiple **Peer** nodes. (The default number is **2**, and it can be configured). Each peer node has the **Fabric Gateway** enabled (available in version 2.4.6 and newer). Adding a gateway component to the Fabric peer provides a single entry point to a Fabric network, and removes much of the transaction submission logic from the client application. The Gateway component in the Fabric Peer exposes a simple gRPC interface to client applications and manages the lifecycle of transaction invocation on behalf of the client. This minimizes the network traffic passing between the client and the blockchain network, as well as minimizing the number of network ports that need to be opened.
- Each **Organization Peer** is registered as **Anchor Peer**.
- **Gossip Protocol** is deactivated, thus block dissemination is being done by **Anchor Peers** by retrieving blocks directly from the ordering service.
- For convenience, a **Peer** as well as its associated **CouchDB** are deployed within a single Kubernetes Deployment, thus located in a single Pod.
- Every time a **Fabric CA Server** is deployed, a custom **MSP** filesystem is created. After that, the **Identity registration** is completed by the **Fabric CA Admin** on the host, but the enrollment is performed directly in the **Fabric CA Server** Pod.
- Each organization can register one or multiple **Users**. (The default number is **1**, and it can be configured).
- Multiple organizations can be deployed. In our case, for demo purposes one organization (using the name **Org1**) was deployed, but this can be configured through the **.env** file to handle more organizations.
- All the deployed organizations in the Kubernetes cluster can join only in one channel (The default channel name is **mychannel**, but it can be configured). **At the moment this project cannot handle multiple private channels between organizations.**
- All the **Kubernetes Manifests YAML** files are designed in an agnostic way without holding information regarding the type and the name of the organization. All the configurations that are being made in the **.env** file are eventually populated in the Kubernetes Manifests.
- Chaincodes are deployed using the **Chaincode As A Service** method (**CCAAS**).
- Applications that require interacting with the Hyperledger Fabric network, must have been designed using the **Fabric Gateway Client API**, which is available for several programming languages. In our case the Fabric Gateway Client API written in GO was used to interact with the Hyperledger Fabric network deployed in the Kubernetes.

There are four main bash scripts that operate the whole project as listed below:

- **minikube.sh:** This script is responsible for spawning or destroying a Minikube instance as well as checking the status of the Minikube instance.

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./minikube.sh --help
Usage: ./minikube.sh MODE

MODE:
  up: Spawn a new Minikube network up
  status: Show the status of the Minikube
  down: Delete the Running Minikube network
```

Figure 22: Outcome of './minikube.sh --help' command

- **nfsv4.sh:** This script is responsible for deploying a new NFSv4 Server as well as its required components to Kubernetes.

- **networkStart.sh:** This script is the most important script and responsible for numerous operations such as:
  - Deploying organizations to Kubernetes,
  - Creating channel artifacts,
  - Creating a channel,
  - Joining organization Peers to the channel,
  - Deploying chaincode to Kubernetes,
  - Installing chaincode to Peers,
  - Approving chaincode for organization,
  - Committing chaincode to the channel.

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh --help
Usage: ./networkStart.sh Mode Subcommand

Mode:
  └── Subcommand (Strong Dependency on .env)

Commands:
  deploy: Deploy an Organization
  artifacts:
      └── create: Create the channel artifacts
  channel:
      ├── create: Create the channel and join Orderers
      ├── join: Join Organization Peers to the channel
      └── delete: Unjoin Orderers from channel
  chaincode:
      ├── deploy: Deploy Chaincode
      ├── install: Install Chaincode to Peers
      ├── approve: Approve Chaincode for an Organization
      └── commit: Commit Chaincode to channel
```

Figure 23: Outcome of './networkStart.sh --help' command

- **networkDown.sh:** This script is responsible for tearing the Hyperledger Fabric Blockchain network down.

4.3.2. Environmental Variables

As stated earlier, the **multihost_k8s** project requires the presence of a **.env** file that holds all the necessary environmental variables, in order to seamlessly operate. The values of the environmental variables are later reflected within the **Kubernetes Manifests YAML** files and consumed by the **Kubernetes**. These environmental variables are also used by the scripts in order to understand the organization profile that is being used during each operation. Some operations must be performed only by an **Orderer Organization**, thus the **Orderer profile** must be selected and other operations must be performed only by **Org Organizations**, thus the **Organization profile** must be selected. Even though there are numerous environmental

variables available within the .env file, the only modifications that are required for the project to be viable are the following environment variables:

```
#☣️ Minimum modifications that are required to execute this script ☣️
ORG=... #👉 Organization Name
DOMAIN=... #👉 Organization Name
NODE_TYPE=... #👉 NODE_TYPE is either 'peer' or 'orderer'
#👉 If ${NODE_TYPE} is orderer, the PEER must be an odd number and
#👉 greater than or equal to 3. If ${NODE_TYPE} is 'orderer', PEER
#👉 is equal to the number of orderers
PEER=... #👉 Number of Peers.
USER=... #👉 Number of Users
CHANNEL_NAME=... #👉 The name of the channel that is being created by Orderer
PATH=...         #👉 Export Path of the Fabric Binaries
#☣️ Minimum modifications that are required to execute this script ☣️
```

Table 13: Minimum Viable Modifications

In the following table, the two different aforementioned profiles that are used in the project as well as their environmental variables default but totally configurable values are presented:

Multihost_k8s Organization Profiles	
Orderer	Organization
ORG=<Orderer Name> DOMAIN=<Orderer Domain Name> NODE_TYPE=orderer PEER=3 USER=1	ORG=<Organization Name> DOMAIN=<Organization Domain Name> NODE_TYPE=peer PEER=2 USER=1

Table 14: Multihost_k8s Organization Profiles

Switching profiles must be done with extreme **CAUTION** due to the fact that decisions are being made according to the value of the environmental variable and according to the operation that needs to be executed. Even if protection mechanisms are being deployed in the scripts to detect such errors, it may be feasible that some errors may not be detected from the protection mechanisms if misuse or typos took place to the aforementioned environmental variables.

4.3.3. Start Minikube

In order to be able to prepare the ground to deploy the Hyperledger Fabric Blockchain network in to Kubernetes some actions had to be made prior to that, such as building the Dockerfiles (located in the **multihost_k8s/Dockerfiles** directory) of the official Hyperledger Fabric Images along with some additional packages that are required from the project in order to operate. In our case the minikube.sh script was used and in order to create a new Minikube instance the following command was used:

```
$ cd multihost_k8s
$ ./minikube.sh up
```

And the following actions are taking place:

\	Start Minikube Flow[4]
	minikube.sh

[4] The entire content of **minikube.sh** appears in **Appendix B** in the Start/Destroy Minikube Instance table.

```
1.  Check Prerequisites
2.  Deleting Previous Minikube Instance
3.  Enabling Metrics Server
4.  Enabling Ingress Plugin
5.  Enabling Ingress DNS Plugin
6.  Patching Ingress tin order to Enable SSL Passthrough
7.  Building alpine-envsubst Dockerfile[5]
8.  Loading alpine-envsubst to Minikube Registry
9.  Building hyperledger/fabric-ca:1.5.5 Dockerfile[6]
10. Loading hyperledger/fabric-ca:1.5.5 to Minikube Registry
11. Building hyperledger/fabric-peer:2.4.6 Dockerfile[7]
12. Loading hyperledger/fabric-peer:2.4.6 to Minikube Registry
13. Building hyperledger/fabric-orderer:2.4.6 Dockerfile[8]
14. Loading hyperledger/fabric-orderer:2.4.6 to Minikube Registry
15. Remove builded docker images from the host
16. Initializing Dashboard
```

Table 15: Start Minikube Process

The result of the execution is shown in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./minikube.sh up
✓ Deleting Previous Instances
✓ Starting Minikube
✓ Enabling Metrics Server
✓ Enabling Ingress Plugin
✓ Enabling Ingress DNS Plugin
✓ Patching Ingress in order to Enable SSL Passthrough
✓ Building alpine-envsubst Dockerfile
✓ Building hyperledger/fabric-ca:1.5.5 Dockerfile
✓ Building hyperledger/fabric-peer:2.4.6 Dockerfile
✓ Building hyperledger/fabric-orderer:2.4.6 Dockerfile
\ Initializing Dashboard
```

Figure 24: Outcome of './minikube.sh up' command

If the script was seamlessly executed, a new window **(Kubernetes Dashboard)** should be opened in the browser, that looks like the following image:

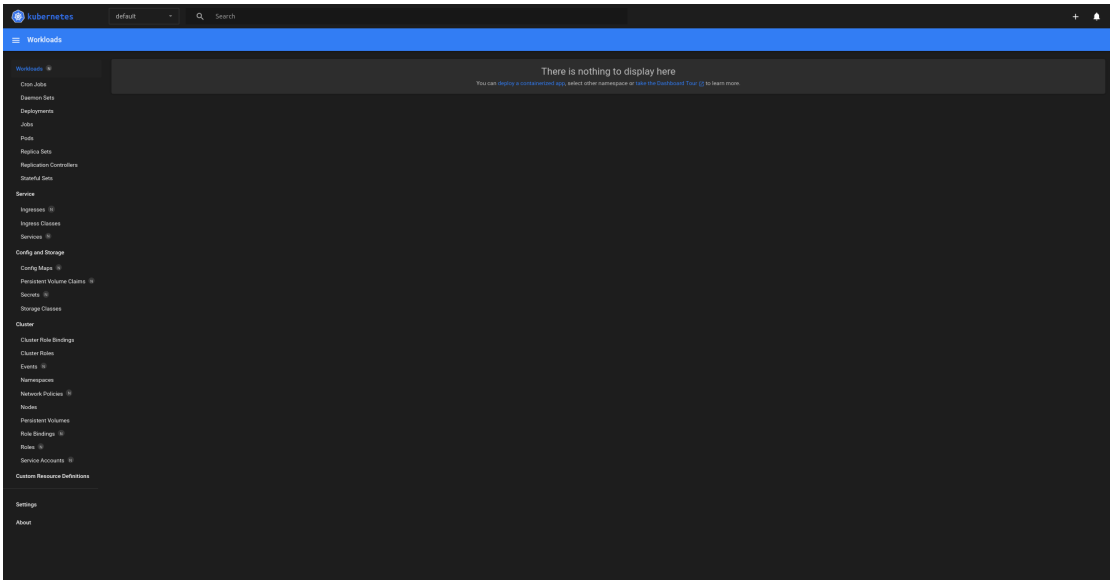

Figure 25: Kubernetes Dashboard

[5] The entire content of **alpine-envsubst Dockerfile** appears in **Appendix B** in the Alpine-Envsubst Dockerfile table.

[6] The entire content of **fabric-ca Dockerfile** appears in **Appendix B** in the Fabric CA Dockerfile table.

[7] The entire content of **fabric-peer Dockerfile** appears in **Appendix B** in the Fabric Peer Dockerfile table.

[8] The entire content of **fabric-orderer Dockerfile** appears in **Appendix B** in the Fabric Orderer Dockerfile table.

In order to verify, if everything is up and running, the following command can be executed:

```
./minikube.sh status
```

And the result of the aforementioned command is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./minikube.sh status
✓ Checking the status of the Minikube Instance
Name: minikube
Type: Control Plane
Host: ✓
Kubelet: ✓
APIServer: ✓
Kubeconfig: Configured
```

Figure 26: Outcome of './minikube.sh status' command when Minikube is up

As depicted in the aforementioned image above, **the Minikube Kubernetes instance is up and running**. At this point the Minikube Kubernetes is ready to receive deployment instructions and thus is ready to start deploying the Hyperledger Fabric components.[9]

4.3.4. Deploy Organization

The next step is to deploy the **Org1 Organization**, thus it is imperative to switch to the **organization profile** in the .env file as mentioned in the Environmental Variables section above. To deploy the **Org1 Organization** the following environmental variables were used:

```
ORG=org1
DOMAIN=example.com
NODE_TYPE=peer
PEER=2
USER=1
```

After the environmental variables are set, in order to be able to deploy the Organization components in the Kubernetes, first the **nfsv4.sh** bash script must be executed, using the following command:

```
$ cd multihost_k8s
$ ./nfsv4.sh
```

And the following actions are taking place for the first command:

\	Start NFSv4 Server Flow[10]
	nfsv4.sh

```
1. Creating namespace fabric-network-org1-nfsv4[11]
2. Provisioning volume storage[12]
3. Starting NFSv4 Server[13]
4. Waiting until NFS Server installation is completed
```

Table 16: Start NFSv4 Server Process for Organization Org1

[9] **NOTE**: *At the moment the **"./minikube.sh up"** command spawns a single master node (also known as a Control Plane), having access to **4 CPU cores** and **8gb RAM**. Although it is possible to add more worker nodes in **Minikube Kubernetes**, at the moment only one node is supported.*

[10] The entire content of **nfsv4.sh** appears in **Appendix B** in the Deploy NFSv4 Server table.

[11] The entire content of **ns.yaml** appears in **Appendix B** in the NFSv4 Namespace Manifest table.

[12] The entire content of **pvc-nfsv4.yaml** appears in **Appendix B** in the NFSv4 Persistent Volume Claim Manifest table.

[13] The entire content of **nfsv4-server.yaml** appears in **Appendix B** in the NFSv4 Server Deployment/Service Manifest table.

The result of the execution is shown in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./nfsv4.sh
✓ Creating namespace fabric-network-org1-nfsv4
namespace/fabric-network-org1-nfsv4 created
✓ Provisioning volume storage
persistentvolumeclaim/fabric-org1-nfsv4-pvc created
✓ Starting NFSv4 server
deployment.apps/org1-nfsv4 created
service/org1-nfsv4 created
✓ Waiting until NFS Server installation is completed
```

Figure 27: Outcome of './nfsv4.sh' command for Organization Org1

And to verify that the NFSv4 Server is up and running, open the **Kubernetes Dashboard**, switch to **fabric-network-org1-nfsv4** namespace, navigate to the **Pods** section and the following image should be appeared:

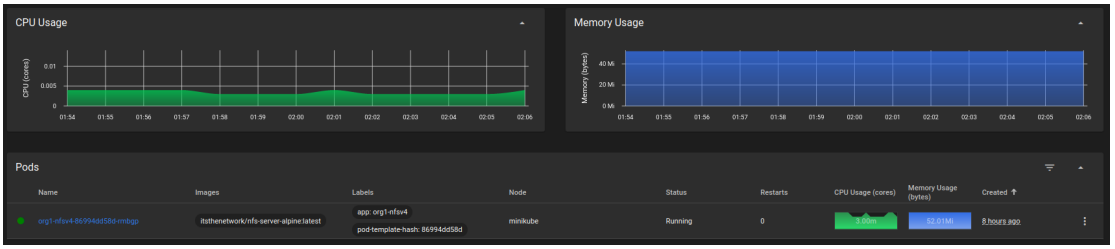

Figure 28: Organization Org1 NFSv4 Server successfully deployed

As depicted in the image above, one **NFSv4 Server** is successfully deployed.

After the execution of the aforementioned command is concluded, then the following command is executed, in order to start deploying organization **Hyperledger Fabric** components in **Kubernetes**:

```
$ cd multihost_k8s
$ ./networkStart.sh deploy
```

And the following actions are taking place for the first command:

\	Deploy Org1 Organization in Kubernetes[14]
	networkStart.sh

```
1.  Deploying Cert-Manager
2.  Waiting until Cert-Manager Deployment is completed
3.  Waiting until Cert-Manager Ca-Injector Deployment is completed
4.  Waiting until Cert-Manager Webhook Deployment is completed
5.  Waiting until Cert-Manager Installation is completed
6.  Check if Directory org1.example.com exists
7.  Creating the missing directory
8.  Creating namespace fabric-network-org1[15]
9.  Provisioning volume storage[16]
10. Creating Org1 Root CAs
11. Creating Org1 TLS Root CA[17]
12. Creating Org1 Identity Root CA[18]
13. Creating and signing TLS Intermediate CA Cert
14. Creating and signing Identity Intermediate CA Cert
15. Creating Intermediate TLS CA Cert Secret
```

[14] The entire content of **networkStart.sh** appears in **Appendix B** in the Deploy Organization table.
[15] The entire content of **ns.yaml** appears in **Appendix B** in the Organization Namespace Manifest table.
[16] The entire content of **org-pvc.yaml** appears in **Appendix B** in the Organization Persistent Volume Claim Manifest table.
[17] The entire content of **openssl_root-tls.cnf** appears in **Appendix B** in the Organization Root CA OpenSSL Configuration table.
[18] The entire content of **openssl_root-identity.cnf** appears in **Appendix B** in the Organization Root CA OpenSSL Configuration table.

```
16. Waiting until Intermediate TLS CA Cert Secret Installation is Completed
17. Loading Intermediate TLS CA Issuer to CERT MANAGER[19]
18. Loading Fabric CA Server Secrets
19. Loading Org1 Fabric CA Server Config Files[20]
20. Waiting until Fabric CA Server Config Files Installation is Completed
21. Loading .env file in configmap[21]
22. Launching org1.example.com Fabric CA Server[22]
23. Waiting until org1.example.com Fabric CA Server Deployment is Completed
24. Retrieving org1.example.com Fabric CA Client Certs
25. Retrieving org1.example.com Fabric CA Operations Client Certs
26. Adding org1.example.com FQDNs entry to /etc/hosts
27. Checking if org1.example.com Fabric CA Server is Listening
28. Creating org1.example.com Peers and Users Certificates in Host
29. Enrolling org1.example.com Fabric CA Admin
30. Registering Peers
31. Registering User for org1.example.com
32. Registering the org1.example.com Admin
33. Create local MSP config.yaml
34. Enrolling the org1.example.com Admin
35. Enrolling the org1.example.com TLS CA Admin
36. Finalizing Org1 MSP
37. Creating org1.example.com Peers and Users Certificates in K8s
38. Create local MSP config.yaml in Fabric CA Server Pod
39. Generating the Peers MSP
40. Generating User MSP
41. Generating the Peer TLS
42. Finalizing org1.example.com MSP
43. Loading Peer CouchDB Secrets
44. Waiting until Peer CouchDB Secrets Installation is Completed
45. Deploying Org1 Gateway Service[23]
46. Loading Peer Config Files[24]
47. Waiting until Peer Config Files Installation is Completed
48. Loading Peer ConfigMap[25]
49. Waiting until Peer ConfigMap is Completed
50. Deploying Peer Manifest[26]
51. Waiting until Peer Deployment is Completed
52. Waiting until Peer Installation is Completed
```

Table 17: Deploy Org1 Organization in Kubernetes Flow

The result of the execution is shown in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh deploy
--- Executing with the following:
- ORGANIZATION HOSTNAME: org1
- ORGANIZATION DOMAIN: example.com
- ORGANIZATION TYPE: peer
- NUMBER OF PEERS: 2
- NUMBER OF USERS: 1
- CHANNEL NAME: mychannel
- NAMESPACE: fabric-network-org1
- FABRIC VERSION: 2.4.6
- FABRIC CA VERSION: 1.5.5
- COUCHDB VERSION: 3.2.2

✓ Deploying Cert-Manager
✓ Waiting until Cert-Manager Deployment is completed
✓ Waiting until Cert-Manager Ca-Injector Deployment is completed
✓ Waiting until Cert-Manager Webhook Deployment is completed
✓ Waiting until Cert-Manager Installation is completed
✗ Directory org1.example.com DOES NOT exists
✓ Creating the missing directory
✓ Creating namespace fabric-network-org1
namespace/fabric-network-org1 created
```

[19] The entire content of **ica-tls-issuer.yaml** appears in **Appendix B** in the Organization Intermediate TLSCA Issuer Manifest table.
[20] The entire content of **fabric-ca-server.yaml** appears in **Appendix B** in the Organization Fabric CA Server Config table
[21] The entire content of **org-ca-env.yaml** appears in **Appendix B** in the Organization Fabric CA Server ConfigMap Manifest table
[22] The entire content of **org-ca.yaml** appears in **Appendix B** in the Organization Fabric CA Deployment Manifest table
[23] The entire content of **peer-gateway.yaml** appears in **Appendix B** in the Organization Gateway Service Manifest table
[24] The entire content of **core.yaml** appears in **Appendix B** in the Organization Peer Config table
[25] The entire content of **peer-env.yaml** appears in **Appendix B** in the Organization Peer ConfigMap Manifest table
[26] The entire content of **peer.yaml** appears in **Appendix B** in the Organization Peer Deployment Manifest table

```
✓ Provisioning volume storage
persistentvolume/fabric-org1-nfsv4-pv created
persistentvolumeclain/fabric-org1 created
✓ Creating Org1 Root CAs

✓ Creating Org1 TLS Root CA
req: No value provided for subject name attribute "OU", skipped
✓ Creating Org1 Identity Root CA
req: No value provided for subject name attribute "OU", skipped
✓ Creating and signing TLS Intermediate CA Cert..
req: No value provided for subject name attribute "OU", skipped
Using configuration from /dev/fd/63
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Feb 28 13:24:10 2023 GMT
            Not After : Feb 27 13:42:10 2028 GMT
        Subject:
            countryName               = GR
            stateOrProvinceName       = Athens
            organizationName          = org1.example.com
            commonName                = ica.tls.org1.example.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                67:4B:32:B2:B2:45:B7:9E:BE:63:49:1A:8A:C3:49:BA:67:7C:9D:E9
            X509v3 Authority Key Identifier:
                2A:EA:21:45:E7:42:33:AC:33:23:AB:CA:B7:00:FE:8F:67:DC:6E:EA
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                DNS:rca.tls.org1.example.com
Certificate is to be certified until Feb 27 13:42:10 2028 GMT (1825 days)

Write out database with 1 new entries
Data Base Updated
✓ Creating and signing Identity Intermediate CA Cert..
req: No value provided for subject name attribute "OU", skipped
Using configuration from /dev/fd/63
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Feb 28 13:42:10 2023 GMT
            Not After : Feb 27 13:42:10 2028 GMT
        Subject:
            countryName               = GR
            stateOrProvinceName       = Athens
            organizationName          = org1.example.com
            commonName                = ica.identity.org1.example.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                6C:DD:B5:FE:DC:DA:FE:5E:CE:29:50:A2:32:5E:41:3B:8A:76:44:10
            X509v3 Authority Key Identifier:
                00:0B:AE:04:6A:66:A0:E5:67:95:8A:E8:CB:81:81:B3:54:F4:F5:4B
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                DNS:rca.identity.org1.example.com
Certificate is to be certified until Feb 27 13:42:10 2028 GMT (1825 days)

Write out database with 1 new entries
Data Base Updated
✓ Creating Intermediate TLS CA Cert Secret..
secret/org1-intermediate-tlsca-key-pair created
secret/org1-intermediate-identity-key-pair created
✓ Waiting until Intermediate TLS CA Cert Secret Installation is Completed
✓ Loading Intermediate TLS CA Issuer to CERT MANAGER..
issuer.cert-manager.io/org1-ica-tls-issuer created
issuer.cert-manager.io/org1-ica-tls-issuer condition met
✓ Loading Fabric CA Server Secrets
secret/fabric-org1-ca-env-secrets created
✓ Loading Org1 Fabric CA Server Config Files
configmap/fabric-ca-server-config created
configmap/fabric-tlsca-server-config created
✓ Waiting until Fabric CA Server Config Files Installation is Completed
✓ Loading .env file in configmap
configmap/fabric-org1-ca-env created
```

```
✓ Launching org1.example.com Fabric CA Server
certificate.cert-manager.io/org1-fabric-ca-server-tls-cert created
certificate.cert-manager.io/org1-fabric-ca-client-tls-cert created
certificate.cert-manager.io/org1-fabric-ca-operations-server-tls-cert created
certificate.cert-manager.io/org1-fabric-ca-operations-client-tls-cert created
deployment.apps/org1-ca created
service/org1-ca created
ingress.networking.k8s.io/org1-ca created
✓ Waiting until org1.example.com Fabric CA Server Deployment is Completed
✓ Retrieving org1.example.com Fabric CA Client Certs
✓ Retrieving org1.example.com Fabric CA Operations Client Certs
✓ Adding org1.example.com FQDNs entry to /etc/hosts
[sudo] password for ubuntu:
✓ Checking if org1.example.com Fabric CA Server is Listening
✓ Fabric CA Server is Listening

✓ Creating org1.example.com Peers and Users Certificates in Host
✓ Enrolling org1.example.com Fabric CA Admin
+ fabric-ca-client enroll --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizatio
n/org1.example.com/users/admin --csr.names C=GR,ST=Athens,L=Athens,O=org1.example.com --csr.keyrequest.al
go ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url https://admin:63bdf7cf625146138053d94
60bf538d8@ica.org1.example.com:443 --tls.certfiles /home/ubuntu/fabric-samples/multihost_k8s/organization
/org1.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /home/ubuntu/fabric-samples/multihost
_k8s/organization/org1.example.com/fabric-ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabr
ic-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.key
2023/02/28 15:43:01 [INFO] Created a default configuration file at /home/ubuntu/fabric-samples/multihost_
k8s/organization/org1.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 15:43:01 [INFO] TLS Enabled
2023/02/28 15:43:01 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 15:43:01 [INFO] encoded CSR
2023/02/28 15:43:01 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/admin/signcerts/cert.pem
2023/02/28 15:43:01 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/org1.example.com/users/admin/cacerts/ica-org1-example-com-443-org1-ca.pem
2023/02/28 15:43:01 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/admin/intermediatecerts/ica-org1-example-com-443-org1-ca.pem
2023/02/28 15:43:01 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/org1.example.com/users/admin/IssuerPublicKey
2023/02/28 15:43:01 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/admin/IssuerRevocationPublicKey
+ set +x

✓ Registering peer0.org1.example.com
+ fabric-ca-client register --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizat
ion/org1.example.com/users/admin --id.name peer0.org1.example.com --id.secret peer0 --id.type peer --id.a
ttrs hf.Registrar.Roles=peer --url https://ica.org1.example.com:443 --tls.certfiles /home/ubuntu/fabric-s
amples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /hom
e/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.crt --tls
.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/
tls/client.key
2023/02/28 15:43:01 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 15:43:01 [INFO] TLS Enabled
2023/02/28 15:43:01 [INFO] TLS Enabled
Password: peer0
+ set +x

✓ Registering peer1.org1.example.com
+ fabric-ca-client register --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizat
ion/org1.example.com/users/admin --id.name peer1.org1.example.com --id.secret peer1 --id.type peer --id.a
ttrs hf.Registrar.Roles=peer --url https://ica.org1.example.com:443 --tls.certfiles /home/ubuntu/fabric-s
amples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /hom
e/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.crt --tls
.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/
tls/client.key
2023/02/28 15:43:01 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 15:43:01 [INFO] TLS Enabled
2023/02/28 15:43:01 [INFO] TLS Enabled
Password: peer1
+ set +x
✓ Registering User1 for org1.example.com
+ fabric-ca-client register --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizat
ion/org1.example.com/users/admin --id.name User1@org1.example.com --id.secret userpw1.org1.example.com --
id.type client --id.attrs hf.Registrar.Roles=client --url https://ica.org1.example.com:443 --tls.certfile
s /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/ca.crt --t
ls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-clie
nt/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.exampl
e.com/fabric-ca-client/tls/client.key
2023/02/28 15:43:01 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 15:43:01 [INFO] TLS Enabled
2023/02/28 15:43:01 [INFO] TLS Enabled
Password: userpw1.org1.example.com
+ set +x
```

```
✓ Registering the org1.example.com Admin
+ fabric-ca-client register --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizat
ion/org1.example.com/users/admin --id.name Admin@org1.example.com --id.secret 26e172d798af44f69cbb2d93082
e9f --id.type admin --id.attrs hf.Registrar.Roles=admin --url https://ica.org1.example.com:443 --tls.cert
files /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/ca.crt
 --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-
client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.ex
ample.com/fabric-ca-client/tls/client.key
2023/02/28 15:43:01 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 15:43:01 [INFO] TLS Enabled
2023/02/28 15:43:01 [INFO] TLS Enabled
Password: 26e172d798af44f69cbb2d93082e9f
+ set +x

✓ Create local MSP config.yaml
+ echo 'NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: orderer'
+ set +x
✓ Enrolling the org1.example.com Admin
+ fabric-ca-client enroll --caname org1-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organizatio
n/org1.example.com/users/Admin@org1.example.com/msp --csr.names C=GR,ST=Athens,L=Athens,O=org1.example.co
m --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url https://Admin@or
g1.example.com:26e172d798af44f69cbb2d93082e9f@ica.org1.example.com:443 --tls.certfiles /home/ubuntu/fabri
c-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /
home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.crt --
tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-clie
nt/tls/client.key
2023/02/28 15:43:02 [INFO] TLS Enabled
2023/02/28 15:43:02 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 15:43:02 [INFO] encoded CSR
2023/02/28 15:43:02 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/Admin@org1.example.com/msp/signcerts/cert.pem
2023/02/28 15:43:02 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/org1.example.com/users/Admin@org1.example.com/msp/cacerts/ica-org1-example-com-443-org1-ca.pem
2023/02/28 15:43:02 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/Admin@org1.example.com/msp/intermediatecerts/ica-org1-example-com-
443-org1-ca.pem
2023/02/28 15:43:02 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/org1.example.com/users/Admin@org1.example.com/msp/IssuerPublicKey
2023/02/28 15:43:02 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/Admin@org1.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Enrolling the org1.example.com TLS CA Admin
+ fabric-ca-client enroll --caname org1-tlsca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/org1.example.com/users/tlsadmin --csr.names C=GR,ST=Athens,L=Athens,O=org1.example.com --csr.keyrequ
est.algo ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url https://admin:63bdf7cf625146138
053d9460bf538d8@ica.org1.example.com:443 --tls.certfiles /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/org1.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /home/ubuntu/fabric-samples/mul
tihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.crt --tls.client.keyfile /home/ubunt
u/fabric-samples/multihost_k8s/organization/org1.example.com/fabric-ca-client/tls/client.key
2023/02/28 15:43:02 [INFO] Created a default configuration file at /home/ubuntu/fabric-samples/multihost_
k8s/organization/org1.example.com/users/tlsadmin/fabric-ca-client-config.yaml
2023/02/28 15:43:02 [INFO] TLS Enabled
2023/02/28 15:43:02 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 15:43:02 [INFO] encoded CSR
2023/02/28 15:43:02 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/org1.example.com/users/tlsadmin/signcerts/cert.pem
2023/02/28 15:43:02 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/org1.example.com/users/tlsadmin/cacerts/ica-org1-example-com-443-org1-tlsca.pem
2023/02/28 15:43:02 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/tlsadmin/intermediatecerts/ica-org1-example-com-443-org1-tlsca.pem
2023/02/28 15:43:02 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/org1.example.com/users/tlsadmin/IssuerPublicKey
2023/02/28 15:43:02 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/org1.example.com/users/tlsadmin/IssuerRevocationPublicKey
+ set +x
✓ Finalizing Org1 MSP
```

```
✓ Creating org1.example.com Peers and Users Certificates in K8s
Defaulted container "org1-ca" out of: org1-ca, org1-ca-init (init)

✓ Create local MSP config.yaml
+ echo 'NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: orderer'
+ set +x

✓ Generating the peer0.org1.example.com msp
+ fabric-ca-client enroll --caname org1-ca --mspdir /var/hyperledger/fabric/crypto-config/peerOrganizatio
ns/org1.example.com/peers/peer0.org1.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=org1.example.
com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts peer0.org1.example.com,peer0-org1,
peer0-org1.fabric-network-org1,peer0.org1.example.com -m peer0.org1.example.com --url https://peer0.org1.
example.com:peer0@ica.org1.example.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/
ca.crt --tls.client.certfile /var/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile
 /var/hyperledger/fabric-ca-client/tls/client/client.key
2023/02/28 13:43:04 [INFO] Created a default configuration file at /var/hyperledger/fabric-ca-client/fabr
ic-ca-client-config.yaml
2023/02/28 13:43:04 [INFO] TLS Enabled
2023/02/28 13:43:04 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 13:43:04 [INFO] encoded CSR
2023/02/28 13:43:05 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/peerOrganiz
ations/org1.example.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem
2023/02/28 13:43:05 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/peerOrgani
zations/org1.example.com/peers/peer0.org1.example.com/msp/cacerts/ica-org1-example-com-7054-org1-ca.pem
2023/02/28 13:43:05 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/intermediatecerts/ica-org1-example-com
-7054-org1-ca.pem
2023/02/28 13:43:05 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer0.org1.example.com/msp/IssuerPublicKey
2023/02/28 13:43:05 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Generating the peer1.org1.example.com msp
+ fabric-ca-client enroll --caname org1-ca --mspdir /var/hyperledger/fabric/crypto-config/peerOrganizatio
ns/org1.example.com/peers/peer1.org1.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=org1.example.
com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts peer1.org1.example.com,peer1-org1,
peer1-org1.fabric-network-org1,peer1.org1.example.com -m peer1.org1.example.com --url https://peer1.org1.
example.com:peer1@ica.org1.example.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/
ca.crt --tls.client.certfile /var/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile
 /var/hyperledger/fabric-ca-client/tls/client/client.key
2023/02/28 13:43:05 [INFO] TLS Enabled
2023/02/28 13:43:05 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 13:43:05 [INFO] encoded CSR
2023/02/28 13:43:05 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/peerOrganiz
ations/org1.example.com/peers/peer1.org1.example.com/msp/signcerts/cert.pem
2023/02/28 13:43:05 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/peerOrgani
zations/org1.example.com/peers/peer1.org1.example.com/msp/cacerts/ica-org1-example-com-7054-org1-ca.pem
2023/02/28 13:43:05 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp/intermediatecerts/ica-org1-example-com
-7054-org1-ca.pem
2023/02/28 13:43:05 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer1.org1.example.com/msp/IssuerPublicKey
2023/02/28 13:43:05 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Generating User1 msp
+ fabric-ca-client enroll --caname org1-ca --mspdir /var/hyperledger/fabric/crypto-config/peerOrganizatio
ns/org1.example.com/users/User1@org1.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=org1.example.
com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts User1@org1.example.com,User1-org1,
User1-org1.fabric-network-org1,User1.org1.example.com -m User1@org1.example.com --url https://User1@org1.
example.com:userpw1.org1.example.com@ica.org1.example.com:7054 --tls.certfiles /var/hyperledger/fabric-ca
-client/tls/client/ca.crt --tls.client.certfile /var/hyperledger/fabric-ca-client/tls/client/client.crt -
-tls.client.keyfile /var/hyperledger/fabric-ca-client/tls/client/client.key
2023/02/28 13:43:05 [INFO] TLS Enabled
2023/02/28 13:43:05 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 13:43:05 [INFO] encoded CSR
2023/02/28 13:43:05 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/peerOrganiz
ations/org1.example.com/users/User1@org1.example.com/msp/signcerts/cert.pem
2023/02/28 13:43:05 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/peerOrgani
zations/org1.example.com/users/User1@org1.example.com/msp/cacerts/ica-org1-example-com-7054-org1-ca.pem
2023/02/28 13:43:05 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/users/User1@org1.example.com/msp/intermediatecerts/ica-org1-example-com
-7054-org1-ca.pem
```

```
2023/02/28 13:43:05 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/users/User1@org1.example.com/msp/IssuerPublicKey
2023/02/28 13:43:05 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/users/User1@org1.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Generating the peer0.org1.example.com TLS
+ fabric-ca-client enroll --caname org1-tlsca --mspdir /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer0.org1.example.com/tls/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=org1.e
xample.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts peer0.org1.example.com,peer
0-org1,peer0-org1.fabric-network-org1,peer0.org1.example.com -m peer0.org1.example.com --url https://peer
0.org1.example.com:peer0@ica.org1.example.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/
client/ca.crt --tls.client.certfile /var/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.
keyfile /var/hyperledger/fabric-ca-client/tls/client/client.key
2023/02/28 13:43:05 [INFO] TLS Enabled
2023/02/28 13:43:05 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 13:43:05 [INFO] encoded CSR
2023/02/28 13:43:06 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/peerOrganiz
ations/org1.example.com/peers/peer0.org1.example.com/tls/msp/signcerts/cert.pem
2023/02/28 13:43:06 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/peerOrgani
zations/org1.example.com/peers/peer0.org1.example.com/tls/msp/cacerts/ica-org1-example-com-7054-org1-tlsc
a.pem
2023/02/28 13:43:06 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/msp/intermediatecerts/ica-org1-example
-com-7054-org1-tlsca.pem
2023/02/28 13:43:06 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer0.org1.example.com/tls/msp/IssuerPublicKey
2023/02/28 13:43:06 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/msp/IssuerRevocationPublicKey
+ set +x

✓ Generating the peer1.org1.example.com TLS
+ fabric-ca-client enroll --caname org1-tlsca --mspdir /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer1.org1.example.com/tls/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=org1.e
xample.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts peer1.org1.example.com,peer
1-org1,peer1-org1.fabric-network-org1,peer1.org1.example.com -m peer1.org1.example.com --url https://peer
1.org1.example.com:peer1@ica.org1.example.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/
client/ca.crt --tls.client.certfile /var/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.
keyfile /var/hyperledger/fabric-ca-client/tls/client/client.key
2023/02/28 13:43:06 [INFO] TLS Enabled
2023/02/28 13:43:06 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 13:43:06 [INFO] encoded CSR
2023/02/28 13:43:06 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/peerOrganiz
ations/org1.example.com/peers/peer1.org1.example.com/tls/msp/signcerts/cert.pem
2023/02/28 13:43:06 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/peerOrgani
zations/org1.example.com/peers/peer1.org1.example.com/tls/msp/cacerts/ica-org1-example-com-7054-org1-tlsc
a.pem
2023/02/28 13:43:06 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/msp/intermediatecerts/ica-org1-example
-com-7054-org1-tlsca.pem
2023/02/28 13:43:06 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/peerOrganiza
tions/org1.example.com/peers/peer1.org1.example.com/tls/msp/IssuerPublicKey
2023/02/28 13:43:06 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/p
eerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/msp/IssuerRevocationPublicKey
+ set +x

✓ Finalizing org1.example.com MSP
✓ Finished

✓ Loading Peer CouchDB Secrets
secret/peer-couchdb-env-secrets created
✓ Waiting until Peer CouchDB Secrets Installation is Completed
✓ Deploying Org1 Gateway Service
service/org1-gateway created
ingress.networking.k8s.io/org1-gateway created
✓ Loading Peer Config Files
configmap/peer-config created
✓ Waiting until Peer Config Files Installation is Completed
✓ Loading peer0.org1.example.com ConfigMap
configmap/peer0-env created
✓ Waiting until peer0.org1.example.com ConfigMap is Completed
✓ Deploying peer0.org1.example.com Manifest
certificate.cert-manager.io/peer0-tls-cert created
certificate.cert-manager.io/peer0-operations-tls-cert created
certificate.cert-manager.io/peer0-operations-client-tls-cert created
deployment.apps/peer0-org1 created
service/peer0-org1 created
ingress.networking.k8s.io/peer0-org1 created
✓ Deploying peer1.org1.example.com Manifest
certificate.cert-manager.io/peer1-tls-cert created
certificate.cert-manager.io/peer1-operations-tls-cert created
certificate.cert-manager.io/peer1-operations-client-tls-cert created
deployment.apps/peer1-org1 created
service/peer1-org1 created
ingress.networking.k8s.io/peer1-org1 created
```

```
✓ Waiting until peer1.org1.example.com Deployment is Completed
✓ Waiting until Peer Installation is Completed
✓ Peer Installation is Completed
```

Figure 29: Outcome of './networkStart.sh deploy' command for Organization Org1

Finally, in order to verify if everything is up and running, open the **Kubernetes Dashboard**, switch to **fabric-network-org1** namespace and navigate to the **Pods** section and the following image should be appeared:

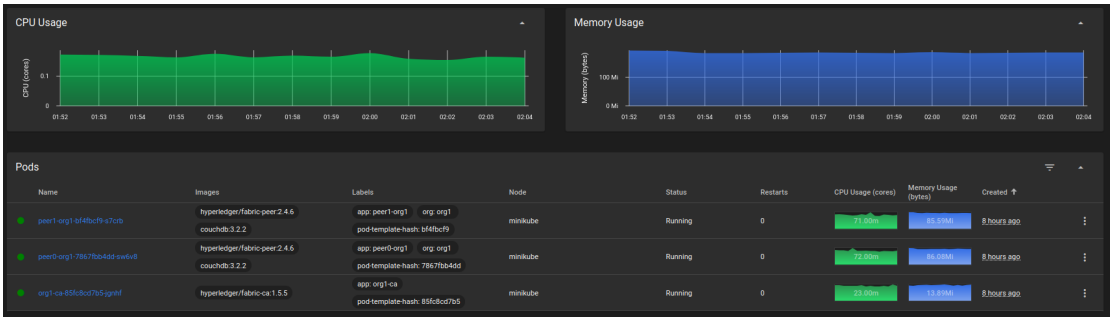

Figure 30: Organization Org1 successfully deployed

As depicted in the image above, two Peers Nodes (consisting of two Pods **peer0-org1** and **peer1-org1**) and a Fabric CA (consisting of one Pod **org1-ca**) are successfully deployed in Kubernetes[27].

4.3.5. Deploy Orderer

The next step is to deploy the **Orderer Organization**, thus it is imperative to switch to the **orderer profile** in the .env file as mentioned in the Environmental Variables section above. To deploy the **Orderer Organization** the following environmental variables were used:

```
ORG=orderer
DOMAIN=example.com
NODE_TYPE=orderer
PEER=3
USER=1
```

After the environmental variables are set, in order to be able to deploy the Organization components in the Kubernetes, first the **nfsv4.sh** bash script must be executed, using the following command:

```
$ cd multihost_k8s
$ ./nfsv4.sh
```

And the following actions are taking place for the first command:

\	Start NFSv4 Server Flow[28]
	nfsv4.sh

```
5. Creating namespace fabric-network-orderer-nfsv4[29]
6. Provisioning volume storage[30]
```

[27] **NOTE**: *If more than one organization must be deployed, the same procedure as described in the section Deploy Organization must be followed.*

[28] The entire content of **nfsv4.sh** appears in **Appendix B** in the Deploy NFSv4 Server table.

[29] The entire content of **ns.yaml** appears in **Appendix B** in the NFSv4 Namespace Manifest table.

[30] The entire content of **pvc-nfsv4.yaml** appears in **Appendix B** in the NFSv4 Persistent Volume Claim Manifest table.

```
7.  Starting NFSv4 Server[31]
8.  Waiting until NFS Server installation is completed
```

Table 18: Start NFSv4 Server Process for Organization Orderer

The result of the execution is shown in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./nfsv4.sh
✓ Creating namespace fabric-network-orderer-nfsv4
namespace/fabric-network-orderer-nfsv4 created
✓ Provisioning volume storage
persistentvolumeclaim/fabric-orderer-nfsv4-pvc created
✓ Starting NFSv4 server
deployment.apps/orderer-nfsv4 created
service/orderer-nfsv4 created
✓ Waiting until NFS Server installation is completed
```

Figure 31: Outcome of './nfsv4.sh' command for Organization Orderer

And to verify that the NFSv4 Server is up and running, open the **Kubernetes Dashboard**, switch to **fabric-network-orderer-nfsv4** namespace and navigate to the **Pods** section and the following image should be appeared:

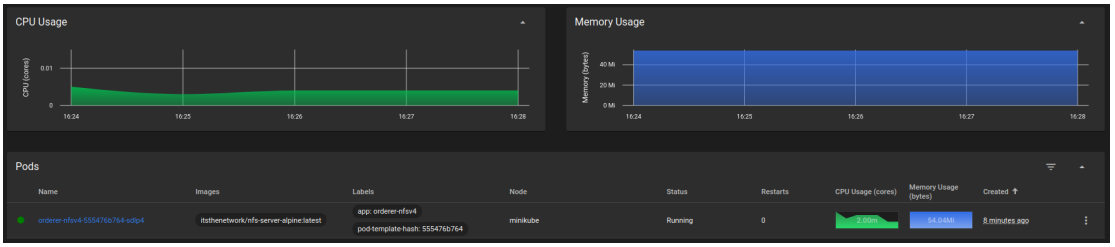

Figure 32: Organization Orderer NFSv4 Server successfully deployed

As depicted in the image above, one **NFSv4 Server** is successfully deployed.

After the execution of the aforementioned command is concluded, then the following command is executed, in order to start deploying organization **Hyperledger Fabric** components in **Kubernetes**:

```
$ cd multihost_k8s
$ ./networkStart.sh deploy
```

And the following actions are taking place for the first command:

\	Deploy Orderer Organization in Kubernetes[32]
	networkStart.sh

```
1.  Deploying Cert-Manager
2.  Waiting until Cert-Manager Deployment is completed
3.  Waiting until Cert-Manager Ca-Injector Deployment is completed
4.  Waiting until Cert-Manager Webhook Deployment is completed
5.  Waiting until Cert-Manager Installation is completed
6.  Check if Directory orderer.example.com exists
7.  Creating the missing directory
8.  Creating namespace fabric-network-orderer[33]
9.  Provisioning volume storage[34]
10. Creating Orderer Root CAs
11. Creating Orderer TLS Root CA[35]
```

[31] The entire content of **nfsv4-server.yaml** appears in **Appendix B** in the NFSv4 Server Deployment/Service Manifest table.
[32] The entire content of **networkStart.sh** appears in **Appendix B** in the Deploy Organization table.
[33] The entire content of **ns.yaml** appears in **Appendix B** in the Organization Namespace Manifest table.
[34] The entire content of **org-pvc.yaml** appears in **Appendix B** in the Organization Persistent Volume Claim Manifest table.
[35] The entire content of **openssl_root-tls.cnf** appears in **Appendix B** in the Organization Root CA OpenSSL Configuration table.

```
12. Creating Orderer Identity Root CA[36]
13. Creating and signing TLS Intermediate CA Cert
14. Creating and signing Identity Intermediate CA Cert
15. Creating Intermediate TLS CA Cert Secret
16. Waiting until Intermediate TLS CA Cert Secret Installation is Completed
17. Loading Intermediate TLS CA Issuer to CERT MANAGER[37]
18. Loading Fabric CA Server Secrets
19. Loading Orderer Fabric CA Server Config Files[38]
20. Waiting until Fabric CA Server Config Files Installation is Completed
21. Loading .env file in configmap[39]
22. Launching orderer.example.com Fabric CA Server[40]
23. Waiting until orderer.example.com Fabric CA Server Deployment is Completed
24. Retrieving orderer.example.com Fabric CA Client Certs
25. Retrieving orderer.example.com Fabric CA Operations Client Certs
26. Adding orderer.example.com FQDNs entry to /etc/hosts
27. Checking if orderer.example.com Fabric CA Server is Listening
28. Creating orderer.example.com Orderers and Users Certificates in Host
29. Enrolling orderer.example.com Fabric CA Admin
30. Registering Orderers
31. Registering User for orderer.example.com
32. Registering the orderer.example.com Admin
33. Create local MSP config.yaml
34. Enrolling the orderer.example.com Admin
35. Enrolling the orderer.example.com TLS CA Admin
36. Finalizing Orderer MSP
37. Creating orderer.example.com Peers and Users Certificates in K8s
38. Create local MSP config.yaml in Fabric CA Server Pod
39. Generating the Orderer MSP
40. Generating User MSP
41. Generating the Orderer TLS
42. Finalizing orderer.example.com MSP
43. Loading Orderer Config File[41]
44. Waiting until Orderer Config File Installation is Completed
45. Loading Orderer ConfigMap[42]
46. Waiting until Orderer ConfigMap is Completed
47. Deploying Orderer Manifest[43]
48. Waiting until Orderer Deployment is Completed
49. Waiting until Orderer Installation is Completed
```

Table 19: Deploy Orderer Organization in Kubernetes Flow

The result of the execution is shown in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh deploy
--- Executing with the following:
- ORGANIZATION HOSTNAME: orderer
- ORGANIZATION DOMAIN: example.com
- ORGANIZATION TYPE: orderer
- NUMBER OF PEERS: 3
- NUMBER OF USERS: 1
- CHANNEL NAME: mychannel
- NAMESPACE: fabric-network-orderer
- FABRIC VERSION: 2.4.6
- FABRIC CA VERSION: 1.5.5
- COUCHDB VERSION: 3.2.2

✓ Cert-Manager is already deployed. Proceed...
X Directory orderer.example.com DOES NOT exists
✓ Creating the missing directory
✓ Creating namespace fabric-network-orderer
namespace/fabric-network-orderer created
✓ Provisioning volume storage
persistentvolume/fabric-orderer-nfsv4-pv created
persistentvolumeclaim/fabric-orderer created
```

[36] The entire content of **openssl_root-identity.cnf** appears in **Appendix B** in the Organization Root CA OpenSSL Configuration table.
[37] The entire content of **ica-tls-issuer.yaml** appears in **Appendix B** in the Organization Intermediate TLSCA Issuer Manifest table.
[38] The entire content of **fabric-ca-server.yaml** appears in **Appendix B** in the Organization Fabric CA Server Config table
[39] The entire content of **org-ca-env.yaml** appears in **Appendix B** in the Organization Fabric CA Server ConfigMap Manifest table
[40] The entire content of **org-ca.yaml** appears in **Appendix B** in the Organization Fabric CA Deployment Manifest table
[41] The entire content of **orderer.yaml** appears in **Appendix B** in the Organization Orderer Config table
[42] The entire content of **orderer-env.yaml** appears in **Appendix B** in the Organization Orderer ConfigMap Manifest table
[43] The entire content of **orderer.yaml** appears in **Appendix B** in the Organization Orderer Deployment Manifest table

```
✓ Creating Orderer Root CAs

✓ Creating Orderer TLS Root CA
req: No value provided for subject name attribute "OU", skipped
✓ Creating Orderer Identity Root CA
req: No value provided for subject name attribute "OU", skipped
✓ Creating and signing TLS Intermediate CA Cert..
req: No value provided for subject name attribute "OU", skipped
Using configuration from /dev/fd/63
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Feb 28 14:34:04 2023 GMT
            Not After : Feb 27 14:34:04 2028 GMT
        Subject:
            countryName               = GR
            stateOrProvinceName       = Athens
            organizationName          = orderer.example.com
            commonName                = ica.tls.orderer.example.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                96:86:7B:DD:5E:F8:A1:00:B6:80:53:E7:76:2F:89:8F:CF:85:91:A8
            X509v3 Authority Key Identifier:
                37:09:2B:66:25:A2:85:2B:33:FF:09:86:F2:DD:C2:90:6C:86:89:24
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                DNS:rca.tls.orderer.example.com
Certificate is to be certified until Feb 27 14:34:04 2028 GMT (1825 days)

Write out database with 1 new entries
Data Base Updated
✓ Creating and signing Identity Intermediate CA Cert..
req: No value provided for subject name attribute "OU", skipped
Using configuration from /dev/fd/63
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Feb 28 14:34:04 2023 GMT
            Not After : Feb 27 14:34:04 2028 GMT
        Subject:
            countryName               = GR
            stateOrProvinceName       = Athens
            organizationName          = orderer.example.com
            commonName                = ica.identity.orderer.example.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                1E:48:A1:73:B5:BE:38:83:B8:A4:F3:39:53:8D:5A:0A:69:FC:18:8B
            X509v3 Authority Key Identifier:
                80:8C:7B:37:C1:89:F3:18:F4:D5:53:27:70:7B:67:A6:BD:16:E8:04
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                DNS:rca.identity.orderer.example.com
Certificate is to be certified until Feb 27 14:34:04 2028 GMT (1825 days)

Write out database with 1 new entries
Data Base Updated
✓ Creating Intermediate TLS CA Cert Secret..
secret/orderer-intermediate-tlsca-key-pair created
secret/orderer-intermediate-identity-key-pair created
✓ Waiting until Intermediate TLS CA Cert Secret Installation is Completed
✓ Loading Intermediate TLS CA Issuer to CERT MANAGER..
issuer.cert-manager.io/orderer-ica-tls-issuer created
issuer.cert-manager.io/orderer-ica-tls-issuer condition met
✓ Loading Fabric CA Server Secrets
secret/fabric-orderer-ca-env-secrets created
✓ Loading Orderer Fabric CA Server Config Files
configmap/fabric-ca-server-config created
configmap/fabric-tlsca-server-config created
✓ Waiting until Fabric CA Server Config Files Installation is Completed
✓ Loading .env file in configmap
configmap/fabric-orderer-ca-env created
✓ Launching orderer.example.com Fabric CA Server
```

```
certificate.cert-manager.io/orderer-fabric-ca-server-tls-cert created
certificate.cert-manager.io/orderer-fabric-ca-client-tls-cert created
certificate.cert-manager.io/orderer-fabric-ca-operations-server-tls-cert created
certificate.cert-manager.io/orderer-fabric-ca-operations-client-tls-cert created
deployment.apps/orderer-ca created
service/orderer-ca created
ingress.networking.k8s.io/orderer-ca created
✓ Waiting until orderer.example.com Fabric CA Server Deployment is Completed
✓ Retrieving orderer.example.com Fabric CA Client Certs
✓ Retrieving orderer.example.com Fabric CA Operations Client Certs
✓ Adding orderer.example.com FQDNs entry to /etc/hosts
✓ Checking if orderer.example.com Fabric CA Server is Listening
✓ Fabric CA Server is Listening

✓ Creating orderer.example.com Orderers and Users Certificates in Host

✓ Enrolling orderer.example.com Fabric CA Admin
+ fabric-ca-client enroll --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/orderer.example.com/users/admin --csr.names C=GR,ST=Athens,L=Athens,O=orderer.example.com --csr.keyr
equest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url https://admin:53f68e0f32ac4b
7297bdd8da35eee9df@ica.orderer.example.com:443 --tls.certfiles /home/ubuntu/fabric-samples/multihost_k8s/
organization/orderer.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /home/ubuntu/fabric-sa
mples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/client.crt --tls.client.keyfile
 /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/client.k
ey
2023/02/28 16:35:58 [INFO] Created a default configuration file at /home/ubuntu/fabric-samples/multihost_
k8s/organization/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:58 [INFO] TLS Enabled
2023/02/28 16:35:58 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 16:35:58 [INFO] encoded CSR
2023/02/28 16:35:58 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/signcerts/cert.pem
2023/02/28 16:35:58 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin/cacerts/ica-orderer-example-com-443-orderer-ca.pem
2023/02/28 16:35:58 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/admin/intermediatecerts/ica-orderer-example-com-443-orderer-ca.
pem
2023/02/28 16:35:58 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/orderer.example.com/users/admin/IssuerPublicKey
2023/02/28 16:35:58 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/admin/IssuerRevocationPublicKey
+ set +x

✓ Registering orderer0.orderer.example.com
+ fabric-ca-client register --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin --id.name orderer0.orderer.example.com --id.secret orderer0 --id.t
ype orderer --id.attrs hf.Registrar.Roles=orderer --url https://ica.orderer.example.com:443 --tls.certfil
es /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/ca.crt
 --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-
ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orde
rer.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:58 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:58 [INFO] TLS Enabled
2023/02/28 16:35:58 [INFO] TLS Enabled
Password: orderer0
+ set +x

✓ Registering orderer1.orderer.example.com
+ fabric-ca-client register --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin --id.name orderer1.orderer.example.com --id.secret orderer1 --id.t
ype orderer --id.attrs hf.Registrar.Roles=orderer --url https://ica.orderer.example.com:443 --tls.certfil
es /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/ca.crt
 --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-
ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orde
rer.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:58 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:58 [INFO] TLS Enabled
2023/02/28 16:35:58 [INFO] TLS Enabled
Password: orderer1
+ set +x

✓ Registering orderer2.orderer.example.com
+ fabric-ca-client register --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin --id.name orderer2.orderer.example.com --id.secret orderer2 --id.t
ype orderer --id.attrs hf.Registrar.Roles=orderer --url https://ica.orderer.example.com:443 --tls.certfil
es /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/ca.crt
 --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-
ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orde
rer.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:58 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:58 [INFO] TLS Enabled
2023/02/28 16:35:58 [INFO] TLS Enabled
Password: orderer2
+ set +x
```

```
✓ Registering User1 for orderer.example.com
+ fabric-ca-client register --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin --id.name User1@orderer.example.com --id.secret userpw1.orderer.ex
ample.com --id.type client --id.attrs hf.Registrar.Roles=client --url https://ica.orderer.example.com:443
 --tls.certfiles /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-cli
ent/tls/ca.crt --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.examp
le.com/fabric-ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/org
anization/orderer.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:59 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:59 [INFO] TLS Enabled
2023/02/28 16:35:59 [INFO] TLS Enabled
Password: userpw1.orderer.example.com
+ set +x
✓ Registering the orderer.example.com Admin
+ fabric-ca-client register --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/admin --id.name Admin@orderer.example.com --id.secret d1f67f3a93d64214bc
e6e8e825503a --id.type admin --id.attrs hf.Registrar.Roles=admin --url https://ica.orderer.example.com:44
3 --tls.certfiles /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-cl
ient/tls/ca.crt --tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.exam
ple.com/fabric-ca-client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/or
ganization/orderer.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:59 [INFO] Configuration file location: /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/admin/fabric-ca-client-config.yaml
2023/02/28 16:35:59 [INFO] TLS Enabled
2023/02/28 16:35:59 [INFO] TLS Enabled
Password: d1f67f3a93d64214bce6e8e825503a
+ set +x

✓ Create local MSP config.yaml
+ echo 'NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: orderer'
+ set +x

✓ Enrolling the orderer.example.com Admin
+ fabric-ca-client enroll --caname orderer-ca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/orderer.example.com/users/Admin@orderer.example.com/msp --csr.names C=GR,ST=Athens,L=Athens,O=ordere
r.example.com --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url http
s://Admin@orderer.example.com:d1f67f3a93d64214bce6e8e825503a@ica.orderer.example.com:443 --tls.certfiles
/home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/ca.crt --
tls.client.certfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-
client/tls/client.crt --tls.client.keyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer
.example.com/fabric-ca-client/tls/client.key
2023/02/28 16:35:59 [INFO] TLS Enabled
2023/02/28 16:35:59 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 16:35:59 [INFO] encoded CSR
2023/02/28 16:35:59 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/Admin@orderer.example.com/msp/signcerts/cert.pem
2023/02/28 16:35:59 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/Admin@orderer.example.com/msp/cacerts/ica-orderer-example-com-443-ordere
r-ca.pem
2023/02/28 16:35:59 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/Admin@orderer.example.com/msp/intermediatecerts/ica-orderer-exa
mple-com-443-orderer-ca.pem
2023/02/28 16:35:59 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/orderer.example.com/users/Admin@orderer.example.com/msp/IssuerPublicKey
2023/02/28 16:35:59 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/Admin@orderer.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Enrolling the orderer.example.com TLS CA Admin
+ fabric-ca-client enroll --caname orderer-tlsca --mspdir /home/ubuntu/fabric-samples/multihost_k8s/organ
ization/orderer.example.com/users/tlsadmin --csr.names C=GR,ST=Athens,L=Athens,O=orderer.example.com --cs
r.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts admin -m admin --url https://admin:53f68e0f
32ac4b7297bdd8da35eee9df@ica.orderer.example.com:443 --tls.certfiles /home/ubuntu/fabric-samples/multihos
t_k8s/organization/orderer.example.com/fabric-ca-client/tls/ca.crt --tls.client.certfile /home/ubuntu/fab
ric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/client.crt --tls.client.k
eyfile /home/ubuntu/fabric-samples/multihost_k8s/organization/orderer.example.com/fabric-ca-client/tls/cl
ient.key
```

```
2023/02/28 16:35:59 [INFO] Created a default configuration file at /home/ubuntu/fabric-samples/multihost_
k8s/organization/orderer.example.com/users/tlsadmin/fabric-ca-client-config.yaml
2023/02/28 16:35:59 [INFO] TLS Enabled
2023/02/28 16:35:59 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 16:35:59 [INFO] encoded CSR
2023/02/28 16:35:59 [INFO] Stored client certificate at /home/ubuntu/fabric-samples/multihost_k8s/organiz
ation/orderer.example.com/users/tlsadmin/signcerts/cert.pem
2023/02/28 16:35:59 [INFO] Stored root CA certificate at /home/ubuntu/fabric-samples/multihost_k8s/organi
zation/orderer.example.com/users/tlsadmin/cacerts/ica-orderer-example-com-443-orderer-tlsca.pem
2023/02/28 16:35:59 [INFO] Stored intermediate CA certificates at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/tlsadmin/intermediatecerts/ica-orderer-example-com-443-orderer-
tlsca.pem
2023/02/28 16:35:59 [INFO] Stored Issuer public key at /home/ubuntu/fabric-samples/multihost_k8s/organiza
tion/orderer.example.com/users/tlsadmin/IssuerPublicKey
2023/02/28 16:35:59 [INFO] Stored Issuer revocation public key at /home/ubuntu/fabric-samples/multihost_k
8s/organization/orderer.example.com/users/tlsadmin/IssuerRevocationPublicKey
+ set +x
✓ Finalizing Orderer MSP

✓ Creating orderer.example.com Orderers and Users Certificates in K8s
Defaulted container "orderer-ca" out of: orderer-ca, orderer-ca-init (init)
✓ Create local MSP config.yaml
+ echo 'NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: orderer'

✓ Generating the orderer0.orderer.example.com msp
+ set +x
+ fabric-ca-client enroll --caname orderer-ca --mspdir /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer0.orderer.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athe
ns,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts orderer0.orde
rer.example.com,orderer0-orderer,orderer0-orderer.fabric-network-orderer,orderer0.orderer.example.com -m
orderer0.orderer.example.com --url https://orderer0.orderer.example.com:orderer0@ica.orderer.example.com:
7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /var/hyper
ledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-client/tls/
client/client.key
2023/02/28 14:36:02 [INFO] Created a default configuration file at /var/hyperledger/fabric-ca-client/fabr
ic-ca-client-config.yaml
2023/02/28 14:36:02 [INFO] TLS Enabled
2023/02/28 14:36:02 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:02 [INFO] encoded CSR
2023/02/28 14:36:02 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer0.orderer.example.com/msp/signcerts/cert.pem
2023/02/28 14:36:02 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer0.orderer.example.com/msp/cacerts/ica-orderer-example-com-
7054-orderer-ca.pem
2023/02/28 14:36:02 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer0.orderer.example.com/msp/intermediatecerts/ica-o
rderer-example-com-7054-orderer-ca.pem
2023/02/28 14:36:02 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer0.orderer.example.com/msp/IssuerPublicKey
2023/02/28 14:36:02 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer0.orderer.example.com/msp/IssuerRevocationPublicK
ey
+ set +x


✓ Generating the orderer1.orderer.example.com msp
+ fabric-ca-client enroll --caname orderer-ca --mspdir /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer1.orderer.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athe
ns,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts orderer1.orde
rer.example.com,orderer1-orderer,orderer1-orderer.fabric-network-orderer,orderer1.orderer.example.com -m
orderer1.orderer.example.com --url https://orderer1.orderer.example.com:orderer1@ica.orderer.example.com:
7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /var/hyper
ledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-client/tls/
client/client.key
2023/02/28 14:36:02 [INFO] TLS Enabled
2023/02/28 14:36:02 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:02 [INFO] encoded CSR
2023/02/28 14:36:02 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer1.orderer.example.com/msp/signcerts/cert.pem
2023/02/28 14:36:02 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer1.orderer.example.com/msp/cacerts/ica-orderer-example-com-
7054-orderer-ca.pem
```

```
2023/02/28 14:36:02 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer1.orderer.example.com/msp/intermediatecerts/ica-o
rderer-example-com-7054-orderer-ca.pem
2023/02/28 14:36:02 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer1.orderer.example.com/msp/IssuerPublicKey
2023/02/28 14:36:02 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer1.orderer.example.com/msp/IssuerRevocationPublicK
ey
+ set +x

✓ Generating the orderer2.orderer.example.com msp
+ fabric-ca-client enroll --caname orderer-ca --mspdir /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer2.orderer.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athe
ns,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts orderer2.orde
rer.example.com,orderer2-orderer,orderer2-orderer.fabric-network-orderer,orderer2.orderer.example.com -m
orderer2.orderer.example.com --url https://orderer2.orderer.example.com:orderer2@ica.orderer.example.com:
7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /var/hyper
ledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-client/tls/
client/client.key
2023/02/28 14:36:02 [INFO] TLS Enabled
2023/02/28 14:36:02 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:03 [INFO] encoded CSR
2023/02/28 14:36:03 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer2.orderer.example.com/msp/signcerts/cert.pem
2023/02/28 14:36:03 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer2.orderer.example.com/msp/cacerts/ica-orderer-example-com-
7054-orderer-ca.pem
2023/02/28 14:36:03 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer2.orderer.example.com/msp/intermediatecerts/ica-o
rderer-example-com-7054-orderer-ca.pem
2023/02/28 14:36:03 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer2.orderer.example.com/msp/IssuerPublicKey
2023/02/28 14:36:03 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer2.orderer.example.com/msp/IssuerRevocationPublicK
ey
+ set +x
✓ Generating User1 msp
+ fabric-ca-client enroll --caname orderer-ca --mspdir /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/users/User1@orderer.example.com/msp --csr.names 'C=GR,ST=Athens,L=Athens,O=o
rderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts User1@orderer.examp
le.com,User1-orderer,User1-orderer.fabric-network-orderer,User1.orderer.example.com -m User1@orderer.exam
ple.com --url https://User1@orderer.example.com:userpw1.orderer.example.com@ica.orderer.example.com:7054
--tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /var/hyperledge
r/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-client/tls/clien
t/client.key
2023/02/28 14:36:03 [INFO] TLS Enabled
2023/02/28 14:36:03 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:03 [INFO] encoded CSR
2023/02/28 14:36:03 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/users/User1@orderer.example.com/msp/signcerts/cert.pem
2023/02/28 14:36:03 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/users/User1@orderer.example.com/msp/cacerts/ica-orderer-example-com-7054-o
rderer-ca.pem
2023/02/28 14:36:03 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/users/User1@orderer.example.com/msp/intermediatecerts/ica-orderer
-example-com-7054-orderer-ca.pem
2023/02/28 14:36:03 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/users/User1@orderer.example.com/msp/IssuerPublicKey
2023/02/28 14:36:03 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/users/User1@orderer.example.com/msp/IssuerRevocationPublicKey
+ set +x

✓ Generating the orderer0.orderer.example.com TLS
+ fabric-ca-client enroll --caname orderer-tlsca --mspdir /var/hyperledger/fabric/crypto-config/ordererOr
ganizations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp --csr.names 'C=GR,ST=Athens
,L=Athens,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts ordere
r0.orderer.example.com,orderer0-orderer,orderer0-orderer.fabric-network-orderer,orderer0.orderer.example.
com -m orderer0.orderer.example.com --url https://orderer0.orderer.example.com:orderer0@ica.orderer.examp
le.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /va
r/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-clie
nt/tls/client/client.key
2023/02/28 14:36:03 [INFO] TLS Enabled
2023/02/28 14:36:03 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:03 [INFO] encoded CSR
2023/02/28 14:36:03 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp/signcerts/cert.pem
2023/02/28 14:36:03 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp/cacerts/ica-orderer-example-
com-7054-orderer-tlsca.pem
2023/02/28 14:36:03 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp/intermediatecerts/i
ca-orderer-example-com-7054-orderer-tlsca.pem
2023/02/28 14:36:03 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp/IssuerPublicKey
2023/02/28 14:36:03 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer0.orderer.example.com/tls/msp/IssuerRevocationPub
licKey
+ set +x
```

```
✓ Generating the orderer1.orderer.example.com TLS
+ fabric-ca-client enroll --caname orderer-tlsca --mspdir /var/hyperledger/fabric/crypto-config/ordererOr
ganizations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp --csr.names 'C=GR,ST=Athens
,L=Athens,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts ordere
r1.orderer.example.com,orderer1-orderer,orderer1-orderer.fabric-network-orderer,orderer1.orderer.example.
com -m orderer1.orderer.example.com --url https://orderer1.orderer.example.com:orderer1@ica.orderer.examp
le.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /va
r/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-clie
nt/tls/client/client.key
2023/02/28 14:36:04 [INFO] TLS Enabled
2023/02/28 14:36:04 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:04 [INFO] encoded CSR
2023/02/28 14:36:04 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp/signcerts/cert.pem
2023/02/28 14:36:04 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp/cacerts/ica-orderer-example-
com-7054-orderer-tlsca.pem
2023/02/28 14:36:04 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp/intermediatecerts/i
ca-orderer-example-com-7054-orderer-tlsca.pem
2023/02/28 14:36:04 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp/IssuerPublicKey
2023/02/28 14:36:04 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer1.orderer.example.com/tls/msp/IssuerRevocationPub
licKey
+ set +x

✓ Generating the orderer2.orderer.example.com TLS
+ fabric-ca-client enroll --caname orderer-tlsca --mspdir /var/hyperledger/fabric/crypto-config/ordererOr
ganizations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp --csr.names 'C=GR,ST=Athens
,L=Athens,O=orderer.example.com' --csr.keyrequest.algo ecdsa --csr.keyrequest.size 384 --csr.hosts ordere
r2.orderer.example.com,orderer2-orderer,orderer2-orderer.fabric-network-orderer,orderer2.orderer.example.
com -m orderer2.orderer.example.com --url https://orderer2.orderer.example.com:orderer2@ica.orderer.examp
le.com:7054 --tls.certfiles /var/hyperledger/fabric-ca-client/tls/client/ca.crt --tls.client.certfile /va
r/hyperledger/fabric-ca-client/tls/client/client.crt --tls.client.keyfile /var/hyperledger/fabric-ca-clie
nt/tls/client/client.key
2023/02/28 14:36:04 [INFO] TLS Enabled
2023/02/28 14:36:04 [INFO] generating key: &{A:ecdsa S:384}
2023/02/28 14:36:04 [INFO] encoded CSR
2023/02/28 14:36:04 [INFO] Stored client certificate at /var/hyperledger/fabric/crypto-config/ordererOrga
nizations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp/signcerts/cert.pem
2023/02/28 14:36:04 [INFO] Stored root CA certificate at /var/hyperledger/fabric/crypto-config/ordererOrg
anizations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp/cacerts/ica-orderer-example-
com-7054-orderer-tlsca.pem
2023/02/28 14:36:04 [INFO] Stored intermediate CA certificates at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp/intermediatecerts/i
ca-orderer-example-com-7054-orderer-tlsca.pem
2023/02/28 14:36:04 [INFO] Stored Issuer public key at /var/hyperledger/fabric/crypto-config/ordererOrgan
izations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp/IssuerPublicKey
2023/02/28 14:36:04 [INFO] Stored Issuer revocation public key at /var/hyperledger/fabric/crypto-config/o
rdererOrganizations/orderer.example.com/orderers/orderer2.orderer.example.com/tls/msp/IssuerRevocationPub
licKey
+ set +x
✓ Finalizing orderer.example.com MSP
✓ Finished
✓ Loading Orderer Config Files
configmap/orderer-config created
✓ Waiting until Orderer Config Files Installation is Completed
✓ Loading orderer0.orderer.example.com ConfigMap
configmap/orderer0-env created
✓ Waiting until orderer0.orderer.example.com ConfigMap is Completed
✓ Deploying orderer0.orderer.example.com Manifest
certificate.cert-manager.io/orderer0-tls-cert created
certificate.cert-manager.io/orderer0-admin-tls-cert created
certificate.cert-manager.io/orderer0-admin-client-tls-cert created
certificate.cert-manager.io/orderer0-operations-tls-cert created
certificate.cert-manager.io/orderer0-operations-client-tls-cert created
deployment.apps/orderer0-orderer created
service/orderer0-orderer created
ingress.networking.k8s.io/orderer0-orderer created
✓ Waiting until orderer0.orderer.example.com Deployment is Completed
✓ Loading orderer1.orderer.example.com ConfigMap
configmap/orderer1-env created
✓ Waiting until orderer1.orderer.example.com ConfigMap is Completed
✓ Deploying orderer1.orderer.example.com Manifest
certificate.cert-manager.io/orderer1-tls-cert created
certificate.cert-manager.io/orderer1-admin-tls-cert created
certificate.cert-manager.io/orderer1-admin-client-tls-cert created
certificate.cert-manager.io/orderer1-operations-tls-cert created
certificate.cert-manager.io/orderer1-operations-client-tls-cert created
deployment.apps/orderer1-orderer created
service/orderer1-orderer created
ingress.networking.k8s.io/orderer1-orderer created
✓ Waiting until orderer1.orderer.example.com Deployment is Completed
✓ Loading orderer2.orderer.example.com ConfigMap
configmap/orderer2-env created
✓ Waiting until orderer2.orderer.example.com ConfigMap is Completed
```

```
✓ Deploying orderer2.orderer.example.com Manifest
certificate.cert-manager.io/orderer2-tls-cert created
certificate.cert-manager.io/orderer2-admin-tls-cert created
certificate.cert-manager.io/orderer2-admin-client-tls-cert created
certificate.cert-manager.io/orderer2-operations-tls-cert created
certificate.cert-manager.io/orderer2-operations-client-tls-cert created
deployment.apps/orderer2-orderer created
service/orderer2-orderer created
ingress.networking.k8s.io/orderer2-orderer created
✓ Waiting until orderer2.orderer.example.com Deployment is Completed
✓ Waiting until Orderer Installation is Completed
✓ Orderer Installation is Completed
```

Figure 33: Outcome of './networkStart.sh deploy' command for Organization Orderer

Finally, in order to verify if everything is up and running, open **Kubernetes Dashboard**, switch to **fabric-network-orderer** namespace and navigate to the **Pods** section and the following image should be appeared:

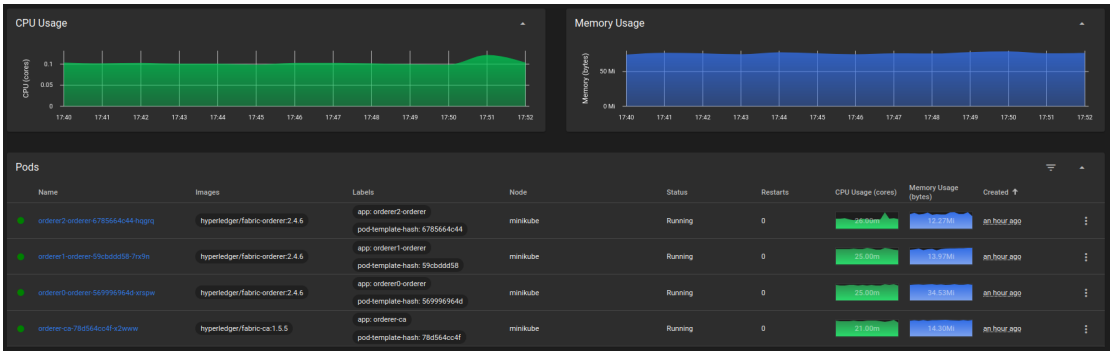

Figure 34: Organization Orderer successfully deployed

As depicted in the image above, three Orderers Nodes (consisting of three Pods **orderer0-orderer**, **orderer1-orderer** and **orderer2-orderer**) and a Fabric CA (consisting of one Pod **orderer-ca**) are successfully deployed in Kubernetes.

4.3.6. Create Channel Artifacts

At this point, all the necessary components of the **Org1 Organization** and the **Orderer Organization** are deployed in Kubernetes. Since all the components are up and running the next step is to start formulating the Hyperledger Fabric network, by collecting MSP information from the deployed organizations and creating the channel artifacts **(genesis block)**, by leveraging the official binary **configtxgen**. This is an operation that can be only performed by the **Orderer Organization**, thus the **orderer profile** must be selected.

To create the **genesis block** the following command is issued:

```
cd multihost_k8s
./networkStart.sh artifacts create
```

And the result is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh artifacts create
✓ Creating Channel Artifacts
✓ Creating Channel mychannel genesis block
✓ Channel Artifacts are Ready
```

Figure 35: Outcome of './networkStart.sh artifacts create' command

4.3.7. Create Channel

Since the **genesis block** is created, the next step is to create a channel, by using the channel artifacts that were created before. This is an operation that can be only performed by the **Orderer Organization**, thus the **orderer profile** must be selected.

A Hyperledger Fabric channel **[48]** serves as a secluded communication "**subnet**" connecting two or more designated network members, with the primary aim of facilitating confidential and private transactions. A channel's definition includes the participating members (organizations), anchor peers representing each member, the shared ledger, one or more chaincode applications, and the ordering service node(s). In the Fabric network, every transaction takes place within the context of a channel. Authentication and authorization are mandatory for all participants involved in transactions on a specific channel. When a peer joins a channel, it is assigned a unique identity by a Membership Services Provider (MSP). This identity serves to authenticate the peer to other channel peers and services. This ensures that every participant in a channel is recognized and authorized appropriately.

In our case, only one organization (**Org1 Organization**) is deployed, thus the **genesis block** has MSP information only for this particular organization. The channel name that is going to be used is declared in the .env file. This means that only members of the **Org1 Organization** can submit transactions to the channel **mychannel**[44].

To create the channel, the orderer organization relies on the official binary **osnadmin**, under the condition that the **Admin API** is enabled on the Orderers. The following command is issued:

```
cd multihost_k8s
./networkStart.sh channel create
```

And the result is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh channel create
✓ Creating Channel mychannel
✓ Joining orderer.example.com Orderers to channel mychannel
✓ Retrieving orderer0.orderer.example.com Admin Certs
✓ Adding FQDNs entry to /etc/hosts
[sudo] password for ubuntu:
✓ Joining orderer orderer0.orderer.example.com to channel mychannel
✓ Checking if orderer orderer0.orderer.example.com has joined the channel mychannel
✓ Orderer orderer0.orderer.example.com has joined the channel mychannel
{
  "name": "mychannel",
  "url": "/participation/v1/channels/mychannel",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
✓ Retrieving orderer1.orderer.example.com Admin Certs
✓ Adding FQDNs entry to /etc/hosts
✓ Joining orderer orderer1.orderer.example.com to channel mychannel
✓ Checking if orderer orderer1.orderer.example.com has joined the channel mychannel
✓ Orderer orderer1.orderer.example.com has joined the channel mychannel
{
  "name": "mychannel",
  "url": "/participation/v1/channels/mychannel",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
✓ Retrieving orderer2.orderer.example.com Admin Certs
✓ Adding FQDNs entry to /etc/hosts
✓ Joining orderer orderer2.orderer.example.com to channel mychannel
✓ Checking if orderer orderer2.orderer.example.com has joined the channel mychannel
✓ Orderer orderer2.orderer.example.com has joined the channel mychannel
```

[44] **NOTE**: *When the channel is already created and a new Organization wants to be a member of the channel, then specific operations must be performed, that falls under the Updating a channel configuration section. At the moment, this project assumes that the members of the channels are all the deployed organizations that have in their namespace a specific **PREFIX** that is declared in the .env*

Figure 36: Outcome of './networkStart.sh channel create' command

In the aforementioned image it is clearly depicted that all the orderer nodes have successfully joined the channel **mychannel**.

4.3.8. Join Organization

At the moment, the channel **mychannel** is **ACTIVE** but the organization is not a member of the channel yet. Thus it is necessary that the peers of the organization become a member of the channel. This is an operation that can be performed only by the **organization**, thus the **organization profile** must be selected. This can be done by issuing the command below:

```
cd multihost_k8s
./networkStart.sh channel join
```

And the result is presented below:

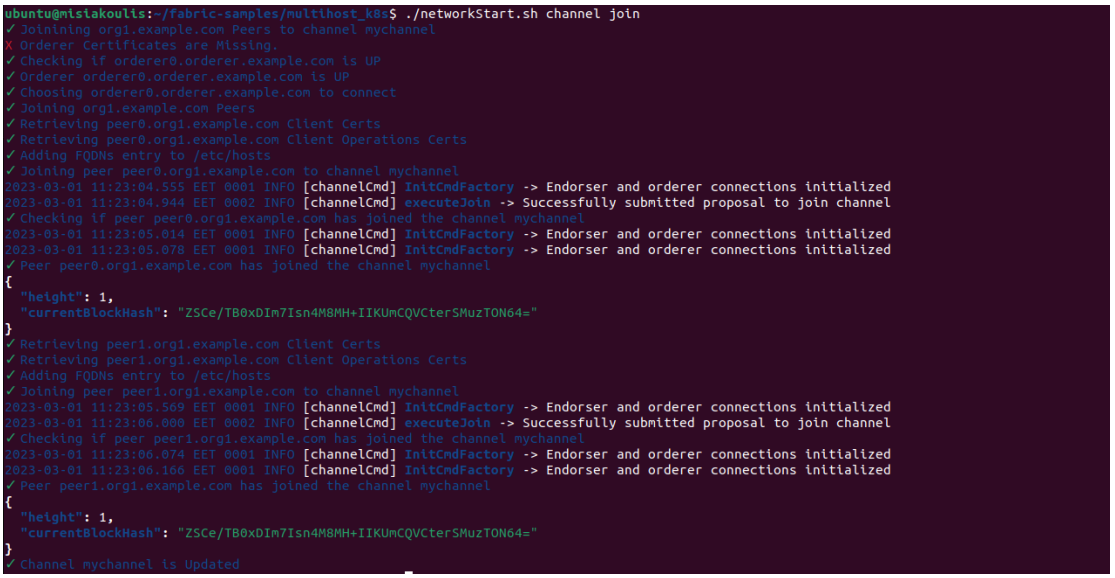

Figure 37: Outcome of './networkStart.sh channel join' command

In the aforementioned image it is clearly depicted that all the peer nodes of the organization have successfully joined the channel **mychannel**.

4.3.9. Deploy Chaincode[45]

Since all peers are now members of the channel, the next step is to deploy the chaincode server into the Kubernetes. The chaincode is going to be deployed using the **CCAAS** method. This is an operation that can be performed only by the **organization**, thus the **organization profile** must be selected.

[45] ***ATTENTION: This page is for advanced users who choose to configure chaincode as a service external to the Fabric peer. Proceed with caution!***

In Fabric v2.0 **[49]**, there is support for deploying and executing chaincode independently from the Fabric peer. This feature allows users to manage the chaincode runtime separately, offering flexibility in deploying chaincode within Fabric cloud environments like **Kubernetes**. Rather than building and launching chaincode on each individual peer, chaincode can be executed as a service external to Fabric. This functionality leverages the Fabric v2.0 external builder and launcher capabilities, which empower operators to enhance a peer with tools for building, launching, and discovering chaincode. This approach simplifies the deployment and management of chaincode in Fabric cloud deployments and similar environments.

Before the introduction of external builders, chaincode packages needed to include a collection of source code files in a specific programming language that could be compiled and executed as a chaincode binary. However, the new external build and launcher functionality offers users the flexibility to customize the build process optionally. When it comes to running chaincode as an external service, the build process enables you to specify the endpoint details of the server where the chaincode is hosted and running. Consequently, the package primarily comprises information about the externally hosted chaincode server endpoint and the necessary TLS (Transport Layer Security) artifacts for ensuring secure connections. This allows for more dynamic and adaptable deployment and execution of chaincode in Fabric environments[46] [47].

Before proceeding to chaincode deployment, it is essential to modify the smart contract in order to be able to connect externally with the organization peers using **mutual TLS**. The modifications of the **fabricvdr.go** smart contract are depicted below:

Smart Contract as a CCAAS

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
package main

import (
    "io/ioutil"
    "strconv"
    "strings"
    "fmt"
    "log"
    "os"

    "github.com/hyperledger/fabric-chaincode-go/shim"
```

[46] NOTE: TLS is optional but highly recommended for all environments except a simple test environment. In this project, *TLS* is enabled by default.

[47] NOTE: This is an advanced feature that will likely require custom packaging of the peer image.

```
    "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

// SmartContract provides functions for managing a main
type SmartContract struct {
    contractapi.Contract
}

// CreateDoc creates a new DocResolution struct and stores it in the ledger.
func (s *SmartContract) CreateDoc(ctx contractapi.TransactionContextInterface, clientId
string) error {
 // Deducted from brevity. Defined in the Smart Contracts Section
}

// ResolveDid returns the DocResolution record stored in the world state under specified
did
func (s *SmartContract) ResolveDid(ctx contractapi.TransactionContextInterface, docId
string) (string, error) {
 // Deducted from brevity. Defined in the Smart Contracts Section
}

////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////// Helper Functions ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
// getState returns the DocResolution record stored in the world state with given input
func getState(ctx contractapi.TransactionContextInterface, input string) ([]byte, error) {
 // Deducted from brevity. Defined in the Smart Contracts Section
}

////////////////////////////////////////////////////////////////////////////////////////
// From this line and below, leave it as it is. Copy the chaicode above this line and
// the developer is responsible to solve the dependencies before proceed
////////////////////////////////////////////////////////////////////////////////////////
func main() {
 ${CHAINCODE_NAME}, err := contractapi.NewChaincode(new(SmartContract))
 if err != nil {
  log.Panicf("Error creating ${CHAINCODE_NAME} chaincode: %s", err)
 }

 server := &shim.ChaincodeServer{
  CCID: os.Getenv("CHAINCODE_ID"),
  Address: os.Getenv("CHAINCODE_SERVER_ADDRESS"),
  CC: ${CHAINCODE_NAME},
  TLSProps: getTLSProperties(),
 }

 if err := server.Start(); err != nil {
  log.Panicf("Error starting ${CHAINCODE_NAME} chaincode: %s", err)
 }
}

func getTLSProperties() shim.TLSProperties {
 // Check if chaincode is TLS enabled
 tlsDisabledStr := getEnvOrDefault("CHAINCODE_TLS_DISABLED", "true")
 key := getEnvOrDefault("CHAINCODE_TLS_KEY", "")
 cert := getEnvOrDefault("CHAINCODE_TLS_CERT", "")
 clientCACert := getEnvOrDefault("CHAINCODE_CLIENT_CA_CERT", "")

 // Convert tlsDisabledStr to boolean
 tlsDisabled := getBoolOrDefault(tlsDisabledStr, false)
 var keyBytes, certBytes, clientCACertBytes []byte
 var err error
 if !tlsDisabled {
  keyBytes, err = ioutil.ReadFile(key)
  if err != nil {
   log.Panicf("Error while reading the crypto file: %s", err)
  }
  certBytes, err = ioutil.ReadFile(cert)
  if err != nil {
   log.Panicf("Error while reading the crypto file: %s", err)
  }
 }
```

```
 // Did not request for the peer cert verification
 if clientCACert != "" {
  clientCACertBytes, err = ioutil.ReadFile(clientCACert)
  if err != nil {
   log.Panicf("Error while reading the crypto file: %s", err)
  }
 }

 return shim.TLSProperties{
  Disabled:     tlsDisabled,
  Key:          keyBytes,
  Cert:         certBytes,
  ClientCACerts: clientCACertBytes,
 }
}

func getEnvOrDefault(env, defaultVal string) string {
 value, ok := os.LookupEnv(env)
 if !ok {
  value = defaultVal
 }
 return value
}

// Note that the method returns default value if the string
// cannot be parsed!
func getBoolOrDefault(value string, defaultVal bool) bool {
 parsed, err := strconv.ParseBool(value)
 if err != nil {
  return defaultVal
 }
 return parsed
}
```

Table 20: Smart Contract as a CCAAS

Place the modified smart contract within the **organization/ccExternalTemplate/chaincode** directory or modify the **CHAINCODE_FILE** environmental variable from the .env to reflect the actual absolute path of the smart contract.

Either way, the necessary environmental variables that handle the chaincode operations are the following:

```
CHAINCODE_FILE=${CHAINCODE_EXTERNAL_TEMPLATE}/chaincode/fabricvdr.go
CHAINCODE_NAME=fabricvdr
CHAINCODE_VERSION=1.0
CHAINCODE_SEQUENCE=1
```

Since the chaincode environmental variables are set, then in order to deploy the chaincode server in the **Kubernetes** the following command is issued:

```
cd multihost_k8s
./networkStart.sh chaincode deploy
```

And the result is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/multihost_k8s$ ./networkStart.sh chaincode deploy
--- Executing with the following:
- ORGANIZATION HOSTNAME: org1
- ORGANIZATION DOMAIN: example.com
- ORGANIZATION TYPE: peer
- NUMBER OF PEERS: 2
- CHANNEL NAME: mychannel
- NAMESPACE: fabric-network-org1
- CHAINCODE NAME: fabricvdr
- CHAINCODE GO FILE: /home/ubuntu/fabric-samples/multihost_k8s/organization/ccExternalTemplate/chaincode/fabricvdr.go
- CHAINCODE VERSION: 1.0
- CHAINCODE LABEL: fabricvdr_1.0
- CHAINCODE TLS DISABLED: false
- CHAINCODE BUILDER ccaas
```

```
✓ Deploying Chaincode to K8s
✓ Copying Chaincode into org1.example.com
✓ Checking the Chaincode Package Name
✓ Creating Chaincode Dockerfile
✓ Building org1.example.com Dockerfile
✓ Container Digest: sha256:11ecdc1b93021d3b540df2db95f9d2e66dd41e9615ea970829cc301ddc4431c8
✓ Publishing image to Minikube Local docker registry
✓ Packaging CCAAS Chaincode fabricvdr

certificate.cert-manager.io/org1-cc-client-tls-cert configured
✓ Deploying Chaincode Client Certificates
✓ Retrieving Chaincode Client Certificates
✓ Creating connection.json for org1.example.com Peers
✓ Creating metadata.json for org1.example.com Peers
✓ Creating fabricvdr.tar.gz for org1.example.com Peers
✓ Exporting Package ID for org1.example.com Peers

✓ Finding the Deployed Peers for org1.example.com
✓ Launching peer0.org1.example.com Chaincode ConfigMap
configmap/peer0-org1-fabricvdr-env created
✓ Waiting until peer0.org1.example.com Chaincode ConfigMap is Completed
✓ Launching peer0.org1.example.com Chaincode Manifest
certificate.cert-manager.io/peer0-org1-cc-tls-cert configured
deployment.apps/peer0-org1-fabricvdr created
service/peer0-org1-fabricvdr created
✓ Waiting until peer0.org1.example.com Chaincode Deployment is Completed
✓ Launching peer1.org1.example.com Chaincode ConfigMap
configmap/peer1-org1-fabricvdr-env created
✓ Waiting until peer1.org1.example.com Chaincode ConfigMap is Completed
✓ Launching peer1.org1.example.com Chaincode Manifest
certificate.cert-manager.io/peer1-org1-cc-tls-cert configured
deployment.apps/peer1-org1-fabricvdr created
service/peer1-org1-fabricvdr created
✓ Waiting until peer1.org1.example.com Chaincode Deployment is Completed
✓ Chaincode is Deployed
```

Figure 38: Outcome of './networkStart.sh chaincode deploy' command

In the aforementioned image it is clearly depicted that the chaincode package is automatically created by the script and deployed as Pod in the Kubernetes, thus now it is feasible to install the chaincode on the peers.

To verify whether the chaincode server is deployed, open the **Kubernetes Dashboard**, select the **fabric-network-org1** namespace and navigate to the **Pods** section. The following image should be appeared:

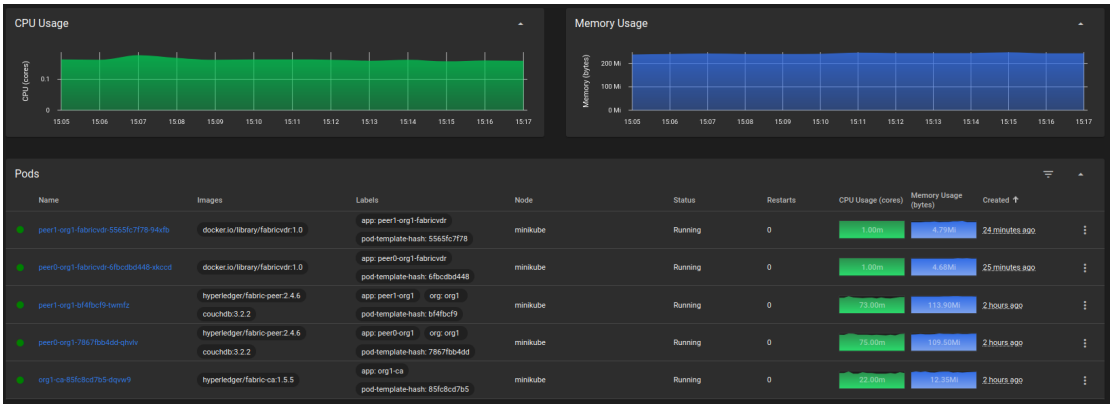

Figure 39: Chaincode fabricvdr successfully deployed

In the aforementioned image it is clearly depicted that two additional **Pods** (consisting of **peer0-org1-fabricvdr** and **peer1-org1-fabricvdr**) were utilized, denoting that two chaincode servers (one per peer) are successfully deployed.

4.3.10. Install Chaincode

At the moment, the chaincode servers are deployed in the **Kubernetes** as **Pods**, but the peers are unaware of the chaincode's existence. The chaincode needs to be installed on every peer that will endorse a transaction. This is an operation that can be performed only by the **organization**, thus the **organization profile** must be selected. The following command is issued:

```
cd multihost_k8s
./networkStart.sh chaincode install
```

And the result is presented below:

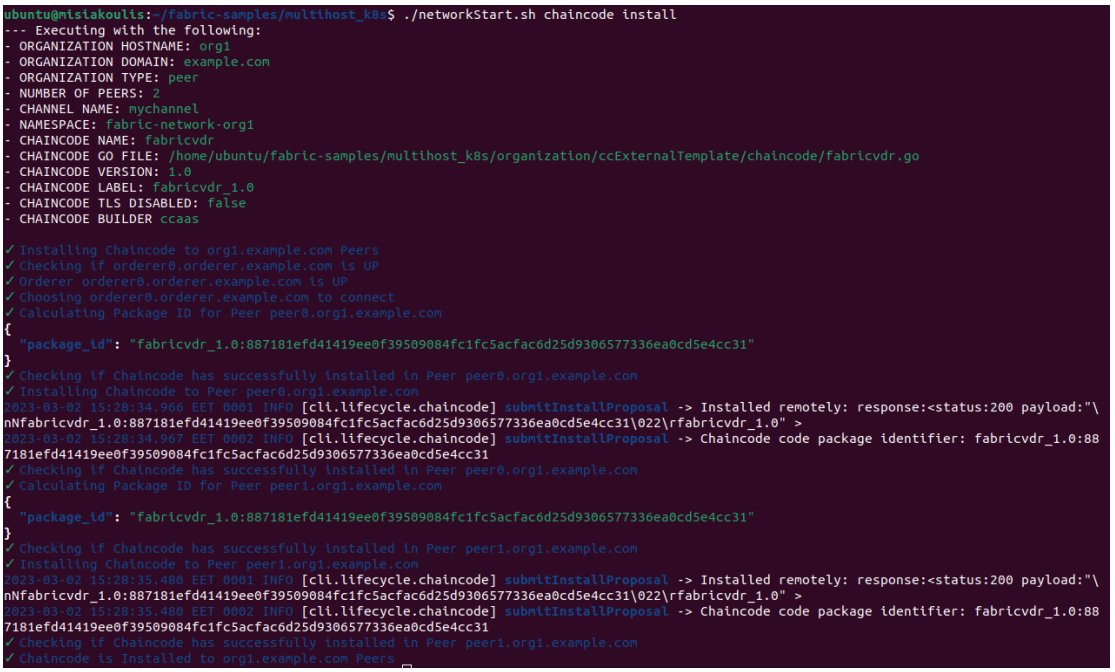

Figure 40: Outcome of './networkStart.sh chaincode install' command

In the aforementioned image it is clearly depicted that the chaincode is installed in both peers. After that, the peers are able to discover the new chaincode and connect with it using **mutual TLS**.

4.3.11. Approve Chaincode

After the chaincode package is installed, the chaincode definition must be approved for the organization. This is an operation that can be performed only by the **organization**, thus the **organization profile** must be selected. The definition of chaincode governance encompasses crucial parameters such as the chaincode **name**, **version**, and the **endorsement policy** that determines who needs to approve the chaincode.

The endorsement policy for determining the set of channel members required to approve a chaincode before deployment is governed by the **/Channel/Application/LifecycleEndorsement** policy. By default, this policy mandates that a majority of channel members must provide their approval for a chaincode to become operational on the channel. In the given scenario, where there's only one organization participating in the channel, and the majority within that context is just one, the chaincode approval process is simplified. It means that the chaincode only needs to be approved by the **Org1 Organization** for it to be deployed and utilized on the channel.

To approve the chaincode definition by the organization, the following command is issued:

```
cd multihost_k8s
./networkStart.sh chaincode approve
```

And the result is presented below:

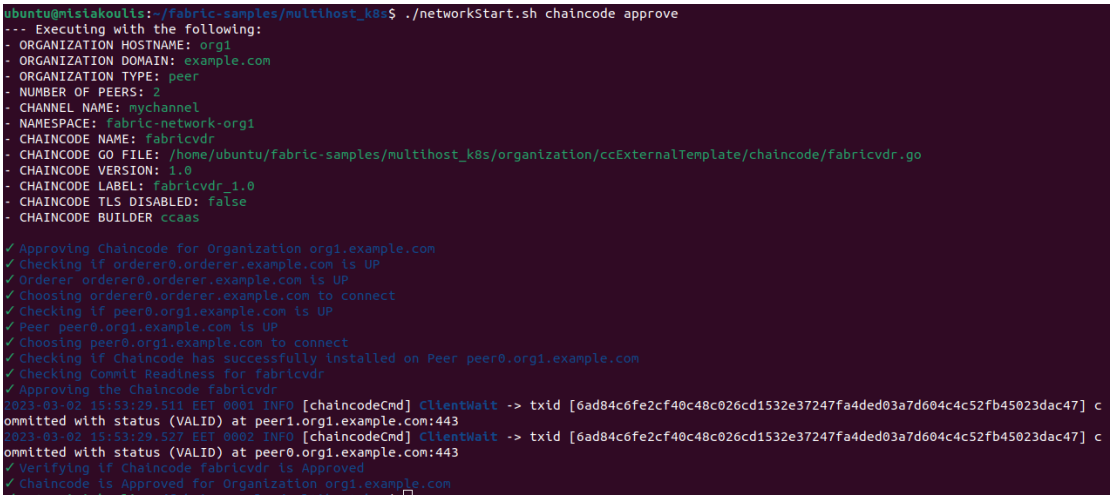

Figure 41: Outcome of './networkStart.sh chaincode approve' command

In the aforementioned image it is clearly depicted that the chaincode is approved by both peers in the organization, thus the organization has approved the chaincode definition. After that, the chaincode is ready to be committed in the channel **mychannel**[48].

4.3.12. Commit Chaincode

Once an adequate number of organizations have given their approval to a chaincode definition, any single organization within the channel can initiate the process of committing the chaincode definition to the channel. If a majority of channel members have indeed endorsed the definition, the commit transaction will be successful. As a result, the parameters and rules outlined in the chaincode definition will be implemented and become effective on the channel for all participating organizations to use.

This process ensures that there is consensus and agreement among the majority of channel members before new chaincode is introduced, enhancing the security and trustworthiness of the network. This is an operation that can be performed only by the **organization**, thus the **organization profile** must be selected.

In the given scenario, because we have only one organization on the channel, and a majority of **one** is **one**, then the chaincode must be committed by the **Org1 Organization**[49].

The following command is issued:

```
cd multihost_k8s
./networkStart.sh chaincode commit
```

And the result is presented below:

[48] **NOTE: If more than one organization exists in the network, then the majority of the organization must have approved the chaincode definition, before proceeding to the next step, which is committing the chaincode to the channel.**

[49] **NOTE: From the majority of the participants that have approved the chaincode definition, only one is responsible to commit the chaincode definition into the channel.**

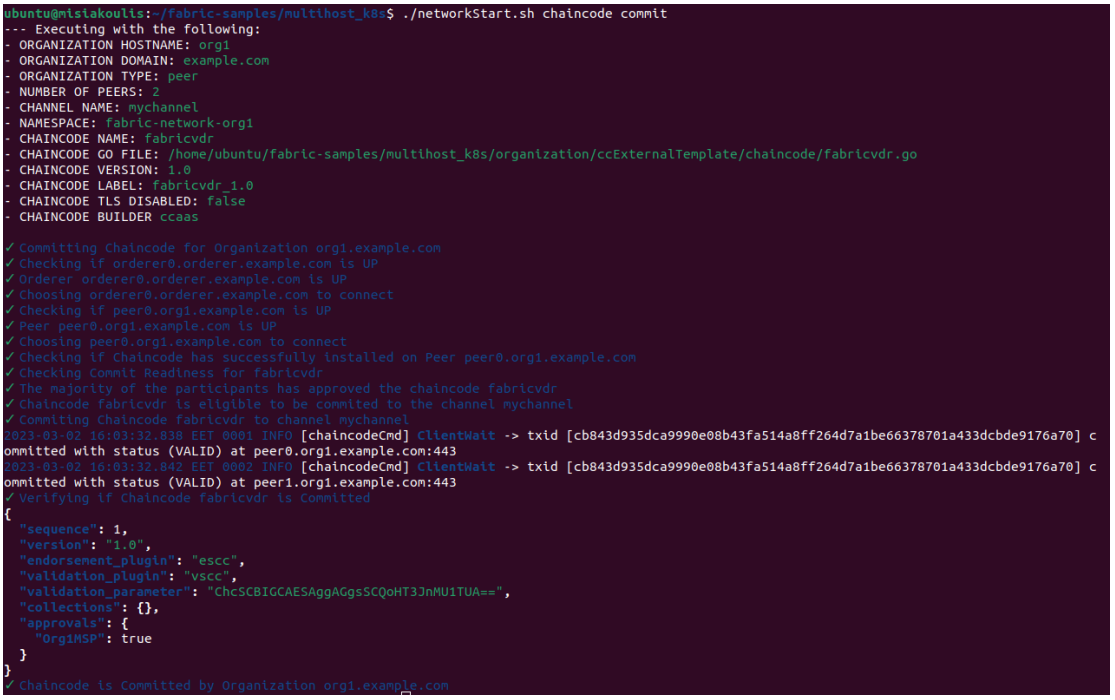

Figure 42: Outcome of './networkStart.sh chaincode commit' command

In the aforementioned image it is clearly depicted that the chaincode is successfully committed in the channel by the **organization**. After that, all the participants of the network that have approved the chaincode definition will be able to submit transactions in the channel **mychannel**[50].

4.3.13. (Optional) Destroy Network

If something went terribly wrong, and the restart of the network is unavoidable, then the following command must be issued:

```
cd multihost_k8s
./networkDown.sh
```

And the result is presented below:

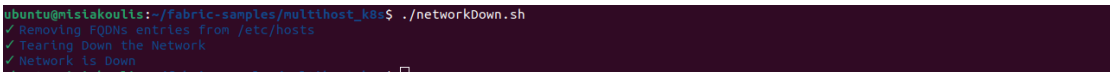

Figure 43: Outcome of './networkDown.sh' command

This command will eventually delete all the Hyperledger Fabric Blockchain network components from the Kubernetes.

4.3.14. (Optional) Destroy Minikube

If **Minikube Kubernetes** at some point becomes unresponsive or it is necessary to destroy the running Minikube instance then the following command must be issued:

[50] **NOTE: It is possible to update the business logic of the smart contract to another version by increasing the *CHAINCODE_VERSION* and *CHAINCODE_SEQUENCE* environmental variables each time by 1. Then the same procedure applies as if a new chaincode was deployed in the Kubernetes. With the chaincode endpoint deployed to the peer and if the chaincode is running, you can continue the normal process of committing the chaincode definition to the channel and invoking the chaincode.**

```
cd multihost_k8s
./minikube.sh down
```

And the result is presented below:

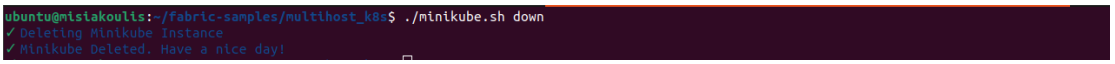

Figure 44: Outcome of './minikube.sh down' command

To verify in Minikube instance is actually destroyed, the following command must be issued:

```
cd multihost_k8s
./minikube.sh status
```

And the result is presented below:

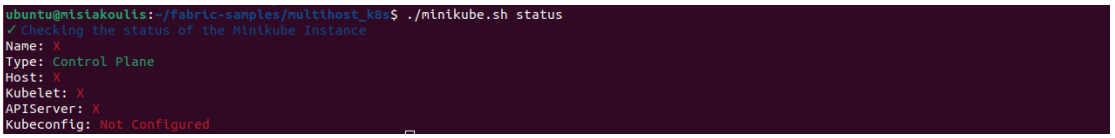

Figure 45: Outcome of './minikube.sh status' command when Minikube is down

As depicted, the **Minikube** instance is no longer running.

4.4. VDR gRPC Server

4.4.1. Design

In our case, in order to be able to facilitate all the communications between a VDR gRPC client and the VDR gRPC server, a long-lived gRPC bidirectional streaming Server was utilized with a custom **GOB** Encoder/Decoder, instead of the traditional Protobuf. Thus, a proto file was defined, that holds information regarding the type of messages and the type of services that this gRPC server will eventually handle.

After the proto file definition is finalized, the proto file must be compiled (using the Proto Compiler or **Protoc**[51]), in order to be able to produce the necessary Protobufs. Usually, depending on the language that the compiler compiles the proto file, two files are created. One file defines the messages and the structs that are exchanged between the bidirectional channel, and the second file holds all the necessary functions that will eventually be used by the gRPC server and the gRPC client, in order to establish connection and to connect to the exposed RPC. Since we used a custom **GOB** Encoder/Decoder, the first file that has information regarding the structs and the messages is going to be omitted.

In the following table the **.proto** file that was used in our case is presented:

Protofile
grpcFabricVdrAPI.proto

```
syntax = "proto3";
option go_package = "./grpcFabricVdrAPI";
package grpcFabricVdrAPI;

// Message Streaming
```

[51] Installation Instructions may be required and appeared in **Appendix C**, in the Proto Compiler Installation Guide table

```
message StreamingRawBytes {}

// Services
service GrpcFabricVdrEndpoints {
    // Services Endpoint
    rpc Services(stream StreamingRawBytes) returns (stream StreamingRawBytes);
}
```

Table 21: VDR gRPC Server Protofile

As presented in the table above, in the **service** section of the proto file, only one Service is declared which is called **Services**. The **Services** is actually a bidirectional streaming RPC, meaning that upon connection, a bidirectional channel is established between the client and the server that each one of them is responsible for keeping it alive on their side, by utilizing "Keepalive Settings" which pings each other to determine the state of the stream, and therefore the state of the service.

Upon compilation, the **grpcFabricVdrAPI_grpc.pb.go**[52] file was generated that holds the information regarding the functions that both client and server will eventually use in order to be able to connect to the bidirectional RPC service.

Since, we are using stream of **GOBs** as custom **Encoder/Decoder** instead of using **Protobuf** the following custom **Codec** was utilized in the client as well as in the server that Marshals and Unmarshal messages from the bidirectional channel:

Custom Codec
codec.go

```
package codec

import (
    "bytes"
    "encoding/gob"
)

// GOB Codec
// Implements grpc.Codec compatible gob codec.
type Codec struct{}

// Register function the map[string]interface in GOB
func Register() bool {
 gob.Register(map[string]interface{}{})
 return true
}

// Marshal function converts messages to bytes using GOB encoder interface
func (c *Codec) Marshal(v interface{}) ([]byte, error) {
 var b bytes.Buffer
 enc := gob.NewEncoder(&b)
 err := enc.Encode(v)
 if err != nil {
  return nil, err
 }
 return b.Bytes(), nil
}

// Unmarshal function converts bytes to message using GOB decoder interface
func (c *Codec) Unmarshal(b []byte, v interface{}) error {
 r := bytes.NewReader(b)
 dec := gob.NewDecoder(r)
 return dec.Decode(v)
}
```

[52] The entire content of the **grpcFabricVdrAPI_grpc.pb.go** appears in the **Appendix C**, in the GRPC Protobuf table.

```
// Converts to string. Unimplemented since we did not handle string, but required from GOB
func (c *Codec) String() string {
 return "Custom Marshal-Unmarshal Codec"
}
```

Table 22: VDR gRPC Server GOB Custom Codec

As already stated above, the Protobuf messages is going to be omitted, thus in order to be able to transmit and receive messages between a client and a server in a bidirectional channel, a universal custom struct was defined that consists of multiple nested structs where each sub-struct is responsible for accommodating a single gRPC request. This universal struct will eventually be used by the bidirectional service (named Services) to determine the type of the service that the gRPC server is going to forward the incoming request to.

The universal struct is named **StreamingRawBytes** and it is presented below:

FabricVdr Data Structures[53]

fabricVdrStruct.go

```
type StreamingRawBytes struct {
    Subscribe            *Subscribe              `json:"subscribe,omitempty"`
    SubscribeEvent       *SubscribeEvent         `json:"subscribeEvent,omitempty"`
    CreateDoc            *CreateDoc              `json:"createDoc,omitempty"`
    CreateDocEvent       *CreateDocEvent         `json:"createDocEvent,omitempty"`
    ResolveDID           *ResolveDID             `json:"resolveDID,omitempty"`
    ResolveDIDEvent      *ResolveDIDEvent        `json:"resolveDIDEvent,omitempty"`
    Error                *status.Status          `json:"error,omitempty"`
}
```

Table 23: VDR gRPC Server Data Structures

4.4.2. Implementation

As described above, in order to be able to facilitate all the communications between a VDR gRPC client and the VDR gRPC server, a long-lived gRPC bidirectional streaming Server was utilized, integrated with a custom **GOB** Encoder/Decoder, instead of the traditional **Protobuf**. This gRPC server is responsible to handle multiple bidirectional channels (one channel per VDR gRPC Client), as well as a significant number of different requests, depending on the agent's needs.

Depending on the type of the sub-struct, nested within the **StreamingRawBytes** universal struct, the gRPC server can determine the type of the service that the agent requests and forward the request accordingly. The gRPC server as well as the VDR gRPC client, both use Keepalive Settings to keep the streams (in the bidirectional channel) alive, (called long-lived streams), in order to avoid any delays that can occur during multiple TLS handshakes.

As a result, the agent performs TLS handshake only once at the beginning of the connection, and after that, an encrypted bidirectional channel is established. This encrypted bidirectional channel can be used at any time to request or receive events from the Blockchain.

Below, the mapping between the available services that the gRPC server can handle and the messages that are used is presented:

[53] The entire content of the **fabricVdrStruct.go** appear in the **Appendix C**, in the FabricVdr Data Structures table

\	Description	I/O

Subscribe	This service is used at the beginning of a connection. The agent provides a **clientId** to the server and the server uses this **clientId** to subscribe to the new incoming **stream connection details**. This will later be used by the server to disseminate events back to the client.	**Input**
		`//Formulate a Subscribe Request`
		`var req *fabricVDR.StreamingRawBytes`
		`req = &fabricVDR.StreamingRawBytes{`
		`Subscribe: &fabricVDR.Subscribe{`
		`ClientId: string,`
		`},`
		`}`
		Output
		`// Formulate SubscribeEvent Response`
		`var msg *fabricVDR.StreamingRawBytes`
		`msg = &fabricVDR.StreamingRawBytes{`
		`SubscribeEvent: &fabricVDR.SubscribeEvent{`
		`Event: string,`
		`},`
		`}`

CreateDoc	This service is used to create a new **DID** by registering a new **DID Document** into the ledger. This service listens for chaincode events from the **CreateDoc** smart contract function and disseminates the DID Document back to the intended device.	**Input**[54]
		`// Formulate a CreateDoc Request`
		`var msg *fabricVDR.StreamingRawBytes`
		`msg = &fabricVDR.StreamingRawBytes{`
		`CreateDoc: &fabricVDR.CreateDoc {`
		`Doc: &did.Doc{},`
		`},`
		`}`
		Output[55]
		`// Formulate a CreateDocEvent Response`
		`var msg *fabricVDR.StreamingRawBytes`
		`msg = &fabricVDR.StreamingRawBytes{`
		`CreateDocEvent: &fabricVDR.CreateDocEvent{`
		`DocResolution: &did.DocResolution{},`
		`},`
		`}`

ResolveDID	This service is used to fetch a **DID Document** if it exists from the world state given a DID, by calling the **ResolveDid** smart contract function.	**Input**
		`// Formulate a ResolveDID Request`
		`var msg *fabricVDR.StreamingRawBytes`
		`msg = &fabricVDR.StreamingRawBytes{`
		`ResolveDID: &fabricVDR.ResolveDID{`
		`DocId: string,`
		`},`
		`}`
		Output
		`// Formulate a ResolveDIDEvent Response`
		`var msg *fabricVDR.StreamingRawBytes`
		`msg = &fabricVDR.StreamingRawBytes{`
		`ResolveDIDEvent: &fabricVDR.ResolveDIDEvent {`
		`DocResolution: &did.DocResolution{},`
		`},`
		`}`

Table 24: VDR gRPC Server I/O

As the design of the gRPC Server is completed and the rules that required to be applied are established, the aforementioned table is extended to withhold information regarding the implementation flow of the gRPC server for each service:

[54] **DID Document** (*did.Doc{}) struct is appeared here

[55] **DID Document Resolution** (*did.DocResolution{}) struct is appeared here

\	GRPC Server Implementation Flow[56]
	server.go

Subscribe	`func (s *server) subscribe(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer, clientId string) error {`
	1. Checks if the `clientId` is empty or already subscribed.
	2. If the `clientId` is valid and not already subscribed, it saves the `stream` and `clientId` in the server's subscription maps.
	3. It sends a subscribe event to the client indicating successful subscription.
	4. If there are any missed events for the client (stored in the `unsuccessfulEvents` map), it sends those events to the client.
	5. It starts a goroutine to monitor the stream and handle potential disconnections or closure of the client's subscription.
	6. If an error occurs during the stream, it adds the remaining events to the `unsuccessfulEvents` map and unsubscribes the client.
	`}`

CreateDoc	`func (s *server) createDoc(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer, didDoc *did.Doc) error {`
	1. It receives a gRPC stream request containing a `did.Doc` object representing the DID document to be created.
	2. It extracts the client ID from the stream, which identifies the caller of the function.
	3. It formulates a `did.DocResolution` struct, which includes the context, the DID document itself, and document metadata.
	4. It checks if the `docId` (the ID of the DID document) is empty. If it is empty, it logs a warning and sends an error response back to the client.
	5. It locks the `docResolutionEvents` map to ensure safe concurrent access and stores the `docResolution` in the map using the `docId` as the key.
	6. It converts the `docResolution` to an array of bytes using the `gob` package for encoding.
	7. It spawns a goroutine `listenCreateDocEvents` to listen for events related to the creation of the DID document.
	a. This goroutine listens for events from the `Hyperledger Fabric Blockchain Network` using the `docResolutionEvents` channel.
	b. When an event is received, it checks if the event matches the `docId` of the created document.
	c. If it matches, it sends the event back to the client via the gRPC stream.
	d. This allows the client to receive updates or notifications about the created document asynchronously.
	8. It sends the encoded `docResolution` as a response back to the client via the gRPC stream.
	`}`

ResolveDID	`func (s *server) resolveDID(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer, docId string) error {`
	1. It receives a gRPC request containing a `ResolveDID` request object, which includes the DID to be resolved.
	2. It extracts the client ID from the stream, which identifies the caller of the function.
	3. It retrieves the DID document associated with the provided DID from the `Hyperledger Fabric Blockchain Network`.
	4. If the DID document is found in the `Hyperledger Fabric Blockchain Network`, it formulates a `ResolveDID` response object, which includes the resolved `did.Doc` and any additional metadata.
	5. It converts the `ResolveDID` response to an array of bytes using the `gob` package for encoding.
	6. It sends the encoded `ResolveDID` response as a response back to the client via the gRPC stream.
	7. If the DID document is not found in the `Hyperledger Fabric Blockchain Network`, it logs a warning and sends an error response back to the client.
	`}`

Table 25: VDR gRPC Server Implementation Flow

To start the **VDR gRPC Server, that connects to the Kubernetes deployed Hyperledger Fabric Network**, the following command must be issued:

[56] The entire content of **server.go** appears in **Appendix C** in the GRPC Server table.

```
cd <path to VDR gRPC server parent directory>
go run server/server.go --tls
```

And the result is depicted in the image below:

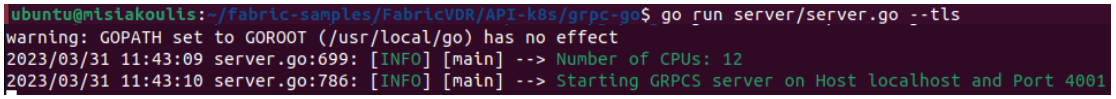

Figure 46: Starting VDR gRPC Server

Now, the **VDR gRPC Server** is ready to receive bidirectional gRPC Streaming requests from VDR gRPC Clients.

4.4.3. Unit Tests

Following the well-known and well-established best practices in the field of software engineering, regarding the VDR gRPC server implementation, table driven unit tests were also created and provided that covers the server's logic extensively, by spawning a new **test Hyperledger Fabric network in Kubernetes** (as described in section 4.3) and a **test VDR gRPC Server Instance** (as described in section 4.4.2).

These unit tests are based on a real bidirectional gRPC Client consisting of two main goroutines that handle the listening from as well as the writing to the stream of GOBs at the same time and not in the official testing Go package. This unit test design eventually constitutes the base of the **VDR gRPC Client** that will be imported as a package in the **Hyperledger Aries**.

The following tests were evaluated against the VDR gRPC Server:

\	GRPC Server Unit Tests Scenarios	
	Scenario	Description
Subscribe	ClientID_Empty	This unit test evaluates the gRPC Server's behavior when an empty ClientID is provided as input.
	Success	This unit test evaluates the GRPC Server's behavior when a valid ClientID is provided as input.
	ClientID_Already_Exists	This unit test evaluates the GRPC Server's behavior when a ClientID that is already registered is provided as input.
CreateDoc	Empty_DocId	This unit test evaluates the GRPC Server's behavior when an empty DID is provided as input.
	Success	This unit test evaluates the GRPC Server's behavior when a valid DidDoc is provided as input and a DocResolution is successfully created.
ResolveDID	Empty_DocId	This unit test evaluates the GRPC Server's behavior when an empty DID is provided as input.
	Success	This unit test evaluates the GRPC Server's behavior when a valid DID is provided and the DocResolution is successfully returned.

Table 26: VDR gRPC Server Unit Tests Scenarios Description

To run the Unit Tests, the following command is issued:

```
go run client/unitTest.go --tls --clientID=dev123
```

And the results are depicted in the table below:

\	VDR gRPC Server Unit Test Results[57]
Subscribe CreateDoc ResolveDID	ubuntu@misiakoulis:~/fabric-samples/FabricVDR/API-k8s/grpc-go$ go run client/unitTest.go --tls --clientID=dev123 warning: GOPATH set to GOROOT (/usr/local/go) has no effect 2023/03/31 17:50:02 unitTest.go:357: [INFO] [main] --> Number of CPUs: 12 2023/03/31 17:50:02 unitTest.go:409: [INFO] [main] --> Starting GRPCS client on host localhost:4001 2023/03/31 17:50:02 unitTest.go:420: [INFO] [main] --> Client dev123 is connected === RUN TestSubscribe === RUN TestSubscribe/ClientID_Empty === RUN TestSubscribe/Success === RUN TestSubscribe/ClientID_Already_Exists --- PASS: TestSubscribe --- PASS: TestSubscribe/ClientID_Empty --- PASS: TestSubscribe/Success --- PASS: TestSubscribe/ClientID_Already_Exists === RUN TestCreateDoc === RUN TestCreateDoc/Empty_DocId === RUN TestCreateDoc/Success --- PASS: TestCreateDoc --- PASS: TestCreateDoc/Empty_DocId --- PASS: TestCreateDoc/Success === RUN TestResolveDID === RUN TestResolveDID/Empty_DocId === RUN TestResolveDID/Success --- PASS: TestResolveDID --- PASS: TestResolveDID/Empty_DocId --- PASS: TestResolveDID/Success PASS

Table 27: VDR gRPC Server Unit Tests

As depicted in the results above, all the VDR gRPC Server Unit Tests have passed successfully.

4.5. Hyperledger Aries Framework Go

4.5.1. Design

According to the aforementioned sections, all the necessary steps from the design, the implementation of the Fabric VDR Smart Contract as well as the Kubernetes deployed Hyperledger Fabric network are being made, thus the time has come to create, package and deploy the Fabric VDR Client package and eventually integrate it into the Hyperledger Aries Framework GO.

As mentioned in the previous section (4.4.3), the VDR gRPC Client relies on the use of two impeccable and perfect synchronized goroutines that handle the reading from and writing to the stream of GOBs simultaneously. The VDR gRPC Client must also implement all the functions that are described here in the official Hyperledger Aries Framework GO.

More specifically, since we are following the smart contract implementation of a Hyperledger Aries Framework GO VDR, the following functions were also implemented for the VDR gRPC Client:

Fabric VDR Function Signatures				
\	Function	Signature	Description	Done
1	Create **(Register)**	`Create(did *did.Doc, opts ...DIDMethodOption) (*did.DocResolution, error)`	**A create operation for a DID Document** **Follows the same principles as CreateDoc**	✔

[57] The entire content of **unitTest.go** appears in **Appendix C** in the GRPC Server Unit Tests table.

2	Read **(Resolve)**	`Read(did string, opts ...DIDMethodOption) (*did.DocResolution, error)`	**A read operation for DID Documents** **Follows the same principles as ResolveDID**	✔
3	Update **(Replace)**	`Update(did *did.Doc, opts ...DIDMethodOption) error`	**An update operation for a DID Document.** **Follows the same principles as UpdateDoc**	✘
4	Delete **(Deactivate)**	`Deactivate(did string, opts ...DIDMethodOption) error`	**A delete operation for a DID Document** **Follows the same principles as DeleteDid**	✘
5	Accept	`Accept(method string, opts ...DIDMethodOption) bool`	**Accepts the did:fabricvdr method**	✔
6	Close	`Close() error`	**Frees resources being maintained by the VDR**	✔
7	New	`New(vdr *VDR)`	**Initialize the VDR gRPC Client source code to interact with the VDR gRPC Server**	✔

Table 28: Aries Framework Go FabricVdr Function Signatures

4.5.2. Implementation

Since the design of the VDR gRPC Client package is concluded, it is time to start constructing the VDR gRPC Client Project Filesystem and implementing the aforementioned functions. The package filesystem must follow the structure as denoted in the Appendix D - Project Filesystem Structure.

In order to start creating the filesystem the following commands were issued:

```
cd ~
git clone -b v0.1.8 https://github.com/hyperledger/aries-framework-go.git
cd aries-framework-go/cmd/aries-agent-rest
mkdir fabricvdr
cd fabricvdr
go mod init fabricvdr
mkdir -p packages/{codec,cryptoutil,request}
touch packages/codec/codec.go #👉 Copy the code from here, under Custom Codec table
touch packages/cryptoutil/{legacy_utils,utils}.go #👉 Copy the code from here and here
touch packages/request/fabricVdrStruct.go  #👉 Copy the code from here
mkdir -p grpcFabricVdrAPI
touch grpcFabricVdrAPI/grpcFabricVdrAPI_grpc.pb.go #👉 Copy the code from here
touch {vdr,creator,resolve}.go #👉 Copy the code from here, here and here
```

The three (3) main go files are the following:

1. **vdr.go** - This file implements the following important functions:
    a. **New()** - This function contains the source code of the VDR gRPC Client, capable of creating a bidirectional gRPC connection with the VDR gRPC Server. Also this function exposes a number of important environmental variables that can be used to interact with the VDR gRPC Server. The environmental variables are listed below:
        i. **GRPC_SERVER_ADDRESS** - (string) - The address of the gRPC Server
        ii. **GRPC_SERVER_PORT** - (string) - The port of the gRPC Server

iii. **GRPC_SERVER_HOST_OVERRIDE** - (string) - The name that is used during TLS handshake and sent along with the client's certificate and must match the server's certificate Common Name
iv. **GRPC_TLS_ENABLED** - (boolen) - Enable TLS
v. **GRPC_TLS_CERT_FILE** - (string) - If TLS enabled, it is used to fetch the client's Certificate from path
vi. **GRPC_TLS_KEY_FILE** - (string) - If TLS enabled, it is used to fetch the client's Private Key from path
vii. **GRPC_TLS_CA_FILE** - (string) - If TLS enabled, it is used to fetch the client's CA Certificate from path
viii. **GRPC_CLIENT_ID** - (string) - Must be a unique identifier and it is used during the **Subscribe** request to the VDR gRPC Server. This later will be used by the VDR gRPC Server to disseminate events back to the VDR gRPC Client

b. **Accept() -** This function is used by the Hyperledger Aries Framework Go to accept the **fabricvdr** method.
c. **Close()** - This function is used to free resources that are being maintained by the VDR
d. **Update()** - This function when used the "NOT SUPPORTED" message is returned
e. **Deactivate()** - This function when used the "NOT SUPPORTED" message is returned

2. **creator.go -** This file implements the Create function and is responsible for sending the CreateDoc request through the bidirectional channel to the VDR gRPC Server and eventually waiting until the DocResolution response is returned from the Kubernetes deployed Hyperledger Fabric network.
3. **resolve.go -** This file implements the Read function and is responsible for sending the ResolveDID requests through the bidirectional channel to the VDR gRPC Server and eventually waiting until the DocResolution response is returned from the Kubernetes deployed Hyperledger Fabric network.

4.5.3. Integration

In order to be able to integrate the newly created **fabricvdr** package into the Hyperledger Aries Framework Go, some modifications must be performed prior to executing the Hyperledger Aries agent and are listed below:

1. In order to add the **fabricvdr** package as a new VDR package, the following line of code must be added in the image (Dockerfile) of the Hyperledger Aries Agent which is located in `~/aries-framework-go/images/agent-rest/Dockerfile` at line **20**:

```
ADD cmd/aries-agent-rest/fabricvdr /usr/local/go/src/fabricvdr
```

2. In order for the **fabricvdr** to be accepted as a new VDR by the Hyperledger Aries Agent, the following lines of code must be added in `~/aries-framework-go/pkg/framework/aries/framework.go` at the following lines:
   a. Line **15**: Import the **fabricvdr** package

```
fabricvdr "fabricvdr"
```

b. Line **503**: In **CreateVDR** function enable the **fabricvdr** method as a new VDR

```
var e *fabricvdr.VDR
go fabricvdr.New(e)
opts = append(opts, vdr.WithVDR(e))
```

3. In order to pass the appropriate environmental variables from the Hyperledger Aries Agent to the **fabricvdr** package, the following piece of code must be added in `~/aries-framework-go/test/bdd/fixtures/agent-rest/docker-compose.yml` after removing the lines 10-39:

```
alice.agent.example.com:
  container_name: alice.aries.example.com
  image: ${AGENT_REST_IMAGE}:${AGENT_REST_IMAGE_TAG}
  environment:
    - ARIESD_API_HOST=${ALICE_HOST}:${ALICE_API_PORT}
    - ARIESD_INBOUND_HOST=${HTTP_SCHEME}@${ALICE_HOST}:${ALICE_INBOUND_PORT}
    -
ARIESD_INBOUND_HOST_EXTERNAL=${HTTP_SCHEME}@https://alice.aries.example.com:${ALICE_INBOUND
_PORT}
    - ARIESD_WEBHOOK_URL=http://${ALICE_WEBHOOK_CONTAINER_NAME}:${ALICE_WEBHOOK_PORT}
    - ARIESD_DEFAULT_LABEL=alice-agent
    - ARIESD_DATABASE_TYPE=leveldb
    - ARIESD_DATABASE_PREFIX=alice
    - ARIESD_DATABASE_TIMEOUT=60
    - ARIESD_HTTP_RESOLVER=${HTTP_DID_RESOLVER}
    - ARIESD_CONTEXT_PROVIDER_URL=${CONTEXT_PROVIDER_URL}
    - ARIESD_MEDIA_TYPE_PROFILES=${DEFAULT_MEDIA_TYPE_PROFILES}
    - ARIESD_KEY_TYPE=${DEFAULT_KEY_TYPE}
    - ARIESD_KEY_AGREEMENT_TYPE=${DEFAULT_KEY_AGREEMENT_TYPE}
    - TLS_CERT_FILE=/etc/tls/ec-pubCert.pem
    - TLS_KEY_FILE=/etc/tls/ec-key.pem

    # GRPC-CLIENT CONNECTION DETAILS
    - GRPC_SERVER_ADDRESS=${GRPC_SERVER_ADDRESS}
    - GRPC_SERVER_PORT=${GRPC_SERVER_PORT}
    - GRPC_SERVER_HOST_OVERRIDE=${GRPC_SERVER_HOST_OVERRIDE}
    # GRPC-CLIENT TLS
    - GRPC_TLS_ENABLED=${GRPC_TLS_ENABLED}
    - GRPC_TLS_CERT_FILE=/etc/tls/grpc-client/clientCert.pem
    - GRPC_TLS_KEY_FILE=/etc/tls/grpc-client/clientUnencryptedKey.pem
    - GRPC_TLS_CA_FILE=/etc/tls/grpc-client/CAcert.pem
    # CLIENT ID
    - GRPC_CLIENT_ID=${GRPC_CLIENT_ID}
  volumes:
    - ${GRPC_CLIENT_TLS_PATH}/:/etc/tls/grpc-client
    - ../keys/tls:/etc/tls
  ports:
    - ${ALICE_INBOUND_PORT}:${ALICE_INBOUND_PORT}
    - ${ALICE_API_PORT}:${ALICE_API_PORT}
  entrypoint: ""
  command:  /bin/sh -c "cp /etc/tls/*
/usr/local/share/ca-certificates/;update-ca-certificates; aries-agent-rest start"
  networks:
    - bdd_net
  depends_on:
    - file-server.example.com
  extra_hosts:
    - ${GRPC_SERVER_ADDRESS}:${GRPC_SERVER_IP}
```

4. Finally, to pass the actual values of the environmental variables from the Hyperledger Aries Agent to the **fabricvdr** package, the following lines must be added

in `~/aries-framework-go/test/bdd/fixtures/agent-rest/.env` at line 18:

```
# GRPC Client Environmental variables
GRPC_CLIENT_TLS_PATH=<Path to the Client TLS Certificates>
GRPC_SERVER_ADDRESS=host.docker.internal # Or any other Domain Name, here
host.docker.internal was used since grpc server relies on the host and aries agents as
docker containers
GRPC_SERVER_IP=host-gateway # Or any other IP, here host-gateway was used since grpc server
relies on the host and aries agents as docker containers
GRPC_SERVER_PORT=4001
GRPC_SERVER_HOST_OVERRIDE=host.docker.internal # It is only used during gRPCs, to match the
client's certificate Common Name.
GRPC_TLS_ENABLED=true
# GRPC CLIENT ID (Unique Identifier)
GRPC_CLIENT_ID=<A Unique Identifier>
```

The aforementioned modifications are required in order to be able to import the **new fabric VDR** package into **Hyperledger Aries Framework GO**.

5. Validation

With the successful deployment of Hyperledger Fabric in Kubernetes, the next crucial step is the validation of the solution, showcasing the integration of the new Verifiable Data Registry (VDR). The successful deployment of the VDR gRPC server and the creation of the VDR gRPC client in the Hyperledger Aries Framework GO lays the foundation for this validation. By leveraging the capabilities of the VDR, the solution aims to demonstrate its ability to securely store and retrieve verifiable data, facilitating trust and immutability within the blockchain network.

During the validation process, it is essential to thoroughly test the functionalities and features of the integrated VDR. This includes verifying the seamless interaction between the VDR gRPC server and the Hyperledger Fabric network, ensuring that data is correctly stored, retrieved, and updated.

The Hyperledger Aries Framework Go is a powerful tool that provides a foundation for building interoperable and privacy-enhancing decentralized identity solutions. As part of its comprehensive testing capabilities, the framework includes a specific test case called **"run-openapi-demo"**.

The **"run-openapi-demo"** test case serves as a practical demonstration of the Hyperledger Aries Framework Go's integration with the OpenAPI specification. OpenAPI allows for the standardized description and documentation of RESTful APIs, promoting interoperability and ease of integration across different systems.

By executing the **"run-openapi-demo"** test case, developers can assess the functionality and compatibility of the framework's OpenAPI implementation. This test case likely involves spinning up a local environment and deploying a sample RESTful API based on the OpenAPI specification. Through this demonstration, developers can explore various capabilities provided by the Hyperledger Aries Framework Go, such as authentication, authorization, data exchange, and secure messaging.

In order to be able to test the run-openapi-demo the following command must be executed:

```
cd <path to Hyperledger Aries Framework GO>
make run-openapi-demo
```

And the outcome of the aforementioned execution is depicted below:

```
ubuntu@misiakoulis:~$ cd aries-framework-go/
ubuntu@misiakoulis:~/aries-framework-go$ make run-openapi-demo
Unable to find image 'frapsoft/openssl:latest' locally
latest: Pulling from frapsoft/openssl
3690ec4760f9: Pull complete
0a18daaf2655: Pull complete
Digest: sha256:79be11ae4f91fc3e79408f303ab53f6328c4fcbb16f408edb2778a1de2050cd9
Status: Downloaded newer image for frapsoft/openssl:latest
Generating Aries-Framework-Go Test PKI
Signature ok
subject=/C=CA/ST=ON/O=Example Inc.:Aries-Framework-Go/OU=Aries-Framework-Go/CN=*.example.com
Getting CA Private Key
done generating Aries-Framework-Go PKI
Generating and validating controller API specifications using Open API
Generating Open API spec
Unable to find image 'quay.io/goswagger/swagger:v0.23.0' locally
v0.23.0: Pulling from goswagger/swagger
c9b1b535fdd9: Pull complete
cbb0d8da1b30: Pull complete
d909eff28200: Pull complete
8b9d9d6824f5: Pull complete
a50ef8b76e53: Pull complete
6c7434552e52: Pull complete
64fe71c224c3: Pull complete
a7577df27822: Pull complete
Digest: sha256:dd7f8f2e94ce74d91aba0e6719adce752e1aa68810fd734f1562d925d474bb5c
Status: Downloaded newer image for quay.io/goswagger/swagger:v0.23.0
Validating generated spec
2023/05/19 09:22:56
The swagger spec at "build/rest/openapi/spec/openAPI.yml" is valid against swagger specificati
on 2.0
2023/05/19 09:22:56
The swagger spec at "build/rest/openapi/spec/openAPI.yml" showed up some valid but possibly un
wanted constructs.
2023/05/19 09:22:56 See warnings below:
2023/05/19 09:22:56 - WARNING: response "#/responses/docResResponse" is not used anywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/CreateOrUpdateProfileRequest" is not
used anywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/RequestCredentialV3Body" is not used
anywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/DIDArgs" is not used anywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/IssueCredentialV3Body" is not used an
ywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/ProposeCredentialV3Body" is not used
anywhere
2023/05/19 09:22:56 - WARNING: definition "#/definitions/OfferCredentialV3Body" is not used an
ywhere
Building aries agent rest docker image
Sending build context to Docker daemon  28.17MB
Step 1/17 : ARG GO_VER
Step 2/17 : ARG ALPINE_VER
Step 3/17 : FROM golang:${GO_VER}-alpine${ALPINE_VER} as golang
 ---> 270c4f58750f
Step 4/17 : RUN apk add --no-cache      gcc     musl-dev        git     libtool         bash m
ake;
 ---> Running in a3e8de0fe732
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/25) Upgrading musl (1.2.3-r0 -> 1.2.3-r2)
(2/25) Installing ncurses-terminfo-base (6.3_p20220521-r0)
(3/25) Installing ncurses-libs (6.3_p20220521-r0)
(4/25) Installing readline (8.1.2-r0)
(5/25) Installing bash (5.1.16-r2)
Executing bash-5.1.16-r2.post-install
(6/25) Installing libgcc (11.2.1_git20220219-r2)
(7/25) Installing libstdc++ (11.2.1_git20220219-r2)
```

```
(8/25) Installing binutils (2.38-r3)
(9/25) Installing libgomp (11.2.1_git20220219-r2)
(10/25) Installing libatomic (11.2.1_git20220219-r2)
(11/25) Installing gmp (6.2.1-r2)
(12/25) Installing isl22 (0.22-r0)
(13/25) Installing mpfr4 (4.1.0-r0)
(14/25) Installing mpc1 (1.2.1-r0)
(15/25) Installing gcc (11.2.1_git20220219-r2)
(16/25) Installing brotli-libs (1.0.9-r6)
(17/25) Installing nghttp2-libs (1.47.0-r0)
(18/25) Installing libcurl (8.1.0-r0)
(19/25) Installing expat (2.5.0-r0)
(20/25) Installing pcre2 (10.40-r0)
(21/25) Installing git (2.36.6-r0)
(22/25) Installing libltdl (2.4.7-r0)
(23/25) Installing libtool (2.4.7-r0)
(24/25) Installing make (4.3-r0)
(25/25) Installing musl-dev (1.2.3-r2)
Executing busybox-1.35.0-r15.trigger
OK: 137 MiB in 39 packages
Removing intermediate container a3e8de0fe732
 ---> 195ecb090cab
Step 5/17 : ADD . $GOPATH/src/github.com/hyperledger/aries-framework-go
 ---> 15caf7b73eac
Step 6/17 : WORKDIR $GOPATH/src/github.com/hyperledger/aries-framework-go
 ---> Running in ecf620a61274
Removing intermediate container ecf620a61274
 ---> 6a9f45f76c15
Step 7/17 : ADD cmd/aries-agent-rest/fabricvdr /usr/local/go/src/fabricvdr
 ---> 4533c072d1c9
Step 8/17 : ENV EXECUTABLES go git
 ---> Running in de67d81020fa
Removing intermediate container de67d81020fa
 ---> 0b97296bd2bb
Step 9/17 : FROM golang as aries-framework
 ---> 0b97296bd2bb
Step 10/17 : LABEL org.opencontainers.image.source https://github.com/hyperledger/aries-framew
ork-go
 ---> Running in 711c80117915
Removing intermediate container 711c80117915
 ---> 9d23ef304e43
Step 11/17 : ARG GO_TAGS
 ---> Running in cb170959c539
Removing intermediate container cb170959c539
 ---> a03d6424cad7
Step 12/17 : ARG GOPROXY
 ---> Running in 448b722e3fc6
Removing intermediate container 448b722e3fc6
 ---> 6dff567dcda9
Step 13/17 : RUN GO_TAGS=${GO_TAGS} GOPROXY=${GOPROXY} make agent-rest
 ---> Running in 7fbc93bfad82
Building aries-agent-rest
go: downloading github.com/spf13/cobra v1.0.0
go: downloading github.com/cenkalti/backoff/v4 v4.1.0
go: downloading github.com/gorilla/mux v1.7.3
go: downloading github.com/rs/cors v1.7.0
go: downloading github.com/stretchr/testify v1.7.0
go: downloading github.com/piprate/json-gold v0.4.1-0.20210813112359-33b90c4ca86c
go: downloading nhooyr.io/websocket v1.8.3
go: downloading github.com/syndtr/goleveldb v1.0.0
go: downloading github.com/google/uuid v1.1.2
go: downloading github.com/hyperledger/aries-framework-go/component/storage/edv v0.0.0-2022032
2085443-50e8f9bd208b
go: downloading github.com/mitchellh/mapstructure v1.1.2
go: downloading github.com/btcsuite/btcutil v1.0.3-0.20201208143702-a53e38424cce
go: downloading github.com/multiformats/go-multibase v0.0.1
```

```
go: downloading github.com/xeipuuv/gojsonschema v1.2.0
go: downloading github.com/square/go-jose/v3 v3.0.0-20200630053402-0a67ce9b0693
go: downloading github.com/jinzhu/copier v0.0.0-20190924061706-b57f9002281a
go: downloading github.com/multiformats/go-multihash v0.0.13
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading google.golang.org/grpc v1.48.0
go: downloading github.com/golang/protobuf v1.5.2
go: downloading github.com/google/tink/go v1.6.1-0.20210519071714-58be99b3c4d0
go: downloading golang.org/x/crypto v0.0.0-20210616213533-5ff15b29337e
go: downloading github.com/pkg/errors v0.9.1
go: downloading github.com/btcsuite/btcd v0.22.0-beta
go: downloading github.com/PaesslerAG/gval v1.1.0
go: downloading github.com/PaesslerAG/jsonpath v0.1.1
go: downloading github.com/VictoriaMetrics/fastcache v1.5.7
go: downloading github.com/bluele/gcache v0.0.0-20190518031135-bc40bd653833
go: downloading github.com/klauspost/compress v1.10.0
go: downloading google.golang.org/genproto v0.0.0-20220713161829-9c7dac0a6568
go: downloading github.com/mr-tron/base58 v1.1.3
go: downloading github.com/multiformats/go-base32 v0.0.3
go: downloading github.com/spf13/pflag v1.0.5
go: downloading github.com/minio/blake2b-simd v0.0.0-20160723061019-3f5f724cb5b1
go: downloading github.com/minio/sha256-simd v0.1.1
go: downloading github.com/multiformats/go-varint v0.0.5
go: downloading github.com/spaolacci/murmur3 v1.1.0
go: downloading github.com/xeipuuv/gojsonreference v0.0.0-20180127040603-bd5ef7bd5415
go: downloading github.com/pquerna/cachecontrol v0.0.0-20180517163645-1555304b9b35
go: downloading google.golang.org/protobuf v1.28.0
go: downloading golang.org/x/net v0.0.0-20220708220712-1185a9018129
go: downloading github.com/teserakt-io/golang-ed25519 v0.0.0-20210104091850-3888c087a4c8
go: downloading github.com/kawamuray/jsonpath v0.0.0-20201211160320-7483bafabd7e
go: downloading github.com/tidwall/sjson v1.1.4
go: downloading github.com/tidwall/gjson v1.6.7
go: downloading github.com/kilic/bls12-381 v0.1.1-0.20210503002446-7b7597926c69
go: downloading github.com/cespare/xxhash/v2 v2.1.1
go: downloading github.com/golang/snappy v0.0.3
go: downloading github.com/pmezard/go-difflib v1.0.0
go: downloading gopkg.in/yaml.v3 v3.0.0-20210107192922-496545a6307b
go: downloading golang.org/x/sys v0.0.0-20220712014510-0a85c31ab51e
go: downloading github.com/xeipuuv/gojsonpointer v0.0.0-20190905194746-02993c407bfb
go: downloading github.com/tidwall/match v1.0.3
go: downloading github.com/tidwall/pretty v1.0.2
go: downloading golang.org/x/text v0.3.7
Removing intermediate container 7fbc93bfad82
 ---> f1fe997db375
Step 14/17 : FROM alpine:${ALPINE_VER} as base
 ---> 9c6f07244728
Step 15/17 : RUN apk add -U --no-cache ca-certificates
 ---> Running in 665a7eda8d9f
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/1) Installing ca-certificates (20230506-r0)
Executing busybox-1.35.0-r17.trigger
Executing ca-certificates-20230506-r0.trigger
OK: 6 MiB in 15 packages
Removing intermediate container 665a7eda8d9f
 ---> d5120ffd87af
Step 16/17 : COPY --from=aries-framework /go/src/github.com/hyperledger/aries-framework-go/bui
ld/bin/aries-agent-rest /usr/local/bin
 ---> e8c5178e73e2
Step 17/17 : ENTRYPOINT ["aries-agent-rest"]
 ---> Running in f6bf1d08b63f
Removing intermediate container f6bf1d08b63f
 ---> dc932ff494c9
Successfully built dc932ff494c9
Successfully tagged aries-framework-go/agent-rest:latest
Building sample webhook server docker image
Sending build context to Docker daemon  28.17MB
Step 1/14 : ARG GO_VER
Step 2/14 : ARG ALPINE_VER
Step 3/14 : FROM golang:${GO_VER}-alpine${ALPINE_VER} as golang
 ---> 270c4f58750f
Step 4/14 : RUN apk add --no-cache      gcc     musl-dev        git     libtool         bash m
ake;
```

```
 ---> Running in 2d45b65f8f68
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/25) Upgrading musl (1.2.3-r0 -> 1.2.3-r2)
(2/25) Installing ncurses-terminfo-base (6.3_p20220521-r0)
(3/25) Installing ncurses-libs (6.3_p20220521-r0)
(4/25) Installing readline (8.1.2-r0)
(5/25) Installing bash (5.1.16-r2)
Executing bash-5.1.16-r2.post-install
(6/25) Installing libgcc (11.2.1_git20220219-r2)
(7/25) Installing libstdc++ (11.2.1_git20220219-r2)
(8/25) Installing binutils (2.38-r3)
(9/25) Installing libgomp (11.2.1_git20220219-r2)
(10/25) Installing libatomic (11.2.1_git20220219-r2)
(11/25) Installing gmp (6.2.1-r2)
(12/25) Installing isl22 (0.22-r0)
(13/25) Installing mpfr4 (4.1.0-r0)
(14/25) Installing mpc1 (1.2.1-r0)
(15/25) Installing gcc (11.2.1_git20220219-r2)
(16/25) Installing brotli-libs (1.0.9-r6)
(17/25) Installing nghttp2-libs (1.47.0-r0)
(18/25) Installing libcurl (8.1.0-r0)
(19/25) Installing expat (2.5.0-r0)
(20/25) Installing pcre2 (10.40-r0)
(21/25) Installing git (2.36.6-r0)
(22/25) Installing libltdl (2.4.7-r0)
(23/25) Installing libtool (2.4.7-r0)
(24/25) Installing make (4.3-r0)
(25/25) Installing musl-dev (1.2.3-r2)
Executing busybox-1.35.0-r15.trigger
OK: 137 MiB in 39 packages
Removing intermediate container 2d45b65f8f68
 ---> 9fa1226975d8
Step 5/14 : ADD . $GOPATH/src/github.com/hyperledger/aries-framework-go
 ---> 8992fe7046b0
Step 6/14 : WORKDIR $GOPATH/src/github.com/hyperledger/aries-framework-go
 ---> Running in bb36347594c3
Removing intermediate container bb36347594c3
 ---> c9fc3798a4af
Step 7/14 : ENV EXECUTABLES go git
 ---> Running in 65547a21414f
Removing intermediate container 65547a21414f
 ---> 2ecd006c94f9
Step 8/14 : FROM golang as aries-framework
 ---> 2ecd006c94f9
Step 9/14 : ARG GO_TAGS
 ---> Running in 0d034f8af1fa
Removing intermediate container 0d034f8af1fa
 ---> a239d392972f
Step 10/14 : ARG GOPROXY
 ---> Running in 31807a56a096
Removing intermediate container 31807a56a096
 ---> 4d33de132d8f
Step 11/14 : RUN GO_TAGS=${GO_TAGS} GOPROXY=${GOPROXY} make sample-webhook
 ---> Running in 0d6af5aeef7a
Building sample webhook server
go: downloading github.com/gorilla/mux v1.7.3
go: downloading github.com/stretchr/testify v1.7.0
go: downloading github.com/hyperledger/aries-framework-go/spi v0.0.0-20220322085443-50e8f9bd20
8b
go: downloading github.com/pmezard/go-difflib v1.0.0
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading gopkg.in/yaml.v3 v3.0.0-20210107192922-496545a6307b
Removing intermediate container 0d6af5aeef7a
 ---> 7e5c19a1faa2
Step 12/14 : FROM alpine:${ALPINE_VER} as base
 ---> 9c6f07244728
Step 13/14 : COPY --from=aries-framework /go/src/github.com/hyperledger/aries-framework-go/bui
ld/bin/webhook-server /usr/local/bin
 ---> 8c9d057008d1
Step 14/14 : CMD WEBHOOK_PORT=${WEBHOOK_PORT} webhook-server
 ---> Running in 95326d622280
Removing intermediate container 95326d622280
 ---> 53a2606df427
Successfully built 53a2606df427
```

```
Successfully tagged aries-framework-go/sample-webhook:latest
Generate demo agent rest controller API specifications using Open API
2023/05/19 09:24:24 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:24 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:10081.yml]
2023/05/19 09:24:26 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:26 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:10073.yml]
2023/05/19 09:24:28 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:28 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:8082.yml]
2023/05/19 09:24:30 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:30 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:9082.yml]
2023/05/19 09:24:32 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:32 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:10061.yml]
2023/05/19 09:24:34 args[0] = build/rest/openapi/spec/openAPI.yml
2023/05/19 09:24:34 args[1:] = [test/bdd/fixtures/demo/openapi/specs/localhost:10093.yml]
Starting demo agent rest containers ...
Creating network "agent-rest_bdd_net" with driver "bridge"
Pulling couchdb.example.com (couchdb:3.1.0)...
3.1.0: Pulling from library/couchdb
d121f8d1c412: Pull complete
36f0ba3ab6ff: Pull complete
4ad755530f10: Pull complete
85b93b47d5de: Pull complete
914762042288: Pull complete
a19d0d042d73: Pull complete
e5072937c22e: Pull complete
3a2c37f3e964: Pull complete
cec4e3b21e82: Pull complete
06823e20f7d4: Pull complete
87dc384bcdb0: Pull complete
Digest: sha256:b604d056d8024f10346eab768de7aea06bc0a1b7c55d6087e1b1cd4328c8061c
Status: Downloaded newer image for couchdb:3.1.0
Pulling kms.example.com (ghcr.io/trustbloc-cicd/kms:v0.1.8-snapshot-3f3ef05)...
v0.1.8-snapshot-3f3ef05: Pulling from trustbloc-cicd/kms
939c5235e7e1: Pull complete
1fc87a5794a6: Pull complete
e45fe1f3e94a: Pull complete
Digest: sha256:04ebd5d0ac8972fd0966778a2450cedee7b669bf488d8a42695d77e448f4c208
Status: Downloaded newer image for ghcr.io/trustbloc-cicd/kms:v0.1.8-snapshot-3f3ef05
Pulling file-server.example.com (halverneus/static-file-server:latest)...
latest: Pulling from halverneus/static-file-server
c3073daec123: Pull complete
45169b830aec: Pull complete
Digest: sha256:4b706c068008e60162775c2569225c089e55e65a5afdb08f88c66611028d720b
Status: Downloaded newer image for halverneus/static-file-server:latest
Creating erin.webhook.example.com        ... done
Creating dave.webhook.example.com        ... done
Creating file-server.example.com         ... done
Creating alice.webhook.example.com       ... done
Creating dave.router.webhook.example.com ... done
Creating dave.router.aries.example.com   ... done
Creating carl.webhook.example.com        ... done
Creating carl.router.webhook.example.com ... done
Creating bob.webhook.example.com         ... done
Creating couchdb.example.com             ... done
Creating carl.router.aries.example.com   ... done
Creating dave.aries.example.com          ... done
Creating carl.aries.example.com          ... done
Creating alice.aries.example.com         ... done
Creating bob.aries.example.com           ... done
Creating erin.aries.example.com          ... done
Creating kms.example.com                 ... done
Pulling aries.bdd.sidetree.mock (ghcr.io/trustbloc/sidetree-mock:0.6.0)...
0.6.0: Pulling from trustbloc/sidetree-mock
8464c5956bbe: Pull complete
e51ecefc8a6e: Pull complete
Digest: sha256:1f031219bb006ea6123f55679faaa695a2d3b75a1853dd3f0b7981cf1f40403d
Status: Downloaded newer image for ghcr.io/trustbloc/sidetree-mock:0.6.0
Creating aries.bdd.sidetree.mock ... done
Pulling alice.openapi.demo.com (swaggerapi/swagger-ui:)...
latest: Pulling from swaggerapi/swagger-ui
f56be85fc22e: Pull complete
2ce963c369bc: Pull complete
59b9d2200e63: Pull complete
3e1e579c95fe: Pull complete
547a97583f72: Pull complete
```

```
1f21f983520d: Pull complete
c23b4f8cf279: Pull complete
fb93c9fcfd35: Pull complete
c884fc508d10: Pull complete
d88f1067c596: Pull complete
50378355bb50: Pull complete
b81a431bb409: Pull complete
5bdf63fb12f6: Pull complete
Digest: sha256:a2184d66ad4427b26dd406d4ae8aadd5236894b6cacdb087ff1f08949aca87b1
Status: Downloaded newer image for swaggerapi/swagger-ui:latest
Creating carl.router.openapi.demo.com ... done
Creating dave.openapi.demo.com        ... done
Creating carl.openapi.demo.com        ... done
Creating dave.router.openapi.demo.com ... done
Creating bob.openapi.demo.com         ... done
Creating alice.openapi.demo.com       ... done
```

Figure 47: Outcome of 'make run-openapi-demo' command

After the aforementioned command is concluded, and the containers of the demo are successfully deployed, the VDR gRPC Client will try to establish a mutual TLS bidirectional streaming gRPC connection with the VDR gRPC Server, by sending a Subscribe Request to the channel and eventually receiving the SubscribeEvent Response back.

Since we provided environmental variables only to the **alice.agent.example.com** container, only this particular container will be able to establish a secure bidirectional gRPC connection with the VDR gRPC Server, leaving other containers complaining about the absence of these important environmental variables.

In order to validate that, the following command must be executed:

```
docker logs -f alice.agent.example.com
```

The outcome of the aforementioned command is depicted below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ docker logs -f alice.aries.e
xample.com
cp: omitting directory '/etc/tls/grpc-client'
WARNING: ca-cert-ec-key.csr.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-ec-cacert.srl.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-secret-lock.key.pem does not contain exactly one certificate or CRL: skipping
2023/05/19 09:25:20 vdr.go:316: [INFO] [new] --> Number of CPUs: 12
2023/05/19 09:25:20 vdr.go:367: [INFO] [new] --> Starting GRPCS client on host host.docker.int
ernal:4001
2023/05/19 09:25:20 vdr.go:378: [INFO] [new] --> Client 213 is connected
2023/05/19 09:25:20 vdr.go:150: [INFO] [subscriptionEvent] --> Client Subscribed
 [aries-framework/agent-rest] 2023/05/19 09:25:20 UTC - startcmd.startAgent -> INFO Starting a
ries agent rest on host [0.0.0.0:8082]
```

Figure 48: Alice Agent Subscribe Request

And the VDR gRPC Server response is depicted in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ go run server/server.go --tl
s --host 0.0.0.0
warning: GOPATH set to GOROOT (/usr/local/go) has no effect
2023/05/19 10:54:41 server.go:705: [INFO] [main] --> Number of CPUs: 12
2023/05/19 10:54:41 server.go:792: [INFO] [main] --> Starting GRPCS server on Host 0.0.0.0 and
 Port 4001
(*request.StreamingRawBytes)(0xc00019e980)({
 Subscribe: (*request.Subscribe)(0xc000584a50)({
  ClientId: (string) (len=3) "213"
 }),
 SubscribeEvent: (*request.SubscribeEvent)(<nil>),
 CreateDoc: (*request.CreateDoc)(<nil>),
 CreateDocEvent: (*request.CreateDocEvent)(<nil>),
 ResolveDID: (*request.ResolveDID)(<nil>),
 ResolveDIDEvent: (*request.ResolveDIDEvent)(<nil>),
```

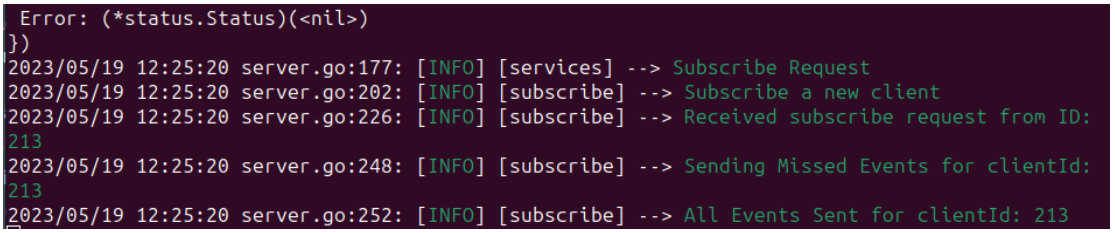

Figure 49: VDR gRPC Server Response to Alice Agent Subscribe Request

This response that depicted in the aforementioned images, denotes that the **alice.agent.example.com** container is successfully subscribed with **ClientId** equals to **213**. Unfortunately, other containers for example **bob.agent.example.com** will produce error messages when trying to connect to the VDR gRPC Server due to the fact that bob has not registered his environmental variables when the container was initiating, as depicted in the image below:

Figure 50: Bob Agent Error Message

Since Alice agent has already subscribed to the VDR gRPC Server and Bob not, we can now proceed with the validation of the remaining **CreateDoc** and **ResolveDID** capabilities of the new fabric VDR, through the use of Alice's Agent.

In the Aries Framework Go, the creation of **DIDs** (Decentralized Identifiers) is typically implemented using the Create method. This method handles the generation and storage of new DIDs based on specific requirements.

To provide an interface for this functionality, Aries Framework Go exposes an **HTTP** endpoint at **https://{{aries-agent-url}}/vdr/did/create**. When a request is made to this endpoint, the associated HTTP handler invokes the **Create** method within the codebase and eventually translated into a **CreateDoc** request.

Within the **Create** method, the logic for DID creation is implemented. This includes generating cryptographic key pairs, constructing the DID document, and potentially registering the DID with a designated DID method or blockchain network.

The following **CURL** command must be issued, in order to be able to register the new DID **did:fabricvdr:1234567890** to our new fabric VDR package:

```
curl -k -X POST -H "Accept: application/json" -H "Content-Type: application/json" -d '{
  "method": "fabricvdr",
  "did": {
   "id": "did:fabricvdr:1234567890",
   "@context": ["https://w3id.org/did/v1"],
   "verificationMethod": [
     {
      "controller": "did:fabricvdr:1234567890",
      "id": "id-123-456-789",
      "publicKeyBase58": "7qf5xCRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP",
      "type": "Ed25519VerificationKey2018"
     }
   ]
  },
  "opts": {
   "store": true
  }
}' https://0.0.0.0:8082/vdr/did/create
```

And the outcome of the aforementioned command is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ curl -vvv -k -X POST -H "acc
ept: application/json" -H "Content-Type: application/json" -d '{"method": "fabricvdr", "did":{
"id": "did:fabricvdr:1234567890", "@context": ["https://w3id.org/did/v1"], "verificationMethod
": [{"controller": "did:fabricvdr:1234567890", "id": "id-123-456-789", "publicKeyBase58": "7qf
5xCRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP", "type": "Ed25519VerificationKey2018"}]},"opts": {"
store": true}}' https://0.0.0.0:8082/vdr/did/create
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 0.0.0.0:8082...
* Connected to 0.0.0.0 (127.0.0.1) port 8082 (#0)
* ALPN: offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN: server accepted h2
* Server certificate:
*  subject: C=CA; ST=ON; O=Example Inc.:Aries-Framework-Go; OU=Aries-Framework-Go; CN=*.exampl
e.com
*  start date: May 19 09:21:44 2023 GMT
*  expire date: May 18 09:21:44 2024 GMT
*  issuer: C=CA; ST=ON; O=Example Internet CA Inc.:CA Sec; OU=CA Sec
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing anyw
ay.
* using HTTP/2
* h2h3 [:method: POST]
* h2h3 [:path: /vdr/did/create]
* h2h3 [:scheme: https]
* h2h3 [:authority: 0.0.0.0:8082]
* h2h3 [user-agent: curl/7.88.1]
* h2h3 [accept: application/json]
* h2h3 [content-type: application/json]
* h2h3 [content-length: 327]
* Using Stream ID: 1 (easy handle 0x556b188e9680)
> POST /vdr/did/create HTTP/2
> Host: 0.0.0.0:8082
> user-agent: curl/7.88.1
> accept: application/json
> content-type: application/json
> content-length: 327
```

```
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN: server accepted h2
* Server certificate:
*  subject: C=CA; ST=ON; O=Example Inc.:Aries-Framework-Go; OU=Aries-Framework-Go; CN=*.exampl
e.com
*  start date: May 19 09:21:44 2023 GMT
*  expire date: May 18 09:21:44 2024 GMT
*  issuer: C=CA; ST=ON; O=Example Internet CA Inc.:CA Sec; OU=CA Sec
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing anyw
ay.
* using HTTP/2
* h2h3 [:method: POST]
* h2h3 [:path: /vdr/did/create]
* h2h3 [:scheme: https]
* h2h3 [:authority: 0.0.0.0:8082]
* h2h3 [user-agent: curl/7.88.1]
* h2h3 [accept: application/json]
* h2h3 [content-type: application/json]
* h2h3 [content-length: 327]
* Using Stream ID: 1 (easy handle 0x556b188e9680)
> POST /vdr/did/create HTTP/2
> Host: 0.0.0.0:8082
> user-agent: curl/7.88.1
> accept: application/json
> content-type: application/json
> content-length: 327
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* We are completely uploaded and fine
< HTTP/2 200
< content-type: application/json
< vary: Origin
< content-length: 269
< date: Fri, 19 May 2023 12:51:19 GMT
<
{"did":{"@context":["https://w3id.org/did/v1"],"id":"did:fabricvdr:1234567890","verificationMe
thod":[{"controller":"did:fabricvdr:1234567890","id":"id-123-456-789","publicKeyBase58":"7qf5x
CRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP","type":"Ed25519VerificationKey2018"}]}}
* Connection #0 to host 0.0.0.0 left intact
```

Figure 51: Alice Agent CreateDoc Request

The VDR gRPC Server response is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ go run server/server.go --tl
s --host 0.0.0.0
warning: GOPATH set to GOROOT (/usr/local/go) has no effect
2023/05/19 10:54:41 server.go:705: [INFO] [main] --> Number of CPUs: 12
2023/05/19 10:54:41 server.go:792: [INFO] [main] --> Starting GRPCS server on Host 0.0.0.0 and
 Port 4001
(*request.StreamingRawBytes)(0xc00019e980)({
 Subscribe: (*request.Subscribe)(0xc000584a50)({
  ClientId: (string) (len=3) "213"
 }),
 SubscribeEvent: (*request.SubscribeEvent)(<nil>),
 CreateDoc: (*request.CreateDoc)(<nil>),
 CreateDocEvent: (*request.CreateDocEvent)(<nil>),
 ResolveDID: (*request.ResolveDID)(<nil>),
 ResolveDIDEvent: (*request.ResolveDIDEvent)(<nil>),
 Error: (*status.Status)(<nil>)
})
2023/05/19 12:25:20 server.go:177: [INFO] [services] --> Subscribe Request
2023/05/19 12:25:20 server.go:202: [INFO] [subscribe] --> Subscribe a new client
2023/05/19 12:25:20 server.go:226: [INFO] [subscribe] --> Received subscribe request from ID:
213
```

```
2023/05/19 12:25:20 server.go:248: [INFO] [subscribe] --> Sending Missed Events for clientId:
213
2023/05/19 12:25:20 server.go:252: [INFO] [subscribe] --> All Events Sent for clientId: 213
(*request.StreamingRawBytes)(0xc000033980)({
 Subscribe: (*request.Subscribe)(<nil>),
 SubscribeEvent: (*request.SubscribeEvent)(<nil>),
 CreateDoc: (*request.CreateDoc)(0xc00048e458)({
  Doc: (*did.Doc)(0xc0003b6240)({
   Context: ([]string) (len=1 cap=1) {
    (string) (len=23) "https://w3id.org/did/v1"
   },
   ID: (string) (len=24) "did:fabricvdr:1234567890",
   VerificationMethod: ([]did.VerificationMethod) (len=1 cap=1) {
    (did.VerificationMethod) {
     ID: (string) (len=14) "id-123-456-789",
     Type: (string) (len=26) "Ed25519VerificationKey2018",
     Controller: (string) (len=24) "did:fabricvdr:1234567890",
     Value: ([]uint8) (len=32 cap=32) {
      00000000  65 9d 10 4c 37 83 db be  60 8e f8 eb 12 fd 3c 6a  |e..L7...`.....<j|
      00000010  2a ed d8 bd ca 57 a5 7d  e2 90 0b 6b b7 df 5a 7c  |*....W.}...k..Z||
     },
     jsonWebKey: (*jwk.JWK)(<nil>),
     relativeURL: (bool) false,
     multibaseEncoding: (multibase.Encoding) 0
    }
   },
   Service: ([]did.Service) <nil>,
   Authentication: ([]did.Verification) <nil>,
   AssertionMethod: ([]did.Verification) <nil>,
   CapabilityDelegation: ([]did.Verification) <nil>,
   CapabilityInvocation: ([]did.Verification) <nil>,
   KeyAgreement: ([]did.Verification) <nil>,
   Created: (*time.Time)(<nil>),
   Updated: (*time.Time)(<nil>),
   Proof: ([]did.Proof) <nil>,
   processingMeta: (did.processingMeta) {
    baseURI: (string) ""
   }
  })
 }),
 CreateDocEvent: (*request.CreateDocEvent)(<nil>),
 ResolveDID: (*request.ResolveDID)(<nil>),
 ResolveDIDEvent: (*request.ResolveDIDEvent)(<nil>),
 Error: (*status.Status)(<nil>)
})
2023/05/19 15:51:17 server.go:182: [INFO] [services] --> Create DID DOC Request
2023/05/19 15:51:17 server.go:386: [INFO] [createDoc] --> Create a new DID DOC
2023/05/19 15:51:17 server.go:390: [INFO] [createDoc] --> Received Create DidDoc request from
ID: 213
2023/05/19 15:51:19 server.go:630: [INFO] [createDocEvents] --> Received Chaincode Event Paylo
ad: {{DeviceID=213}, {DID=did:fabricvdr:1234567890}}
2023/05/19 15:51:19 server.go:645: [INFO] [createDocEvents] --> Disseminate Events ...
```

Figure 52: VDR gRPC Server CreateDoc Response

And the **CreateDoc** event disseminated back to the Alice agent, as presented in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ docker logs -f alice.aries.e
xample.com
cp: omitting directory '/etc/tls/grpc-client'
WARNING: ca-cert-ec-key.csr.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-ec-cacert.srl.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-secret-lock.key.pem does not contain exactly one certificate or CRL: skipping
2023/05/19 09:25:20 vdr.go:316: [INFO] [new] --> Number of CPUs: 12
2023/05/19 09:25:20 vdr.go:367: [INFO] [new] --> Starting GRPCS client on host host.docker.int
ernal:4001
2023/05/19 09:25:20 vdr.go:378: [INFO] [new] --> Client 213 is connected
2023/05/19 09:25:20 vdr.go:150: [INFO] [subscriptionEvent] --> Client Subscribed
 [aries-framework/agent-rest] 2023/05/19 09:25:20 UTC - startcmd.startAgent -> INFO Starting a
ries agent rest on host [0.0.0.0:8082]
```

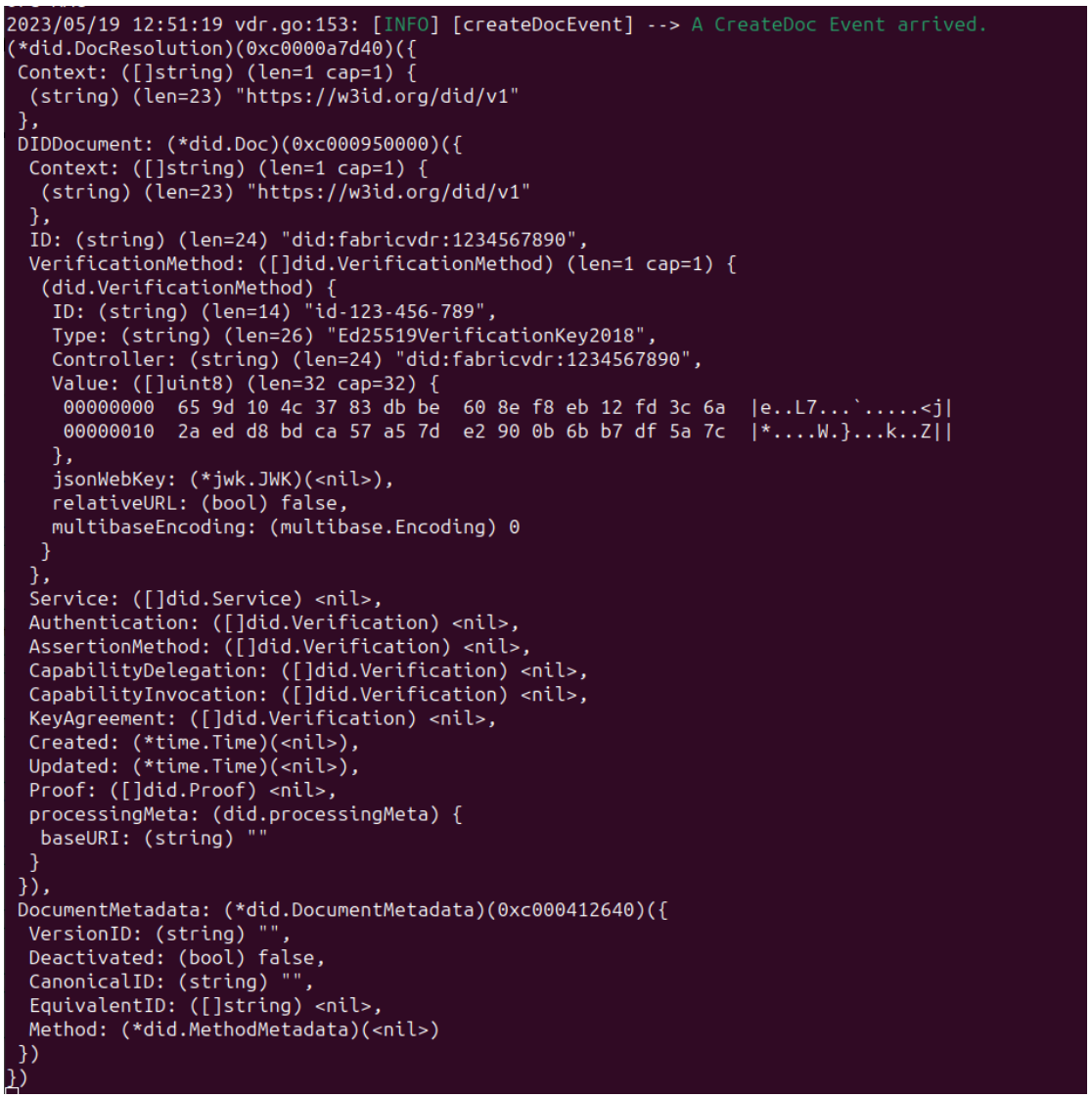

Figure 53: Alice Agent CreateDocEvent Response

In the Aries Framework Go, the resolution of DIDs (Decentralized Identifiers) is typically implemented using the **Read** method. This method is responsible for retrieving the DID document associated with a given DID URL.

To facilitate the resolution of DIDs, Aries Framework Go exposes an HTTP endpoint at **https://{{aries-agent-url}}/vdr/did/resolve/<base64 encode of DID URL>**. When a request is made to this endpoint, the associated HTTP handler invokes the **Read** method by forwarding the base64-encoded DID URL to the gRPC server through the established connection and eventually translated into a **ResolveDID** request.

The VDR gRPC server receives the DID URL from the VDR gRPC client and processes the resolution request. If the DID document is found in the Hyperledger Fabric network, the VDR gRPC server returns a response containing the retrieved DID Document (**DocResolution**) to the VDR gRPC client in a JSON format.

The following **CURL** command must be issued, in order to be able to resolve an existing DID **did:fabricvdr:1234567890** to our new fabric VDR package:

```
curl -k -X GET https://0.0.0.0:8082/vdr/did/resolve/$(echo -n "did:fabricvdr:1234567890" |
base64)
```

And the outcome of the aforementioned command is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ curl -vvv -k -X GET https://
0.0.0.0:8082/vdr/did/resolve/$(echo -n "did:fabricvdr:1234567890" | base64)
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 0.0.0.0:8082...
* Connected to 0.0.0.0 (127.0.0.1) port 8082 (#0)
* ALPN: offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN: server accepted h2
* Server certificate:
*  subject: C=CA; ST=ON; O=Example Inc.:Aries-Framework-Go; OU=Aries-Framework-Go; CN=*.exampl
e.com
*  start date: May 19 09:21:44 2023 GMT
*  expire date: May 18 09:21:44 2024 GMT
*  issuer: C=CA; ST=ON; O=Example Internet CA Inc.:CA Sec; OU=CA Sec
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing anyw
ay.
* using HTTP/2
* h2h3 [:method: GET]
* h2h3 [:path: /vdr/did/resolve/ZGlkOmZhYnJpY3ZkcjoxMjM0NTY3ODkw]
* h2h3 [:scheme: https]
* h2h3 [:authority: 0.0.0.0:8082]
* h2h3 [user-agent: curl/7.88.1]
* h2h3 [accept: */*]
* Using Stream ID: 1 (easy handle 0x55c56293b680)
> GET /vdr/did/resolve/ZGlkOmZhYnJpY3ZkcjoxMjM0NTY3ODkw HTTP/2
> Host: 0.0.0.0:8082
> user-agent: curl/7.88.1
> accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
< HTTP/2 200
< content-type: application/json
< vary: Origin
< content-length: 340
< date: Fri, 19 May 2023 21:02:57 GMT
<
* Connection #0 to host 0.0.0.0 left intact
{"@context":["https://w3id.org/did/v1"],"didDocument":{"@context":["https://w3id.org/did/v1"],
"id":"did:fabricvdr:1234567890","verificationMethod":[{"controller":"did:fabricvdr:1234567890"
,"id":"id-123-456-789","publicKeyBase58":"7qf5xCRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP","type"
:"Ed25519VerificationKey2018"}]},"didDocumentMetadata":{}}ubuntu@misiakoulis:~/fabric-samples/
Eratosthenes/API-k8s/grpc-go$
```

Figure 54: Alice Agent ResolveDID Request

The VDR gRPC Server response is presented below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ go run server/server.go --tl
s --host 0.0.0.0
warning: GOPATH set to GOROOT (/usr/local/go) has no effect
2023/05/19 10:54:41 server.go:705: [INFO] [main] --> Number of CPUs: 12
2023/05/19 10:54:41 server.go:792: [INFO] [main] --> Starting GRPCS server on Host 0.0.0.0 and
 Port 4001
(*request.StreamingRawBytes)(0xc00019e980)({
 Subscribe: (*request.Subscribe)(0xc000584a50)({
  ClientId: (string) (len=3) "213"
 }),
 SubscribeEvent: (*request.SubscribeEvent)(<nil>),
 CreateDoc: (*request.CreateDoc)(<nil>),
```

```
 CreateDocEvent: (*request.CreateDocEvent)(<nil>),
 ResolveDID: (*request.ResolveDID)(<nil>),
 ResolveDIDEvent: (*request.ResolveDIDEvent)(<nil>),
 Error: (*status.Status)(<nil>)
})
2023/05/19 12:25:20 server.go:177: [INFO] [services] --> Subscribe Request
2023/05/19 12:25:20 server.go:202: [INFO] [subscribe] --> Subscribe a new client
2023/05/19 12:25:20 server.go:226: [INFO] [subscribe] --> Received subscribe request from ID:
213
2023/05/19 12:25:20 server.go:248: [INFO] [subscribe] --> Sending Missed Events for clientId:
213
2023/05/19 12:25:20 server.go:252: [INFO] [subscribe] --> All Events Sent for clientId: 213
(*request.StreamingRawBytes)(0xc000400bc0)({
 Subscribe: (*request.Subscribe)(<nil>),
 SubscribeEvent: (*request.SubscribeEvent)(<nil>),
 CreateDoc: (*request.CreateDoc)(<nil>),
 CreateDocEvent: (*request.CreateDocEvent)(<nil>),
 ResolveDID: (*request.ResolveDID)(0xc0004949f0)({
  DocId: (string) (len=24) "did:fabricvdr:1234567890"
 }),
 ResolveDIDEvent: (*request.ResolveDIDEvent)(<nil>),
 Error: (*status.Status)(<nil>)
})
2023/05/20 00:09:23 server.go:186: [INFO] [services] --> Resolve DID Request
2023/05/20 00:09:23 server.go:475: [INFO] [resolveDID] --> Resolve a DID
2023/05/20 00:09:23 server.go:479: [INFO] [resolveDID] --> Received Resolve DID request from I
D: 213
```

Figure 55: VDR gRPC Server ResolveDID Response

And finally **ResolveDID** event disseminated back to the Alice agent, as presented in the image below:

```
ubuntu@misiakoulis:~/fabric-samples/Eratosthenes/API-k8s/grpc-go$ docker logs -f alice.aries.e
xample.com
cp: omitting directory '/etc/tls/grpc-client'
WARNING: ca-cert-ec-key.csr.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-ec-cacert.srl.pem does not contain exactly one certificate or CRL: skipping
WARNING: ca-cert-secret-lock.key.pem does not contain exactly one certificate or CRL: skipping
2023/05/19 09:25:20 vdr.go:316: [INFO] [new] --> Number of CPUs: 12
2023/05/19 09:25:20 vdr.go:367: [INFO] [new] --> Starting GRPCS client on host host.docker.int
ernal:4001
2023/05/19 09:25:20 vdr.go:378: [INFO] [new] --> Client 213 is connected
2023/05/19 09:25:20 vdr.go:150: [INFO] [subscriptionEvent] --> Client Subscribed
 [aries-framework/agent-rest] 2023/05/19 09:25:20 UTC - startcmd.startAgent -> INFO Starting a
ries agent rest on host [0.0.0.0:8082]
2023/05/19 21:09:23 vdr.go:157: [INFO] [resolveDIDEvent] --> A ResolveDID Event arrived.
(*did.DocResolution)(0xc000473e90)({
 Context: ([]string) (len=1 cap=1) {
  (string) (len=23) "https://w3id.org/did/v1"
 },
 DIDDocument: (*did.Doc)(0xc000360240)({
  Context: ([]string) (len=1 cap=1) {
   (string) (len=23) "https://w3id.org/did/v1"
  },
  ID: (string) (len=24) "did:fabricvdr:1234567890",
  VerificationMethod: ([]did.VerificationMethod) (len=1 cap=1) {
   (did.VerificationMethod) {
    ID: (string) (len=14) "id-123-456-789",
    Type: (string) (len=26) "Ed25519VerificationKey2018",
    Controller: (string) (len=24) "did:fabricvdr:1234567890",
    Value: ([]uint8) (len=32 cap=32) {
     00000000  65 9d 10 4c 37 83 db be  60 8e f8 eb 12 fd 3c 6a  |e..L7...`.....<j|
     00000010  2a ed d8 bd ca 57 a5 7d  e2 90 0b 6b b7 df 5a 7c  |*....W.}...k..Z||
    },
    jsonWebKey: (*jwk.JWK)(<nil>),
    relativeURL: (bool) false,
    multibaseEncoding: (multibase.Encoding) 0
   }
  },
  Service: ([]did.Service) <nil>,
  Authentication: ([]did.Verification) <nil>,
```

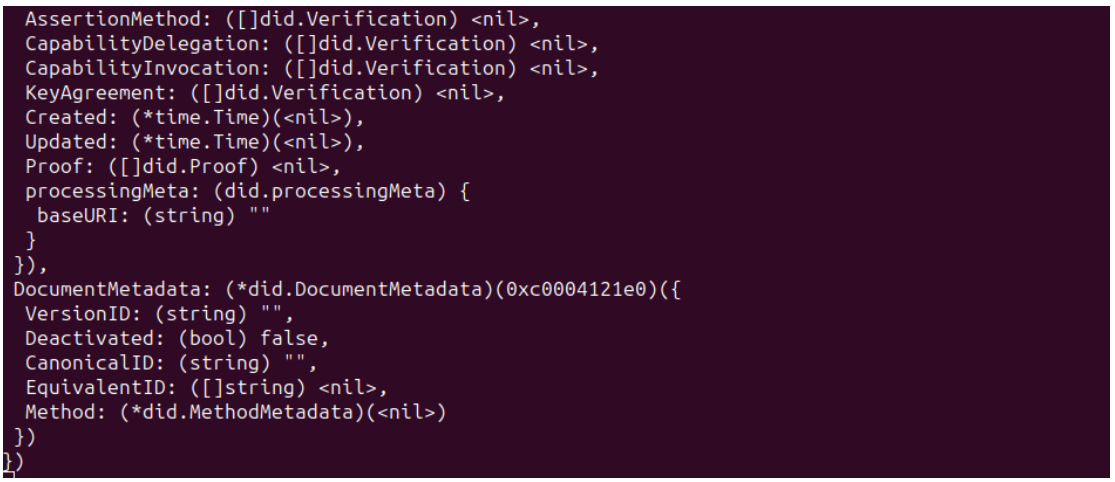

Figure 56: Alice Agent ResolveDIDEvent Response

From the aforementioned showcases, it is clearly depicted that the main capabilities **CreateDoc** and **ResolveDID** of the new VDR package, as described above, operates seamlessly with the Hyperledger Aries Framework GO, paving the way of integrating new permissioned Blockchain networks such as Hyperledger Fabric into Hyperledger Aries as a new VDR.

6. Discussion and Conclusion

The integration of the VDR (Verifiable Data Registry) within the Hyperledger Aries Framework Go opens up possibilities for exploring interoperability with other decentralized identity frameworks. This interoperability enables seamless exchange and verification of verifiable credentials across different platforms, promoting decentralized and self-sovereign identity solutions.

The successful validation of the integrated VDR brings confidence in its reliability, security, and efficiency. This validation process establishes a foundation for organizations to adopt and utilize the VDR in real-world scenarios. These scenarios can include supply chain management, healthcare systems, financial transactions, and various other use cases where data integrity and authenticity are crucial.

By validating the capabilities of the integrated VDR, organizations can harness its potential to enhance data management within blockchain networks. This validation contributes to the wider adoption and acceptance of the VDR solution. Furthermore, the integration of Hyperledger Fabric in Kubernetes adds to the robustness and scalability of the solution.

Through rigorous testing and assessment, the integrated VDR can prove its effectiveness in ensuring secure and trusted data management. This opens up new opportunities for organizations to leverage the benefits of blockchain technology and decentralized identity solutions.

In conclusion, the validation of the integrated VDR, alongside the successful deployment of Hyperledger Fabric in Kubernetes, demonstrates the power and potential of this solution. It sets the stage for secure and trusted data management in various industries, enabling organizations to leverage the advantages of blockchain networks effectively.

7. Bibliography

[1] Verifiable Credentials Data Model v1.1 «Terminology».
URL: https://www.w3.org/TR/vc-data-model/#terminology

[2] Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard.
URL: https://www.rfc-editor.org/rfc/rfc3986.
Other RFCs available at: https://www.w3.org/TR/vc-data-model/#references

[3] A. Tobin and D. Reed, «The inevitable rise of self-sovereign identity,» The Sovrin Foundation, pp. 1-18, 2016.

[4] C. Allen, «The Path to Self-Sovereign Identity,» April 2016. URL: https://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html. [Πρόσβαση 27 10 2022].

[5] K. Cameron, «A user-centric identity metasystem,» Microsoft Corp, 2008.

[6] R. Soltani, U. T. Nguyen A. An, «A Survey of Self-Sovereign Identity Ecosystem,» Security and Communication Networks, 2021

[7] G. Kondova and J. Erbguth, «Self-sovereign identity on public blockchains and the GDPR,» σε Proceedings of the 35th Annual ACM Symposium on Applied Computing, 2020.

[8] D. Van Bokkem, R. Hageman, G. Koning, L. Nguyen and N. Zarin, «Self-sovereign identity solutions: The necessity of blockchain technology,» arXiv preprint arXiv:1904.12816, 2019.

[9] P. Bartolomeu, E. Vieira, S. Hosseini and J. Ferreira, «Self-sovereign identity: Use-cases, technologies, and challenges for industrial IoT» in 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, 2019.

[10] **ISCC 2022 Tutorial Proposal**, Nikos Fotiou, George C. Polyzos and Vasilios A. Siris «Self-Sovereign Identity (SSI), Decentralized Identifiers, and Verifiable Credentials with applications to Access Control, Web Services, the Internet of Things, and Smart Cities». URL: https://iscc2022.unipi.gr/files/Self-Sovereign-Identity.pdf

[11] Verifiable Credentials Data Model v1.1 «What is a Verifiable Credential?». URL: https://www.w3.org/TR/vc-data-model/#what-is-a-verifiable-credential

[12] Verifiable Credentials Data Model v1.1 «Ecosystem Overview». URL: https://www.w3.org/TR/vc-data-model/#ecosystem-overview

[13] Verifiable Credentials Data Model v1.1 «Use Cases and Requirements». URL: https://www.w3.org/TR/vc-data-model/#use-cases-and-requirements

[14] Verifiable Credentials Data Model v1.1 «Claims». URL: https://www.w3.org/TR/vc-data-model/#claims

[15] Verifiable Credentials Data Model v1.1 «Credentials». URL: https://www.w3.org/TR/vc-data-model/#credentials

[16] Verifiable Credentials Data Model v1.1 «Presentations». URL: https://www.w3.org/TR/vc-data-model/#presentations

[17] Verifiable Credentials Data Model v1.1 «Concrete Lifecycle Example». URL: https://www.w3.org/TR/vc-data-model/#concrete-lifecycle-example

[18] Amazon, «What is Hyperledger Fabric?». URL: https://aws.amazon.com/blockchain/what-is-hyperledger-fabric/

[19] IBM, «Kubernetes». URL: https://www.ibm.com/topics/kubernetes

[20] Cloud Native Computing Foundation, « How Kubernetes works». URL: https://www.cncf.io/blog/2019/08/19/how-kubernetes-works/

[21] gRPC, «A high performance, open source universal RPC framework». URL: https://grpc.io/

[22] wallarm, «What is gRPC? Meaning, Architecture, Advantages». URL: https://www.wallarm.com/what/the-concept-of-grpc

[23] Yair Fernando, gRPC «How to make Effective Unary Calls», URL: https://levelup.gitconnected.com/grpc-how-to-make-effective-unary-calls-4c9fa68cd9d5

[24] Yair Fernando, gRPC «How to make Server Streaming Calls», URL: https://levelup.gitconnected.com/grpc-how-to-make-server-streaming-calls-763b42895481

[25] Yair Fernando, gRPC «How to make Client Streaming Calls», URL: https://levelup.gitconnected.com/grpc-how-to-make-client-streaming-calls-5c731197585

[26] Yair Fernando, gRPC «How to make Bidirectional Streaming Calls», URL: https://levelup.gitconnected.com/grpc-how-to-make-bi-directional-streaming-calls-70b4a0569b5b

[27] Anand Rathod, «Serlialize Using GOB in Golang», URL: https://tech.shaadi.com/2021/10/05/serialize-using-gob-in-golang/

[28] E. Bandara et al., “Promize - Blockchain and Self Sovereign Identity Empowered Mobile ATM Platform,” Lecture Notes in Networks and Systems, vol. 284, pp. 891–911, 2021, doi: 10.1007/978-3-030-80126-7_63.

[29] A. Norta, A. Kormiltsyn, C. Udokwu, V. Dwivedi, S. Aroh, and I. Nikolajev, “A Blockchain Implementation for Configurable Multi-Factor Challenge-Set Self-Sovereign Identity Authentication,” in Proceedings - 2022 IEEE International Conference on Blockchain, Blockchain 2022, 2022, pp. 455–461. doi: 10.1109/Blockchain55522.2022.00070.

[30] C. Dong, F. Jiang, X. Li, A. Yao, G. Li, and X. Liu, “A blockchain-aided self-sovereign identity framework for edge-based UAV delivery system,” in Proceedings - 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2021, 2021, pp. 622–624. doi: 10.1109/CCGrid51090.2021.00074.

[31] A. Dixit, M. Smith-Creasey, and M. Rajarajan, “A Decentralized IIoT Identity Framework based on Self-Sovereign Identity using Blockchain,” in Proceedings - Conference on Local Computer Networks, LCN, 2022, pp. 335–338. doi: 10.1109/LCN53696.2022.9843700.

[32] Y. Chen, C. Liu, Y. Wang, and Y. Wang, “A Self-Sovereign Decentralized Identity Platform Based on Blockchain,” in Proceedings - IEEE Symposium on Computers and Communications, 2021. doi: 10.1109/ISCC53001.2021.9631518.

[33] A. Abid, S. Cheikhrouhou, S. Kallel, and M. Jmaiel, “A Blockchain-Based Self-Sovereign Identity Approach for Inter-Organizational Business Processes,” in Proceedings of the 17th Conference on Computer Science and Intelligence Systems, FedCSIS 2022, 2022, pp. 685–694. doi: 10.15439/2022F194.

[34] T. Rathee and P. Singh, “A Self-Sovereign Identity Management System Using Blockchain,” Lecture Notes on Data Engineering and Communications Technologies, vol. 73, pp. 371–379, 2022, doi: 10.1007/978-981-16-3961-6_31.

[35] L. Cocco, R. Tonelli, and M. Marchesi, “Blockchain and self sovereign identity to support quality in the food supply chain,” Future Internet, vol. 13, no. 12, 2021, doi: 10.3390/fi13120301.

[36] K. A. M. Ahmed, S. F. Saraya, J. F. Wanis, and A. M. T. Ali-Eldin, “A Self-Sovereign Identity Architecture Based on Blockchain and the Utilization of Customer’s Banking Cards : The Case of Bank Scam Calls Prevention,” in Proceedings of ICCES 2020 - 2020 15th International Conference on Computer Engineering and Systems, 2020. doi: 10.1109/ICCES51560.2020.9334648.

[37] E. Bandara et al., “A blockchain empowered and privacy preserving digital contact tracing platform,” Information Processing and Management, vol. 58, no. 4, 2021, doi: 10.1016/j.ipm.2021.102572.

[38] A. B. Chavan and K. Rajeswari, “Design and development of self-sovereign identity using ethereum blockchain,” Lecture Notes on Data Engineering and Communications Technologies, vol. 39, pp. 523–531, 2020, doi: 10.1007/978-3-030-34515-0_55.

[39] H. Tadjik, J. Geng, M. G. Jaatun, and C. Rong, “Blockchain Empowered and Self-sovereign Access Control System,” in Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2022, pp. 74–82. doi: 10.1109/CloudCom55334.2022.00021.

[40] E. Bandara et al., “Connect - Blockchain and Self-Sovereign Identity Empowered Contact Tracing Platform,” Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 362 LNICST, pp. 208–223, 2021, doi: 10.1007/978-3-030-70569-5_13.

[41] M. P. Bhattacharya, P. Zavarsky, and S. Butakov, “Enhancing the security and privacy of self-sovereign identities on hyperledger indy blockchain,” in 2020 International Symposium on Networks, Computers and Communications, ISNCC 2020, 2020. doi: 10.1109/ISNCC49221.2020.9297357.

[42] T. Zhou, X. Li, and H. Zhao, “EverSSDI: Blockchain-based framework for verification, authorisation and recovery of self-sovereign identity using smart contracts,” International Journal of Computer Applications in Technology, vol. 60, no. 3, pp. 281–295, 2019, doi: 10.1504/IJCAT.2019.100300.

[43] H. Gulati and C.-T. Huang, “Self-Sovereign Dynamic Digital Identities based on Blockchain Technology,” in Conference Proceedings - IEEE SOUTHEASTCON, 2019. doi: 10.1109/SoutheastCon42311.2019.9020518.

[44] D. N. Kirupanithi and A. Antonidoss, “Self-sovereign identity creation on blockchain using identity based encryption,” in Proceedings - 5th International Conference on Intelligent Computing and Control Systems, ICICCS 2021, 2021, pp. 299–304. doi: 10.1109/ICICCS51141.2021.9432099.

[45] D. N. Kirupanithi and A. Antonidoss, “Self-Sovereign Identity Management System on blockchain based applications using Chameleon Hash,” in Proceedings - 2nd International Conference on Smart Electronics and Communication, ICOSEC 2021, 2021, pp. 386–390. doi: 10.1109/ICOSEC51865.2021.9591734.

[46] K. Papageorgiou and G. Spathoulas, “Self-sovereign, verifiable, ubiquitous and privacy preserving public entity documents through the use of blockchain technology,” in 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference, SEEDA-CECNSM 2022, 2022. doi: 10.1109/SEEDA-CECNSM57760.2022.9932938.

[47] D. T. Harrell et al., “Technical Design and Development of a Self-Sovereign Identity Management Platform for Patient-Centric Health Care using Blockchain Technology,” Blockchain in Healthcare Today, vol. 5, no. Special issue, 2022, doi: 10.30953/bhty.v5.196.

[48] Hyperledger Fabric Docs, «Channels», URL: https://hyperledger-fabric.readthedocs.io/en/latest/channels.html

[49] Hyperledger Fabric Docs, «Running Chaincode as an External Service», URL: https://hyperledger-fabric.readthedocs.io/en/release-2.5/cc_service.html

Appendix A

This Appendix is dedicated to demonstrate the codes that were used during the creation of the smart contracts.

Smart Contract

smartcontract.go

```
package chaincode

import (
 "strings"
 "fmt"
 "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

// SmartContract provides functions for managing a chaincode
type SmartContract struct {
 contractapi.Contract
}

// CreateDoc creates a new DocResolution struct and stores it in the ledger.
func (s *SmartContract) CreateDoc(ctx contractapi.TransactionContextInterface, clientId
string) error {
 if len(clientId) == 0 {
  return fmt.Errorf("Empty string as Input is not supported.")
 }

 // Get new object from Transient map
 transientMap, err := ctx.GetStub().GetTransient()
 if err != nil {
  return fmt.Errorf("Error while getting transient data: %v", err)
 }

 docIDAsBytes := transientMap["DID"]
 docId := string(docIDAsBytes)
 docResolutionAsBytes := transientMap["docResolution"]
 if len(docId) == 0 {
  return fmt.Errorf("Empty DocId as Input is not supported.")
 }

 // Check if a DocResolution exists already with this DID
 docResolutionIfExistsAsBytes, err := getState(ctx, docId)
 if err != nil && !strings.Contains(err.Error(), "Query: Record with id " + docId + " does
not exist") {
  return fmt.Errorf("Error while checking whether the Did exists: %s", err)
 }
 if docResolutionIfExistsAsBytes != nil {
  return fmt.Errorf("DocResolution with Did %s is already exists", docId)
 }

 // Since it is already in bytes there is no need to change it
 err = ctx.GetStub().PutState(docId, docResolutionAsBytes)
 if err != nil {
  return fmt.Errorf("Error while putting data into the ledger: %v", err)
 }

 // Set Event to return Nonce
 disseminateDocResolution := "{DeviceID=" + clientId + "}, {DID=" + docId + "}"
 err = ctx.GetStub().SetEvent("disseminateDocResolution", []byte(disseminateDocResolution))
 if err != nil {
  return fmt.Errorf("Error while Setting Event: %s", err)
 }
 return nil
}

// ResolveDid returns the DocResolution record stored in the world state under specified
did
func (s *SmartContract) ResolveDid(ctx contractapi.TransactionContextInterface, docId
```

```
string) (string, error) {
 valAsbytes, err := getState(ctx, docId)
 if err != nil {
  return "", fmt.Errorf("Error while handling ResolveDID request: %s", err)
 }
 return string(valAsbytes), nil
}

////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////// Helper Functions ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
// getState returns the DocResolution record stored in the world state with given input
func getState(ctx contractapi.TransactionContextInterface, input string) ([]byte, error) {
 //Check if the input is empty
 if len(input) == 0 {
  return nil, fmt.Errorf("Empty string as Input is not supported.")
 }

 //Get the object from chaincode state
 valAsbytes, err := ctx.GetStub().GetState(input)
 if err != nil {
  return nil, fmt.Errorf("Query: Failed to get state with key %s, error: %v", input, err)
 } else if valAsbytes == nil {
  return nil, fmt.Errorf("Query: Record with id %s does not exist", input)
 }
 return valAsbytes, nil
}
```

Smart Contract Main

fabricVdr.go

```
package main

import (
 "log"
 "github.com/hyperledger/fabric-contract-api-go/contractapi"
 "github.com/hyperledger/fabric-samples/chaincode/fabricVdr/chaincode-go/chaincode"
)

func main() {
 fabricVdr, err := contractapi.NewChaincode(new(chaincode.SmartContract))
 if err != nil {
  log.Panicf("Error create Fabric VDR chaincode: %v", err)
  return
 }
 if err := fabricVdr.Start(); err != nil {
  log.Panicf("Error starting Fabric VDR chaincode: %v", err)
 }
}
```

Dependency Requirements

go.mod

```
module github.com/hyperledger/fabric-samples/chaincode/fabricVdr/chaincode-go

go 1.17
```

Mock Stubs for Chaincode Unit Tests

chaincodestub.go

```
// Code generated by counterfeiter. DO NOT EDIT.
```

```
package mocks

import (
    "sync"

    "github.com/hyperledger/fabric-chaincode-go/shim"
    "github.com/hyperledger/fabric-protos-go/peer"
    "google.golang.org/protobuf/types/known/timestamppb"
)

type ChaincodeStub struct {
    CreateCompositeKeyStub       func(string, []string) (string, error)
    createCompositeKeyMutex      sync.RWMutex
    createCompositeKeyArgsForCall []struct {
         arg1 string
         arg2 []string
    }
    createCompositeKeyReturns struct {
         result1 string
         result2 error
    }
    createCompositeKeyReturnsOnCall map[int]struct {
         result1 string
         result2 error
    }
    DelPrivateDataStub           func(string, string) error
    delPrivateDataMutex          sync.RWMutex
    delPrivateDataArgsForCall []struct {
         arg1 string
         arg2 string
    }
    delPrivateDataReturns struct {
         result1 error
    }
    delPrivateDataReturnsOnCall map[int]struct {
         result1 error
    }
    DelStateStub         func(string) error
    delStateMutex        sync.RWMutex
    delStateArgsForCall []struct {
         arg1 string
    }
    delStateReturns struct {
         result1 error
    }
    delStateReturnsOnCall map[int]struct {
         result1 error
    }
    GetArgsStub          func() [][]byte
    getArgsMutex         sync.RWMutex
    getArgsArgsForCall []struct {
    }
    getArgsReturns struct {
         result1 [][]byte
    }
    getArgsReturnsOnCall map[int]struct {
         result1 [][]byte
    }
    GetArgsSliceStub     func() ([]byte, error)
    getArgsSliceMutex    sync.RWMutex
    getArgsSliceArgsForCall []struct {
    }
    getArgsSliceReturns struct {
         result1 []byte
         result2 error
    }
    getArgsSliceReturnsOnCall map[int]struct {
         result1 []byte
         result2 error
    }
    GetBindingStub       func() ([]byte, error)
    getBindingMutex      sync.RWMutex
    getBindingArgsForCall []struct {
```

```
    }
    getBindingReturns struct {
        result1 []byte
        result2 error
    }
    getBindingReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetChannelIDStub     func() string
    getChannelIDMutex    sync.RWMutex
    getChannelIDArgsForCall []struct {
    }
    getChannelIDReturns struct {
        result1 string
    }
    getChannelIDReturnsOnCall map[int]struct {
        result1 string
    }
    GetCreatorStub       func() ([]byte, error)
    getCreatorMutex      sync.RWMutex
    getCreatorArgsForCall []struct {
    }
    getCreatorReturns struct {
        result1 []byte
        result2 error
    }
    getCreatorReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetDecorationsStub           func() map[string][]byte
    getDecorationsMutex          sync.RWMutex
    getDecorationsArgsForCall []struct {
    }
    getDecorationsReturns struct {
        result1 map[string][]byte
    }
    getDecorationsReturnsOnCall map[int]struct {
        result1 map[string][]byte
    }
    GetFunctionAndParametersStub         func() (string, []string)
    getFunctionAndParametersMutex        sync.RWMutex
    getFunctionAndParametersArgsForCall []struct {
    }
    getFunctionAndParametersReturns struct {
        result1 string
        result2 []string
    }
    getFunctionAndParametersReturnsOnCall map[int]struct {
        result1 string
        result2 []string
    }
    GetHistoryForKeyStub         func(string) (shim.HistoryQueryIteratorInterface, error)
    getHistoryForKeyMutex        sync.RWMutex
    getHistoryForKeyArgsForCall []struct {
        arg1 string
    }
    getHistoryForKeyReturns struct {
        result1 shim.HistoryQueryIteratorInterface
        result2 error
    }
    getHistoryForKeyReturnsOnCall map[int]struct {
        result1 shim.HistoryQueryIteratorInterface
        result2 error
    }
    GetPrivateDataStub           func(string, string) ([]byte, error)
    getPrivateDataMutex          sync.RWMutex
    getPrivateDataArgsForCall []struct {
        arg1 string
        arg2 string
    }
    getPrivateDataReturns struct {
```

```
        result1 []byte
        result2 error
    }
    getPrivateDataReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetPrivateDataByPartialCompositeKeyStub      func(string, string, []string)
(shim.StateQueryIteratorInterface, error)
    getPrivateDataByPartialCompositeKeyMutex     sync.RWMutex
    getPrivateDataByPartialCompositeKeyArgsForCall []struct {
        arg1 string
        arg2 string
        arg3 []string
    }
    getPrivateDataByPartialCompositeKeyReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getPrivateDataByPartialCompositeKeyReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetPrivateDataByRangeStub            func(string, string, string)
(shim.StateQueryIteratorInterface, error)
    getPrivateDataByRangeMutex           sync.RWMutex
    getPrivateDataByRangeArgsForCall []struct {
        arg1 string
        arg2 string
        arg3 string
    }
    getPrivateDataByRangeReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getPrivateDataByRangeReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetPrivateDataHashStub       func(string, string) ([]byte, error)
    getPrivateDataHashMutex      sync.RWMutex
    getPrivateDataHashArgsForCall []struct {
        arg1 string
        arg2 string
    }
    getPrivateDataHashReturns struct {
        result1 []byte
        result2 error
    }
    getPrivateDataHashReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetPrivateDataQueryResultStub        func(string, string)
(shim.StateQueryIteratorInterface, error)
    getPrivateDataQueryResultMutex       sync.RWMutex
    getPrivateDataQueryResultArgsForCall []struct {
        arg1 string
        arg2 string
    }
    getPrivateDataQueryResultReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getPrivateDataQueryResultReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetPrivateDataValidationParameterStub        func(string, string) ([]byte, error)
    getPrivateDataValidationParameterMutex       sync.RWMutex
    getPrivateDataValidationParameterArgsForCall []struct {
        arg1 string
        arg2 string
```

```
    }
    getPrivateDataValidationParameterReturns struct {
        result1 []byte
        result2 error
    }
    getPrivateDataValidationParameterReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetQueryResultStub           func(string) (shim.StateQueryIteratorInterface, error)
    getQueryResultMutex          sync.RWMutex
    getQueryResultArgsForCall []struct {
        arg1 string
    }
    getQueryResultReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getQueryResultReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetQueryResultWithPaginationStub     func(string, int32, string)
(shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)
    getQueryResultWithPaginationMutex    sync.RWMutex
    getQueryResultWithPaginationArgsForCall []struct {
        arg1 string
        arg2 int32
        arg3 string
    }
    getQueryResultWithPaginationReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    getQueryResultWithPaginationReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    GetSignedProposalStub        func() (*peer.SignedProposal, error)
    getSignedProposalMutex       sync.RWMutex
    getSignedProposalArgsForCall []struct {
    }
    getSignedProposalReturns struct {
        result1 *peer.SignedProposal
        result2 error
    }
    getSignedProposalReturnsOnCall map[int]struct {
        result1 *peer.SignedProposal
        result2 error
    }
    GetStateStub         func(string) ([]byte, error)
    getStateMutex        sync.RWMutex
    getStateArgsForCall []struct {
        arg1 string
    }
    getStateReturns struct {
        result1 []byte
        result2 error
    }
    getStateReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
    GetStateByPartialCompositeKeyStub            func(string, []string)
(shim.StateQueryIteratorInterface, error)
    getStateByPartialCompositeKeyMutex           sync.RWMutex
    getStateByPartialCompositeKeyArgsForCall []struct {
        arg1 string
        arg2 []string
    }
    getStateByPartialCompositeKeyReturns struct {
```

```
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getStateByPartialCompositeKeyReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetStateByPartialCompositeKeyWithPaginationStub      func(string, []string, int32,
string) (shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)
    getStateByPartialCompositeKeyWithPaginationMutex     sync.RWMutex
    getStateByPartialCompositeKeyWithPaginationArgsForCall []struct {
        arg1 string
        arg2 []string
        arg3 int32
        arg4 string
    }
    getStateByPartialCompositeKeyWithPaginationReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    getStateByPartialCompositeKeyWithPaginationReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    GetStateByRangeStub          func(string, string) (shim.StateQueryIteratorInterface,
error)
    getStateByRangeMutex         sync.RWMutex
    getStateByRangeArgsForCall []struct {
        arg1 string
        arg2 string
    }
    getStateByRangeReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    getStateByRangeReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }
    GetStateByRangeWithPaginationStub            func(string, string, int32, string)
(shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)
    getStateByRangeWithPaginationMutex           sync.RWMutex
    getStateByRangeWithPaginationArgsForCall []struct {
        arg1 string
        arg2 string
        arg3 int32
        arg4 string
    }
    getStateByRangeWithPaginationReturns struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    getStateByRangeWithPaginationReturnsOnCall map[int]struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }
    GetStateValidationParameterStub      func(string) ([]byte, error)
    getStateValidationParameterMutex     sync.RWMutex
    getStateValidationParameterArgsForCall []struct {
        arg1 string
    }
    getStateValidationParameterReturns struct {
        result1 []byte
        result2 error
    }
    getStateValidationParameterReturnsOnCall map[int]struct {
        result1 []byte
        result2 error
    }
```

```
    GetStringArgsStub           func() []string
    getStringArgsMutex          sync.RWMutex
    getStringArgsArgsForCall []struct {
    }
    getStringArgsReturns struct {
         result1 []string
    }
    getStringArgsReturnsOnCall map[int]struct {
         result1 []string
    }
    GetTransientStub     func() (map[string][]byte, error)
    getTransientMutex    sync.RWMutex
    getTransientArgsForCall []struct {
    }
    getTransientReturns struct {
         result1 map[string][]byte
         result2 error
    }
    getTransientReturnsOnCall map[int]struct {
         result1 map[string][]byte
         result2 error
    }
    GetTxIDStub          func() string
    getTxIDMutex         sync.RWMutex
    getTxIDArgsForCall []struct {
    }
    getTxIDReturns struct {
         result1 string
    }
    getTxIDReturnsOnCall map[int]struct {
         result1 string
    }
    GetTxTimestampStub          func() (*timestamppb.Timestamp, error)
    getTxTimestampMutex         sync.RWMutex
    getTxTimestampArgsForCall []struct {
    }
    getTxTimestampReturns struct {
         result1 *timestamppb.Timestamp
         result2 error
    }
    getTxTimestampReturnsOnCall map[int]struct {
         result1 *timestamppb.Timestamp
         result2 error
    }
    InvokeChaincodeStub         func(string, [][]byte, string) peer.Response
    invokeChaincodeMutex        sync.RWMutex
    invokeChaincodeArgsForCall []struct {
         arg1 string
         arg2 [][]byte
         arg3 string
    }
    invokeChaincodeReturns struct {
         result1 peer.Response
    }
    invokeChaincodeReturnsOnCall map[int]struct {
         result1 peer.Response
    }
    PurgePrivateDataStub        func(string, string) error
    purgePrivateDataMutex       sync.RWMutex
    purgePrivateDataArgsForCall []struct {
         arg1 string
         arg2 string
    }
    purgePrivateDataReturns struct {
         result1 error
    }
    purgePrivateDataReturnsOnCall map[int]struct {
         result1 error
    }
    PutPrivateDataStub          func(string, string, []byte) error
    putPrivateDataMutex         sync.RWMutex
    putPrivateDataArgsForCall []struct {
         arg1 string
```

```
        arg2 string
        arg3 []byte
    }
    putPrivateDataReturns struct {
        result1 error
    }
    putPrivateDataReturnsOnCall map[int]struct {
        result1 error
    }
    PutStateStub         func(string, []byte) error
    putStateMutex        sync.RWMutex
    putStateArgsForCall []struct {
        arg1 string
        arg2 []byte
    }
    putStateReturns struct {
        result1 error
    }
    putStateReturnsOnCall map[int]struct {
        result1 error
    }
    SetEventStub         func(string, []byte) error
    setEventMutex        sync.RWMutex
    setEventArgsForCall []struct {
        arg1 string
        arg2 []byte
    }
    setEventReturns struct {
        result1 error
    }
    setEventReturnsOnCall map[int]struct {
        result1 error
    }
    SetPrivateDataValidationParameterStub        func(string, string, []byte) error
    setPrivateDataValidationParameterMutex       sync.RWMutex
    setPrivateDataValidationParameterArgsForCall []struct {
        arg1 string
        arg2 string
        arg3 []byte
    }
    setPrivateDataValidationParameterReturns struct {
        result1 error
    }
    setPrivateDataValidationParameterReturnsOnCall map[int]struct {
        result1 error
    }
    SetStateValidationParameterStub      func(string, []byte) error
    setStateValidationParameterMutex     sync.RWMutex
    setStateValidationParameterArgsForCall []struct {
        arg1 string
        arg2 []byte
    }
    setStateValidationParameterReturns struct {
        result1 error
    }
    setStateValidationParameterReturnsOnCall map[int]struct {
        result1 error
    }
    SplitCompositeKeyStub        func(string) (string, []string, error)
    splitCompositeKeyMutex       sync.RWMutex
    splitCompositeKeyArgsForCall []struct {
        arg1 string
    }
    splitCompositeKeyReturns struct {
        result1 string
        result2 []string
        result3 error
    }
    splitCompositeKeyReturnsOnCall map[int]struct {
        result1 string
        result2 []string
        result3 error
    }
```

```
    invocations          map[string][][]interface{}
    invocationsMutex sync.RWMutex
}

func (fake *ChaincodeStub) CreateCompositeKey(arg1 string, arg2 []string) (string, error) {
    var arg2Copy []string
    if arg2 != nil {
         arg2Copy = make([]string, len(arg2))
         copy(arg2Copy, arg2)
    }
    fake.createCompositeKeyMutex.Lock()
    ret, specificReturn :=
fake.createCompositeKeyReturnsOnCall[len(fake.createCompositeKeyArgsForCall)]
    fake.createCompositeKeyArgsForCall = append(fake.createCompositeKeyArgsForCall, struct
{
         arg1 string
         arg2 []string
    }{arg1, arg2Copy})
    fake.recordInvocation("CreateCompositeKey", []interface{}{arg1, arg2Copy})
    fake.createCompositeKeyMutex.Unlock()
    if fake.CreateCompositeKeyStub != nil {
         return fake.CreateCompositeKeyStub(arg1, arg2)
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.createCompositeKeyReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) CreateCompositeKeyCallCount() int {
    fake.createCompositeKeyMutex.RLock()
    defer fake.createCompositeKeyMutex.RUnlock()
    return len(fake.createCompositeKeyArgsForCall)
}

func (fake *ChaincodeStub) CreateCompositeKeyCalls(stub func(string, []string) (string,
error)) {
    fake.createCompositeKeyMutex.Lock()
    defer fake.createCompositeKeyMutex.Unlock()
    fake.CreateCompositeKeyStub = stub
}

func (fake *ChaincodeStub) CreateCompositeKeyArgsForCall(i int) (string, []string) {
    fake.createCompositeKeyMutex.RLock()
    defer fake.createCompositeKeyMutex.RUnlock()
    argsForCall := fake.createCompositeKeyArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) CreateCompositeKeyReturns(result1 string, result2 error) {
    fake.createCompositeKeyMutex.Lock()
    defer fake.createCompositeKeyMutex.Unlock()
    fake.CreateCompositeKeyStub = nil
    fake.createCompositeKeyReturns = struct {
         result1 string
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) CreateCompositeKeyReturnsOnCall(i int, result1 string, result2
error) {
    fake.createCompositeKeyMutex.Lock()
    defer fake.createCompositeKeyMutex.Unlock()
    fake.CreateCompositeKeyStub = nil
    if fake.createCompositeKeyReturnsOnCall == nil {
         fake.createCompositeKeyReturnsOnCall = make(map[int]struct {
                 result1 string
                 result2 error
         })
    }
    fake.createCompositeKeyReturnsOnCall[i] = struct {
         result1 string
```

```
            result2 error
        }{result1, result2}
}

func (fake *ChaincodeStub) DelPrivateData(arg1 string, arg2 string) error {
        fake.delPrivateDataMutex.Lock()
        ret, specificReturn :=
fake.delPrivateDataReturnsOnCall[len(fake.delPrivateDataArgsForCall)]
        fake.delPrivateDataArgsForCall = append(fake.delPrivateDataArgsForCall, struct {
                arg1 string
                arg2 string
        }{arg1, arg2})
        fake.recordInvocation("DelPrivateData", []interface{}{arg1, arg2})
        fake.delPrivateDataMutex.Unlock()
        if fake.DelPrivateDataStub != nil {
                return fake.DelPrivateDataStub(arg1, arg2)
        }
        if specificReturn {
                return ret.result1
        }
        fakeReturns := fake.delPrivateDataReturns
        return fakeReturns.result1
}

func (fake *ChaincodeStub) DelPrivateDataCallCount() int {
        fake.delPrivateDataMutex.RLock()
        defer fake.delPrivateDataMutex.RUnlock()
        return len(fake.delPrivateDataArgsForCall)
}

func (fake *ChaincodeStub) DelPrivateDataCalls(stub func(string, string) error) {
        fake.delPrivateDataMutex.Lock()
        defer fake.delPrivateDataMutex.Unlock()
        fake.DelPrivateDataStub = stub
}

func (fake *ChaincodeStub) DelPrivateDataArgsForCall(i int) (string, string) {
        fake.delPrivateDataMutex.RLock()
        defer fake.delPrivateDataMutex.RUnlock()
        argsForCall := fake.delPrivateDataArgsForCall[i]
        return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) DelPrivateDataReturns(result1 error) {
        fake.delPrivateDataMutex.Lock()
        defer fake.delPrivateDataMutex.Unlock()
        fake.DelPrivateDataStub = nil
        fake.delPrivateDataReturns = struct {
                result1 error
        }{result1}
}

func (fake *ChaincodeStub) DelPrivateDataReturnsOnCall(i int, result1 error) {
        fake.delPrivateDataMutex.Lock()
        defer fake.delPrivateDataMutex.Unlock()
        fake.DelPrivateDataStub = nil
        if fake.delPrivateDataReturnsOnCall == nil {
                fake.delPrivateDataReturnsOnCall = make(map[int]struct {
                        result1 error
                })
        }
        fake.delPrivateDataReturnsOnCall[i] = struct {
                result1 error
        }{result1}
}

func (fake *ChaincodeStub) DelState(arg1 string) error {
        fake.delStateMutex.Lock()
        ret, specificReturn := fake.delStateReturnsOnCall[len(fake.delStateArgsForCall)]
        fake.delStateArgsForCall = append(fake.delStateArgsForCall, struct {
                arg1 string
        }{arg1})
        fake.recordInvocation("DelState", []interface{}{arg1})
```

```
    fake.delStateMutex.Unlock()
    if fake.DelStateStub != nil {
        return fake.DelStateStub(arg1)
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.delStateReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) DelStateCallCount() int {
    fake.delStateMutex.RLock()
    defer fake.delStateMutex.RUnlock()
    return len(fake.delStateArgsForCall)
}

func (fake *ChaincodeStub) DelStateCalls(stub func(string) error) {
    fake.delStateMutex.Lock()
    defer fake.delStateMutex.Unlock()
    fake.DelStateStub = stub
}

func (fake *ChaincodeStub) DelStateArgsForCall(i int) string {
    fake.delStateMutex.RLock()
    defer fake.delStateMutex.RUnlock()
    argsForCall := fake.delStateArgsForCall[i]
    return argsForCall.arg1
}

func (fake *ChaincodeStub) DelStateReturns(result1 error) {
    fake.delStateMutex.Lock()
    defer fake.delStateMutex.Unlock()
    fake.DelStateStub = nil
    fake.delStateReturns = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) DelStateReturnsOnCall(i int, result1 error) {
    fake.delStateMutex.Lock()
    defer fake.delStateMutex.Unlock()
    fake.DelStateStub = nil
    if fake.delStateReturnsOnCall == nil {
        fake.delStateReturnsOnCall = make(map[int]struct {
            result1 error
        })
    }
    fake.delStateReturnsOnCall[i] = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) GetArgs() [][]byte {
    fake.getArgsMutex.Lock()
    ret, specificReturn := fake.getArgsReturnsOnCall[len(fake.getArgsArgsForCall)]
    fake.getArgsArgsForCall = append(fake.getArgsArgsForCall, struct {
    }{})
    fake.recordInvocation("GetArgs", []interface{}{})
    fake.getArgsMutex.Unlock()
    if fake.GetArgsStub != nil {
        return fake.GetArgsStub()
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.getArgsReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) GetArgsCallCount() int {
    fake.getArgsMutex.RLock()
    defer fake.getArgsMutex.RUnlock()
```

```
    return len(fake.getArgsArgsForCall)
}

func (fake *ChaincodeStub) GetArgsCalls(stub func() [][]byte) {
    fake.getArgsMutex.Lock()
    defer fake.getArgsMutex.Unlock()
    fake.GetArgsStub = stub
}

func (fake *ChaincodeStub) GetArgsReturns(result1 [][]byte) {
    fake.getArgsMutex.Lock()
    defer fake.getArgsMutex.Unlock()
    fake.GetArgsStub = nil
    fake.getArgsReturns = struct {
        result1 [][]byte
    }{result1}
}

func (fake *ChaincodeStub) GetArgsReturnsOnCall(i int, result1 [][]byte) {
    fake.getArgsMutex.Lock()
    defer fake.getArgsMutex.Unlock()
    fake.GetArgsStub = nil
    if fake.getArgsReturnsOnCall == nil {
        fake.getArgsReturnsOnCall = make(map[int]struct {
            result1 [][]byte
        })
    }
    fake.getArgsReturnsOnCall[i] = struct {
        result1 [][]byte
    }{result1}
}

func (fake *ChaincodeStub) GetArgsSlice() ([]byte, error) {
    fake.getArgsSliceMutex.Lock()
    ret, specificReturn :=
fake.getArgsSliceReturnsOnCall[len(fake.getArgsSliceArgsForCall)]
    fake.getArgsSliceArgsForCall = append(fake.getArgsSliceArgsForCall, struct {
    }{})
    fake.recordInvocation("GetArgsSlice", []interface{}{})
    fake.getArgsSliceMutex.Unlock()
    if fake.GetArgsSliceStub != nil {
        return fake.GetArgsSliceStub()
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getArgsSliceReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetArgsSliceCallCount() int {
    fake.getArgsSliceMutex.RLock()
    defer fake.getArgsSliceMutex.RUnlock()
    return len(fake.getArgsSliceArgsForCall)
}

func (fake *ChaincodeStub) GetArgsSliceCalls(stub func() ([]byte, error)) {
    fake.getArgsSliceMutex.Lock()
    defer fake.getArgsSliceMutex.Unlock()
    fake.GetArgsSliceStub = stub
}

func (fake *ChaincodeStub) GetArgsSliceReturns(result1 []byte, result2 error) {
    fake.getArgsSliceMutex.Lock()
    defer fake.getArgsSliceMutex.Unlock()
    fake.GetArgsSliceStub = nil
    fake.getArgsSliceReturns = struct {
        result1 []byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetArgsSliceReturnsOnCall(i int, result1 []byte, result2 error)
```

```
{
    fake.getArgsSliceMutex.Lock()
    defer fake.getArgsSliceMutex.Unlock()
    fake.GetArgsSliceStub = nil
    if fake.getArgsSliceReturnsOnCall == nil {
         fake.getArgsSliceReturnsOnCall = make(map[int]struct {
                 result1 []byte
                 result2 error
         })
    }
    fake.getArgsSliceReturnsOnCall[i] = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetBinding() ([]byte, error) {
    fake.getBindingMutex.Lock()
    ret, specificReturn := fake.getBindingReturnsOnCall[len(fake.getBindingArgsForCall)]
    fake.getBindingArgsForCall = append(fake.getBindingArgsForCall, struct {
    }{})
    fake.recordInvocation("GetBinding", []interface{}{})
    fake.getBindingMutex.Unlock()
    if fake.GetBindingStub != nil {
         return fake.GetBindingStub()
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.getBindingReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetBindingCallCount() int {
    fake.getBindingMutex.RLock()
    defer fake.getBindingMutex.RUnlock()
    return len(fake.getBindingArgsForCall)
}

func (fake *ChaincodeStub) GetBindingCalls(stub func() ([]byte, error)) {
    fake.getBindingMutex.Lock()
    defer fake.getBindingMutex.Unlock()
    fake.GetBindingStub = stub
}

func (fake *ChaincodeStub) GetBindingReturns(result1 []byte, result2 error) {
    fake.getBindingMutex.Lock()
    defer fake.getBindingMutex.Unlock()
    fake.GetBindingStub = nil
    fake.getBindingReturns = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetBindingReturnsOnCall(i int, result1 []byte, result2 error) {
    fake.getBindingMutex.Lock()
    defer fake.getBindingMutex.Unlock()
    fake.GetBindingStub = nil
    if fake.getBindingReturnsOnCall == nil {
         fake.getBindingReturnsOnCall = make(map[int]struct {
                 result1 []byte
                 result2 error
         })
    }
    fake.getBindingReturnsOnCall[i] = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetChannelID() string {
    fake.getChannelIDMutex.Lock()
```

```
    ret, specificReturn :=
fake.getChannelIDReturnsOnCall[len(fake.getChannelIDArgsForCall)]
    fake.getChannelIDArgsForCall = append(fake.getChannelIDArgsForCall, struct {
    }{})
    fake.recordInvocation("GetChannelID", []interface{}{})
    fake.getChannelIDMutex.Unlock()
    if fake.GetChannelIDStub != nil {
         return fake.GetChannelIDStub()
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.getChannelIDReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) GetChannelIDCallCount() int {
    fake.getChannelIDMutex.RLock()
    defer fake.getChannelIDMutex.RUnlock()
    return len(fake.getChannelIDArgsForCall)
}

func (fake *ChaincodeStub) GetChannelIDCalls(stub func() string) {
    fake.getChannelIDMutex.Lock()
    defer fake.getChannelIDMutex.Unlock()
    fake.GetChannelIDStub = stub
}

func (fake *ChaincodeStub) GetChannelIDReturns(result1 string) {
    fake.getChannelIDMutex.Lock()
    defer fake.getChannelIDMutex.Unlock()
    fake.GetChannelIDStub = nil
    fake.getChannelIDReturns = struct {
         result1 string
    }{result1}
}

func (fake *ChaincodeStub) GetChannelIDReturnsOnCall(i int, result1 string) {
    fake.getChannelIDMutex.Lock()
    defer fake.getChannelIDMutex.Unlock()
    fake.GetChannelIDStub = nil
    if fake.getChannelIDReturnsOnCall == nil {
         fake.getChannelIDReturnsOnCall = make(map[int]struct {
                result1 string
         })
    }
    fake.getChannelIDReturnsOnCall[i] = struct {
         result1 string
    }{result1}
}

func (fake *ChaincodeStub) GetCreator() ([]byte, error) {
    fake.getCreatorMutex.Lock()
    ret, specificReturn := fake.getCreatorReturnsOnCall[len(fake.getCreatorArgsForCall)]
    fake.getCreatorArgsForCall = append(fake.getCreatorArgsForCall, struct {
    }{})
    fake.recordInvocation("GetCreator", []interface{}{})
    fake.getCreatorMutex.Unlock()
    if fake.GetCreatorStub != nil {
         return fake.GetCreatorStub()
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.getCreatorReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetCreatorCallCount() int {
    fake.getCreatorMutex.RLock()
    defer fake.getCreatorMutex.RUnlock()
    return len(fake.getCreatorArgsForCall)
```

```
}

func (fake *ChaincodeStub) GetCreatorCalls(stub func() ([]byte, error)) {
    fake.getCreatorMutex.Lock()
    defer fake.getCreatorMutex.Unlock()
    fake.GetCreatorStub = stub
}

func (fake *ChaincodeStub) GetCreatorReturns(result1 []byte, result2 error) {
    fake.getCreatorMutex.Lock()
    defer fake.getCreatorMutex.Unlock()
    fake.GetCreatorStub = nil
    fake.getCreatorReturns = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetCreatorReturnsOnCall(i int, result1 []byte, result2 error) {
    fake.getCreatorMutex.Lock()
    defer fake.getCreatorMutex.Unlock()
    fake.GetCreatorStub = nil
    if fake.getCreatorReturnsOnCall == nil {
         fake.getCreatorReturnsOnCall = make(map[int]struct {
                 result1 []byte
                 result2 error
         })
    }
    fake.getCreatorReturnsOnCall[i] = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetDecorations() map[string][]byte {
    fake.getDecorationsMutex.Lock()
    ret, specificReturn :=
fake.getDecorationsReturnsOnCall[len(fake.getDecorationsArgsForCall)]
    fake.getDecorationsArgsForCall = append(fake.getDecorationsArgsForCall, struct {
    }{})
    fake.recordInvocation("GetDecorations", []interface{}{})
    fake.getDecorationsMutex.Unlock()
    if fake.GetDecorationsStub != nil {
         return fake.GetDecorationsStub()
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.getDecorationsReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) GetDecorationsCallCount() int {
    fake.getDecorationsMutex.RLock()
    defer fake.getDecorationsMutex.RUnlock()
    return len(fake.getDecorationsArgsForCall)
}

func (fake *ChaincodeStub) GetDecorationsCalls(stub func() map[string][]byte) {
    fake.getDecorationsMutex.Lock()
    defer fake.getDecorationsMutex.Unlock()
    fake.GetDecorationsStub = stub
}

func (fake *ChaincodeStub) GetDecorationsReturns(result1 map[string][]byte) {
    fake.getDecorationsMutex.Lock()
    defer fake.getDecorationsMutex.Unlock()
    fake.GetDecorationsStub = nil
    fake.getDecorationsReturns = struct {
         result1 map[string][]byte
    }{result1}
}
```

```
func (fake *ChaincodeStub) GetDecorationsReturnsOnCall(i int, result1 map[string][]byte) {
    fake.getDecorationsMutex.Lock()
    defer fake.getDecorationsMutex.Unlock()
    fake.GetDecorationsStub = nil
    if fake.getDecorationsReturnsOnCall == nil {
         fake.getDecorationsReturnsOnCall = make(map[int]struct {
                 result1 map[string][]byte
         })
    }
    fake.getDecorationsReturnsOnCall[i] = struct {
         result1 map[string][]byte
    }{result1}
}

func (fake *ChaincodeStub) GetFunctionAndParameters() (string, []string) {
    fake.getFunctionAndParametersMutex.Lock()
    ret, specificReturn :=
fake.getFunctionAndParametersReturnsOnCall[len(fake.getFunctionAndParametersArgsForCall)]
    fake.getFunctionAndParametersArgsForCall =
append(fake.getFunctionAndParametersArgsForCall, struct {
    }{})
    fake.recordInvocation("GetFunctionAndParameters", []interface{}{})
    fake.getFunctionAndParametersMutex.Unlock()
    if fake.GetFunctionAndParametersStub != nil {
         return fake.GetFunctionAndParametersStub()
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.getFunctionAndParametersReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetFunctionAndParametersCallCount() int {
    fake.getFunctionAndParametersMutex.RLock()
    defer fake.getFunctionAndParametersMutex.RUnlock()
    return len(fake.getFunctionAndParametersArgsForCall)
}

func (fake *ChaincodeStub) GetFunctionAndParametersCalls(stub func() (string, []string)) {
    fake.getFunctionAndParametersMutex.Lock()
    defer fake.getFunctionAndParametersMutex.Unlock()
    fake.GetFunctionAndParametersStub = stub
}

func (fake *ChaincodeStub) GetFunctionAndParametersReturns(result1 string, result2
[]string) {
    fake.getFunctionAndParametersMutex.Lock()
    defer fake.getFunctionAndParametersMutex.Unlock()
    fake.GetFunctionAndParametersStub = nil
    fake.getFunctionAndParametersReturns = struct {
         result1 string
         result2 []string
    }{result1, result2}
}

func (fake *ChaincodeStub) GetFunctionAndParametersReturnsOnCall(i int, result1 string,
result2 []string) {
    fake.getFunctionAndParametersMutex.Lock()
    defer fake.getFunctionAndParametersMutex.Unlock()
    fake.GetFunctionAndParametersStub = nil
    if fake.getFunctionAndParametersReturnsOnCall == nil {
         fake.getFunctionAndParametersReturnsOnCall = make(map[int]struct {
                 result1 string
                 result2 []string
         })
    }
    fake.getFunctionAndParametersReturnsOnCall[i] = struct {
         result1 string
         result2 []string
    }{result1, result2}
}
```

```
func (fake *ChaincodeStub) GetHistoryForKey(arg1 string)
(shim.HistoryQueryIteratorInterface, error) {
    fake.getHistoryForKeyMutex.Lock()
    ret, specificReturn :=
fake.getHistoryForKeyReturnsOnCall[len(fake.getHistoryForKeyArgsForCall)]
    fake.getHistoryForKeyArgsForCall = append(fake.getHistoryForKeyArgsForCall, struct {
        arg1 string
    }{arg1})
    fake.recordInvocation("GetHistoryForKey", []interface{}{arg1})
    fake.getHistoryForKeyMutex.Unlock()
    if fake.GetHistoryForKeyStub != nil {
        return fake.GetHistoryForKeyStub(arg1)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getHistoryForKeyReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetHistoryForKeyCallCount() int {
    fake.getHistoryForKeyMutex.RLock()
    defer fake.getHistoryForKeyMutex.RUnlock()
    return len(fake.getHistoryForKeyArgsForCall)
}

func (fake *ChaincodeStub) GetHistoryForKeyCalls(stub func(string)
(shim.HistoryQueryIteratorInterface, error)) {
    fake.getHistoryForKeyMutex.Lock()
    defer fake.getHistoryForKeyMutex.Unlock()
    fake.GetHistoryForKeyStub = stub
}

func (fake *ChaincodeStub) GetHistoryForKeyArgsForCall(i int) string {
    fake.getHistoryForKeyMutex.RLock()
    defer fake.getHistoryForKeyMutex.RUnlock()
    argsForCall := fake.getHistoryForKeyArgsForCall[i]
    return argsForCall.arg1
}

func (fake *ChaincodeStub) GetHistoryForKeyReturns(result1
shim.HistoryQueryIteratorInterface, result2 error) {
    fake.getHistoryForKeyMutex.Lock()
    defer fake.getHistoryForKeyMutex.Unlock()
    fake.GetHistoryForKeyStub = nil
    fake.getHistoryForKeyReturns = struct {
        result1 shim.HistoryQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetHistoryForKeyReturnsOnCall(i int, result1
shim.HistoryQueryIteratorInterface, result2 error) {
    fake.getHistoryForKeyMutex.Lock()
    defer fake.getHistoryForKeyMutex.Unlock()
    fake.GetHistoryForKeyStub = nil
    if fake.getHistoryForKeyReturnsOnCall == nil {
        fake.getHistoryForKeyReturnsOnCall = make(map[int]struct {
                result1 shim.HistoryQueryIteratorInterface
                result2 error
        })
    }
    fake.getHistoryForKeyReturnsOnCall[i] = struct {
        result1 shim.HistoryQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateData(arg1 string, arg2 string) ([]byte, error) {
    fake.getPrivateDataMutex.Lock()
    ret, specificReturn :=
fake.getPrivateDataReturnsOnCall[len(fake.getPrivateDataArgsForCall)]
    fake.getPrivateDataArgsForCall = append(fake.getPrivateDataArgsForCall, struct {
```

```
            arg1 string
            arg2 string
        }{arg1, arg2})
        fake.recordInvocation("GetPrivateData", []interface{}{arg1, arg2})
        fake.getPrivateDataMutex.Unlock()
        if fake.GetPrivateDataStub != nil {
            return fake.GetPrivateDataStub(arg1, arg2)
        }
        if specificReturn {
            return ret.result1, ret.result2
        }
        fakeReturns := fake.getPrivateDataReturns
        return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataCallCount() int {
        fake.getPrivateDataMutex.RLock()
        defer fake.getPrivateDataMutex.RUnlock()
        return len(fake.getPrivateDataArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataCalls(stub func(string, string) ([]byte, error)) {
        fake.getPrivateDataMutex.Lock()
        defer fake.getPrivateDataMutex.Unlock()
        fake.GetPrivateDataStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataArgsForCall(i int) (string, string) {
        fake.getPrivateDataMutex.RLock()
        defer fake.getPrivateDataMutex.RUnlock()
        argsForCall := fake.getPrivateDataArgsForCall[i]
        return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetPrivateDataReturns(result1 []byte, result2 error) {
        fake.getPrivateDataMutex.Lock()
        defer fake.getPrivateDataMutex.Unlock()
        fake.GetPrivateDataStub = nil
        fake.getPrivateDataReturns = struct {
            result1 []byte
            result2 error
        }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataReturnsOnCall(i int, result1 []byte, result2
error) {
        fake.getPrivateDataMutex.Lock()
        defer fake.getPrivateDataMutex.Unlock()
        fake.GetPrivateDataStub = nil
        if fake.getPrivateDataReturnsOnCall == nil {
            fake.getPrivateDataReturnsOnCall = make(map[int]struct {
                    result1 []byte
                    result2 error
            })
        }
        fake.getPrivateDataReturnsOnCall[i] = struct {
            result1 []byte
            result2 error
        }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKey(arg1 string, arg2 string,
arg3 []string) (shim.StateQueryIteratorInterface, error) {
        var arg3Copy []string
        if arg3 != nil {
            arg3Copy = make([]string, len(arg3))
            copy(arg3Copy, arg3)
        }
        fake.getPrivateDataByPartialCompositeKeyMutex.Lock()
        ret, specificReturn :=
fake.getPrivateDataByPartialCompositeKeyReturnsOnCall[len(fake.getPrivateDataByPartialCompo
siteKeyArgsForCall)]
        fake.getPrivateDataByPartialCompositeKeyArgsForCall =
```

```
append(fake.getPrivateDataByPartialCompositeKeyArgsForCall, struct {
        arg1 string
        arg2 string
        arg3 []string
    }{arg1, arg2, arg3Copy})
    fake.recordInvocation("GetPrivateDataByPartialCompositeKey", []interface{}{arg1, arg2,
arg3Copy})
    fake.getPrivateDataByPartialCompositeKeyMutex.Unlock()
    if fake.GetPrivateDataByPartialCompositeKeyStub != nil {
        return fake.GetPrivateDataByPartialCompositeKeyStub(arg1, arg2, arg3)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getPrivateDataByPartialCompositeKeyReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKeyCallCount() int {
    fake.getPrivateDataByPartialCompositeKeyMutex.RLock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.RUnlock()
    return len(fake.getPrivateDataByPartialCompositeKeyArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKeyCalls(stub func(string,
string, []string) (shim.StateQueryIteratorInterface, error)) {
    fake.getPrivateDataByPartialCompositeKeyMutex.Lock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.Unlock()
    fake.GetPrivateDataByPartialCompositeKeyStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKeyArgsForCall(i int) (string,
string, []string) {
    fake.getPrivateDataByPartialCompositeKeyMutex.RLock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.RUnlock()
    argsForCall := fake.getPrivateDataByPartialCompositeKeyArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKeyReturns(result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataByPartialCompositeKeyMutex.Lock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.Unlock()
    fake.GetPrivateDataByPartialCompositeKeyStub = nil
    fake.getPrivateDataByPartialCompositeKeyReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataByPartialCompositeKeyReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataByPartialCompositeKeyMutex.Lock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.Unlock()
    fake.GetPrivateDataByPartialCompositeKeyStub = nil
    if fake.getPrivateDataByPartialCompositeKeyReturnsOnCall == nil {
        fake.getPrivateDataByPartialCompositeKeyReturnsOnCall = make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 error
        })
    }
    fake.getPrivateDataByPartialCompositeKeyReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataByRange(arg1 string, arg2 string, arg3 string)
(shim.StateQueryIteratorInterface, error) {
    fake.getPrivateDataByRangeMutex.Lock()
    ret, specificReturn :=
fake.getPrivateDataByRangeReturnsOnCall[len(fake.getPrivateDataByRangeArgsForCall)]
```

```
    fake.getPrivateDataByRangeArgsForCall = append(fake.getPrivateDataByRangeArgsForCall,
struct {
        arg1 string
        arg2 string
        arg3 string
    }{arg1, arg2, arg3})
    fake.recordInvocation("GetPrivateDataByRange", []interface{}{arg1, arg2, arg3})
    fake.getPrivateDataByRangeMutex.Unlock()
    if fake.GetPrivateDataByRangeStub != nil {
        return fake.GetPrivateDataByRangeStub(arg1, arg2, arg3)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getPrivateDataByRangeReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataByRangeCallCount() int {
    fake.getPrivateDataByRangeMutex.RLock()
    defer fake.getPrivateDataByRangeMutex.RUnlock()
    return len(fake.getPrivateDataByRangeArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataByRangeCalls(stub func(string, string, string)
(shim.StateQueryIteratorInterface, error)) {
    fake.getPrivateDataByRangeMutex.Lock()
    defer fake.getPrivateDataByRangeMutex.Unlock()
    fake.GetPrivateDataByRangeStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataByRangeArgsForCall(i int) (string, string, string)
{
    fake.getPrivateDataByRangeMutex.RLock()
    defer fake.getPrivateDataByRangeMutex.RUnlock()
    argsForCall := fake.getPrivateDataByRangeArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
}

func (fake *ChaincodeStub) GetPrivateDataByRangeReturns(result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataByRangeMutex.Lock()
    defer fake.getPrivateDataByRangeMutex.Unlock()
    fake.GetPrivateDataByRangeStub = nil
    fake.getPrivateDataByRangeReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataByRangeReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataByRangeMutex.Lock()
    defer fake.getPrivateDataByRangeMutex.Unlock()
    fake.GetPrivateDataByRangeStub = nil
    if fake.getPrivateDataByRangeReturnsOnCall == nil {
        fake.getPrivateDataByRangeReturnsOnCall = make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 error
        })
    }
    fake.getPrivateDataByRangeReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataHash(arg1 string, arg2 string) ([]byte, error) {
    fake.getPrivateDataHashMutex.Lock()
    ret, specificReturn :=
fake.getPrivateDataHashReturnsOnCall[len(fake.getPrivateDataHashArgsForCall)]
    fake.getPrivateDataHashArgsForCall = append(fake.getPrivateDataHashArgsForCall, struct
{
```

```
            arg1 string
            arg2 string
        }{arg1, arg2})
        fake.recordInvocation("GetPrivateDataHash", []interface{}{arg1, arg2})
        fake.getPrivateDataHashMutex.Unlock()
        if fake.GetPrivateDataHashStub != nil {
            return fake.GetPrivateDataHashStub(arg1, arg2)
        }
        if specificReturn {
            return ret.result1, ret.result2
        }
        fakeReturns := fake.getPrivateDataHashReturns
        return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataHashCallCount() int {
        fake.getPrivateDataHashMutex.RLock()
        defer fake.getPrivateDataHashMutex.RUnlock()
        return len(fake.getPrivateDataHashArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataHashCalls(stub func(string, string) ([]byte,
error)) {
        fake.getPrivateDataHashMutex.Lock()
        defer fake.getPrivateDataHashMutex.Unlock()
        fake.GetPrivateDataHashStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataHashArgsForCall(i int) (string, string) {
        fake.getPrivateDataHashMutex.RLock()
        defer fake.getPrivateDataHashMutex.RUnlock()
        argsForCall := fake.getPrivateDataHashArgsForCall[i]
        return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetPrivateDataHashReturns(result1 []byte, result2 error) {
        fake.getPrivateDataHashMutex.Lock()
        defer fake.getPrivateDataHashMutex.Unlock()
        fake.GetPrivateDataHashStub = nil
        fake.getPrivateDataHashReturns = struct {
            result1 []byte
            result2 error
        }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataHashReturnsOnCall(i int, result1 []byte, result2
error) {
        fake.getPrivateDataHashMutex.Lock()
        defer fake.getPrivateDataHashMutex.Unlock()
        fake.GetPrivateDataHashStub = nil
        if fake.getPrivateDataHashReturnsOnCall == nil {
            fake.getPrivateDataHashReturnsOnCall = make(map[int]struct {
                    result1 []byte
                    result2 error
            })
        }
        fake.getPrivateDataHashReturnsOnCall[i] = struct {
            result1 []byte
            result2 error
        }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataQueryResult(arg1 string, arg2 string)
(shim.StateQueryIteratorInterface, error) {
        fake.getPrivateDataQueryResultMutex.Lock()
        ret, specificReturn :=
fake.getPrivateDataQueryResultReturnsOnCall[len(fake.getPrivateDataQueryResultArgsForCall)]
        fake.getPrivateDataQueryResultArgsForCall =
append(fake.getPrivateDataQueryResultArgsForCall, struct {
            arg1 string
            arg2 string
        }{arg1, arg2})
        fake.recordInvocation("GetPrivateDataQueryResult", []interface{}{arg1, arg2})
```

```
    fake.getPrivateDataQueryResultMutex.Unlock()
    if fake.GetPrivateDataQueryResultStub != nil {
        return fake.GetPrivateDataQueryResultStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getPrivateDataQueryResultReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataQueryResultCallCount() int {
    fake.getPrivateDataQueryResultMutex.RLock()
    defer fake.getPrivateDataQueryResultMutex.RUnlock()
    return len(fake.getPrivateDataQueryResultArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataQueryResultCalls(stub func(string, string)
(shim.StateQueryIteratorInterface, error)) {
    fake.getPrivateDataQueryResultMutex.Lock()
    defer fake.getPrivateDataQueryResultMutex.Unlock()
    fake.GetPrivateDataQueryResultStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataQueryResultArgsForCall(i int) (string, string) {
    fake.getPrivateDataQueryResultMutex.RLock()
    defer fake.getPrivateDataQueryResultMutex.RUnlock()
    argsForCall := fake.getPrivateDataQueryResultArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetPrivateDataQueryResultReturns(result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataQueryResultMutex.Lock()
    defer fake.getPrivateDataQueryResultMutex.Unlock()
    fake.GetPrivateDataQueryResultStub = nil
    fake.getPrivateDataQueryResultReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataQueryResultReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getPrivateDataQueryResultMutex.Lock()
    defer fake.getPrivateDataQueryResultMutex.Unlock()
    fake.GetPrivateDataQueryResultStub = nil
    if fake.getPrivateDataQueryResultReturnsOnCall == nil {
        fake.getPrivateDataQueryResultReturnsOnCall = make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 error
        })
    }
    fake.getPrivateDataQueryResultReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameter(arg1 string, arg2 string)
([]byte, error) {
    fake.getPrivateDataValidationParameterMutex.Lock()
    ret, specificReturn :=
fake.getPrivateDataValidationParameterReturnsOnCall[len(fake.getPrivateDataValidationParame
terArgsForCall)]
    fake.getPrivateDataValidationParameterArgsForCall =
append(fake.getPrivateDataValidationParameterArgsForCall, struct {
        arg1 string
        arg2 string
    }{arg1, arg2})
    fake.recordInvocation("GetPrivateDataValidationParameter", []interface{}{arg1, arg2})
    fake.getPrivateDataValidationParameterMutex.Unlock()
    if fake.GetPrivateDataValidationParameterStub != nil {
```

```
        return fake.GetPrivateDataValidationParameterStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getPrivateDataValidationParameterReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameterCallCount() int {
    fake.getPrivateDataValidationParameterMutex.RLock()
    defer fake.getPrivateDataValidationParameterMutex.RUnlock()
    return len(fake.getPrivateDataValidationParameterArgsForCall)
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameterCalls(stub func(string, string)
([]byte, error)) {
    fake.getPrivateDataValidationParameterMutex.Lock()
    defer fake.getPrivateDataValidationParameterMutex.Unlock()
    fake.GetPrivateDataValidationParameterStub = stub
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameterArgsForCall(i int) (string,
string) {
    fake.getPrivateDataValidationParameterMutex.RLock()
    defer fake.getPrivateDataValidationParameterMutex.RUnlock()
    argsForCall := fake.getPrivateDataValidationParameterArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameterReturns(result1 []byte, result2
error) {
    fake.getPrivateDataValidationParameterMutex.Lock()
    defer fake.getPrivateDataValidationParameterMutex.Unlock()
    fake.GetPrivateDataValidationParameterStub = nil
    fake.getPrivateDataValidationParameterReturns = struct {
        result1 []byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetPrivateDataValidationParameterReturnsOnCall(i int, result1
[]byte, result2 error) {
    fake.getPrivateDataValidationParameterMutex.Lock()
    defer fake.getPrivateDataValidationParameterMutex.Unlock()
    fake.GetPrivateDataValidationParameterStub = nil
    if fake.getPrivateDataValidationParameterReturnsOnCall == nil {
        fake.getPrivateDataValidationParameterReturnsOnCall = make(map[int]struct {
                result1 []byte
                result2 error
        })
    }
    fake.getPrivateDataValidationParameterReturnsOnCall[i] = struct {
        result1 []byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetQueryResult(arg1 string) (shim.StateQueryIteratorInterface,
error) {
    fake.getQueryResultMutex.Lock()
    ret, specificReturn :=
fake.getQueryResultReturnsOnCall[len(fake.getQueryResultArgsForCall)]
    fake.getQueryResultArgsForCall = append(fake.getQueryResultArgsForCall, struct {
        arg1 string
    }{arg1})
    fake.recordInvocation("GetQueryResult", []interface{}{arg1})
    fake.getQueryResultMutex.Unlock()
    if fake.GetQueryResultStub != nil {
        return fake.GetQueryResultStub(arg1)
    }
    if specificReturn {
        return ret.result1, ret.result2
```

```
    }
    fakeReturns := fake.getQueryResultReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetQueryResultCallCount() int {
    fake.getQueryResultMutex.RLock()
    defer fake.getQueryResultMutex.RUnlock()
    return len(fake.getQueryResultArgsForCall)
}

func (fake *ChaincodeStub) GetQueryResultCalls(stub func(string)
(shim.StateQueryIteratorInterface, error)) {
    fake.getQueryResultMutex.Lock()
    defer fake.getQueryResultMutex.Unlock()
    fake.GetQueryResultStub = stub
}

func (fake *ChaincodeStub) GetQueryResultArgsForCall(i int) string {
    fake.getQueryResultMutex.RLock()
    defer fake.getQueryResultMutex.RUnlock()
    argsForCall := fake.getQueryResultArgsForCall[i]
    return argsForCall.arg1
}

func (fake *ChaincodeStub) GetQueryResultReturns(result1 shim.StateQueryIteratorInterface,
result2 error) {
    fake.getQueryResultMutex.Lock()
    defer fake.getQueryResultMutex.Unlock()
    fake.GetQueryResultStub = nil
    fake.getQueryResultReturns = struct {
         result1 shim.StateQueryIteratorInterface
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetQueryResultReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getQueryResultMutex.Lock()
    defer fake.getQueryResultMutex.Unlock()
    fake.GetQueryResultStub = nil
    if fake.getQueryResultReturnsOnCall == nil {
         fake.getQueryResultReturnsOnCall = make(map[int]struct {
                 result1 shim.StateQueryIteratorInterface
                 result2 error
         })
    }
    fake.getQueryResultReturnsOnCall[i] = struct {
         result1 shim.StateQueryIteratorInterface
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetQueryResultWithPagination(arg1 string, arg2 int32, arg3
string) (shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error) {
    fake.getQueryResultWithPaginationMutex.Lock()
    ret, specificReturn :=
fake.getQueryResultWithPaginationReturnsOnCall[len(fake.getQueryResultWithPaginationArgsFor
Call)]
    fake.getQueryResultWithPaginationArgsForCall =
append(fake.getQueryResultWithPaginationArgsForCall, struct {
         arg1 string
         arg2 int32
         arg3 string
    }{arg1, arg2, arg3})
    fake.recordInvocation("GetQueryResultWithPagination", []interface{}{arg1, arg2, arg3})
    fake.getQueryResultWithPaginationMutex.Unlock()
    if fake.GetQueryResultWithPaginationStub != nil {
         return fake.GetQueryResultWithPaginationStub(arg1, arg2, arg3)
    }
    if specificReturn {
         return ret.result1, ret.result2, ret.result3
    }
```

```
    fakeReturns := fake.getQueryResultWithPaginationReturns
    return fakeReturns.result1, fakeReturns.result2, fakeReturns.result3
}

func (fake *ChaincodeStub) GetQueryResultWithPaginationCallCount() int {
    fake.getQueryResultWithPaginationMutex.RLock()
    defer fake.getQueryResultWithPaginationMutex.RUnlock()
    return len(fake.getQueryResultWithPaginationArgsForCall)
}

func (fake *ChaincodeStub) GetQueryResultWithPaginationCalls(stub func(string, int32,
string) (shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)) {
    fake.getQueryResultWithPaginationMutex.Lock()
    defer fake.getQueryResultWithPaginationMutex.Unlock()
    fake.GetQueryResultWithPaginationStub = stub
}

func (fake *ChaincodeStub) GetQueryResultWithPaginationArgsForCall(i int) (string, int32,
string) {
    fake.getQueryResultWithPaginationMutex.RLock()
    defer fake.getQueryResultWithPaginationMutex.RUnlock()
    argsForCall := fake.getQueryResultWithPaginationArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
}

func (fake *ChaincodeStub) GetQueryResultWithPaginationReturns(result1
shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3 error) {
    fake.getQueryResultWithPaginationMutex.Lock()
    defer fake.getQueryResultWithPaginationMutex.Unlock()
    fake.GetQueryResultWithPaginationStub = nil
    fake.getQueryResultWithPaginationReturns = struct {
         result1 shim.StateQueryIteratorInterface
         result2 *peer.QueryResponseMetadata
         result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetQueryResultWithPaginationReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3 error) {
    fake.getQueryResultWithPaginationMutex.Lock()
    defer fake.getQueryResultWithPaginationMutex.Unlock()
    fake.GetQueryResultWithPaginationStub = nil
    if fake.getQueryResultWithPaginationReturnsOnCall == nil {
         fake.getQueryResultWithPaginationReturnsOnCall = make(map[int]struct {
                 result1 shim.StateQueryIteratorInterface
                 result2 *peer.QueryResponseMetadata
                 result3 error
         })
    }
    fake.getQueryResultWithPaginationReturnsOnCall[i] = struct {
         result1 shim.StateQueryIteratorInterface
         result2 *peer.QueryResponseMetadata
         result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetSignedProposal() (*peer.SignedProposal, error) {
    fake.getSignedProposalMutex.Lock()
    ret, specificReturn :=
fake.getSignedProposalReturnsOnCall[len(fake.getSignedProposalArgsForCall)]
    fake.getSignedProposalArgsForCall = append(fake.getSignedProposalArgsForCall, struct {
    }{})
    fake.recordInvocation("GetSignedProposal", []interface{}{})
    fake.getSignedProposalMutex.Unlock()
    if fake.GetSignedProposalStub != nil {
         return fake.GetSignedProposalStub()
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.getSignedProposalReturns
    return fakeReturns.result1, fakeReturns.result2
}
```

```
func (fake *ChaincodeStub) GetSignedProposalCallCount() int {
    fake.getSignedProposalMutex.RLock()
    defer fake.getSignedProposalMutex.RUnlock()
    return len(fake.getSignedProposalArgsForCall)
}

func (fake *ChaincodeStub) GetSignedProposalCalls(stub func() (*peer.SignedProposal,
error)) {
    fake.getSignedProposalMutex.Lock()
    defer fake.getSignedProposalMutex.Unlock()
    fake.GetSignedProposalStub = stub
}

func (fake *ChaincodeStub) GetSignedProposalReturns(result1 *peer.SignedProposal, result2
error) {
    fake.getSignedProposalMutex.Lock()
    defer fake.getSignedProposalMutex.Unlock()
    fake.GetSignedProposalStub = nil
    fake.getSignedProposalReturns = struct {
         result1 *peer.SignedProposal
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetSignedProposalReturnsOnCall(i int, result1
*peer.SignedProposal, result2 error) {
    fake.getSignedProposalMutex.Lock()
    defer fake.getSignedProposalMutex.Unlock()
    fake.GetSignedProposalStub = nil
    if fake.getSignedProposalReturnsOnCall == nil {
         fake.getSignedProposalReturnsOnCall = make(map[int]struct {
                 result1 *peer.SignedProposal
                 result2 error
         })
    }
    fake.getSignedProposalReturnsOnCall[i] = struct {
         result1 *peer.SignedProposal
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetState(arg1 string) ([]byte, error) {
    fake.getStateMutex.Lock()
    ret, specificReturn := fake.getStateReturnsOnCall[len(fake.getStateArgsForCall)]
    fake.getStateArgsForCall = append(fake.getStateArgsForCall, struct {
         arg1 string
    }{arg1})
    fake.recordInvocation("GetState", []interface{}{arg1})
    fake.getStateMutex.Unlock()
    if fake.GetStateStub != nil {
         return fake.GetStateStub(arg1)
    }
    if specificReturn {
         return ret.result1, ret.result2
    }
    fakeReturns := fake.getStateReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetStateCallCount() int {
    fake.getStateMutex.RLock()
    defer fake.getStateMutex.RUnlock()
    return len(fake.getStateArgsForCall)
}

func (fake *ChaincodeStub) GetStateCalls(stub func(string) ([]byte, error)) {
    fake.getStateMutex.Lock()
    defer fake.getStateMutex.Unlock()
    fake.GetStateStub = stub
}

func (fake *ChaincodeStub) GetStateArgsForCall(i int) string {
```

```
    fake.getStateMutex.RLock()
    defer fake.getStateMutex.RUnlock()
    argsForCall := fake.getStateArgsForCall[i]
    return argsForCall.arg1
}

func (fake *ChaincodeStub) GetStateReturns(result1 []byte, result2 error) {
    fake.getStateMutex.Lock()
    defer fake.getStateMutex.Unlock()
    fake.GetStateStub = nil
    fake.getStateReturns = struct {
        result1 []byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateReturnsOnCall(i int, result1 []byte, result2 error) {
    fake.getStateMutex.Lock()
    defer fake.getStateMutex.Unlock()
    fake.GetStateStub = nil
    if fake.getStateReturnsOnCall == nil {
        fake.getStateReturnsOnCall = make(map[int]struct {
                result1 []byte
                result2 error
        })
    }
    fake.getStateReturnsOnCall[i] = struct {
        result1 []byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKey(arg1 string, arg2 []string)
(shim.StateQueryIteratorInterface, error) {
    var arg2Copy []string
    if arg2 != nil {
        arg2Copy = make([]string, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.getStateByPartialCompositeKeyMutex.Lock()
    ret, specificReturn :=
fake.getStateByPartialCompositeKeyReturnsOnCall[len(fake.getStateByPartialCompositeKeyArgsF
orCall)]
    fake.getStateByPartialCompositeKeyArgsForCall =
append(fake.getStateByPartialCompositeKeyArgsForCall, struct {
        arg1 string
        arg2 []string
    }{arg1, arg2Copy})
    fake.recordInvocation("GetStateByPartialCompositeKey", []interface{}{arg1, arg2Copy})
    fake.getStateByPartialCompositeKeyMutex.Unlock()
    if fake.GetStateByPartialCompositeKeyStub != nil {
        return fake.GetStateByPartialCompositeKeyStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getStateByPartialCompositeKeyReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyCallCount() int {
    fake.getStateByPartialCompositeKeyMutex.RLock()
    defer fake.getStateByPartialCompositeKeyMutex.RUnlock()
    return len(fake.getStateByPartialCompositeKeyArgsForCall)
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyCalls(stub func(string, []string)
(shim.StateQueryIteratorInterface, error)) {
    fake.getStateByPartialCompositeKeyMutex.Lock()
    defer fake.getStateByPartialCompositeKeyMutex.Unlock()
    fake.GetStateByPartialCompositeKeyStub = stub
}
```

```
func (fake *ChaincodeStub) GetStateByPartialCompositeKeyArgsForCall(i int) (string,
[]string) {
    fake.getStateByPartialCompositeKeyMutex.RLock()
    defer fake.getStateByPartialCompositeKeyMutex.RUnlock()
    argsForCall := fake.getStateByPartialCompositeKeyArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyReturns(result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getStateByPartialCompositeKeyMutex.Lock()
    defer fake.getStateByPartialCompositeKeyMutex.Unlock()
    fake.GetStateByPartialCompositeKeyStub = nil
    fake.getStateByPartialCompositeKeyReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getStateByPartialCompositeKeyMutex.Lock()
    defer fake.getStateByPartialCompositeKeyMutex.Unlock()
    fake.GetStateByPartialCompositeKeyStub = nil
    if fake.getStateByPartialCompositeKeyReturnsOnCall == nil {
        fake.getStateByPartialCompositeKeyReturnsOnCall = make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 error
        })
    }
    fake.getStateByPartialCompositeKeyReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPagination(arg1 string, arg2
[]string, arg3 int32, arg4 string) (shim.StateQueryIteratorInterface,
*peer.QueryResponseMetadata, error) {
    var arg2Copy []string
    if arg2 != nil {
        arg2Copy = make([]string, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.getStateByPartialCompositeKeyWithPaginationMutex.Lock()
    ret, specificReturn :=
fake.getStateByPartialCompositeKeyWithPaginationReturnsOnCall[len(fake.getStateByPartialCom
positeKeyWithPaginationArgsForCall)]
    fake.getStateByPartialCompositeKeyWithPaginationArgsForCall =
append(fake.getStateByPartialCompositeKeyWithPaginationArgsForCall, struct {
        arg1 string
        arg2 []string
        arg3 int32
        arg4 string
    }{arg1, arg2Copy, arg3, arg4})
    fake.recordInvocation("GetStateByPartialCompositeKeyWithPagination",
[]interface{}{arg1, arg2Copy, arg3, arg4})
    fake.getStateByPartialCompositeKeyWithPaginationMutex.Unlock()
    if fake.GetStateByPartialCompositeKeyWithPaginationStub != nil {
        return fake.GetStateByPartialCompositeKeyWithPaginationStub(arg1, arg2, arg3,
arg4)
    }
    if specificReturn {
        return ret.result1, ret.result2, ret.result3
    }
    fakeReturns := fake.getStateByPartialCompositeKeyWithPaginationReturns
    return fakeReturns.result1, fakeReturns.result2, fakeReturns.result3
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPaginationCallCount() int {
    fake.getStateByPartialCompositeKeyWithPaginationMutex.RLock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.RUnlock()
```

```
    return len(fake.getStateByPartialCompositeKeyWithPaginationArgsForCall)
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPaginationCalls(stub
func(string, []string, int32, string) (shim.StateQueryIteratorInterface,
*peer.QueryResponseMetadata, error)) {
    fake.getStateByPartialCompositeKeyWithPaginationMutex.Lock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.Unlock()
    fake.GetStateByPartialCompositeKeyWithPaginationStub = stub
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPaginationArgsForCall(i int)
(string, []string, int32, string) {
    fake.getStateByPartialCompositeKeyWithPaginationMutex.RLock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.RUnlock()
    argsForCall := fake.getStateByPartialCompositeKeyWithPaginationArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3, argsForCall.arg4
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPaginationReturns(result1
shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3 error) {
    fake.getStateByPartialCompositeKeyWithPaginationMutex.Lock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.Unlock()
    fake.GetStateByPartialCompositeKeyWithPaginationStub = nil
    fake.getStateByPartialCompositeKeyWithPaginationReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetStateByPartialCompositeKeyWithPaginationReturnsOnCall(i int,
result1 shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3
error) {
    fake.getStateByPartialCompositeKeyWithPaginationMutex.Lock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.Unlock()
    fake.GetStateByPartialCompositeKeyWithPaginationStub = nil
    if fake.getStateByPartialCompositeKeyWithPaginationReturnsOnCall == nil {
        fake.getStateByPartialCompositeKeyWithPaginationReturnsOnCall =
make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 *peer.QueryResponseMetadata
                result3 error
        })
    }
    fake.getStateByPartialCompositeKeyWithPaginationReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetStateByRange(arg1 string, arg2 string)
(shim.StateQueryIteratorInterface, error) {
    fake.getStateByRangeMutex.Lock()
    ret, specificReturn :=
fake.getStateByRangeReturnsOnCall[len(fake.getStateByRangeArgsForCall)]
    fake.getStateByRangeArgsForCall = append(fake.getStateByRangeArgsForCall, struct {
        arg1 string
        arg2 string
    }{arg1, arg2})
    fake.recordInvocation("GetStateByRange", []interface{}{arg1, arg2})
    fake.getStateByRangeMutex.Unlock()
    if fake.GetStateByRangeStub != nil {
        return fake.GetStateByRangeStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getStateByRangeReturns
    return fakeReturns.result1, fakeReturns.result2
}
```

```
func (fake *ChaincodeStub) GetStateByRangeCallCount() int {
    fake.getStateByRangeMutex.RLock()
    defer fake.getStateByRangeMutex.RUnlock()
    return len(fake.getStateByRangeArgsForCall)
}

func (fake *ChaincodeStub) GetStateByRangeCalls(stub func(string, string)
(shim.StateQueryIteratorInterface, error)) {
    fake.getStateByRangeMutex.Lock()
    defer fake.getStateByRangeMutex.Unlock()
    fake.GetStateByRangeStub = stub
}

func (fake *ChaincodeStub) GetStateByRangeArgsForCall(i int) (string, string) {
    fake.getStateByRangeMutex.RLock()
    defer fake.getStateByRangeMutex.RUnlock()
    argsForCall := fake.getStateByRangeArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) GetStateByRangeReturns(result1 shim.StateQueryIteratorInterface,
result2 error) {
    fake.getStateByRangeMutex.Lock()
    defer fake.getStateByRangeMutex.Unlock()
    fake.GetStateByRangeStub = nil
    fake.getStateByRangeReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateByRangeReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 error) {
    fake.getStateByRangeMutex.Lock()
    defer fake.getStateByRangeMutex.Unlock()
    fake.GetStateByRangeStub = nil
    if fake.getStateByRangeReturnsOnCall == nil {
        fake.getStateByRangeReturnsOnCall = make(map[int]struct {
            result1 shim.StateQueryIteratorInterface
            result2 error
        })
    }
    fake.getStateByRangeReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateByRangeWithPagination(arg1 string, arg2 string, arg3
int32, arg4 string) (shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)
{
    fake.getStateByRangeWithPaginationMutex.Lock()
    ret, specificReturn :=
fake.getStateByRangeWithPaginationReturnsOnCall[len(fake.getStateByRangeWithPaginationArgsF
orCall)]
    fake.getStateByRangeWithPaginationArgsForCall =
append(fake.getStateByRangeWithPaginationArgsForCall, struct {
        arg1 string
        arg2 string
        arg3 int32
        arg4 string
    }{arg1, arg2, arg3, arg4})
    fake.recordInvocation("GetStateByRangeWithPagination", []interface{}{arg1, arg2, arg3,
arg4})
    fake.getStateByRangeWithPaginationMutex.Unlock()
    if fake.GetStateByRangeWithPaginationStub != nil {
        return fake.GetStateByRangeWithPaginationStub(arg1, arg2, arg3, arg4)
    }
    if specificReturn {
        return ret.result1, ret.result2, ret.result3
    }
    fakeReturns := fake.getStateByRangeWithPaginationReturns
    return fakeReturns.result1, fakeReturns.result2, fakeReturns.result3
```

```
}

func (fake *ChaincodeStub) GetStateByRangeWithPaginationCallCount() int {
    fake.getStateByRangeWithPaginationMutex.RLock()
    defer fake.getStateByRangeWithPaginationMutex.RUnlock()
    return len(fake.getStateByRangeWithPaginationArgsForCall)
}

func (fake *ChaincodeStub) GetStateByRangeWithPaginationCalls(stub func(string, string,
int32, string) (shim.StateQueryIteratorInterface, *peer.QueryResponseMetadata, error)) {
    fake.getStateByRangeWithPaginationMutex.Lock()
    defer fake.getStateByRangeWithPaginationMutex.Unlock()
    fake.GetStateByRangeWithPaginationStub = stub
}

func (fake *ChaincodeStub) GetStateByRangeWithPaginationArgsForCall(i int) (string, string,
int32, string) {
    fake.getStateByRangeWithPaginationMutex.RLock()
    defer fake.getStateByRangeWithPaginationMutex.RUnlock()
    argsForCall := fake.getStateByRangeWithPaginationArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3, argsForCall.arg4
}

func (fake *ChaincodeStub) GetStateByRangeWithPaginationReturns(result1
shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3 error) {
    fake.getStateByRangeWithPaginationMutex.Lock()
    defer fake.getStateByRangeWithPaginationMutex.Unlock()
    fake.GetStateByRangeWithPaginationStub = nil
    fake.getStateByRangeWithPaginationReturns = struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetStateByRangeWithPaginationReturnsOnCall(i int, result1
shim.StateQueryIteratorInterface, result2 *peer.QueryResponseMetadata, result3 error) {
    fake.getStateByRangeWithPaginationMutex.Lock()
    defer fake.getStateByRangeWithPaginationMutex.Unlock()
    fake.GetStateByRangeWithPaginationStub = nil
    if fake.getStateByRangeWithPaginationReturnsOnCall == nil {
        fake.getStateByRangeWithPaginationReturnsOnCall = make(map[int]struct {
                result1 shim.StateQueryIteratorInterface
                result2 *peer.QueryResponseMetadata
                result3 error
        })
    }
    fake.getStateByRangeWithPaginationReturnsOnCall[i] = struct {
        result1 shim.StateQueryIteratorInterface
        result2 *peer.QueryResponseMetadata
        result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) GetStateValidationParameter(arg1 string) ([]byte, error) {
    fake.getStateValidationParameterMutex.Lock()
    ret, specificReturn :=
fake.getStateValidationParameterReturnsOnCall[len(fake.getStateValidationParameterArgsForCa
ll)]
    fake.getStateValidationParameterArgsForCall =
append(fake.getStateValidationParameterArgsForCall, struct {
        arg1 string
    }{arg1})
    fake.recordInvocation("GetStateValidationParameter", []interface{}{arg1})
    fake.getStateValidationParameterMutex.Unlock()
    if fake.GetStateValidationParameterStub != nil {
        return fake.GetStateValidationParameterStub(arg1)
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getStateValidationParameterReturns
    return fakeReturns.result1, fakeReturns.result2
```

```
}

func (fake *ChaincodeStub) GetStateValidationParameterCallCount() int {
    fake.getStateValidationParameterMutex.RLock()
    defer fake.getStateValidationParameterMutex.RUnlock()
    return len(fake.getStateValidationParameterArgsForCall)
}

func (fake *ChaincodeStub) GetStateValidationParameterCalls(stub func(string) ([]byte,
error)) {
    fake.getStateValidationParameterMutex.Lock()
    defer fake.getStateValidationParameterMutex.Unlock()
    fake.GetStateValidationParameterStub = stub
}

func (fake *ChaincodeStub) GetStateValidationParameterArgsForCall(i int) string {
    fake.getStateValidationParameterMutex.RLock()
    defer fake.getStateValidationParameterMutex.RUnlock()
    argsForCall := fake.getStateValidationParameterArgsForCall[i]
    return argsForCall.arg1
}

func (fake *ChaincodeStub) GetStateValidationParameterReturns(result1 []byte, result2
error) {
    fake.getStateValidationParameterMutex.Lock()
    defer fake.getStateValidationParameterMutex.Unlock()
    fake.GetStateValidationParameterStub = nil
    fake.getStateValidationParameterReturns = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStateValidationParameterReturnsOnCall(i int, result1 []byte,
result2 error) {
    fake.getStateValidationParameterMutex.Lock()
    defer fake.getStateValidationParameterMutex.Unlock()
    fake.GetStateValidationParameterStub = nil
    if fake.getStateValidationParameterReturnsOnCall == nil {
         fake.getStateValidationParameterReturnsOnCall = make(map[int]struct {
                 result1 []byte
                 result2 error
         })
    }
    fake.getStateValidationParameterReturnsOnCall[i] = struct {
         result1 []byte
         result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetStringArgs() []string {
    fake.getStringArgsMutex.Lock()
    ret, specificReturn :=
fake.getStringArgsReturnsOnCall[len(fake.getStringArgsArgsForCall)]
    fake.getStringArgsArgsForCall = append(fake.getStringArgsArgsForCall, struct {
    }{})
    fake.recordInvocation("GetStringArgs", []interface{}{})
    fake.getStringArgsMutex.Unlock()
    if fake.GetStringArgsStub != nil {
         return fake.GetStringArgsStub()
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.getStringArgsReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) GetStringArgsCallCount() int {
    fake.getStringArgsMutex.RLock()
    defer fake.getStringArgsMutex.RUnlock()
    return len(fake.getStringArgsArgsForCall)
}
```

```
func (fake *ChaincodeStub) GetStringArgsCalls(stub func() []string) {
    fake.getStringArgsMutex.Lock()
    defer fake.getStringArgsMutex.Unlock()
    fake.GetStringArgsStub = stub
}

func (fake *ChaincodeStub) GetStringArgsReturns(result1 []string) {
    fake.getStringArgsMutex.Lock()
    defer fake.getStringArgsMutex.Unlock()
    fake.GetStringArgsStub = nil
    fake.getStringArgsReturns = struct {
        result1 []string
    }{result1}
}

func (fake *ChaincodeStub) GetStringArgsReturnsOnCall(i int, result1 []string) {
    fake.getStringArgsMutex.Lock()
    defer fake.getStringArgsMutex.Unlock()
    fake.GetStringArgsStub = nil
    if fake.getStringArgsReturnsOnCall == nil {
        fake.getStringArgsReturnsOnCall = make(map[int]struct {
                result1 []string
        })
    }
    fake.getStringArgsReturnsOnCall[i] = struct {
        result1 []string
    }{result1}
}

func (fake *ChaincodeStub) GetTransient() (map[string][]byte, error) {
    fake.getTransientMutex.Lock()
    ret, specificReturn :=
fake.getTransientReturnsOnCall[len(fake.getTransientArgsForCall)]
    fake.getTransientArgsForCall = append(fake.getTransientArgsForCall, struct {
    }{})
    fake.recordInvocation("GetTransient", []interface{}{})
    fake.getTransientMutex.Unlock()
    if fake.GetTransientStub != nil {
        return fake.GetTransientStub()
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getTransientReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetTransientCallCount() int {
    fake.getTransientMutex.RLock()
    defer fake.getTransientMutex.RUnlock()
    return len(fake.getTransientArgsForCall)
}

func (fake *ChaincodeStub) GetTransientCalls(stub func() (map[string][]byte, error)) {
    fake.getTransientMutex.Lock()
    defer fake.getTransientMutex.Unlock()
    fake.GetTransientStub = stub
}

func (fake *ChaincodeStub) GetTransientReturns(result1 map[string][]byte, result2 error) {
    fake.getTransientMutex.Lock()
    defer fake.getTransientMutex.Unlock()
    fake.GetTransientStub = nil
    fake.getTransientReturns = struct {
        result1 map[string][]byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetTransientReturnsOnCall(i int, result1 map[string][]byte,
result2 error) {
    fake.getTransientMutex.Lock()
```

```
    defer fake.getTransientMutex.Unlock()
    fake.GetTransientStub = nil
    if fake.getTransientReturnsOnCall == nil {
        fake.getTransientReturnsOnCall = make(map[int]struct {
            result1 map[string][]byte
            result2 error
        })
    }
    fake.getTransientReturnsOnCall[i] = struct {
        result1 map[string][]byte
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetTxID() string {
    fake.getTxIDMutex.Lock()
    ret, specificReturn := fake.getTxIDReturnsOnCall[len(fake.getTxIDArgsForCall)]
    fake.getTxIDArgsForCall = append(fake.getTxIDArgsForCall, struct {
    }{})
    fake.recordInvocation("GetTxID", []interface{}{})
    fake.getTxIDMutex.Unlock()
    if fake.GetTxIDStub != nil {
        return fake.GetTxIDStub()
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.getTxIDReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) GetTxIDCallCount() int {
    fake.getTxIDMutex.RLock()
    defer fake.getTxIDMutex.RUnlock()
    return len(fake.getTxIDArgsForCall)
}

func (fake *ChaincodeStub) GetTxIDCalls(stub func() string) {
    fake.getTxIDMutex.Lock()
    defer fake.getTxIDMutex.Unlock()
    fake.GetTxIDStub = stub
}

func (fake *ChaincodeStub) GetTxIDReturns(result1 string) {
    fake.getTxIDMutex.Lock()
    defer fake.getTxIDMutex.Unlock()
    fake.GetTxIDStub = nil
    fake.getTxIDReturns = struct {
        result1 string
    }{result1}
}

func (fake *ChaincodeStub) GetTxIDReturnsOnCall(i int, result1 string) {
    fake.getTxIDMutex.Lock()
    defer fake.getTxIDMutex.Unlock()
    fake.GetTxIDStub = nil
    if fake.getTxIDReturnsOnCall == nil {
        fake.getTxIDReturnsOnCall = make(map[int]struct {
            result1 string
        })
    }
    fake.getTxIDReturnsOnCall[i] = struct {
        result1 string
    }{result1}
}

func (fake *ChaincodeStub) GetTxTimestamp() (*timestamppb.Timestamp, error) {
    fake.getTxTimestampMutex.Lock()
    ret, specificReturn :=
fake.getTxTimestampReturnsOnCall[len(fake.getTxTimestampArgsForCall)]
    fake.getTxTimestampArgsForCall = append(fake.getTxTimestampArgsForCall, struct {
    }{})
    fake.recordInvocation("GetTxTimestamp", []interface{}{})
```

```
    fake.getTxTimestampMutex.Unlock()
    if fake.GetTxTimestampStub != nil {
        return fake.GetTxTimestampStub()
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.getTxTimestampReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *ChaincodeStub) GetTxTimestampCallCount() int {
    fake.getTxTimestampMutex.RLock()
    defer fake.getTxTimestampMutex.RUnlock()
    return len(fake.getTxTimestampArgsForCall)
}

func (fake *ChaincodeStub) GetTxTimestampCalls(stub func() (*timestamppb.Timestamp, error))
{
    fake.getTxTimestampMutex.Lock()
    defer fake.getTxTimestampMutex.Unlock()
    fake.GetTxTimestampStub = stub
}

func (fake *ChaincodeStub) GetTxTimestampReturns(result1 *timestamppb.Timestamp, result2
error) {
    fake.getTxTimestampMutex.Lock()
    defer fake.getTxTimestampMutex.Unlock()
    fake.GetTxTimestampStub = nil
    fake.getTxTimestampReturns = struct {
        result1 *timestamppb.Timestamp
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) GetTxTimestampReturnsOnCall(i int, result1
*timestamppb.Timestamp, result2 error) {
    fake.getTxTimestampMutex.Lock()
    defer fake.getTxTimestampMutex.Unlock()
    fake.GetTxTimestampStub = nil
    if fake.getTxTimestampReturnsOnCall == nil {
        fake.getTxTimestampReturnsOnCall = make(map[int]struct {
                result1 *timestamppb.Timestamp
                result2 error
        })
    }
    fake.getTxTimestampReturnsOnCall[i] = struct {
        result1 *timestamppb.Timestamp
        result2 error
    }{result1, result2}
}

func (fake *ChaincodeStub) InvokeChaincode(arg1 string, arg2 [][]byte, arg3 string)
peer.Response {
    var arg2Copy [][]byte
    if arg2 != nil {
        arg2Copy = make([][]byte, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.invokeChaincodeMutex.Lock()
    ret, specificReturn :=
fake.invokeChaincodeReturnsOnCall[len(fake.invokeChaincodeArgsForCall)]
    fake.invokeChaincodeArgsForCall = append(fake.invokeChaincodeArgsForCall, struct {
        arg1 string
        arg2 [][]byte
        arg3 string
    }{arg1, arg2Copy, arg3})
    fake.recordInvocation("InvokeChaincode", []interface{}{arg1, arg2Copy, arg3})
    fake.invokeChaincodeMutex.Unlock()
    if fake.InvokeChaincodeStub != nil {
        return fake.InvokeChaincodeStub(arg1, arg2, arg3)
    }
    if specificReturn {
```

```
            return ret.result1
        }
        fakeReturns := fake.invokeChaincodeReturns
        return fakeReturns.result1
    }

    func (fake *ChaincodeStub) InvokeChaincodeCallCount() int {
        fake.invokeChaincodeMutex.RLock()
        defer fake.invokeChaincodeMutex.RUnlock()
        return len(fake.invokeChaincodeArgsForCall)
    }

    func (fake *ChaincodeStub) InvokeChaincodeCalls(stub func(string, [][]byte, string)
    peer.Response) {
        fake.invokeChaincodeMutex.Lock()
        defer fake.invokeChaincodeMutex.Unlock()
        fake.InvokeChaincodeStub = stub
    }

    func (fake *ChaincodeStub) InvokeChaincodeArgsForCall(i int) (string, [][]byte, string) {
        fake.invokeChaincodeMutex.RLock()
        defer fake.invokeChaincodeMutex.RUnlock()
        argsForCall := fake.invokeChaincodeArgsForCall[i]
        return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
    }

    func (fake *ChaincodeStub) InvokeChaincodeReturns(result1 peer.Response) {
        fake.invokeChaincodeMutex.Lock()
        defer fake.invokeChaincodeMutex.Unlock()
        fake.InvokeChaincodeStub = nil
        fake.invokeChaincodeReturns = struct {
            result1 peer.Response
        }{result1}
    }

    func (fake *ChaincodeStub) InvokeChaincodeReturnsOnCall(i int, result1 peer.Response) {
        fake.invokeChaincodeMutex.Lock()
        defer fake.invokeChaincodeMutex.Unlock()
        fake.InvokeChaincodeStub = nil
        if fake.invokeChaincodeReturnsOnCall == nil {
            fake.invokeChaincodeReturnsOnCall = make(map[int]struct {
                    result1 peer.Response
            })
        }
        fake.invokeChaincodeReturnsOnCall[i] = struct {
            result1 peer.Response
        }{result1}
    }

    func (fake *ChaincodeStub) PurgePrivateData(arg1 string, arg2 string) error {
        fake.purgePrivateDataMutex.Lock()
        ret, specificReturn :=
    fake.purgePrivateDataReturnsOnCall[len(fake.purgePrivateDataArgsForCall)]
        fake.purgePrivateDataArgsForCall = append(fake.purgePrivateDataArgsForCall, struct {
            arg1 string
            arg2 string
        }{arg1, arg2})
        fake.recordInvocation("PurgePrivateData", []interface{}{arg1, arg2})
        fake.purgePrivateDataMutex.Unlock()
        if fake.PurgePrivateDataStub != nil {
            return fake.PurgePrivateDataStub(arg1, arg2)
        }
        if specificReturn {
            return ret.result1
        }
        fakeReturns := fake.purgePrivateDataReturns
        return fakeReturns.result1
    }

    func (fake *ChaincodeStub) PurgePrivateDataCallCount() int {
        fake.purgePrivateDataMutex.RLock()
        defer fake.purgePrivateDataMutex.RUnlock()
        return len(fake.purgePrivateDataArgsForCall)
```

```
}

func (fake *ChaincodeStub) PurgePrivateDataCalls(stub func(string, string) error) {
    fake.purgePrivateDataMutex.Lock()
    defer fake.purgePrivateDataMutex.Unlock()
    fake.PurgePrivateDataStub = stub
}

func (fake *ChaincodeStub) PurgePrivateDataArgsForCall(i int) (string, string) {
    fake.purgePrivateDataMutex.RLock()
    defer fake.purgePrivateDataMutex.RUnlock()
    argsForCall := fake.purgePrivateDataArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) PurgePrivateDataReturns(result1 error) {
    fake.purgePrivateDataMutex.Lock()
    defer fake.purgePrivateDataMutex.Unlock()
    fake.PurgePrivateDataStub = nil
    fake.purgePrivateDataReturns = struct {
         result1 error
    }{result1}
}

func (fake *ChaincodeStub) PurgePrivateDataReturnsOnCall(i int, result1 error) {
    fake.purgePrivateDataMutex.Lock()
    defer fake.purgePrivateDataMutex.Unlock()
    fake.PurgePrivateDataStub = nil
    if fake.purgePrivateDataReturnsOnCall == nil {
         fake.purgePrivateDataReturnsOnCall = make(map[int]struct {
                 result1 error
         })
    }
    fake.purgePrivateDataReturnsOnCall[i] = struct {
         result1 error
    }{result1}
}

func (fake *ChaincodeStub) PutPrivateData(arg1 string, arg2 string, arg3 []byte) error {
    var arg3Copy []byte
    if arg3 != nil {
         arg3Copy = make([]byte, len(arg3))
         copy(arg3Copy, arg3)
    }
    fake.putPrivateDataMutex.Lock()
    ret, specificReturn :=
fake.putPrivateDataReturnsOnCall[len(fake.putPrivateDataArgsForCall)]
    fake.putPrivateDataArgsForCall = append(fake.putPrivateDataArgsForCall, struct {
         arg1 string
         arg2 string
         arg3 []byte
    }{arg1, arg2, arg3Copy})
    fake.recordInvocation("PutPrivateData", []interface{}{arg1, arg2, arg3Copy})
    fake.putPrivateDataMutex.Unlock()
    if fake.PutPrivateDataStub != nil {
         return fake.PutPrivateDataStub(arg1, arg2, arg3)
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.putPrivateDataReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) PutPrivateDataCallCount() int {
    fake.putPrivateDataMutex.RLock()
    defer fake.putPrivateDataMutex.RUnlock()
    return len(fake.putPrivateDataArgsForCall)
}

func (fake *ChaincodeStub) PutPrivateDataCalls(stub func(string, string, []byte) error) {
    fake.putPrivateDataMutex.Lock()
    defer fake.putPrivateDataMutex.Unlock()
```

```
    fake.PutPrivateDataStub = stub
}

func (fake *ChaincodeStub) PutPrivateDataArgsForCall(i int) (string, string, []byte) {
    fake.putPrivateDataMutex.RLock()
    defer fake.putPrivateDataMutex.RUnlock()
    argsForCall := fake.putPrivateDataArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
}

func (fake *ChaincodeStub) PutPrivateDataReturns(result1 error) {
    fake.putPrivateDataMutex.Lock()
    defer fake.putPrivateDataMutex.Unlock()
    fake.PutPrivateDataStub = nil
    fake.putPrivateDataReturns = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) PutPrivateDataReturnsOnCall(i int, result1 error) {
    fake.putPrivateDataMutex.Lock()
    defer fake.putPrivateDataMutex.Unlock()
    fake.PutPrivateDataStub = nil
    if fake.putPrivateDataReturnsOnCall == nil {
        fake.putPrivateDataReturnsOnCall = make(map[int]struct {
            result1 error
        })
    }
    fake.putPrivateDataReturnsOnCall[i] = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) PutState(arg1 string, arg2 []byte) error {
    var arg2Copy []byte
    if arg2 != nil {
        arg2Copy = make([]byte, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.putStateMutex.Lock()
    ret, specificReturn := fake.putStateReturnsOnCall[len(fake.putStateArgsForCall)]
    fake.putStateArgsForCall = append(fake.putStateArgsForCall, struct {
        arg1 string
        arg2 []byte
    }{arg1, arg2Copy})
    fake.recordInvocation("PutState", []interface{}{arg1, arg2Copy})
    fake.putStateMutex.Unlock()
    if fake.PutStateStub != nil {
        return fake.PutStateStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.putStateReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) PutStateCallCount() int {
    fake.putStateMutex.RLock()
    defer fake.putStateMutex.RUnlock()
    return len(fake.putStateArgsForCall)
}

func (fake *ChaincodeStub) PutStateCalls(stub func(string, []byte) error) {
    fake.putStateMutex.Lock()
    defer fake.putStateMutex.Unlock()
    fake.PutStateStub = stub
}

func (fake *ChaincodeStub) PutStateArgsForCall(i int) (string, []byte) {
    fake.putStateMutex.RLock()
    defer fake.putStateMutex.RUnlock()
    argsForCall := fake.putStateArgsForCall[i]
```

```
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) PutStateReturns(result1 error) {
    fake.putStateMutex.Lock()
    defer fake.putStateMutex.Unlock()
    fake.PutStateStub = nil
    fake.putStateReturns = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) PutStateReturnsOnCall(i int, result1 error) {
    fake.putStateMutex.Lock()
    defer fake.putStateMutex.Unlock()
    fake.PutStateStub = nil
    if fake.putStateReturnsOnCall == nil {
        fake.putStateReturnsOnCall = make(map[int]struct {
                result1 error
        })
    }
    fake.putStateReturnsOnCall[i] = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) SetEvent(arg1 string, arg2 []byte) error {
    var arg2Copy []byte
    if arg2 != nil {
        arg2Copy = make([]byte, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.setEventMutex.Lock()
    ret, specificReturn := fake.setEventReturnsOnCall[len(fake.setEventArgsForCall)]
    fake.setEventArgsForCall = append(fake.setEventArgsForCall, struct {
        arg1 string
        arg2 []byte
    }{arg1, arg2Copy})
    fake.recordInvocation("SetEvent", []interface{}{arg1, arg2Copy})
    fake.setEventMutex.Unlock()
    if fake.SetEventStub != nil {
        return fake.SetEventStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.setEventReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) SetEventCallCount() int {
    fake.setEventMutex.RLock()
    defer fake.setEventMutex.RUnlock()
    return len(fake.setEventArgsForCall)
}

func (fake *ChaincodeStub) SetEventCalls(stub func(string, []byte) error) {
    fake.setEventMutex.Lock()
    defer fake.setEventMutex.Unlock()
    fake.SetEventStub = stub
}

func (fake *ChaincodeStub) SetEventArgsForCall(i int) (string, []byte) {
    fake.setEventMutex.RLock()
    defer fake.setEventMutex.RUnlock()
    argsForCall := fake.setEventArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) SetEventReturns(result1 error) {
    fake.setEventMutex.Lock()
    defer fake.setEventMutex.Unlock()
    fake.SetEventStub = nil
```

```
    fake.setEventReturns = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) SetEventReturnsOnCall(i int, result1 error) {
    fake.setEventMutex.Lock()
    defer fake.setEventMutex.Unlock()
    fake.SetEventStub = nil
    if fake.setEventReturnsOnCall == nil {
        fake.setEventReturnsOnCall = make(map[int]struct {
                result1 error
        })
    }
    fake.setEventReturnsOnCall[i] = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameter(arg1 string, arg2 string, arg3
[]byte) error {
    var arg3Copy []byte
    if arg3 != nil {
        arg3Copy = make([]byte, len(arg3))
        copy(arg3Copy, arg3)
    }
    fake.setPrivateDataValidationParameterMutex.Lock()
    ret, specificReturn :=
fake.setPrivateDataValidationParameterReturnsOnCall[len(fake.setPrivateDataValidationParame
terArgsForCall)]
    fake.setPrivateDataValidationParameterArgsForCall =
append(fake.setPrivateDataValidationParameterArgsForCall, struct {
        arg1 string
        arg2 string
        arg3 []byte
    }{arg1, arg2, arg3Copy})
    fake.recordInvocation("SetPrivateDataValidationParameter", []interface{}{arg1, arg2,
arg3Copy})
    fake.setPrivateDataValidationParameterMutex.Unlock()
    if fake.SetPrivateDataValidationParameterStub != nil {
        return fake.SetPrivateDataValidationParameterStub(arg1, arg2, arg3)
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.setPrivateDataValidationParameterReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameterCallCount() int {
    fake.setPrivateDataValidationParameterMutex.RLock()
    defer fake.setPrivateDataValidationParameterMutex.RUnlock()
    return len(fake.setPrivateDataValidationParameterArgsForCall)
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameterCalls(stub func(string, string,
[]byte) error) {
    fake.setPrivateDataValidationParameterMutex.Lock()
    defer fake.setPrivateDataValidationParameterMutex.Unlock()
    fake.SetPrivateDataValidationParameterStub = stub
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameterArgsForCall(i int) (string,
string, []byte) {
    fake.setPrivateDataValidationParameterMutex.RLock()
    defer fake.setPrivateDataValidationParameterMutex.RUnlock()
    argsForCall := fake.setPrivateDataValidationParameterArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2, argsForCall.arg3
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameterReturns(result1 error) {
    fake.setPrivateDataValidationParameterMutex.Lock()
    defer fake.setPrivateDataValidationParameterMutex.Unlock()
```

```
    fake.SetPrivateDataValidationParameterStub = nil
    fake.setPrivateDataValidationParameterReturns = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) SetPrivateDataValidationParameterReturnsOnCall(i int, result1
error) {
    fake.setPrivateDataValidationParameterMutex.Lock()
    defer fake.setPrivateDataValidationParameterMutex.Unlock()
    fake.SetPrivateDataValidationParameterStub = nil
    if fake.setPrivateDataValidationParameterReturnsOnCall == nil {
        fake.setPrivateDataValidationParameterReturnsOnCall = make(map[int]struct {
                result1 error
        })
    }
    fake.setPrivateDataValidationParameterReturnsOnCall[i] = struct {
        result1 error
    }{result1}
}

func (fake *ChaincodeStub) SetStateValidationParameter(arg1 string, arg2 []byte) error {
    var arg2Copy []byte
    if arg2 != nil {
        arg2Copy = make([]byte, len(arg2))
        copy(arg2Copy, arg2)
    }
    fake.setStateValidationParameterMutex.Lock()
    ret, specificReturn :=
fake.setStateValidationParameterReturnsOnCall[len(fake.setStateValidationParameterArgsForCa
ll)]
    fake.setStateValidationParameterArgsForCall =
append(fake.setStateValidationParameterArgsForCall, struct {
        arg1 string
        arg2 []byte
    }{arg1, arg2Copy})
    fake.recordInvocation("SetStateValidationParameter", []interface{}{arg1, arg2Copy})
    fake.setStateValidationParameterMutex.Unlock()
    if fake.SetStateValidationParameterStub != nil {
        return fake.SetStateValidationParameterStub(arg1, arg2)
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.setStateValidationParameterReturns
    return fakeReturns.result1
}

func (fake *ChaincodeStub) SetStateValidationParameterCallCount() int {
    fake.setStateValidationParameterMutex.RLock()
    defer fake.setStateValidationParameterMutex.RUnlock()
    return len(fake.setStateValidationParameterArgsForCall)
}

func (fake *ChaincodeStub) SetStateValidationParameterCalls(stub func(string, []byte)
error) {
    fake.setStateValidationParameterMutex.Lock()
    defer fake.setStateValidationParameterMutex.Unlock()
    fake.SetStateValidationParameterStub = stub
}

func (fake *ChaincodeStub) SetStateValidationParameterArgsForCall(i int) (string, []byte) {
    fake.setStateValidationParameterMutex.RLock()
    defer fake.setStateValidationParameterMutex.RUnlock()
    argsForCall := fake.setStateValidationParameterArgsForCall[i]
    return argsForCall.arg1, argsForCall.arg2
}

func (fake *ChaincodeStub) SetStateValidationParameterReturns(result1 error) {
    fake.setStateValidationParameterMutex.Lock()
    defer fake.setStateValidationParameterMutex.Unlock()
    fake.SetStateValidationParameterStub = nil
    fake.setStateValidationParameterReturns = struct {
```

```
          result1 error
     }{result1}
}

func (fake *ChaincodeStub) SetStateValidationParameterReturnsOnCall(i int, result1 error) {
     fake.setStateValidationParameterMutex.Lock()
     defer fake.setStateValidationParameterMutex.Unlock()
     fake.SetStateValidationParameterStub = nil
     if fake.setStateValidationParameterReturnsOnCall == nil {
          fake.setStateValidationParameterReturnsOnCall = make(map[int]struct {
                  result1 error
          })
     }
     fake.setStateValidationParameterReturnsOnCall[i] = struct {
          result1 error
     }{result1}
}

func (fake *ChaincodeStub) SplitCompositeKey(arg1 string) (string, []string, error) {
     fake.splitCompositeKeyMutex.Lock()
     ret, specificReturn :=
fake.splitCompositeKeyReturnsOnCall[len(fake.splitCompositeKeyArgsForCall)]
     fake.splitCompositeKeyArgsForCall = append(fake.splitCompositeKeyArgsForCall, struct {
          arg1 string
     }{arg1})
     fake.recordInvocation("SplitCompositeKey", []interface{}{arg1})
     fake.splitCompositeKeyMutex.Unlock()
     if fake.SplitCompositeKeyStub != nil {
          return fake.SplitCompositeKeyStub(arg1)
     }
     if specificReturn {
          return ret.result1, ret.result2, ret.result3
     }
     fakeReturns := fake.splitCompositeKeyReturns
     return fakeReturns.result1, fakeReturns.result2, fakeReturns.result3
}

func (fake *ChaincodeStub) SplitCompositeKeyCallCount() int {
     fake.splitCompositeKeyMutex.RLock()
     defer fake.splitCompositeKeyMutex.RUnlock()
     return len(fake.splitCompositeKeyArgsForCall)
}

func (fake *ChaincodeStub) SplitCompositeKeyCalls(stub func(string) (string, []string,
error)) {
     fake.splitCompositeKeyMutex.Lock()
     defer fake.splitCompositeKeyMutex.Unlock()
     fake.SplitCompositeKeyStub = stub
}

func (fake *ChaincodeStub) SplitCompositeKeyArgsForCall(i int) string {
     fake.splitCompositeKeyMutex.RLock()
     defer fake.splitCompositeKeyMutex.RUnlock()
     argsForCall := fake.splitCompositeKeyArgsForCall[i]
     return argsForCall.arg1
}

func (fake *ChaincodeStub) SplitCompositeKeyReturns(result1 string, result2 []string,
result3 error) {
     fake.splitCompositeKeyMutex.Lock()
     defer fake.splitCompositeKeyMutex.Unlock()
     fake.SplitCompositeKeyStub = nil
     fake.splitCompositeKeyReturns = struct {
          result1 string
          result2 []string
          result3 error
     }{result1, result2, result3}
}

func (fake *ChaincodeStub) SplitCompositeKeyReturnsOnCall(i int, result1 string, result2
[]string, result3 error) {
     fake.splitCompositeKeyMutex.Lock()
     defer fake.splitCompositeKeyMutex.Unlock()
```

```
    fake.SplitCompositeKeyStub = nil
    if fake.splitCompositeKeyReturnsOnCall == nil {
        fake.splitCompositeKeyReturnsOnCall = make(map[int]struct {
            result1 string
            result2 []string
            result3 error
        })
    }
    fake.splitCompositeKeyReturnsOnCall[i] = struct {
        result1 string
        result2 []string
        result3 error
    }{result1, result2, result3}
}

func (fake *ChaincodeStub) Invocations() map[string][][]interface{} {
    fake.invocationsMutex.RLock()
    defer fake.invocationsMutex.RUnlock()
    fake.createCompositeKeyMutex.RLock()
    defer fake.createCompositeKeyMutex.RUnlock()
    fake.delPrivateDataMutex.RLock()
    defer fake.delPrivateDataMutex.RUnlock()
    fake.delStateMutex.RLock()
    defer fake.delStateMutex.RUnlock()
    fake.getArgsMutex.RLock()
    defer fake.getArgsMutex.RUnlock()
    fake.getArgsSliceMutex.RLock()
    defer fake.getArgsSliceMutex.RUnlock()
    fake.getBindingMutex.RLock()
    defer fake.getBindingMutex.RUnlock()
    fake.getChannelIDMutex.RLock()
    defer fake.getChannelIDMutex.RUnlock()
    fake.getCreatorMutex.RLock()
    defer fake.getCreatorMutex.RUnlock()
    fake.getDecorationsMutex.RLock()
    defer fake.getDecorationsMutex.RUnlock()
    fake.getFunctionAndParametersMutex.RLock()
    defer fake.getFunctionAndParametersMutex.RUnlock()
    fake.getHistoryForKeyMutex.RLock()
    defer fake.getHistoryForKeyMutex.RUnlock()
    fake.getPrivateDataMutex.RLock()
    defer fake.getPrivateDataMutex.RUnlock()
    fake.getPrivateDataByPartialCompositeKeyMutex.RLock()
    defer fake.getPrivateDataByPartialCompositeKeyMutex.RUnlock()
    fake.getPrivateDataByRangeMutex.RLock()
    defer fake.getPrivateDataByRangeMutex.RUnlock()
    fake.getPrivateDataHashMutex.RLock()
    defer fake.getPrivateDataHashMutex.RUnlock()
    fake.getPrivateDataQueryResultMutex.RLock()
    defer fake.getPrivateDataQueryResultMutex.RUnlock()
    fake.getPrivateDataValidationParameterMutex.RLock()
    defer fake.getPrivateDataValidationParameterMutex.RUnlock()
    fake.getQueryResultMutex.RLock()
    defer fake.getQueryResultMutex.RUnlock()
    fake.getQueryResultWithPaginationMutex.RLock()
    defer fake.getQueryResultWithPaginationMutex.RUnlock()
    fake.getSignedProposalMutex.RLock()
    defer fake.getSignedProposalMutex.RUnlock()
    fake.getStateMutex.RLock()
    defer fake.getStateMutex.RUnlock()
    fake.getStateByPartialCompositeKeyMutex.RLock()
    defer fake.getStateByPartialCompositeKeyMutex.RUnlock()
    fake.getStateByPartialCompositeKeyWithPaginationMutex.RLock()
    defer fake.getStateByPartialCompositeKeyWithPaginationMutex.RUnlock()
    fake.getStateByRangeMutex.RLock()
    defer fake.getStateByRangeMutex.RUnlock()
    fake.getStateByRangeWithPaginationMutex.RLock()
    defer fake.getStateByRangeWithPaginationMutex.RUnlock()
    fake.getStateValidationParameterMutex.RLock()
    defer fake.getStateValidationParameterMutex.RUnlock()
    fake.getStringArgsMutex.RLock()
    defer fake.getStringArgsMutex.RUnlock()
    fake.getTransientMutex.RLock()
```

```
    defer fake.getTransientMutex.RUnlock()
    fake.getTxIDMutex.RLock()
    defer fake.getTxIDMutex.RUnlock()
    fake.getTxTimestampMutex.RLock()
    defer fake.getTxTimestampMutex.RUnlock()
    fake.invokeChaincodeMutex.RLock()
    defer fake.invokeChaincodeMutex.RUnlock()
    fake.purgePrivateDataMutex.RLock()
    defer fake.purgePrivateDataMutex.RUnlock()
    fake.putPrivateDataMutex.RLock()
    defer fake.putPrivateDataMutex.RUnlock()
    fake.putStateMutex.RLock()
    defer fake.putStateMutex.RUnlock()
    fake.setEventMutex.RLock()
    defer fake.setEventMutex.RUnlock()
    fake.setPrivateDataValidationParameterMutex.RLock()
    defer fake.setPrivateDataValidationParameterMutex.RUnlock()
    fake.setStateValidationParameterMutex.RLock()
    defer fake.setStateValidationParameterMutex.RUnlock()
    fake.splitCompositeKeyMutex.RLock()
    defer fake.splitCompositeKeyMutex.RUnlock()
    copiedInvocations := map[string][][]interface{}{}
    for key, value := range fake.invocations {
        copiedInvocations[key] = value
    }
    return copiedInvocations
}

func (fake *ChaincodeStub) recordInvocation(key string, args []interface{}) {
    fake.invocationsMutex.Lock()
    defer fake.invocationsMutex.Unlock()
    if fake.invocations == nil {
        fake.invocations = map[string][][]interface{}{}
    }
    if fake.invocations[key] == nil {
        fake.invocations[key] = [][]interface{}{}
    }
    fake.invocations[key] = append(fake.invocations[key], args)
}
```

Mocks State Query Iterator for Chaincode Unit Tests

statequeryiterator.go

```
// Code generated by counterfeiter. DO NOT EDIT.
package mocks

import (
    "sync"

    "github.com/hyperledger/fabric-protos-go/ledger/queryresult"
)

type StateQueryIterator struct {
    CloseStub            func() error
    closeMutex           sync.RWMutex
    closeArgsForCall []struct {
    }
    closeReturns struct {
        result1 error
    }
    closeReturnsOnCall map[int]struct {
        result1 error
    }
    HasNextStub          func() bool
    hasNextMutex         sync.RWMutex
    hasNextArgsForCall []struct {
    }
    hasNextReturns struct {
```

```
		result1 bool
	}
	hasNextReturnsOnCall map[int]struct {
		result1 bool
	}
	NextStub     func() (*queryresult.KV, error)
	nextMutex    sync.RWMutex
	nextArgsForCall []struct {
	}
	nextReturns struct {
		result1 *queryresult.KV
		result2 error
	}
	nextReturnsOnCall map[int]struct {
		result1 *queryresult.KV
		result2 error
	}
	invocations          map[string][][]interface{}
	invocationsMutex sync.RWMutex
}

func (fake *StateQueryIterator) Close() error {
	fake.closeMutex.Lock()
	ret, specificReturn := fake.closeReturnsOnCall[len(fake.closeArgsForCall)]
	fake.closeArgsForCall = append(fake.closeArgsForCall, struct {
	}{})
	fake.recordInvocation("Close", []interface{}{})
	fake.closeMutex.Unlock()
	if fake.CloseStub != nil {
		return fake.CloseStub()
	}
	if specificReturn {
		return ret.result1
	}
	fakeReturns := fake.closeReturns
	return fakeReturns.result1
}

func (fake *StateQueryIterator) CloseCallCount() int {
	fake.closeMutex.RLock()
	defer fake.closeMutex.RUnlock()
	return len(fake.closeArgsForCall)
}

func (fake *StateQueryIterator) CloseCalls(stub func() error) {
	fake.closeMutex.Lock()
	defer fake.closeMutex.Unlock()
	fake.CloseStub = stub
}

func (fake *StateQueryIterator) CloseReturns(result1 error) {
	fake.closeMutex.Lock()
	defer fake.closeMutex.Unlock()
	fake.CloseStub = nil
	fake.closeReturns = struct {
		result1 error
	}{result1}
}

func (fake *StateQueryIterator) CloseReturnsOnCall(i int, result1 error) {
	fake.closeMutex.Lock()
	defer fake.closeMutex.Unlock()
	fake.CloseStub = nil
	if fake.closeReturnsOnCall == nil {
		fake.closeReturnsOnCall = make(map[int]struct {
			result1 error
		})
	}
	fake.closeReturnsOnCall[i] = struct {
		result1 error
	}{result1}
}
```

```
func (fake *StateQueryIterator) HasNext() bool {
    fake.hasNextMutex.Lock()
    ret, specificReturn := fake.hasNextReturnsOnCall[len(fake.hasNextArgsForCall)]
    fake.hasNextArgsForCall = append(fake.hasNextArgsForCall, struct {
    }{})
    fake.recordInvocation("HasNext", []interface{}{})
    fake.hasNextMutex.Unlock()
    if fake.HasNextStub != nil {
        return fake.HasNextStub()
    }
    if specificReturn {
        return ret.result1
    }
    fakeReturns := fake.hasNextReturns
    return fakeReturns.result1
}

func (fake *StateQueryIterator) HasNextCallCount() int {
    fake.hasNextMutex.RLock()
    defer fake.hasNextMutex.RUnlock()
    return len(fake.hasNextArgsForCall)
}

func (fake *StateQueryIterator) HasNextCalls(stub func() bool) {
    fake.hasNextMutex.Lock()
    defer fake.hasNextMutex.Unlock()
    fake.HasNextStub = stub
}

func (fake *StateQueryIterator) HasNextReturns(result1 bool) {
    fake.hasNextMutex.Lock()
    defer fake.hasNextMutex.Unlock()
    fake.HasNextStub = nil
    fake.hasNextReturns = struct {
        result1 bool
    }{result1}
}

func (fake *StateQueryIterator) HasNextReturnsOnCall(i int, result1 bool) {
    fake.hasNextMutex.Lock()
    defer fake.hasNextMutex.Unlock()
    fake.HasNextStub = nil
    if fake.hasNextReturnsOnCall == nil {
        fake.hasNextReturnsOnCall = make(map[int]struct {
            result1 bool
        })
    }
    fake.hasNextReturnsOnCall[i] = struct {
        result1 bool
    }{result1}
}

func (fake *StateQueryIterator) Next() (*queryresult.KV, error) {
    fake.nextMutex.Lock()
    ret, specificReturn := fake.nextReturnsOnCall[len(fake.nextArgsForCall)]
    fake.nextArgsForCall = append(fake.nextArgsForCall, struct {
    }{})
    fake.recordInvocation("Next", []interface{}{})
    fake.nextMutex.Unlock()
    if fake.NextStub != nil {
        return fake.NextStub()
    }
    if specificReturn {
        return ret.result1, ret.result2
    }
    fakeReturns := fake.nextReturns
    return fakeReturns.result1, fakeReturns.result2
}

func (fake *StateQueryIterator) NextCallCount() int {
    fake.nextMutex.RLock()
    defer fake.nextMutex.RUnlock()
    return len(fake.nextArgsForCall)
```

```
}

func (fake *StateQueryIterator) NextCalls(stub func() (*queryresult.KV, error)) {
    fake.nextMutex.Lock()
    defer fake.nextMutex.Unlock()
    fake.NextStub = stub
}

func (fake *StateQueryIterator) NextReturns(result1 *queryresult.KV, result2 error) {
    fake.nextMutex.Lock()
    defer fake.nextMutex.Unlock()
    fake.NextStub = nil
    fake.nextReturns = struct {
         result1 *queryresult.KV
         result2 error
    }{result1, result2}
}

func (fake *StateQueryIterator) NextReturnsOnCall(i int, result1 *queryresult.KV, result2
error) {
    fake.nextMutex.Lock()
    defer fake.nextMutex.Unlock()
    fake.NextStub = nil
    if fake.nextReturnsOnCall == nil {
         fake.nextReturnsOnCall = make(map[int]struct {
                 result1 *queryresult.KV
                 result2 error
         })
    }
    fake.nextReturnsOnCall[i] = struct {
         result1 *queryresult.KV
         result2 error
    }{result1, result2}
}

func (fake *StateQueryIterator) Invocations() map[string][][]interface{} {
    fake.invocationsMutex.RLock()
    defer fake.invocationsMutex.RUnlock()
    fake.closeMutex.RLock()
    defer fake.closeMutex.RUnlock()
    fake.hasNextMutex.RLock()
    defer fake.hasNextMutex.RUnlock()
    fake.nextMutex.RLock()
    defer fake.nextMutex.RUnlock()
    copiedInvocations := map[string][][]interface{}{}
    for key, value := range fake.invocations {
         copiedInvocations[key] = value
    }
    return copiedInvocations
}

func (fake *StateQueryIterator) recordInvocation(key string, args []interface{}) {
    fake.invocationsMutex.Lock()
    defer fake.invocationsMutex.Unlock()
    if fake.invocations == nil {
         fake.invocations = map[string][][]interface{}{}
    }
    if fake.invocations[key] == nil {
         fake.invocations[key] = [][]interface{}{}
    }
    fake.invocations[key] = append(fake.invocations[key], args)
}
```

Mock Transaction Context for Chaincode Unit Tests

transaction.go

```
// Code generated by counterfeiter. DO NOT EDIT.
package mocks
```

```
import (
    "sync"

    "github.com/hyperledger/fabric-chaincode-go/pkg/cid"
    "github.com/hyperledger/fabric-chaincode-go/shim"
)

type TransactionContext struct {
    GetClientIdentityStub        func() cid.ClientIdentity
    getClientIdentityMutex       sync.RWMutex
    getClientIdentityArgsForCall []struct {
    }
    getClientIdentityReturns struct {
         result1 cid.ClientIdentity
    }
    getClientIdentityReturnsOnCall map[int]struct {
         result1 cid.ClientIdentity
    }
    GetStubStub          func() shim.ChaincodeStubInterface
    getStubMutex         sync.RWMutex
    getStubArgsForCall []struct {
    }
    getStubReturns struct {
         result1 shim.ChaincodeStubInterface
    }
    getStubReturnsOnCall map[int]struct {
         result1 shim.ChaincodeStubInterface
    }
    invocations          map[string][][]interface{}
    invocationsMutex sync.RWMutex
}

func (fake *TransactionContext) GetClientIdentity() cid.ClientIdentity {
    fake.getClientIdentityMutex.Lock()
    ret, specificReturn :=
fake.getClientIdentityReturnsOnCall[len(fake.getClientIdentityArgsForCall)]
    fake.getClientIdentityArgsForCall = append(fake.getClientIdentityArgsForCall, struct {
    }{})
    fake.recordInvocation("GetClientIdentity", []interface{}{})
    fake.getClientIdentityMutex.Unlock()
    if fake.GetClientIdentityStub != nil {
         return fake.GetClientIdentityStub()
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.getClientIdentityReturns
    return fakeReturns.result1
}

func (fake *TransactionContext) GetClientIdentityCallCount() int {
    fake.getClientIdentityMutex.RLock()
    defer fake.getClientIdentityMutex.RUnlock()
    return len(fake.getClientIdentityArgsForCall)
}

func (fake *TransactionContext) GetClientIdentityCalls(stub func() cid.ClientIdentity) {
    fake.getClientIdentityMutex.Lock()
    defer fake.getClientIdentityMutex.Unlock()
    fake.GetClientIdentityStub = stub
}

func (fake *TransactionContext) GetClientIdentityReturns(result1 cid.ClientIdentity) {
    fake.getClientIdentityMutex.Lock()
    defer fake.getClientIdentityMutex.Unlock()
    fake.GetClientIdentityStub = nil
    fake.getClientIdentityReturns = struct {
         result1 cid.ClientIdentity
    }{result1}
}

func (fake *TransactionContext) GetClientIdentityReturnsOnCall(i int, result1
```

```
cid.ClientIdentity) {
    fake.getClientIdentityMutex.Lock()
    defer fake.getClientIdentityMutex.Unlock()
    fake.GetClientIdentityStub = nil
    if fake.getClientIdentityReturnsOnCall == nil {
         fake.getClientIdentityReturnsOnCall = make(map[int]struct {
                 result1 cid.ClientIdentity
         })
    }
    fake.getClientIdentityReturnsOnCall[i] = struct {
         result1 cid.ClientIdentity
    }{result1}
}

func (fake *TransactionContext) GetStub() shim.ChaincodeStubInterface {
    fake.getStubMutex.Lock()
    ret, specificReturn := fake.getStubReturnsOnCall[len(fake.getStubArgsForCall)]
    fake.getStubArgsForCall = append(fake.getStubArgsForCall, struct {
    }{})
    fake.recordInvocation("GetStub", []interface{}{})
    fake.getStubMutex.Unlock()
    if fake.GetStubStub != nil {
         return fake.GetStubStub()
    }
    if specificReturn {
         return ret.result1
    }
    fakeReturns := fake.getStubReturns
    return fakeReturns.result1
}

func (fake *TransactionContext) GetStubCallCount() int {
    fake.getStubMutex.RLock()
    defer fake.getStubMutex.RUnlock()
    return len(fake.getStubArgsForCall)
}

func (fake *TransactionContext) GetStubCalls(stub func() shim.ChaincodeStubInterface) {
    fake.getStubMutex.Lock()
    defer fake.getStubMutex.Unlock()
    fake.GetStubStub = stub
}

func (fake *TransactionContext) GetStubReturns(result1 shim.ChaincodeStubInterface) {
    fake.getStubMutex.Lock()
    defer fake.getStubMutex.Unlock()
    fake.GetStubStub = nil
    fake.getStubReturns = struct {
         result1 shim.ChaincodeStubInterface
    }{result1}
}

func (fake *TransactionContext) GetStubReturnsOnCall(i int, result1
shim.ChaincodeStubInterface) {
    fake.getStubMutex.Lock()
    defer fake.getStubMutex.Unlock()
    fake.GetStubStub = nil
    if fake.getStubReturnsOnCall == nil {
         fake.getStubReturnsOnCall = make(map[int]struct {
                 result1 shim.ChaincodeStubInterface
         })
    }
    fake.getStubReturnsOnCall[i] = struct {
         result1 shim.ChaincodeStubInterface
    }{result1}
}

func (fake *TransactionContext) Invocations() map[string][][]interface{} {
    fake.invocationsMutex.RLock()
    defer fake.invocationsMutex.RUnlock()
    fake.getClientIdentityMutex.RLock()
    defer fake.getClientIdentityMutex.RUnlock()
    fake.getStubMutex.RLock()
```

```
    defer fake.getStubMutex.RUnlock()
    copiedInvocations := map[string][][]interface{}{}
    for key, value := range fake.invocations {
        copiedInvocations[key] = value
    }
    return copiedInvocations
}

func (fake *TransactionContext) recordInvocation(key string, args []interface{}) {
    fake.invocationsMutex.Lock()
    defer fake.invocationsMutex.Unlock()
    if fake.invocations == nil {
        fake.invocations = map[string][][]interface{}{}
    }
    if fake.invocations[key] == nil {
        fake.invocations[key] = [][]interface{}{}
    }
    fake.invocations[key] = append(fake.invocations[key], args)
}
```

Smart Contract Unit Tests

smartcontract_test.go

```
package chaincode_test

import (
 "fmt"
 "testing"
 "github.com/hyperledger/fabric-chaincode-go/shim"
 "github.com/hyperledger/fabric-contract-api-go/contractapi"
 "github.com/hyperledger/fabric-samples/chaincode/fabricVdr/chaincode-go/chaincode"
 "github.com/hyperledger/fabric-samples/chaincode/fabricVdr/chaincode-go/chaincode/mocks"
 "github.com/stretchr/testify/require"
)

//go:generate counterfeiter -o mocks/transaction.go -fake-name TransactionContext .
transactionContext
type transactionContext interface {
 contractapi.TransactionContextInterface
}

//go:generate counterfeiter -o mocks/chaincodestub.go -fake-name ChaincodeStub .
chaincodeStub
type chaincodeStub interface {
 shim.ChaincodeStubInterface
}

//go:generate counterfeiter -o mocks/statequeryiterator.go -fake-name StateQueryIterator .
stateQueryIterator
type stateQueryIterator interface {
 shim.StateQueryIteratorInterface
}

var (
 clientId        string = "client123"
 deviceId        string = "dev123"
 did             string = "did:fabric:123456789"
 docResolution   string = "TestDocResolution"
 score           string = "0"
)

// Test Cases for CreateDoc function
func TestCreateDoc(t *testing.T) {
 chaincodeStub := &mocks.ChaincodeStub{}
 transactionContext := &mocks.TransactionContext{}
 transactionContext.GetStubReturns(chaincodeStub)

 fabricVdr := chaincode.SmartContract{}
```

```
 // Empty Input
 t.Run("Empty Input", func(t *testing.T){
  clientId := ""
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Empty string as Input is not supported.")
 })

 // No Error
 t.Run("No Error", func(t *testing.T){
  transient := make(map[string][]byte) // Create transient map
  transient["DID"] = []byte(did)
  transient["docResolution"] = []byte(docResolution)
  chaincodeStub.GetTransientReturns(transient, nil)
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.NoError(t, err)
 })

 // Get State Error
 t.Run("Get State Error", func(t *testing.T){
  chaincodeStub.PutStateReturns(nil)
  chaincodeStub.SetEventReturns(nil)
  chaincodeStub.GetStateReturns(nil, fmt.Errorf("Key not found"))
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Error while checking whether the Did exists: Query: Failed to
get state with key " + did + ", error: Key not found")
 })

 // Get State Returns not nil
 t.Run("Get State Returns Not Nil", func(t *testing.T){
  chaincodeStub.PutStateReturns(nil)
  chaincodeStub.SetEventReturns(nil)
  chaincodeStub.GetStateReturns([]byte(did), nil)
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "DocResolution with Did " + did + " is already exists")
 })

 // Put State Error
 t.Run("Put State Error", func(t *testing.T){
  chaincodeStub.GetStateReturns(nil, nil)
  chaincodeStub.PutStateReturns(fmt.Errorf("Fail inserting Key"))
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Error while putting data into the ledger: Fail inserting
Key")
 })

 // Set Event Error
 t.Run("Set Event Error", func(t *testing.T){
  chaincodeStub.PutStateReturns(nil)
  chaincodeStub.SetEventReturns(fmt.Errorf("Fail to set Event"))
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Error while Setting Event: Fail to set Event")
 })

 // Get Transient Error
 t.Run("Get Transient Error", func(t *testing.T){
  chaincodeStub.GetTransientReturns(nil, fmt.Errorf("Fail to get Transient"))
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Error while getting transient data: Fail to get Transient")
 })

 // Empty DID
 t.Run("Empty DID", func(t *testing.T){
  transient := make(map[string][]byte) // Create transient map
  transient["DID"] = []byte("")
  chaincodeStub.GetTransientReturns(transient, nil)
  err := fabricVdr.CreateDoc(transactionContext, clientId)
  require.EqualError(t, err, "Empty DocId as Input is not supported.")
 })
}

// Test Cases for ResolveDID function
func TestResolveDid(t *testing.T) {
```

```
 chaincodeStub := &mocks.ChaincodeStub{}
 transactionContext := &mocks.TransactionContext{}
 transactionContext.GetStubReturns(chaincodeStub)

 fabricVdr := chaincode.SmartContract{}

 // Empty DocID
 t.Run("Empty DocID", func(t *testing.T) {
  docId := ""
  didDoc, err := fabricVdr.ResolveDid(transactionContext, docId)
  require.Equal(t, didDoc, "")
  require.EqualError(t, err, "Error while handling ResolveDID request: Empty string as
Input is not supported.")
 })

 // No Error
 t.Run("No Error", func(t *testing.T){
  expectedDidDoc := []byte(docResolution)
  chaincodeStub.GetStateReturns(expectedDidDoc, nil)
  didDoc, err := fabricVdr.ResolveDid(transactionContext, did)
  require.Equal(t, string(didDoc), string(expectedDidDoc))
  require.NoError(t, err)
 })

 // DocId does not Exist
 t.Run("DocId Does Not Exist", func(t *testing.T){
  chaincodeStub.GetStateReturns(nil, nil)
  didDoc, err := fabricVdr.ResolveDid(transactionContext, did)
  require.Equal(t, string(didDoc), "")
  require.EqualError(t, err, "Error while handling ResolveDID request: Query: Record with
id " + did + " does not exist")
 })

 // Get State Error
 t.Run("Get State Error", func(t *testing.T){
  chaincodeStub.GetStateReturns(nil, fmt.Errorf("Key not found"))
  didDoc, err := fabricVdr.ResolveDid(transactionContext, did)
  require.Equal(t, string(didDoc), "")
  require.EqualError(t, err, "Error while handling ResolveDID request: Query: Failed to get
state with key " + did + ", error: Key not found")
 })
}
```

Appendix B

This Appendix is dedicated to demonstrate the codes that were used during the Deployment of the Hyperledger Fabric Network in Kubernetes (Minikube).

Project Filesystem Structure

```
.
├── Dockerfiles #☣️ Dockerfiles Directory
│   ├── alpine-envsubst #☣️ Alpine Image with envsubst Directory
│   │   └── Dockerfile
│   ├── fabric-ca #☣️ Fabric CA Image with envsubst-Curl-jq Directory
│   │   └── Dockerfile
│   ├── fabric-orderer #☣️ Orderer Image with curl-jq Directory
│   │   └── Dockerfile
│   └── fabric-peer #☣️ Peer Image with curl-jq Directory
│       └── Dockerfile
├── .env #👉 Environmental Variables
├── minikube.sh      #👉 Script for Initializing/Destroying a Minikube Instance
├── networkDown.sh   #👉 Script for Destroying a Hyperledger Fabric Network in Kubernetes
├── networkStart.sh #👉 Script for Starting a Hyperledger Fabric Network in Kubernetes
├── nfsv4 #☣️ NFSv4 Directory
│   ├── nfsv4-server.yaml #👉 NFSv4 Server Deployment Manifest
│   ├── ns.yaml           #👉 NFSv4 Namespace Manifest
│   └── pvc-nfsv4.yaml    #👉 NFSv4 Persistent Volume Claim Manifest
```

```
├── nfsv4.sh #👉 Script for Initializing a NFSv4 Server
├── organization #☣️ Organization Directory
│   ├── bin #☣️ Fabric Official Binaries
│   │   ├── configtxgen
│   │   ├── configtxlator
│   │   ├── cryptogen
│   │   ├── discover
│   │   ├── fabric-ca-client
│   │   ├── fabric-ca-server
│   │   ├── ledgerutil
│   │   ├── orderer
│   │   ├── osnadmin
│   │   └── peer
│   ├── ccExternalTemplate #☣️ Chaincode External Template
│   │   ├── builders #☣️ Chaincode Builders
│   │   │   └── ccaas #☣️ Chaincode As A Service (CCAAS) Builders
│   │   │       └── bin
│   │   │           ├── build
│   │   │           ├── detect
│   │   │           └── release
│   │   ├── chaincode #☣️ Chaincode Source Code Directory
│   │   │   └── smartcontract.go #👉 FabricVDR Chaincode modified to support CCAAS
│   │   ├── connection.json #👉 Peer Connection Details
│   │   ├── Dockerfile      #👉 Chaincode Dockerfile
│   │   └── metadata.json   #👉 Peer Metadata Details
│   ├── config #☣️ Configuration files Directory
│   │   ├── ca #☣️ CA Configuration files Directory
│   │   │   ├── fabric-ca #☣️ Fabric CA Directory
│   │   │   │   ├── fabric-ca-server-config.yaml #👉 Identity Configuration file
│   │   │   │   └── tlsca #☣️ Tlsca Directory
│   │   │   │       └── fabric-ca-server-config.yaml #👉 Tlsca Configuration file
│   │   │   ├── intermediateCA #☣️ IntermediateCA Directory
│   │   │   │   ├── ica-tls-issuer.yaml #👉 Intermediate TLS Issuer Manifest
│   │   │   │   ├── identity #☣️ Empty Directory
│   │   │   │   └── tlsca #☣️ Empty Directory
│   │   │   └── rootCA #☣️ RootCA Directory
│   │   │       ├── identity #☣️ Identity Directory
│   │   │       │   └── openssl_root-identity.cnf #👉 Identity OPENSSL Configuration file
│   │   │       └── tlsca #☣️ Tlsca Directory
│   │   │           └── openssl_root-tls.cnf #👉 Tlsca OPENSSL Configuration file
│   │   ├── orderer.yaml #👉 Orderer Configuration file
│   │   └── peer.yaml    #👉 Peer Configuration file
│   └── manifests #☣️ Manifests Directory
│       ├── chaincode #☣️ Chaincode Manifests Directory
│       │   ├── ccBuilder #☣️ Chaincode K8s Builder Manifests Directory
│       │   │   ├── fabric-builder-rolebinding.yaml
│       │   │   ├── fabric-builder-role.yaml
│       │   │   └── install-k8s-builder.yaml
│       │   ├── chaincode-env.yaml #👉 Chaincode ConfigMap
│       │   └── chaincode.yaml     #👉 Chaincode Deployment-Service
│       ├── fabric-ca #☣️ Fabric CA Manifests Directory
│       │   ├── org-ca-env.yaml #👉 Fabric CA ConfigMap
│       │   └── org-ca.yaml     #👉 Fabric CA Deployment-Service-Ingress
│       ├── ns.yaml #👉 Namespace Manifest
│       ├── orderer #☣️ Orderer Manifests Directory
│       │   ├── orderer-env.yaml #👉 Orderer ConfigMap
│       │   └── orderer.yaml     #👉 Orderer Deployment-Service-Ingress
│       ├── org-pvc.yaml #👉 Persistent Volume Claim Manifest
│       └── peer #☣️ Peer Manifests Directory
│           ├── peer-env.yaml     #👉 Peer ConfigMap Manifest
│           ├── peer-gateway.yaml #👉 Peer Gateway Manifest
│           └── peer.yaml         #👉 Peer Deployment-Service-Ingress Manifest
└── scripts #☣️ Scripts Directory
    ├── artifacts.sh         #👉 Script for Creating Channel Artifacts
    ├── chaincode.sh         #👉 Script for Handling Chaincode operations
    ├── channel.sh           #👉 Script for Handling Channel operations
    ├── createLocalMSP.sh    #👉 Script for Creating local MSP directory
    ├── exportPeerContext.sh #👉 Script for Exporting Peer context
    └── spinner.sh           #👉 Script for Handling Spinner animation

30 directories, 55 files
```

Environmental Variables

.env

```
#################
#🔝: Headers   #
#👉: Comments  #
#################

############################################################
#####################🔝[COLORS]🔝#########################
############################################################
GREEN='\033[0;32m'
BLUE='\033[0;34m'
RED='\033[0;31m'
YELLOW='\033[1;33m'
NC='\033[0m'
############################################################
#####################🔝[GENERIC]🔝########################
############################################################
#☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️
#☣️ Minimum modifications that are required to execute this script ☣️
ORG=< orgN OR orderer >
DOMAIN=example.com
NODE_TYPE=< peer OR orderer > #👉 NODE_TYPE is either 'peer' or 'orderer'
#👉 If ${NODE_TYPE} is orderer, the PEER must be an odd number and greater than
#👉 or equal to 3
PEER=2 #👉 Number of Peers. If ${NODE_TYPE} is 'orderer', PEER is equal to the number of
orderers
USER=1 #👉 Number of Users
CHANNEL_NAME=mychannel #👉 The name of the channel that is being created by Orderer
PATH=${PWD}/organization/bin:$PATH #👉 Export Path of the Fabric Binaries
#☣️ Minimum modifications that are required to execute this script ☣️
#☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️☣️
NETWORK_NAME_PREFIX=fabric-network- #👉 Network Name Prefix
NETWORK_NAME=${NETWORK_NAME_PREFIX}${ORG} #👉 Network Name
NS=${NETWORK_NAME} #👉 Namespace Name
STORAGE_RESOURCES=2Gi #👉 Adjust the STORAGE_RESOURCES, according to your needs.
#👉 Depending on the type of the cluster (e.g Minikube, Kind etc) STORAGE_CLASS_NAME
#👉 may change
STORAGE_CLASS_NAME=""
#👉 Hyperledger Fabric Infrastructure Properties
FABRIC_CONTAINER_REGISTRY=hyperledger #👉 Hyperledger Fabric Registry
FABRIC_VERSION=2.4.6  #👉 Hyperledger Fabric version
CA_VERSION=1.5.5      #👉 Hyperledger Fabric CA Server version
COUCHDB_IMAGE=couchdb #👉 CouchDB Registry
COUCHDB_VERSION=3.2.2 #👉 Hyperledger Fabric CouchDB version
GO_VERSION=1.18.2     #👉 Go version that is used to build Chaincode
ALPINE_VERSION=3.16   #👉 Alpine Image version that is used by InitContainers and Chaincode
FABRIC_EMPTY_DIR=/tmp/hyperledger
FABRIC_CFG_PATH=${SHARE_DIRECTORY}/fabric/config
FABRIC_LOGGING_SPEC=INFO
############################################################
#####################🔝[OPENSSL]🔝########################
############################################################
#👉 Working Directory
WORKING_DIR=${PWD}/organization
#👉 Scripts Directory
SCRIPTS_DIR=${PWD}/scripts
#👉 Manifests Directory
MANIFESTS_DIR=${WORKING_DIR}/manifests
#👉 Path of the CA dir
CA_DIR=${WORKING_DIR}/config/ca
#👉 Path of the RootCA dir
ROOT_CA_DIR=${WORKING_DIR}/${ORG}.${DOMAIN}/rootCA
#👉 Path of the IntermediateCA dir
INTERMEDIATE_CA_DIR=${WORKING_DIR}/${ORG}.${DOMAIN}/intermediateCA
#👉 Openssl tlsca config dir
OPENSSL_TLSCA_CONFIG=${ROOT_CA_DIR}/tlsca
#👉 Openssl identity config dir
OPENSSL_IDENTITY_CONFIG=${ROOT_CA_DIR}/identity
############################################################
```

```
#####################⬆️[NFSv4]⬆️#########################
############################################################
NFSv4_SUFFIX=nfsv4
NS_NFSv4=${NETWORK_NAME}-${NFSv4_SUFFIX}
NFSv4_PORT=2049
SHARE_DIRECTORY=/var/hyperledger
SYNC=true
############################################################
#####################⬆️[FABRIC-CA]⬆️#####################
############################################################
FABRIC_CA_SERVER_USERNAME=admin   #👉 Pass it in Vault secrets engine
FABRIC_CA_SERVER_PASSWORD=$(cat /proc/sys/kernel/random/uuid | sed 's/[-]//g' | head -c 40;
echo;) #👉 Pass it in Vault secrets engine
ADMIN_CREDENTIALS=${FABRIC_CA_SERVER_USERNAME}:${FABRIC_CA_SERVER_PASSWORD}
FABRIC_CA_CLIENT_HOME=${SHARE_DIRECTORY}/fabric-ca-client
FABRIC_CA_SERVER_HOME=${SHARE_DIRECTORY}/fabric-ca-server
FABRIC_CA_CLIENT_CERTS=${FABRIC_CA_CLIENT_HOME}/tls/client
FABRIC_CA_SERVER_CERTS=${FABRIC_CA_SERVER_HOME}/tls/server
FABRIC_CA_SERVER_OPERATIONS_CERTS=${FABRIC_CA_SERVER_HOME}/operations
#👉 Version of config file
FABRIC_CA_VERSION=${CA_VERSION}
#👉 Servers listening port
FABRIC_CA_SERVER_PORT=7054
#👉 Enables debug logging
FABRIC_CA_SERVER_DEBUG=false
#####################➡️ TLS Section ⬅️#####################
#👉 Enable TLS
FABRIC_CA_SERVER_TLS_ENABLED=true
#👉 TLS for the servers listening port
FABRIC_CA_SERVER_TLS_CERTFILE=${FABRIC_CA_SERVER_CERTS}/server.crt
FABRIC_CA_SERVER_TLS_KEYFILE=${FABRIC_CA_SERVER_CERTS}/server.key
#👉 The Following Types Are Supported For Client Authentication: Noclientcert,
#👉 Requestclientcert, Requireanyclientcert, Verifyclientcertifgiven and
#👉 Requireandverifyclientcert.
FABRIC_CA_SERVER_TLS_CLIENTAUTH_TYPE=RequireAndVerifyClientCert
#👉 Certfiles is a list of root certificate authorities that the server uses
#👉 when verifying client certificates.
FABRIC_CA_SERVER_TLS_CLIENTAUTH_CERTFILES=${FABRIC_CA_SERVER_CERTS}/ca.crt
#####################➡️ CA Section ⬅️#####################
#👉 Name of this CA
FABRIC_CA_SERVER_CA_NAME=${ORG}-ca
#👉 Key file (is only used to import a private key into BCCSP)
FABRIC_CA_SERVER_CA_KEYFILE=${FABRIC_CA_SERVER_HOME}/ca/certs/ica.identity.${ORG}.${DOMAIN}
.key
#👉 Certificate file
FABRIC_CA_SERVER_CA_CERTFILE=${FABRIC_CA_SERVER_HOME}/ca/certs/ica.identity.${ORG}.${DOMAIN
}.crt
#👉 Chain file
FABRIC_CA_SERVER_CA_CHAINFILE=${FABRIC_CA_SERVER_HOME}/ca/certs/chain.identity.${ORG}.${DOM
AIN}.crt
#####################➡️ Database Section ⬅️################
#👉 Supported types are: sqlite3, postgres and mysql.
FABRIC_CA_SERVER_DB_TYPE=sqlite3
FABRIC_CA_SERVER_DB_DATASOURCE=${FABRIC_CA_SERVER_HOME}/ca/fabric-ca-server.db
#👉 Enable TLS
FABRIC_CA_SERVER_DB_TLS_ENABLED=false
#👉 Certificate file
FABRIC_CA_SERVER_DB_TLS_CERTFILES=
#👉 Client Certificate file
FABRIC_CA_SERVER_DB_TLS_CLIENT_CERTFILE=
#👉 Client Key file
FABRIC_CA_SERVER_DB_TLS_CLIENT_KEYFILE=
#####################➡️ Operations Section ⬅️################
FABRIC_CA_SERVER_OPERATIONS_ADDRESS=0.0.0.0
FABRIC_CA_SERVER_OPERATIONS_PORT=9443
FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS=${FABRIC_CA_SERVER_OPERATIONS_ADDRESS}:${FABRIC_C
A_SERVER_OPERATIONS_PORT}
#👉 Enable Operations TLS
FABRIC_CA_SERVER_OPERATIONS_TLS_ENABLED=true
FABRIC_CA_SERVER_OPERATIONS_TLS_KEY_FILE=${FABRIC_CA_SERVER_OPERATIONS_CERTS}/server/server
.key
FABRIC_CA_SERVER_OPERATIONS_TLS_CERT_FILE=${FABRIC_CA_SERVER_OPERATIONS_CERTS}/server/serve
r.crt
```

```
FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES=${FABRIC_CA_SERVER_OPERATIONS_CERTS}/server/ca.crt
#👉 Enable Operations Mutual TLS
FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTAUTHREQUIRED=true
FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTROOTCAS_FILES=${FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES}
############################################################
####################⬆️[FABRIC-TLSCA]⬆️#################
############################################################
FABRIC_TLSCA_SERVER_USERNAME=${FABRIC_CA_SERVER_USERNAME} #👉 Pass it in Vault secrets engine
FABRIC_TLSCA_SERVER_PASSWORD=${FABRIC_CA_SERVER_PASSWORD} #👉 Pass it in Vault secrets engine
ADMIN_CREDENTIALS_TLSCA=${FABRIC_TLSCA_SERVER_USERNAME}:${FABRIC_TLSCA_SERVER_PASSWORD}
#👉 Pass it in Vault secrets engine
#####################➡️ CA Section ⬅️#####################
#👉 Name of this CA
FABRIC_TLSCA_SERVER_CA_NAME=${ORG}-tlsca
#👉 Key file (is only used to import a private key into BCCSP)
FABRIC_TLSCA_SERVER_CA_KEYFILE=${FABRIC_CA_SERVER_HOME}/tlsca/certs/ica.tls.${ORG}.${DOMAIN}.key
#👉 Certificate file
FABRIC_TLSCA_SERVER_CA_CERTFILE=${FABRIC_CA_SERVER_HOME}/tlsca/certs/ica.tls.${ORG}.${DOMAIN}.crt
#👉 Chain file
FABRIC_TLSCA_SERVER_CA_CHAINFILE=${FABRIC_CA_SERVER_HOME}/tlsca/certs/chain.tls.${ORG}.${DOMAIN}.crt
#####################➡️ Database Section ⬅️################
#👉 Supported types are: sqlite3, postgres and mysql.
FABRIC_TLSCA_SERVER_DB_TYPE=${FABRIC_CA_SERVER_DB_TYPE}
FABRIC_TLSCA_SERVER_DB_DATASOURCE=${FABRIC_CA_SERVER_DB_DATASOURCE}
#👉 Enable TLS
FABRIC_TLSCA_SERVER_DB_TLS_ENABLED=${FABRIC_CA_SERVER_DB_TLS_ENABLED}
#👉 Certificate file
FABRIC_TLSCA_SERVER_DB_TLS_CERTFILES=
#👉 Client Certificate file
FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_CERTFILE=
#👉 Client Key file
FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_KEYFILE=
############################################################
####################⬆️[FABRIC-NETWORK]⬆️################
############################################################
ORG_DIR=${SHARE_DIRECTORY}/fabric/crypto-config/${NODE_TYPE}Organizations/${ORG}.${DOMAIN}
IDENTITY_REGISTRAR_DIR=${WORKING_DIR}/${ORG}.${DOMAIN}/users/admin
TLS_REGISTRAR_DIR=${WORKING_DIR}/${ORG}.${DOMAIN}/users/tlsadmin
PEER_DIR=${ORG_DIR}/${NODE_TYPE}s
ADMIN_DIR=${WORKING_DIR}/${ORG}.${DOMAIN}/users/Admin@${ORG}.${DOMAIN}
LOCALMSPID=${ORG^}MSP
NODE_NAME=${NODE_TYPE}\${k} #👉 Pass it in Vault secrets engine
IDENTITY_ADMIN_USERNAME=Admin@${ORG}.${DOMAIN}
IDENTITY_ADMIN_PASSWORD=$(cat /proc/sys/kernel/random/uuid | sed 's/[-]//g' | head -c 30; echo;)
IDENTITY_PEER_USERNAME=${NODE_NAME}.${ORG}.${DOMAIN}
IDENTITY_PEER_PASSWORD=${NODE_NAME}
IDENTITY_USER_USERNAME=User\${k}@${ORG}.${DOMAIN}
IDENTITY_USER_PASSWORD=userpw\${k}.${ORG}.${DOMAIN}
############################################################
####################⬆️[PEER]⬆️############################
############################################################
PEER_GRPC_ADDRESS=0.0.0.0
PEER_GRPC_PORT=7051
CORE_PEER_ID=${NODE_NAME}.${ORG}.${DOMAIN}
CORE_PEER_LISTENADDRESS=${PEER_GRPC_ADDRESS}:${PEER_GRPC_PORT}
CORE_PEER_ADDRESS=${NODE_NAME}-${ORG}:${PEER_GRPC_PORT}
CORE_PEER_ADDRESSAUTODETECT=false
CORE_PEER_LOCALMSPID=${LOCALMSPID}
CORE_PEER_MSPCONFIGPATH=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/msp
#👉 Disable Gossip (Retrieve blocks directly from orderer)
CORE_PEER_GOSSIP_ORGLEADER=true
CORE_PEER_GOSSIP_USELEADERELECTION=false
CORE_PEER_GOSSIP_STATE_ENABLED=false
CORE_PEER_DELIVERYCLIENT_BLOCKGOSSIPENABLED=false
#👉 Enable TLS
```

```
CORE_PEER_TLS_ENABLED=true
CORE_PEER_TLS_KEY_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/server.key
CORE_PEER_TLS_CERT_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/server.crt
CORE_PEER_TLS_ROOTCERT_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/ca.crt
#👉 Enable Mutual TLS
CORE_PEER_TLS_CLIENTAUTHREQUIRED=true
CORE_PEER_TLS_CLIENTKEY_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/client.key
CORE_PEER_TLS_CLIENTCERT_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/client.crt
CORE_PEER_TLS_CLIENTROOTCAS_FILES=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/ca.crt
#👉 Operations
PEER_OPERATIONS_PORT=9443
CORE_OPERATIONS_LISTENADDRESS=${PEER_GRPC_ADDRESS}:${PEER_OPERATIONS_PORT}
#👉 Enable Operations TLS
CORE_OPERATIONS_TLS_ENABLED=true
CORE_OPERATIONS_TLS_KEY_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/server.ke
y
CORE_OPERATIONS_TLS_CERT_FILE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/server.c
rt
CORE_OPERATIONS_TLS_ROOTCAS_FILES=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/ca.c
rt
#👉 Enable Operations Mutual TLS
CORE_OPERATIONS_TLS_CLIENTAUTHREQUIRED=true
CORE_OPERATIONS_TLS_CLIENTROOTCAS_FILES=${CORE_OPERATIONS_TLS_ROOTCAS_FILES}
#👉 Chaincode
PEER_CHAINCODE_PORT=7052
CORE_PEER_CHAINCODEADDRESS=${NODE_NAME}-${ORG}:${PEER_CHAINCODE_PORT}
CORE_PEER_CHAINCODELISTENADDRESS=${PEER_GRPC_ADDRESS}:${PEER_CHAINCODE_PORT}
#👉 Chaincode TLS [Peer nodes act as clients to chaincode server]
CORE_PEER_CHAINCODE_TLS_ENABLED=$([ ${CHAINCODE_TLS_DISABLED:-true} == true ] && echo false
|| echo true)
#👉 Filesystem
CORE_PEER_FILESYSTEMPATH=${SHARE_DIRECTORY}/fabric/data/${NODE_TYPE}s/${NODE_NAME}.${ORG}.$
{DOMAIN}
#👉 CouchDB
COUCHDB_PORT=5984
COUCHDB_USER=admin
COUCHDB_PASSWORD=$(cat /proc/sys/kernel/random/uuid | sed 's/[-]//g' | head -c 30; echo;)
#👉 Ledger
CORE_LEDGER_SNAPSHOTS_ROOTDIR=${CORE_PEER_FILESYSTEMPATH}/snapshots
CORE_LEDGER_STATE_STATEDATABASE=CouchDB
CORE_LEDGER_STATE_COUCHDBCONFIG_MAXRETRIESONSTARTUP=20
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=localhost:${COUCHDB_PORT}
#👉 Gateway
CORE_PEER_GATEWAY_ENABLED=true
############################################################
#####################🔝[ORDERER]🔝#######################
############################################################
ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
ORDERER_GENERAL_LISTENPORT=6050
ORDERER_GENERAL_LOCALMSPID=${LOCALMSPID}
ORDERER_GENERAL_LOCALMSPDIR=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/msp
ORDERER_GENERAL_BOOTSTRAPMETHOD=none
#👉 Enable Channel Participation API
ORDERER_CHANNELPARTICIPATION_ENABLED=true
#👉 Enable TLS
ORDERER_GENERAL_TLS_ENABLED=true
ORDERER_GENERAL_TLS_PRIVATEKEY=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/server.key
ORDERER_GENERAL_TLS_CERTIFICATE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/server.crt
ORDERER_GENERAL_TLS_ROOTCAS=[${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/tls/ca.crt]
#👉 Enable Mutual TLS
ORDERER_GENERAL_TLS_CLIENTAUTHREQUIRED=true
ORDERER_GENERAL_TLS_CLIENTROOTCAS=[${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/ca.crt]
#👉 Cluster Configuration
ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/
client.crt
ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/c
lient.key
ORDERER_GENERAL_CLUSTER_ROOTCAS=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/ca.crt
#👉 The below 4 properties should be either set together, or be unset together.
#👉 If they are set, then the orderer node uses a separate listener for intra-cluster
#👉 communication. If they are unset, then the general orderer listener is used.
#👉 This is useful if you want to use a different TLS server certificates on the
#👉 client-facing and the intra-cluster listeners.
```

```
ORDERER_GENERAL_CLUSTER_LISTENPORT=
ORDERER_GENERAL_CLUSTER_LISTENADDRESS=
ORDERER_GENERAL_CLUSTER_SERVERCERTIFICATE=
ORDERER_GENERAL_CLUSTER_SERVERPRIVATEKEY=
#👉 Admin
ORDERER_ADMIN_ADDRESS=${ORDERER_GENERAL_LISTENADDRESS}
ORDERER_ADMIN_PORT=9443
ORDERER_ADMIN_LISTENADDRESS=${ORDERER_ADMIN_ADDRESS}:${ORDERER_ADMIN_PORT}
#👉 Enable Admin TLS
ORDERER_ADMIN_TLS_ENABLED=true
ORDERER_ADMIN_TLS_PRIVATEKEY=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/admin/admin.key
ORDERER_ADMIN_TLS_CERTIFICATE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/admin/admin.crt
ORDERER_ADMIN_TLS_ROOTCAS=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/admin/ca.crt
#👉 Enable Admin  Mutual TLS
#👉 NOTE: When TLS is enabled, the admin endpoint requires mutual TLS. The
#👉 orderer will panic on startup if this value is set to false.
ORDERER_ADMIN_TLS_CLIENTAUTHREQUIRED=true
ORDERER_ADMIN_TLS_CLIENTROOTCAS=[${ORDERER_ADMIN_TLS_ROOTCAS}]
#👉 Fileledger
ORDERER_FILELEDGER_LOCATION=${SHARE_DIRECTORY}/fabric/data/${NODE_TYPE}s/${NODE_NAME}.${ORG
}.${DOMAIN}
#👉 Consensus
ORDERER_CONSENSUS_WALDIR=${ORDERER_FILELEDGER_LOCATION}/etcdraft/wal
ORDERER_CONSENSUS_SNAPDIR=${ORDERER_FILELEDGER_LOCATION}/etcdraft/snapshot
#👉 Operations
ORDERER_OPERATIONS_ADDRESS=${ORDERER_GENERAL_LISTENADDRESS}
ORDERER_OPERATIONS_PORT=8443
ORDERER_OPERATIONS_LISTENADDRESS=${ORDERER_OPERATIONS_ADDRESS}:${ORDERER_OPERATIONS_PORT}
#👉 Enable Operations TLS
ORDERER_OPERATIONS_TLS_ENABLED=true
ORDERER_OPERATIONS_TLS_PRIVATEKEY=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/oper
ations.key
ORDERER_OPERATIONS_TLS_CERTIFICATE=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/ope
rations.crt
ORDERER_OPERATIONS_TLS_ROOTCAS=${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations/ca.crt
#👉 Enable Operations Mutual TLS
ORDERER_OPERATIONS_TLS_CLIENTAUTHREQUIRED=true
ORDERER_OPERATIONS_TLS_CLIENTROOTCAS=${ORDERER_OPERATIONS_TLS_ROOTCAS}
############################################################
##################🔝[CHAINCODE SERVER]🔝##################
############################################################
K8S_CHAINCODE_BUILDER_IMAGE=ghcr.io/hyperledger-labs/k8s-fabric-peer
K8S_CHAINCODE_BUILDER_VERSION=v0.7.2
CHAINCODE_EXTERNAL_TEMPLATE=${WORKING_DIR}/ccExternalTemplate
CHAINCODE_BUILDER=ccaas #👉 CHAINCODE_BUILDER is either 'ccaas' or 'k8s'
#👉 The CHAINCODE_FILE must be set using the absolute path,
#👉 to point to the location of the cc go file
CHAINCODE_FILE=${CHAINCODE_EXTERNAL_TEMPLATE}/chaincode/smartcontract.go
CHAINCODE_DIR=$(basename ${CHAINCODE_FILE})
CHAINCODE_FILENAME=$(echo ${CHAINCODE_DIR} | awk -F '.' '{print $1}')
CHAINCODE_FILETYPE=$(echo ${CHAINCODE_DIR} | awk -F '.' '{print $2}')
CHAINCODE_NAME=fabricvdr    #👉 Chaincode Name
CHAINCODE_PORT=9999         #👉 Chaincode Port
CHAINCODE_VERSION=1.0       #👉 Chaincode Version
CHAINCODE_SEQUENCE=1        #👉 Chaincode Sequence
CHAINCODE_LABEL=${CHAINCODE_NAME}_${CHAINCODE_VERSION} #👉 Chaincode Label
CHAINCODE_ID=${CHAINCODE_LABEL}:\${PACKAGE_ID} #👉 Chaincode ID
CHAINCODE_SERVER_ADDRESS={{.PEER}}-${ORG}-${CHAINCODE_NAME}:${CHAINCODE_PORT}
#👉 Enable TLS (If false, TLS is enabled)
CHAINCODE_TLS_DISABLED=false
#👉 The following paths are referred from within the chaincode container
CHAINCODE_TLS_CERT=${PEER_DIR}/\${PEER}.${ORG}.${DOMAIN}/chaincode/server.crt
CHAINCODE_TLS_KEY=${PEER_DIR}/\${PEER}.${ORG}.${DOMAIN}/chaincode/server.key
CHAINCODE_CLIENT_CA_CERT=${PEER_DIR}/\${PEER}.${ORG}.${DOMAIN}/chaincode/ca.crt
############################################################
####################🔝[CERT-MANAGER]🔝###################
############################################################
CERT_MANAGER_VERSION=v1.8.2
CERT_MANAGER_MANIFEST=https://github.com/cert-manager/cert-manager/releases/download/${CERT
_MANAGER_VERSION}/cert-manager.yaml
```

Start/Destroy Minikube Instance

minikube.sh

./minikube.sh {up|status|down}

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o allexport
source .env
. scripts/spinner.sh
set +o allexport

MINIKUBE_VERSION=v1.29.0
KUBERNETES_VERSION=v1.26.1
CPU=4
MEMORY=8192

function print_help(){
 echo -e "${RED}Usage: ./minikube.sh MODE                    ${NC}"
 echo
 echo -e "${RED}MODE:                                        ${NC}"
 echo -e "${RED}  up: Spawn a new Minikube network up       ${NC}"
 echo -e "${RED}  status: Check the status of the Minikube  ${NC}"
 echo -e "${RED}  down: Delete the Running Minikube network ${NC}"
}

## Parse mode
if [[ $# -lt 1 ]] ; then
  print_help
  exit 0
else
  MODE=$1
  shift
fi

# Double check that kind, kubectl, docker, and all required images are present.
function checkPrereqs() {
  set +e
  docker version > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No docker binary available${NC}"
   exit 1
  fi

  kubectl > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No kubectl binary available${NC}"
   exit 1
  fi

  jq --version > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No jq binary available${NC}"
   sudo apt install -y jq > /dev/null
  fi

  openssl version > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No openssl binary available${NC}"
   sudo apt-get install -y libssl-dev openssl > /dev/null
  fi

  echo | envsubst > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No envsubst binary available${NC}"
   sudo apt install -y gettext-base > /dev/null
```

```
  fi

  curl --version > /dev/null
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No curl binary available${NC}"
   sudo apt-get install -y curl > /dev/null
  fi

  # Use the local fabric binaries if available.  If not, go get them.
  peer version > /dev/null 2>&1
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No hyperledger fabric binaries available${NC}"
   echo -e "${GREEN}✓ ${BLUE}Downloading LATEST Fabric binaries and config${NC}"
   curl -sSL https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/bootstrap.sh
| bash -s ${FABRIC_VERSION} ${CA_VERSION} -s -d > /dev/null 2>&1 & spinner "Waiting until
Hyperledger Fabric binaries installation is finished"

   # remove sample config files extracted by the installation script
   rm -rf config/ > /dev/null 2>&1
   mv bin/ organization/ > /dev/null 2>&1
   mv builders/ organization/ccExternalTemplate/ > /dev/null 2>&1
  fi

  # Double-check that the binary transfer was OK
  peer version > /dev/null 2>&1
  if [[ $? -ne 0 ]]; then
   echo -e "${RED}X ${BLUE}No peer binary available${NC}"
   exit 1
  fi
  if [[ $(peer version | grep Version | awk -F 'Version: ' '{print $2}') !=
"${FABRIC_VERSION}" ]]; then
   echo -e "${RED}X ${BLUE}Peer binaries Version mismatch${NC}"
   exit 1
  fi

  if [[ ! $(which wget) ]]; then
   echo -e "${RED}X ${BLUE}No wget binary available${NC}"
   sudo apt install -y wget > /dev/null 2>&1
  fi

  minikube version > /dev/null 2>&1
  if [[ $? -ne 0 ]]; then
   installMinikube ${MINIKUBE_VERSION}
  fi
  if [[ ! $(which minikube) ]]; then
   echo -e "${RED}X ${BLUE}No minikube binary available${NC}"
   exit 1
  fi
  if [[ $(minikube version | grep version | awk -F 'minikube version: ' '{print $2}') !=
${MINIKUBE_VERSION} ]]; then
   installMinikube ${MINIKUBE_VERSION}
  fi
  set -e
}

# Install Minikube
function installMinikube() {
 local MINIKUBE_VERSION=$1
 rm /usr/local/bin/minikube > /dev/null 2>&1
 rm /usr/bin/minikube > /dev/null 2>&1
 wget -O minikube
https://github.com/kubernetes/minikube/releases/download/${MINIKUBE_VERSION}/minikube-linux
-amd64 > /dev/null 2>&1
 chmod +x minikube > /dev/null 2>&1
 sudo mv minikube /usr/local/bin/ > /dev/null 2>&1
}

function buildDockerfile() {
 local IMAGE=$1
 local VERSION=$2

 if [[ ! -d "Dockerfiles" ]]; then
  echo -e "${RED}X ${BLUE}The Dockerfiles directory does not exists${NC}"
```

```
  exit 1
 fi
 FILENAME=$(echo ${IMAGE} | grep '/' | awk -F '/' '{print $2}')
 if [[ -z ${FILENAME} ]]; then
  FILENAME=${IMAGE}
  REGISTRY=
 else
  REGISTRY=$(echo ${IMAGE} | grep '/' | awk -F '/' '{print $1}')'/'
 fi
 if [[ ! -d "Dockerfiles/${FILENAME}" ]]; then
   echo -e "${RED}X ${BLUE}The Dockerfiles/${FILENAME} directory does not exists${NC}"
   exit 1
 fi
 # Removing old images
 docker images -a | grep ${FILENAME} | awk '{print $3}' | xargs docker rmi -f - > /dev/null
2>&1 || true
 (cd Dockerfiles/${FILENAME}
  cat Dockerfile | envsubst | docker build --rm=true --force-rm -t
${REGISTRY}${FILENAME}:${VERSION} - > /dev/null 2>&1
  minikube image load ${REGISTRY}${FILENAME}:${VERSION} > /dev/null 2>&1
 )
 # Removing images after were loaded into Minikube
 docker images -a | grep ${FILENAME} | awk '{print $3}' | xargs docker rmi -f - > /dev/null
2>&1 || true
 # Double check if the image has published into Minikube correctly
 if [[ ! $(minikube image ls | grep ${FILENAME}) ]]; then
  echo -e "${RED}X ${BLUE}The ${REGISTRY}/${FILENAME}:${VERSION} image is not published in
Minikube${NC}"
  exit 1
 fi
}

if [ "${MODE}" == "up" ]; then
 sudo chmod 666 /var/run/docker.sock

 checkPrereqs #> /dev/null 2>&1

 minikube delete > /dev/null 2>&1 \
  & spinner "Deleting Previous Instances"
 minikube start --memory ${MEMORY} --cpus ${CPU} --kubernetes-version ${KUBERNETES_VERSION}
--extra-config=kubelet.housekeeping-interval=10s > /dev/null 2>&1 \
  & spinner "Starting Minikube"
 minikube addons enable metrics-server > /dev/null 2>&1 \
  & spinner "Enabling Metrics Server"
 minikube addons enable ingress > /dev/null 2>&1 \
  & spinner "Enabling Ingress Plugin"
 minikube addons enable ingress-dns > /dev/null 2>&1 \
  & spinner "Enabling Ingress DNS Plugin"
 minikube kubectl -- patch deployment -n ingress-nginx ingress-nginx-controller
-p='{"spec":{"template":{"spec":{"containers":[{"name":"controller","args":["/nginx-ingress
-controller","--ingress-class=nginx","--configmap=$(POD_NAMESPACE)/ingress-nginx-controller
","--report-node-internal-ip-address","--tcp-services-configmap=$(POD_NAMESPACE)/tcp-servic
es","--udp-services-configmap=$(POD_NAMESPACE)/udp-services","--validating-webhook=:8443","
--validating-webhook-certificate=/usr/local/certificates/cert","--validating-webhook-key=/u
sr/local/certificates/key","--enable-ssl-passthrough"]}]}}}}' > /dev/null 2>&1 \
  & spinner "Patching Ingress in order to Enable SSL Passthrough"

 docker ps -a | grep Exit | cut -d ' ' -f 1 | xargs sudo docker rm > /dev/null 2>&1 || true
 docker images -a | grep none | awk '{ print $3; }' | xargs docker rmi -f - > /dev/null
2>&1 || true

 buildDockerfile alpine-envsubst latest > /dev/null 2>&1 \
  & spinner "Building alpine-envsubst Dockerfile"
 buildDockerfile ${FABRIC_CONTAINER_REGISTRY}/fabric-ca ${CA_VERSION} > /dev/null 2>&1 \
  & spinner "Building ${FABRIC_CONTAINER_REGISTRY}/fabric-ca:${CA_VERSION} Dockerfile"
 buildDockerfile ${FABRIC_CONTAINER_REGISTRY}/fabric-peer ${FABRIC_VERSION} > /dev/null
2>&1 \
  & spinner "Building ${FABRIC_CONTAINER_REGISTRY}/fabric-peer:${FABRIC_VERSION}
Dockerfile"
 buildDockerfile ${FABRIC_CONTAINER_REGISTRY}/fabric-orderer ${FABRIC_VERSION} > /dev/null
2>&1 \
  & spinner "Building ${FABRIC_CONTAINER_REGISTRY}/fabric-orderer:${FABRIC_VERSION}
Dockerfile"
```

```
 docker ps -a | grep Exit | cut -d ' ' -f 1 | xargs sudo docker rm > /dev/null 2>&1 || true
 docker images -a | grep none | awk '{ print $3; }' | xargs docker rmi -f - > /dev/null
2>&1 || true

 minikube dashboard > /dev/null 2>&1 \
  & spinner "Initializing Dashboard"

elif [ "${MODE}" == "down" ]; then
 minikube delete > /dev/null 2>&1 \
  & spinner "Deleting Minikube Instance"

 if [[ $(docker ps -a | grep minikube) ]]; then
  echo -e "${RED}X ${BLUE}Deleting Minikube failed. Please try again!${NC}"
  exit 1
 fi
 echo -e "${GREEN}✓ ${BLUE}Minikube Deleted. Have a nice day!${NC}"

elif [ "${MODE}" == "status" ]; then
 echo -e "${GREEN}✓ ${BLUE}Checking the status of the Minikube Instance${NC}"
 STATUS=$(minikube status --output json)

 NAME=$(echo ${STATUS} | jq -r .Name)
 CONTROL_PLANE=$(echo -e ${GREEN}Control Plane${NC})
 HOST_RUNNING=$(echo -e ${GREEN}✓${NC})
 KUBELET_RUNNING=$(echo -e ${GREEN}✓${NC})
 APISERVER_RUNNING=$(echo -e ${GREEN}✓${NC})
 KUBECONFIG_RUNNING=$(echo -e ${GREEN}Configured${NC})

 if [[ $(echo ${NAME}) == "null null" ]]; then
  NAME=$(echo -e ${RED}X${NC})
 fi
 if [[ $(echo ${STATUS} | jq -r .Host) != "Running" ]]; then
  HOST_RUNNING=$(echo -e ${RED}X${NC})
 fi
 if [[ $(echo ${STATUS} | jq -r .Kubelet) != "Running" ]]; then
  KUBELET_RUNNING=$(echo -e ${RED}X${NC})
 fi
 if [[ $(echo ${STATUS} | jq -r .APIServer) != "Running" ]]; then
  APISERVER_RUNNING=$(echo -e ${RED}X${NC})
 fi
 if [[ $(echo ${STATUS} | jq -r .Kubeconfig) != "Configured" ]]; then
  KUBECONFIG_RUNNING=$(echo -e ${RED}Not Configured${NC})
 fi
 if [ "$(echo ${STATUS} | jq -r .Worker)" != false ]; then
  :
 fi

 echo -e 'Name: '${NAME}'\nType: '${CONTROL_PLANE}'\nHost: '${HOST_RUNNING}'\nKubelet:
'${KUBELET_RUNNING}'\nAPIServer: '${APISERVER_RUNNING}'\nKubeconfig: '${KUBECONFIG_RUNNING}

else
  print_help
  exit 1
fi
```

Alpine-Envsubst Dockerfile

Dockerfiles/alpine-envsubst/Dockerfile

```
FROM alpine:${ALPINE_VERSION}
LABEL George Misiakoulis
RUN set -x && \
        apk add --update libintl && \
        apk add --virtual build_deps gettext &&  \
        cp /usr/bin/envsubst /usr/local/bin/envsubst && \
        apk del build_deps && \
        rm -rf /var/cache/apk/* && \
        rm -rf /var/lib/apt/lists/*
```

Fabric CA Dockerfile

Dockerfiles/fabric-ca/Dockerfile

```
FROM ${FABRIC_CONTAINER_REGISTRY}/fabric-ca:${CA_VERSION}
LABEL George Misiakoulis
RUN set -x && \
        apk add --update libintl libcurl oniguruma-dev && \
        apk add --virtual build_deps gettext curl jq &&  \
        cp /usr/bin/envsubst /usr/local/bin/envsubst && \
        cp /usr/bin/curl /usr/local/bin/curl && \
        cp /usr/bin/jq /usr/local/bin/jq && \
        apk del build_deps && \
        rm -rf /var/cache/apk/* && \
        rm -rf /var/lib/apt/lists/*
```

Fabric Peer Dockerfile

Dockerfiles/fabric-peer/Dockerfile

```
FROM ${FABRIC_CONTAINER_REGISTRY}/fabric-peer:${FABRIC_VERSION}
LABEL George Misiakoulis
RUN set -x && \
        apk add --update libcurl oniguruma-dev && \
        apk add --virtual build_deps curl jq &&  \
        cp /usr/bin/curl /usr/local/bin/curl && \
        cp /usr/bin/jq /usr/local/bin/jq && \
        apk del build_deps && \
        rm -rf /var/cache/apk/* && \
        rm -rf /var/lib/apt/lists/*
```

Fabric Orderer Dockerfile

Dockerfiles/fabric-orderer/Dockerfile

```
FROM ${FABRIC_CONTAINER_REGISTRY}/fabric-orderer:${FABRIC_VERSION}
LABEL George Misiakoulis
RUN set -x && \
        apk add --update libcurl oniguruma-dev && \
        apk add --virtual build_deps curl jq &&  \
        cp /usr/bin/curl /usr/local/bin/curl && \
        cp /usr/bin/jq /usr/local/bin/jq && \
        apk del build_deps && \
        rm -rf /var/cache/apk/* && \
        rm -rf /var/lib/apt/lists/*
```

Deploy NFSv4 Server

nfsv4.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o errexit
set -o allexport
source .env
. scripts/spinner.sh
set +o allexport
```

```
if [[ ! -d "${PWD}/nfsv4" ]]; then
 echo -e "${RED}X ${BLUE}Directory nfsv4 DOES NOT exists${NC}"
 exit 1
fi

(cd ${PWD}/nfsv4
 echo -e "${GREEN}✓ ${BLUE}Creating namespace ${NS_NFSv4}${NC}"
 cat ns.yaml | envsubst | kubectl apply -f -

 echo -e "${GREEN}✓ ${BLUE}Provisioning volume storage${NC}"
 cat pvc-nfsv4.yaml | envsubst | kubectl -n ${NS_NFSv4} apply -f -

 echo -e "${GREEN}✓ ${BLUE}Starting NFSv4 server${NC}"
 cat nfsv4-server.yaml | envsubst | kubectl -n ${NS_NFSv4} apply -f -
 kubectl -n ${NS_NFSv4} rollout status deploy/${ORG}-nfsv4 > /dev/null 2>&1 \
  & spinner "Waiting until NFS Server installation is completed"
)
```

NFSv4 Namespace Manifest

nfsv4/ns.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
apiVersion: v1
kind: Namespace
metadata:
  name: ${NS_NFSv4}
```

NFSv4 Persistent Volume Claim Manifest

nfsv4/pvc-nfsv4.yaml

```
#
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fabric-${ORG}-nfsv4-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: ${STORAGE_RESOURCES}
```

NFSv4 Server Deployment/Service Manifest

nfsv4/nfsv4-server.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ${ORG}-nfsv4
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ${ORG}-nfsv4
  template:
    metadata:
      labels:
        app: ${ORG}-nfsv4
    spec:
      containers:
        - name: ${ORG}-nfsv4
          image: itsthenetwork/nfs-server-alpine:latest
          imagePullPolicy: IfNotPresent
          securityContext:
            privileged: true
          env:
            - name: SHARED_DIRECTORY
              value: "${SHARE_DIRECTORY}"
            - name: SYNC
              value: "${SYNC}"
          ports:
            - name: nfsv4
              containerPort: ${NFSv4_PORT}
          volumeMounts:
            - name: nfsv4-pvc
              mountPath: ${SHARE_DIRECTORY}
          readinessProbe:
            tcpSocket:
              port: ${NFSv4_PORT}
            initialDelaySeconds: 5
            periodSeconds: 5
      volumes:
        - name: nfsv4-pvc
          persistentVolumeClaim:
            claimName: fabric-${ORG}-nfsv4-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: ${ORG}-nfsv4
  labels:
    app: ${ORG}-nfsv4
spec:
  ports:
    - name: nfsv4
      port: ${NFSv4_PORT}
      targetPort: ${NFSv4_PORT}
      protocol: TCP
  selector:
    app: ${ORG}-nfsv4
```

Deploy Organization
networkStart.sh
./networkStart.sh deploy

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
```

```
set -o errexit

set -o allexport
source .env
. scripts/spinner.sh
. scripts/createLocalMSP.sh
. scripts/artifacts.sh
. scripts/channel.sh
. scripts/chaincode.sh
set +o allexport

function print_help(){
 echo -e "${RED}Usage: ./networkStart.sh Mode Subcommand                ${NC}"
 echo
 echo -e "${RED}Mode:                                                    ${NC}"
 echo -e "${RED}  └── Subcommand (Strong Dependency on .env)             ${NC}"
 echo
 echo -e "${RED}Commands:                                                ${NC}"
 echo -e "${RED}  deploy: Deploy an Organization                         ${NC}"
 echo -e "${RED}  artifacts:                                             ${NC}"
 echo -e "${RED}  └── create: Create the channel artifacts               ${NC}"
 echo -e "${RED}  channel:                                               ${NC}"
 echo -e "${RED}  ├── create: Create the channel and join Orderers       ${NC}"
 echo -e "${RED}  ├── join: Join Organization Peers to the channel       ${NC}"
 echo -e "${RED}  └── delete: Unjoin Orderers from channel               ${NC}"
 echo -e "${RED}  chaincode:                                             ${NC}"
 echo -e "${RED}  ├── deploy: Deploy Chaincode                           ${NC}"
 echo -e "${RED}  ├── install: Install Chaincode to Peers                ${NC}"
 echo -e "${RED}  ├── approve: Approve Chaincode for an Organization     ${NC}"
 echo -e "${RED}  └── commit: Commit Chaincode to channel                ${NC}"
}

## Parse mode
if [[ $# -lt 1 ]] ; then
  print_help
  exit 0
else
  MODE=$1
  shift
fi

if [ "${MODE}" == "deploy" ]; then
 echo --- Executing with the following:
 echo -e - ORGANIZATION HOSTNAME: ${GREEN}${ORG}${NC}
 echo -e - ORGANIZATION DOMAIN:   ${GREEN}${DOMAIN}${NC}
 echo -e - ORGANIZATION TYPE:     ${GREEN}${NODE_TYPE}${NC}
 echo -e - NUMBER OF PEERS:       ${GREEN}${PEER}${NC}
 echo -e - NUMBER OF USERS:       ${GREEN}${USER}${NC}
 echo -e - CHANNEL NAME:          ${GREEN}${CHANNEL_NAME}${NC}
 echo -e - NAMESPACE:             ${GREEN}${NS}${NC}
 echo -e - FABRIC VERSION:        ${GREEN}${FABRIC_VERSION}${NC}
 echo -e - FABRIC CA VERSION:     ${GREEN}${FABRIC_CA_VERSION}${NC}
 echo -e - COUCHDB VERSION:       ${GREEN}${COUCHDB_VERSION}${NC}
 echo

 if [ -z "${ORG}" ]; then
  echo -e "${RED}ORG environmental variable declared in .env file is NOT Defined${NC}"
  exit 1
 fi

 if [ -z "${DOMAIN}" ]; then
  echo -e "${RED}DOMAIN environmental variable declared in .env file is NOT Defined${NC}"
  exit 1
 fi

 if [ ! -z "${NODE_TYPE}" ]; then
  if [ "${NODE_TYPE}" != "peer" ] && [ "${NODE_TYPE}" != "orderer" ]; then
   echo -e "${RED}NODE_TYPE environmental variable declared in .env file must be either
'peer' or 'orderer'${NC}"
   exit 1
  fi
 else
  echo -e "${RED}NODE_TYPE environmental variable declared in .env file is NOT
```

```
Defined${NC}"
  exit 1
 fi

 if [ ! -z "${FABRIC_CA_SERVER_DB_TYPE}" ]; then
  if [ "${FABRIC_CA_SERVER_DB_TYPE}" != "sqlite3" ] && [ "${FABRIC_CA_SERVER_DB_TYPE}" !=
"postgres" ] && [ "${FABRIC_CA_SERVER_DB_TYPE}" != "mysql" ]; then
   echo -e "${RED}FABRIC_CA_SERVER_DB_TYPE environmental variable declared in .env file
must be either 'sqlite3', 'postgres' or 'mysql'${NC}"
   exit 1
  fi
 else
  echo -e "${RED}FABRIC_CA_SERVER_DB_TYPE environmental variable declared in .env file is
NOT Defined${NC}"
  exit 1
 fi

 if [ ! -z "${PEER}" ]; then
  if [[ ${PEER} ]] && [ ${PEER} -eq ${PEER} 2>/dev/null ]; then
   if [[ ${PEER} -le 0 ]]; then
    echo -e "${RED}PEER environmental variable declared in .env file must be >= 1${NC}"
    exit 1
   fi
   if [ "${NODE_TYPE}" == "orderer" ]; then
    if [[ ${PEER} -le 2 ]]; then
     echo -e "${RED}PEER environmental variable declared in .env file must be equal to or
greater than 3, when NODE_TYPE environmental variable is type 'orderer'${NC}"
     exit 1
    fi
    if [ $((${PEER}%2)) -eq 0 ]; then
     echo -e "${RED}PEER environmental variable declared in .env file must be an odd
number, when NODE_TYPE environmental variable is type 'orderer'${NC}"
     exit 1
    fi
   fi
  else
   echo -e "${RED}PEER environmental variable declared in .env file is not an Integer or is
NOT Defined${NC}"
   exit 1
  fi
 fi

 if [ ! -z "${USER}" ]; then
  if [[ ${USER} ]] && [ ${USER} -eq ${USER} 2>/dev/null ]; then
   if [[ ${USER} -le 0 ]]; then
    echo -e "${RED}USER environmental variable declared in .env file must be >= 1${NC}"
    exit 1
   fi
  else
   echo -e "${RED}USER environmental variable declared in .env is not an Integer or is NOT
Defined${NC}"
   exit 1
  fi
 fi

 if [ ! -z "${CHAINCODE_BUILDER}" ]; then
  if [ "${CHAINCODE_BUILDER}" != "ccaas" ] && [ "${CHAINCODE_BUILDER}" != "k8s" ]; then
   echo -e "${RED}CHAINCODE_BUILDER environmental variable declared in .env file must be
either 'ccaas' or 'k8s'${NC}"
   exit 1
  fi
 else
  echo -e "${RED}CHAINCODE_BUILDER environmental variable declared in .env file is NOT
Defined${NC}"
  exit 1
 fi

 set +e
 kubectl get namespace cert-manager > /dev/null 2>&1
 if [[ $? -ne 0 ]]; then
  echo -e "${GREEN}✓ ${BLUE}Deploying Cert-Manager${NC}"
  kubectl apply -f ${CERT_MANAGER_MANIFEST} > /dev/null 2>&1
  kubectl -n cert-manager rollout status deploy/cert-manager > /dev/null 2>&1 \
```

```
  & spinner "Waiting until Cert-Manager Deployment is completed"
  kubectl -n cert-manager rollout status deploy/cert-manager-cainjector > /dev/null 2>&1 \
  & spinner "Waiting until Cert-Manager Ca-Injector Deployment is completed"
  kubectl -n cert-manager rollout status deploy/cert-manager-webhook > /dev/null 2>&1 \
  & spinner "Waiting until Cert-Manager Webhook Deployment is completed"
  sleep 20s & spinner "Waiting until Cert-Manager Installation is completed"
 else
  echo -e "${GREEN}✓ ${BLUE}Cert-Manager is already deployed. Proceed...${NC}"
 fi
 set -e

 mkdir -p ${WORKING_DIR}/
 (cd ${WORKING_DIR}
  if [ ! -d "${ORG}.${DOMAIN}" ]; then
   echo -e "${RED}X ${BLUE}Directory ${ORG}.${DOMAIN} DOES NOT exists${NC}"
   echo -e "${GREEN}✓ ${BLUE}Creating the missing directory${NC}"
   mkdir -p ${ORG}.${DOMAIN} > /dev/null 2>&1 || true
  fi
  cp -a ${CA_DIR}/intermediateCA ${CA_DIR}/rootCA ${ORG}.${DOMAIN}/
  (cd ${ORG}.${DOMAIN}
   mkdir -p config > /dev/null 2>&1 || true
   cat <(cat ${WORKING_DIR}/config/${NODE_TYPE}.yaml | envsubst) | envsubst >
config/${NODE_TYPE}.yaml
  )
 )

 (cd ${MANIFESTS_DIR}
  echo -e "${GREEN}✓ ${BLUE}Creating namespace ${NS}${NC}"
  cat ns.yaml | envsubst | kubectl apply -f -
  echo -e "${GREEN}✓ ${BLUE}Provisioning volume storage${NC}"
  IS_NFSv4_EXISTS=$(kubectl get namespaces | grep ${NS_NFSv4} | awk '{print $1}')
  if [ -z "${IS_NFSv4_EXISTS}" ]; then
   echo -e "${RED}X ${BLUE}NFSv4 server is not Deployed${NC}"
   exit 1
  fi
  export NFSv4_SERVER_URL=$(kubectl -n ${NS_NFSv4} get service/${ORG}-${NFSv4_SUFFIX} -o
jsonpath='{.spec.clusterIP}')
  cat org-pvc.yaml | envsubst | kubectl -n ${NS} apply -f -
  unset NFSv4_SERVER_URL
 )

 if [ "${NODE_TYPE}" == "peer" ] && [ "${CHAINCODE_BUILDER}" == "k8s" ]; then
  echo -e "${GREEN}✓ ${BLUE}Applying ${ORG^} k8s Chaincode Builder Roles${NC}"
  (cd ${MANIFESTS_DIR}/chaincode/ccBuilder
   cat fabric-builder-role.yaml | kubectl -n ${NS} apply -f -
   echo -e "${GREEN}✓ ${BLUE}Applying k8s ${ORG^} Chaincode Builder Role Bindings${NC}"
   cat fabric-builder-rolebinding.yaml | envsubst | kubectl -n ${NS} apply -f -
   echo -e "${GREEN}✓ ${BLUE}Installing k8s ${ORG^} Chaincode Builders${NC}"
   cat install-k8s-builder.yaml | envsubst | kubectl -n ${NS} apply -f -
  )
 fi

 echo
 echo -e "${GREEN}✓ ${BLUE}Creating ${ORG^} Root CAs${NC}"
 echo
 echo -e "${GREEN}✓ ${BLUE}Creating ${ORG^} TLS Root CA${NC}"
 if [ ! -d "${ROOT_CA_DIR}/tlsca" ]; then
  echo -e "${RED}X ${BLUE}Directory ${ROOT_CA_DIR}/tlsca DOES NOT exists${NC}"
  echo -e "${GREEN}✓ ${BLUE}Creating missing directory${NC}"
  mkdir -p ${ROOT_CA_DIR}/tlsca
 fi
 (cd ${ROOT_CA_DIR}/tlsca
  mkdir -p private certs newcerts crl
  touch index.txt serial
  echo 1000 > serial
  echo 1000 > crlnumber
  openssl ecparam -name secp384r1 -genkey -noout -out private/rca.tls.${ORG}.${DOMAIN}.key
  openssl req -config <(cat openssl_root-tls.cnf | envsubst) -new -x509 -sha256 -extensions
v3_ca -key private/rca.tls.${ORG}.${DOMAIN}.key -out certs/rca.tls.${ORG}.${DOMAIN}.crt
-days 3650 -subj
"/C=GR/ST=Athens/L=Athens/O=${ORG}.${DOMAIN}/OU=/CN=rca.tls.${ORG}.${DOMAIN}"
 )
```

```
 echo -e "${GREEN}✓ ${BLUE}Creating ${ORG^} Identity Root CA${NC}"
 if [ ! -d "${ROOT_CA_DIR}/identity" ]; then
  echo -e "${RED}X ${BLUE}Directory ${ROOT_CA_DIR}/identity DOES NOT exists${NC}"
  echo -e "${GREEN}✓ ${BLUE}Creating missing directory${NC}"
  mkdir -p ${ROOT_CA_DIR}/identity
 fi
 (cd ${ROOT_CA_DIR}/identity
  mkdir -p private certs newcerts crl
  touch index.txt serial
  echo 1000 > serial
  echo 1000 > crlnumber
  openssl ecparam -name secp384r1 -genkey -noout -out
private/rca.identity.${ORG}.${DOMAIN}.key
  openssl req -config <(cat openssl_root-identity.cnf | envsubst) -new -x509 -sha256
-extensions v3_ca -key private/rca.identity.${ORG}.${DOMAIN}.key -out
certs/rca.identity.${ORG}.${DOMAIN}.crt -days 3650 -subj
"/C=GR/ST=Athens/L=Athens/O=${ORG}.${DOMAIN}/OU=/CN=rca.identity.${ORG}.${DOMAIN}"
 )

 echo -e "${GREEN}✓ ${BLUE}Creating and signing TLS Intermediate CA Cert..${NC}"
 if [ ! -d "${INTERMEDIATE_CA_DIR}/tlsca" ]; then
  echo -e "${RED}X ${BLUE}Directory ${INTERMEDIATE_CA_DIR}/tlsca DOES NOT exists${NC}"
  echo -e "${GREEN}✓ ${BLUE}Creating missing directory${NC}"
  mkdir -p ${INTERMEDIATE_CA_DIR}/tlsca
 fi
 (cd ${INTERMEDIATE_CA_DIR}/tlsca
  openssl ecparam -name secp384r1 -genkey -noout -out ica.tls.${ORG}.${DOMAIN}.key
  openssl req -new -sha256 -key ica.tls.${ORG}.${DOMAIN}.key -out
ica.tls.${ORG}.${DOMAIN}.csr -subj
"/C=GR/ST=Athens/L=Athens/O=${ORG}.${DOMAIN}/OU=/CN=ica.tls.${ORG}.${DOMAIN}"
  openssl ca -batch -config <(cat ${ROOT_CA_DIR}/tlsca/openssl_root-tls.cnf | envsubst)
-extensions v3_intermediate_ca -days 1825 -notext -md sha256 -in
ica.tls.${ORG}.${DOMAIN}.csr -out ica.tls.${ORG}.${DOMAIN}.crt
  cat ica.tls.${ORG}.${DOMAIN}.crt ${ROOT_CA_DIR}/tlsca/certs/rca.tls.${ORG}.${DOMAIN}.crt
> chain.tls.${ORG}.${DOMAIN}.crt
 )

 echo -e "${GREEN}✓ ${BLUE}Creating and signing Identity Intermediate CA Cert..${NC}"
 if [ ! -d "${INTERMEDIATE_CA_DIR}/identity" ]; then
  echo -e "${RED}X ${BLUE}Directory ${INTERMEDIATE_CA_DIR}/identity DOES NOT exists${NC}"
  echo -e "${GREEN}✓ ${BLUE}Creating missing directory${NC}"
  mkdir -p ${INTERMEDIATE_CA_DIR}/identity
 fi
 (cd ${INTERMEDIATE_CA_DIR}/identity
  openssl ecparam -name secp384r1 -genkey -noout -out ica.identity.${ORG}.${DOMAIN}.key
  openssl req -new -sha256 -key ica.identity.${ORG}.${DOMAIN}.key -out
ica.identity.${ORG}.${DOMAIN}.csr -subj
"/C=GR/ST=Athens/L=Athens/O=${ORG}.${DOMAIN}/OU=/CN=ica.identity.${ORG}.${DOMAIN}"
  openssl ca -batch -config <(cat ${ROOT_CA_DIR}/identity/openssl_root-identity.cnf |
envsubst) -extensions v3_intermediate_ca -days 1825 -notext -md sha256 -in
ica.identity.${ORG}.${DOMAIN}.csr -out ica.identity.${ORG}.${DOMAIN}.crt
  cat ica.identity.${ORG}.${DOMAIN}.crt
${ROOT_CA_DIR}/identity/certs/rca.identity.${ORG}.${DOMAIN}.crt >
chain.identity.${ORG}.${DOMAIN}.crt
 )

echo -e "${GREEN}✓ ${BLUE}Creating Intermediate TLS CA Cert Secret..${NC}"
 cat <<EOF | kubectl -n ${NS} apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: ${ORG}-intermediate-tlsca-key-pair
  namespace: ${NS}
type: kubernetes.io/tls
data:
  tls.crt: $(cat ${INTERMEDIATE_CA_DIR}/tlsca/chain.tls.${ORG}.${DOMAIN}.crt | base64 -w0)
  tls.key: $(cat ${INTERMEDIATE_CA_DIR}/tlsca/ica.tls.${ORG}.${DOMAIN}.key | base64 -w0)
---
apiVersion: v1
kind: Secret
metadata:
  name: ${ORG}-intermediate-identity-key-pair
  namespace: ${NS}
```

```
type: kubernetes.io/tls
data:
  tls.crt: $(cat ${INTERMEDIATE_CA_DIR}/identity/chain.identity.${ORG}.${DOMAIN}.crt |
base64 -w0)
  tls.key: $(cat ${INTERMEDIATE_CA_DIR}/identity/ica.identity.${ORG}.${DOMAIN}.key | base64
-w0)
EOF

 sleep 10s & spinner "Waiting until Intermediate TLS CA Cert Secret Installation is
Completed"

 echo -e "${GREEN}✓ ${BLUE}Loading Intermediate TLS CA Issuer to CERT MANAGER..${NC}"
 (cd ${INTERMEDIATE_CA_DIR}
  cat ica-tls-issuer.yaml | envsubst | kubectl -n ${NS} apply -f -
  kubectl -n ${NS} wait --timeout=50s --for=condition=Ready issuer/${ORG}-ica-tls-issuer
 )

 echo -e "${GREEN}✓ ${BLUE}Loading Fabric CA Server Secrets${NC}"
 cat <<EOF | envsubst | kubectl -n ${NS} apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: fabric-${ORG}-ca-env-secrets
type: Opaque
data:
  FABRIC_CA_SERVER_USERNAME: $(echo -n ${FABRIC_CA_SERVER_USERNAME} | base64 -w0)
  FABRIC_CA_SERVER_PASSWORD: $(echo -n ${FABRIC_CA_SERVER_PASSWORD} | base64 -w0)
  ADMIN_CREDENTIALS: $(echo -n ${ADMIN_CREDENTIALS} | base64 -w0)
  FABRIC_TLSCA_SERVER_USERNAME: $(echo -n ${FABRIC_TLSCA_SERVER_USERNAME} | base64 -w0)
  FABRIC_TLSCA_SERVER_PASSWORD: $(echo -n ${FABRIC_TLSCA_SERVER_PASSWORD} | base64 -w0)
  ADMIN_CREDENTIALS_TLSCA: $(echo -n ${ADMIN_CREDENTIALS_TLSCA} | base64 -w0)
  IDENTITY_ADMIN_USERNAME: $(echo -n ${IDENTITY_ADMIN_USERNAME} | base64 -w0)
  IDENTITY_ADMIN_PASSWORD: $(echo -n ${IDENTITY_ADMIN_PASSWORD} | base64 -w0)
  IDENTITY_PEER_USERNAME: $(echo -n ${IDENTITY_PEER_USERNAME} | base64 -w0)
  IDENTITY_PEER_PASSWORD: $(echo -n ${IDENTITY_PEER_PASSWORD} | base64 -w0)
  IDENTITY_USER_USERNAME: $(echo -n ${IDENTITY_USER_USERNAME} | base64 -w0)
  IDENTITY_USER_PASSWORD: $(echo -n ${IDENTITY_USER_PASSWORD} | base64 -w0)
EOF

 echo -e "${GREEN}✓ ${BLUE}Loading ${ORG^} Fabric CA Server Config Files${NC}"
 kubectl -n ${NS} create configmap fabric-ca-server-config
--from-file=${WORKING_DIR}/config/ca/fabric-ca
 kubectl -n ${NS} create configmap fabric-tlsca-server-config
--from-file=${WORKING_DIR}/config/ca/fabric-ca/tlsca

 sleep 5s & spinner "Waiting until Fabric CA Server Config Files Installation is Completed"

 export FABRIC_CA_ALPINE_ENVSUBST=docker.io/library/alpine-envsubst:latest
 (cd ${WORKING_DIR}/manifests/fabric-ca
  echo -e "${GREEN}✓ ${BLUE}Loading .env file in configmap${NC}"
  cat org-ca-env.yaml | envsubst | kubectl -n ${NS} --validate=false apply -f -
  echo -e "${GREEN}✓ ${BLUE}Launching ${ORG}.${DOMAIN} Fabric CA Server${NC}"
  cat org-ca.yaml | envsubst | kubectl -n ${NS} apply -f -
  kubectl -n ${NS} rollout status deploy/${ORG}-ca > /dev/null 2>&1 \
  & spinner "Waiting until ${ORG}.${DOMAIN} Fabric CA Server Deployment is Completed"
 )
 (cd ${WORKING_DIR}/${ORG}.${DOMAIN}
  mkdir -p fabric-ca-client/tls > /dev/null 2>&1
  mkdir -p fabric-ca-client/operations > /dev/null 2>&1
  echo fabric-ca-client/tls/ca.crt fabric-ca-client/operations/ca.crt | xargs -n 1 cp
intermediateCA/tlsca/chain.tls.${ORG}.${DOMAIN}.crt
  echo -e "${GREEN}✓ ${BLUE}Retrieving ${ORG}.${DOMAIN} Fabric CA Client Certs${NC}"
  (cd fabric-ca-client/tls
   GET_SECRET=$(kubectl -n ${NS} get secrets ${ORG}-fabric-ca-client-tls-cert -o json)
   echo "${GET_SECRET}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
   echo "${GET_SECRET}" | jq -r .data.\"tls.key\" | base64 -d > client.key
  )
  echo -e "${GREEN}✓ ${BLUE}Retrieving ${ORG}.${DOMAIN} Fabric CA Operations Client
Certs${NC}"
  (cd fabric-ca-client/operations
   GET_SECRET=$(kubectl -n ${NS} get secrets ${ORG}-fabric-ca-operations-client-tls-cert -o
json)
   echo "${GET_SECRET}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
```

```
    echo "${GET_SECRET}" | jq -r .data.\"tls.key\" | base64 -d > client.key
   )

   echo -e "${GREEN}✓ ${BLUE}Adding ${ORG}.${DOMAIN} FQDNs entry to /etc/hosts${NC}"
   CA_FQDNS=ica.${ORG}.${DOMAIN}
   sudo sed -i '$ a\'$(minikube ip)' '${CA_FQDNS}'' /etc/hosts
   sudo sed -i '$ a\'$(minikube ip)' operations.'${CA_FQDNS}'' /etc/hosts

   wait() {
    until [[ $(curl -s --cacert fabric-ca-client/operations/ca.crt
https://operations.${CA_FQDNS}:443/healthz | jq -r .status) == "OK" ]]; do
         sleep 5s
    done
   } & spinner "Checking if ${ORG}.${DOMAIN} Fabric CA Server is Listening"
   echo -e "${GREEN}✓ ${BLUE}Fabric CA Server is Listening${NC}"

   export FABRIC_CA_CLIENT_HOME=${IDENTITY_REGISTRAR_DIR}
   export FABRIC_CA_CLIENT_TLS_CERTFILES=${PWD}/fabric-ca-client/tls
   echo
   echo -e "${GREEN}✓ ${BLUE}Creating ${ORG}.${DOMAIN} ${NODE_TYPE^}s and Users
Certificates in Host${NC}"
   set +x
   echo
   echo -e "${GREEN}✓ ${BLUE}Enrolling ${ORG}.${DOMAIN} Fabric CA Admin${NC}"
   ADMIN_CREDS=$(getSecret ADMIN_CREDENTIALS)
   enrollAdmin ca ${IDENTITY_REGISTRAR_DIR} ${ADMIN_CREDS} ${CA_FQDNS}

   k=0
   while [ ${k} -le $((${PEER}-1)) ]; do
    set +x
    echo
    export k=${k}
    echo -e "${GREEN}✓ ${BLUE}Registering ${NODE_TYPE}${k}.${ORG}.${DOMAIN}${NC}"
    USERNAME=$(echo $(getSecret IDENTITY_PEER_USERNAME) | envsubst)
    PASSWORD=$(echo $(getSecret IDENTITY_PEER_PASSWORD) | envsubst)
    registerIdentity ${IDENTITY_REGISTRAR_DIR} ${USERNAME} ${PASSWORD} ${NODE_TYPE}
${CA_FQDNS}
    k=$((${k}+1))
   done

   k=1
   while [ ${k} -le $((${USER})) ]; do
    set +x
    echo
    export k=${k}
    echo -e "${GREEN}✓ ${BLUE}Registering User${k} for ${ORG}.${DOMAIN}${NC}"
    USERNAME=$(echo $(getSecret IDENTITY_USER_USERNAME) | envsubst)
    PASSWORD=$(echo $(getSecret IDENTITY_USER_PASSWORD) | envsubst)
    registerIdentity ${IDENTITY_REGISTRAR_DIR} ${USERNAME} ${PASSWORD} client ${CA_FQDNS}
    k=$((${k}+1))
   done

   set +x
   echo
   echo -e "${GREEN}✓ ${BLUE}Registering the ${ORG}.${DOMAIN} Admin${NC}"
   USERNAME=$(getSecret IDENTITY_ADMIN_USERNAME)
   PASSWORD=$(getSecret IDENTITY_ADMIN_PASSWORD)
   registerIdentity ${IDENTITY_REGISTRAR_DIR} ${USERNAME} ${PASSWORD} admin ${CA_FQDNS}

   set +x
   echo
   echo -e "${GREEN}✓ ${BLUE}Create local MSP config.yaml${NC}"
   mkdir -p msp/admincerts msp/intermediatecerts msp/tlsintermediatecerts msp/cacerts
msp/tlscacerts msp/signcerts msp/keystore
   mkdir -p tlsca
   mkdir -p ${ADMIN_DIR}/msp/admincerts ${ADMIN_DIR}/msp/tlsintermediatecerts
${ADMIN_DIR}/msp/tlscacerts
   set -x
   echo "NodeOUs:
   Enable: true
   ClientOUIdentifier:
     Certificate: chain.crt
     OrganizationalUnitIdentifier: client
```

```
  PeerOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: chain.crt
    OrganizationalUnitIdentifier: orderer" > msp/config.yaml

  set +x
  echo
  echo -e "${GREEN}✓ ${BLUE}Enrolling the ${ORG}.${DOMAIN} Admin${NC}"
  USERNAME=$(getSecret IDENTITY_ADMIN_USERNAME)
  PASSWORD=$(getSecret IDENTITY_ADMIN_PASSWORD)
  ADMIN_CREDS=${USERNAME}:${PASSWORD}
  enrollAdmin ca ${ADMIN_DIR}/msp ${ADMIN_CREDS} ${CA_FQDNS}

  set +x
  echo
  echo -e "${GREEN}✓ ${BLUE}Enrolling the ${ORG}.${DOMAIN} TLS CA Admin${NC}"
  export FABRIC_CA_CLIENT_HOME=${TLS_REGISTRAR_DIR}
  ADMIN_CREDS=$(getSecret ADMIN_CREDENTIALS_TLSCA)
  enrollAdmin tlsca ${TLS_REGISTRAR_DIR} ${ADMIN_CREDS} ${CA_FQDNS}

  echo -e "${GREEN}✓ ${BLUE}Finalizing ${ORG^} MSP${NC}"
  echo ${ADMIN_DIR}/msp/chain.crt msp/chain.crt | xargs -n 1 cp
intermediateCA/identity/chain.identity.${ORG}.${DOMAIN}.crt
  cp msp/config.yaml ${ADMIN_DIR}/msp/config.yaml
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==2"
intermediateCA/tlsca/chain.tls.${ORG}.${DOMAIN}.crt >
msp/tlscacerts/rca.tls.${ORG}.${DOMAIN}.crt
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==1"
intermediateCA/tlsca/chain.tls.${ORG}.${DOMAIN}.crt > tlsca/ica.tls.${ORG}.${DOMAIN}.crt
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==2"
intermediateCA/identity/chain.identity.${ORG}.${DOMAIN}.crt >
msp/cacerts/rca.identity.${ORG}.${DOMAIN}.crt
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==1"
intermediateCA/identity/chain.identity.${ORG}.${DOMAIN}.crt >
msp/intermediatecerts/ica.identity.${ORG}.${DOMAIN}.crt
  echo msp/tlsintermediatecerts/ ${ADMIN_DIR}/msp/tlsintermediatecerts | xargs -n 1 cp
tlsca/ica.tls.${ORG}.${DOMAIN}.crt
  cp ${ADMIN_DIR}/msp/signcerts/*.pem ${ADMIN_DIR}/msp/admincerts
  cp msp/tlscacerts/rca.tls.${ORG}.${DOMAIN}.crt ${ADMIN_DIR}/msp/tlscacerts
  echo msp/admincerts msp/signcerts | xargs -n 1 cp ${ADMIN_DIR}/msp/signcerts/*.pem
  mv ${ADMIN_DIR}/msp/keystore/*_sk ${ADMIN_DIR}/msp/keystore/key.pem
  cp ${ADMIN_DIR}/msp/keystore/key.pem msp/keystore
  set +x

  sleep 2s

  # SCRIPT DESIGNED TO GENERATE CUSTOM MSP FILESYSTEM AFTER FABRIC-CA INSTALLATION IS
FINISHED.
  echo
  echo -e "${GREEN}✓ ${BLUE}Creating ${ORG}.${DOMAIN} ${NODE_TYPE^}s and Users
Certificates in K8s${NC}"
  echo '
  function enrollIdentity(){
   local CA=$1
   local NODE=$2
   local MSPDIR=$3
   local USERNAME=$4
   local PASSWORD=$5
   local CA_FQDNS=ica.${ORG}.${DOMAIN}
   local CA_PORT=${FABRIC_CA_SERVER_PORT}
   set -x
   fabric-ca-client enroll \
    --caname ${ORG}-${CA} \
    --mspdir ${MSPDIR} \
    --csr.names C=GR,ST=Athens,L=Athens,O=${ORG}.${DOMAIN} \
    --csr.keyrequest.algo ecdsa \
    --csr.keyrequest.size 384 \
    --csr.hosts ${USERNAME},${NODE}-${ORG},${NODE}-${ORG}.${NS},${NODE}.${ORG}.${DOMAIN} \
```

```
    -m ${USERNAME} \
    --url https://${USERNAME}:${PASSWORD}@${CA_FQDNS}:${CA_PORT} \
    --tls.certfiles ${FABRIC_CA_CLIENT_CERTS}/ca.crt \
    --tls.client.certfile ${FABRIC_CA_CLIENT_CERTS}/client.crt \
    --tls.client.keyfile ${FABRIC_CA_CLIENT_CERTS}/client.key
   set +x
  }

  set +x
  echo
  echo -e "'${GREEN}✓ ${BLUE}Create local MSP config.yaml${NC}'"
  set -x
  echo "'"$(cat msp/config.yaml)"'"  > '${ORG_DIR}'/msp/config.yaml
  set +x

  k=0
  while [ "${k}" -le $(('${PEER}'-1)) ]; do
   set +x
   echo
   echo -e "'${GREEN}✓ ${BLUE}Generating the ${NODE_NAME}.${ORG}.${DOMAIN} msp${NC}'"
   mkdir -p '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}
   export k=${k}
   USERNAME=$(echo ${IDENTITY_PEER_USERNAME} | envsubst)
   PASSWORD=$(echo ${IDENTITY_PEER_PASSWORD} | envsubst)
   enrollIdentity ca '${NODE_NAME}' '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp
${USERNAME} ${PASSWORD}
   cp '${ORG_DIR}'/ca/chain.identity.${ORG}.${DOMAIN}.crt
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/chain.crt
   cp '${ORG_DIR}'/msp/config.yaml
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/config.yaml
   k=$((${k}+1))
  done

  k=1
  while [ "${k}" -le $(('${USER}')) ]; do
   set +x
   echo
   echo -e "'${GREEN}✓ ${BLUE}Generating User'${k}' msp${NC}'"
   mkdir -p '${ORG_DIR}'/users/User${k}@${ORG}.${DOMAIN}
   export k=${k}
   USERNAME=$(echo ${IDENTITY_USER_USERNAME} | envsubst)
   PASSWORD=$(echo ${IDENTITY_USER_PASSWORD} | envsubst)
   enrollIdentity ca User${k} '${ORG_DIR}'/users/User${k}@${ORG}.${DOMAIN}/msp ${USERNAME}
${PASSWORD}
   cp '${ORG_DIR}'/ca/chain.identity.${ORG}.${DOMAIN}.crt
'${ORG_DIR}'/users/User${k}@${ORG}.${DOMAIN}/msp/chain.crt
   cp '${ORG_DIR}'/msp/config.yaml
'${ORG_DIR}'/users/User${k}@${ORG}.${DOMAIN}/msp/config.yaml
   k=$((${k}+1))
  done

  k=0
  while [ "${k}" -le $(('${PEER}'-1)) ]; do
   set +x
   echo
   echo -e "'${GREEN}✓ ${BLUE}Generating the ${NODE_NAME}.${ORG}.${DOMAIN} TLS${NC}'"
   mkdir -p '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls
   export k=${k}
   USERNAME=$(echo ${IDENTITY_PEER_USERNAME} | envsubst)
   PASSWORD=$(echo ${IDENTITY_PEER_PASSWORD} | envsubst)
   enrollIdentity tlsca '${NODE_NAME}'
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp ${USERNAME} ${PASSWORD}
   cp '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp/signcerts/*.pem
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/server.crt
   cp '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp/keystore/*
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/server.key
   cat '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp/intermediatecerts/*.pem
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp/cacerts/*.pem >
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/ca.crt
   rm -rf '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/msp
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/tls/*.yaml
   k=$((${k}+1))
  done
```

```
  set +x
  echo
  echo -e "'${GREEN}✓ ${BLUE}Finalizing ${ORG}.${DOMAIN} MSP${NC}'"
  k=0
  while [ "${k}" -le $(('${PEER}'-1)) ]; do
   cp '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/cacerts/*.pem
'${ORG_DIR}'/msp/cacerts/
   cp '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/intermediatecerts/*.pem
'${ORG_DIR}'/msp/intermediatecerts/
   mkdir -p '${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/admincerts
   echo "'"$(cat ${ADMIN_DIR}/msp/signcerts/*.pem)"'" >
'${PEER_DIR}'/'${NODE_NAME}'.${ORG}.${DOMAIN}/msp/admincerts/cert.pem
   k=$(($k+1))
  done
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==2"
'${ORG_DIR}'/tlsca/chain.tls.${ORG}.${DOMAIN}.crt >
'${ORG_DIR}'/msp/tlscacerts/rca.tls.${ORG}.${DOMAIN}.crt
  awk "/-----BEGIN CERTIFICATE-----/{i++}i==1"
'${ORG_DIR}'/tlsca/chain.tls.${ORG}.${DOMAIN}.crt >
'${ORG_DIR}'/tlsca/ica.tls.${ORG}.${DOMAIN}.crt
  echo '${ORG_DIR}'/msp/tlsintermediatecerts/ | xargs -n 1 cp
'${ORG_DIR}'/tlsca/ica.tls.${ORG}.${DOMAIN}.crt
  cp '${ORG_DIR}'/ca/chain.identity.${ORG}.${DOMAIN}.crt '${ORG_DIR}'/msp/chain.crt
  echo "'"$(cat ${ADMIN_DIR}/msp/signcerts/*.pem)"'" > '${ORG_DIR}'/msp/admincerts/cert.pem

  echo -e "'${GREEN}✓ ${BLUE}Finished${NC}'"' | exec kubectl -n ${NS} exec
deploy/${ORG}-ca -i -- /bin/sh
 )
 if [ "${NODE_TYPE}" == "peer" ]; then
  echo
  echo -e "${GREEN}✓ ${BLUE}Loading ${NODE_TYPE^} CouchDB Secrets${NC}"
  cat <<EOF | envsubst | kubectl -n ${NS} apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: ${NODE_TYPE}-couchdb-env-secrets
type: Opaque
data:
  COUCHDB_USER: $(echo -n ${COUCHDB_USER} | base64 -w0)
  COUCHDB_PASSWORD: $(echo -n ${COUCHDB_PASSWORD} | base64 -w0)
EOF
  sleep 10s & spinner "Waiting until ${NODE_TYPE^} CouchDB Secrets Installation is
Completed"

  echo -e "${GREEN}✓ ${BLUE}Deploying ${ORG^} Gateway Service${NC}"
  (cd ${MANIFESTS_DIR}/${NODE_TYPE}
   GATEWAY_FQDNS=gateway.${ORG}.${DOMAIN}
   sudo sed -i '$ a\'$(minikube ip)' '${GATEWAY_FQDNS}'' /etc/hosts
   cat ${NODE_TYPE}-gateway.yaml | envsubst | kubectl -n ${NS} apply -f -
  )
 fi

 echo -e "${GREEN}✓ ${BLUE}Loading ${NODE_TYPE^} Config Files${NC}"
 kubectl -n ${NS} create configmap ${NODE_TYPE}-config
--from-file=${WORKING_DIR}/config/${NODE_TYPE}.yaml
 sleep 5s & spinner "Waiting until ${NODE_TYPE^} Config Files Installation is Completed"
 (cd ${MANIFESTS_DIR}/${NODE_TYPE}
  l=0
  while [ ${l} -le $((${PEER}-1)) ]; do
   export k=${l}
   echo -e "${GREEN}✓ ${BLUE}Loading $(cat <(echo ${NODE_NAME}.${ORG}.${DOMAIN} |
envsubst) | envsubst) ConfigMap${NC}"
   cat <(cat ${NODE_TYPE}-env.yaml | envsubst) | envsubst | kubectl -n ${NS}
--validate=false apply -f -
   sleep 5s & spinner "Waiting until $(cat <(echo ${NODE_NAME}.${ORG}.${DOMAIN} | envsubst)
| envsubst) ConfigMap is Completed"
   echo -e "${GREEN}✓ ${BLUE}Deploying $(cat <(echo ${NODE_NAME}.${ORG}.${DOMAIN} |
envsubst) | envsubst) Manifest${NC}"
   cat <(cat ${NODE_TYPE}.yaml | envsubst) | envsubst | kubectl -n ${NS} apply -f -
   kubectl -n ${NS} rollout status deploy/$(cat <(echo ${NODE_NAME}-${ORG} | envsubst) |
envsubst) > /dev/null 2>&1 \
   & spinner "Waiting until $(cat <(echo ${NODE_NAME}.${ORG}.${DOMAIN} | envsubst) |
```

```
envsubst) Deployment is Completed"
   unset k
   l=$(($l+1))
  done
 )
 sleep 10s & spinner "Waiting until ${NODE_TYPE^} Installation is Completed"
 echo -e "${GREEN}✓ ${BLUE}${NODE_TYPE^} Installation is Completed${NC}"

elif [ "${MODE}" == "artifacts" ]; then
 artifacts_command_group $@
elif [ "${MODE}" == "channel" ]; then
 channel_command_group $@
elif [ "${MODE}" == "chaincode" ]; then

 echo --- Executing with the following:
 echo -e - ORGANIZATION HOSTNAME:  ${GREEN}${ORG}${NC}
 echo -e - ORGANIZATION DOMAIN:    ${GREEN}${DOMAIN}${NC}
 echo -e - ORGANIZATION TYPE:      ${GREEN}${NODE_TYPE}${NC}
 echo -e - NUMBER OF PEERS:        ${GREEN}$(kubectl -n ${NS} get deployments | awk -F
''${ORG}'-ca' '{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep
'peer[[:digit:]]-'${ORG}'*' -o | uniq | wc -l)${NC}
 echo -e - CHANNEL NAME:           ${GREEN}${CHANNEL_NAME}${NC}
 echo -e - NAMESPACE:              ${GREEN}${NS}${NC}
 echo -e - CHAINCODE NAME:         ${GREEN}${CHAINCODE_NAME}${NC}
 echo -e - CHAINCODE GO FILE:      ${GREEN}${CHAINCODE_FILE}${NC}
 echo -e - CHAINCODE VERSION:      ${GREEN}${CHAINCODE_VERSION}${NC}
 echo -e - CHAINCODE LABEL:        ${GREEN}${CHAINCODE_LABEL}${NC}
 echo -e - CHAINCODE TLS DISABLED: ${GREEN}${CHAINCODE_TLS_DISABLED}${NC}
 echo -e - CHAINCODE BUILDER       ${GREEN}${CHAINCODE_BUILDER}${NC}
 echo

 if [ -z "${CHAINCODE_NAME}" ]; then
   echo -e "${RED}CHAINCODE_NAME environmental variable declared in .env file is NOT
Defined${NC}"
   exit 1
 fi

 if [ ! -z "${CHAINCODE_FILE}" ]; then
  if [ ! -f "${CHAINCODE_FILE}" ]; then
   echo -e "${RED}File ${CHAINCODE_FILE} is missing${NC}"
   exit 1
  fi
 else
   echo -e "${RED}CHAINCODE_FILE environmental variable declared in .env file is NOT
Defined${NC}"
   exit 1
 fi
 chaincode_command_group $@
else
 print_help
fi
```

Destroy Network
networkDown.sh
./networkDown.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o errexit
set -o allexport
source .env
. scripts/spinner.sh
set +o allexport
```

```
function networkDown(){
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk '{print $1}')
> /dev/null 2>&1 || true
 REMOVE_SUFFIX=$(echo ${GET_NAMESPACES//-${NFSv4_SUFFIX}/})> /dev/null 2>&1 || true
 REMOVE_PREFIX=$(echo ${REMOVE_SUFFIX//${NETWORK_NAME_PREFIX}/})> /dev/null 2>&1 || true
 ORG_NAMES=$(echo ${REMOVE_PREFIX} | awk '{for (i=1;i<=NF;i++) if (!a[$i]++)
printf("%s%s",$i,FS)}{printf("\n")}') > /dev/null 2>&1 || true
 for ORG_NAME in ${ORG_NAMES}; do
  if [ -d ${WORKING_DIR}/${ORG_NAME}.* ]; then
   rm -rf ${WORKING_DIR}/${ORG_NAME}.* > /dev/null 2>&1 || true
  fi
  for NS in ${GET_NAMESPACES}; do
   if [[ "${NS}" != *"${ORG_NAME}"* ]]; then
    continue;
   fi
   kubectl delete namespace ${NS} > /dev/null 2>&1 || true
   GET_PODS=$(kubectl -n ${NS} get pods --no-headers -o custom-columns=":metadata.name") >
/dev/null 2>&1 || true
   kubectl -n ${NS} delete pod ${GET_PODS} --grace-period=0 --force > /dev/null 2>&1 ||
true
   GET_PVCs=$(kubectl -n ${NS} get persistentvolumeclaims | grep fabric-* | awk '{print
$1}')
   kubectl -n ${NS} patch pvc ${GET_PVCs} -p '{"metadata":{"finalizers":null}}' > /dev/null
2>&1 || true
   kubectl -n ${NS} delete pvc ${GET_PVCs} > /dev/null 2>&1 || true
  done
 done

 GET_PVs=$(kubectl get persistentvolumes | grep fabric-* | awk '{print $1}') > /dev/null
2>&1 || true
 kubectl patch pv ${GET_PVs} -p '{"metadata":{"finalizers":null}}' > /dev/null 2>&1 || true
 kubectl delete persistentvolumes ${GET_PVs} --grace-period=0 --force > /dev/null 2>&1 ||
true

 kubectl delete namespace cert-manager > /dev/null 2>&1 || true
 kubectl delete validatingwebhookconfigurations cert-manager-webhook > /dev/null 2>&1 ||
true
 kubectl delete validatingwebhookconfigurations ingress-nginx-admission > /dev/null 2>&1 ||
true
}

minikube image rm docker.io/library/${CHAINCODE_NAME}:latest > /dev/null 2>&1 || true
docker rmi -f ${CHAINCODE_NAME}:latest > /dev/null 2>&1 || true
echo -e "${GREEN}✓ ${BLUE}Removing FQDNs entries from /etc/hosts${NC}"
sudo sed -i '/'$(minikube ip)'/d' /etc/hosts > /dev/null 2>&1 || true
networkDown > /dev/null 2>&1 & spinner "Tearing Down the Network"
echo -e "${GREEN}✓ ${BLUE}Network is Down${NC}"
```

Spinner Animation

scripts/spinner.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
function spinner() {
 local MSG=$1
 setterm -cursor off
 IFS=$'\n'
 PID=$!

 while [ "$(ps a | awk '{print $1}' | grep ${PID})" ]; do
  for i in / - \\ \|;
  do
        echo -ne "\r${GREEN}${i} ${BLUE}${MSG}${NC}\r"
        sleep 0.2
```

```
  done
 done
 echo -e "\r${GREEN}✓ ${BLUE}${MSG}${NC}\r"
 setterm -cursor on
}
```

Export Peer Context

scripts/exportPeerContext.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o errexit
set -o allexport
source .env
set +o allexport

function exportPeerContext(){
 local ORG=$1
 local ORG_DIR=$2
 local PEER=$3
 local ORG_FQDNS=$(basename ${ORG_DIR})
 export FABRIC_CFG_PATH=${ORG_DIR}/config
 export CORE_PEER_LOCALMSPID=${ORG^}MSP
 export CORE_PEER_MSPCONFIGPATH=${ORG_DIR}/msp
 export CORE_PEER_ADDRESS=${PEER}.${ORG_FQDNS}:443
 export CORE_PEER_TLS_ENABLED=${CORE_PEER_TLS_ENABLED}
 export CORE_PEER_TLS_ROOTCERT_FILE=${ORG_DIR}/peers/${PEER}.${ORG_FQDNS}/client/ca.crt
 export CORE_PEER_TLS_CLIENTAUTHREQUIRED=${CORE_PEER_TLS_CLIENTAUTHREQUIRED}
 export
CORE_PEER_TLS_CLIENTCERT_FILE=${ORG_DIR}/peers/${PEER}.${ORG_FQDNS}/client/client.crt
 export
CORE_PEER_TLS_CLIENTKEY_FILE=${ORG_DIR}/peers/${PEER}.${ORG_FQDNS}/client/client.key
}
```

Create Channel Artifacts

scripts/artifacts.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o errexit
set -o allexport
source .env
set +o allexport

function artifacts_command_group(){
 COMMAND=$1
 shift

 if [ "${COMMAND}" == "create" ]; then
  echo -e "${GREEN}✓ ${BLUE}Creating Channel Artifacts${NC}"
  createArtifacts
  echo -e "${GREEN}✓ ${BLUE}Channel Artifacts are Ready${NC}"
 else
  print_help
  exit 1
 fi
}
```

```
function createArtifacts(){
 if [ "${NODE_TYPE}" != "orderer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by Orderer${NC}"
  exit 1
 fi
 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi

 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 if [[ ${ORG_NAME} == *"orderer"* ]]; then
  ORDERER_NS=${NS}
  ORDERER_ID=${ORG_NAME}
  ORDERER_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')

  (cd ${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}
   rm configtx.yaml > /dev/null 2>&1 || true
   rm -rf orderers > /dev/null 2>&1 || true
  )

  ORDERER_MSPDir=${WORKING_DIR}/${ORG_DIRECTORY}/msp
  ORDERER_TLS=${WORKING_DIR}/${ORG_DIRECTORY}/orderers
  ORDERERS=$(kubectl -n ${ORDERER_NS} get deployments | awk -F ''${ORDERER_ID}'-ca' '{print
$1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)

  for ORDERER in ${ORDERERS}; do
   ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
   ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
   mkdir -p ${ORDERER_TLS}/${ORDERER_FQDNS}/tls > /dev/null 2>&1 || true
   mkdir -p ${ORDERER_TLS}/${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
   mkdir -p ${ORDERER_TLS}/${ORDERER_FQDNS}/operations > /dev/null 2>&1 || true

   (cd ${ORDERER_TLS}/${ORDERER_FQDNS}
    echo tls/ca.crt client/ca.crt operations/ca.crt | xargs -n 1 cp
${WORKING_DIR}/${ORG_DIRECTORY}/intermediateCA/tlsca/chain.tls.${ORDERER_ID}.${ORDERER_DOMA
IN}.crt
   )

   (cd ${ORDERER_TLS}/${ORDERER_FQDNS}/tls
    RETRIEVE_SERVER_TLS=$(echo 'cat
'${SHARE_DIRECTORY}'/fabric/crypto-config/ordererOrganizations/'${ORDERER_ID}'.'${ORDERER_D
OMAIN}'/orderers/'${ORDERER}'.'${ORDERER_ID}'.'${ORDERER_DOMAIN}'/tls/server.crt' | exec
kubectl -n ${ORDERER_NS} exec -q deploy/${ORDERER_ID}-ca -i -- /bin/sh)
    echo "${RETRIEVE_SERVER_TLS}" > server.crt
   )

   (cd ${ORDERER_TLS}/${ORDERER_FQDNS}/client
    RETRIEVE_CLIENT_TLS=$(kubectl -n ${ORDERER_NS} get secrets ${ORDERER}-tls-cert -o json)
    echo "${RETRIEVE_CLIENT_TLS}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
    echo "${RETRIEVE_CLIENT_TLS}" | jq -r .data.\"tls.key\" | base64 -d > client.key
   )

   (cd ${ORDERER_TLS}/${ORDERER_FQDNS}/operations
    RETRIEVE_OPERATIONS_TLS=$(kubectl -n ${ORDERER_NS} get secrets
${ORDERER}-operations-client-tls-cert -o json)
    echo "${RETRIEVE_OPERATIONS_TLS}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
```

```
    echo "${RETRIEVE_OPERATIONS_TLS}" | jq -r .data.\"tls.key\" | base64 -d > client.key
   )

  done
 fi

 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)

 # Remove Orderer's Namespace from the list
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${CHECK_NAMESPACE}})
 arr=()
 for NAMESPACE in ${GET_NAMESPACES}; do
  ORG_NAME=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
  ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
  if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
   echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
   exit 1
  fi
  arr=("${ORG_NAME}" "${arr[@]}")
  SIGNATURE+="'"${ORG_NAME^}MSP.peer"'",
 done

 NUM_OF_ORGS=$(echo "${#arr[@]}")
 if [[ -d "${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts" ]]; then
  rm -rf ${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts > /dev/null 2>&1
|| true
 fi
 mkdir -p ${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts > /dev/null 2>&1
|| true
 (cd ${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts
 echo -n '
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

---
################################################################################
#
#   Section: Organizations
#
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
#
################################################################################
Organizations:

  # SampleOrg defines an MSP using the sampleconfig.  It should never be used
  # in production but may be used as a template for other definitions
  - &OrdererOrg
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: '${ORDERER_ID^}'Org

    # ID to load the MSP definition as
    ID: '${ORDERER_ID^}'MSP

    # MSPDir is the filesystem path which contains the MSP configuration
    MSPDir: '${ORDERER_MSPDir}'

    # Policies defines the set of policies at this level of the config tree
    # For organization policies, their canonical path is usually
    #   /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
      Readers:
        Type: Signature
        Rule: "OR('"'${ORDERER_ID^}MSP.member'"')"
      Writers:
        Type: Signature
        Rule: "OR('"'${ORDERER_ID^}MSP.member'"')"
      Admins:
```

```
        Type: Signature
        Rule: "OR('"'${ORDERER_ID^}MSP.admin'"')"

    OrdererEndpoints:' >> configtx.yaml
    for ORDERER in ${ORDERERS}; do
     echo -n '
      - '${ORDERER}'.'${ORDERER_NS}':'${ORDERER_GENERAL_LISTENPORT} >> configtx.yaml
    done


  for (( idx=${#arr[@]}-1 ; idx>=0 ; idx-- )) ; do
   ORG=$(echo "${arr[idx]}")
   PEERS=$(kubectl -n ${NETWORK_NAME_PREFIX}${ORG} get deployments | awk -F ''${ORG}'-ca'
'{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep
'peer[[:digit:]]-'${ORG}'*' -o | uniq)
   echo -n '

  - &'${ORG^}'
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: '${ORG^}'MSP

    # ID to load the MSP definition as
    ID: '${ORG^}'MSP

    MSPDir: '$(cd ${WORKING_DIR}; DIR=$(ls | grep ${ORG}); cd ${DIR}/msp; pwd)'

    # Policies defines the set of policies at this level of the config tree
    # For organization policies, their canonical path is usually
    #   /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
      Readers:
        Type: Signature
        Rule: "OR('"'${ORG^}MSP.admin'"', '"'${ORG^}MSP.peer'"', '"'${ORG^}MSP.client'"')"
      Writers:
        Type: Signature
        Rule: "OR('"'${ORG^}MSP.admin'"', '"'${ORG^}MSP.client'"')"
      Admins:
        Type: Signature
        Rule: "OR('"'${ORG^}MSP.admin'"')"
      Endorsement:
        Type: Signature
        Rule: "OR('"'${ORG^}MSP.peer'"')"

    # leave this flag set to true.
    AnchorPeers:
      # AnchorPeers defines the location of peers which can be used
      # for cross org gossip communication.  Note, this value is only
      # encoded in the genesis block in the Application section context' >> configtx.yaml
      for PEER in ${PEERS}; do
       echo -n '
      - Host: '${PEER}'.'${NETWORK_NAME_PREFIX}${ORG}'
        Port: '${PEER_GRPC_PORT} >> configtx.yaml
      done
  done
 echo -n '

################################################################################
#
#   SECTION: Capabilities
#
#   - This section defines the capabilities of fabric network. This is a new
#   concept as of v1.1.0 and should not be utilized in mixed networks with
#   v1.0.x peers and orderers.  Capabilities define features which must be
#   present in a fabric binary for that binary to safely participate in the
#   fabric network.  For instance, if a new MSP type is added, newer binaries
#   might recognize and validate the signatures from this type, while older
#   binaries without this support would be unable to validate those
#   transactions.  This could lead to different versions of the fabric binaries
#   having different world states.  Instead, defining a capability for a channel
#   informs those binaries without this capability that they must cease
#   processing transactions until they have been upgraded.  For v1.0.x if any
#   capabilities are defined (including a map with all capabilities turned off)
```

```
#   then the v1.0.x peer will deliberately crash.
#
################################################################################
Capabilities:
  # Channel capabilities apply to both the orderers and the peers and must be
  # supported by both.
  # Set the value of the capability to true to require it.
  Channel: &ChannelCapabilities
    # V2_0 capability ensures that orderers and peers behave according
    # to v2.0 channel capabilities. Orderers and peers from
    # prior releases would behave in an incompatible way, and are therefore
    # not able to participate in channels at v2.0 capability.
    # Prior to enabling V2.0 channel capabilities, ensure that all
    # orderers and peers on a channel are at v2.0.0 or later.
    V2_0: true

  # Orderer capabilities apply only to the orderers, and may be safely
  # used with prior release peers.
  # Set the value of the capability to true to require it.
  Orderer: &OrdererCapabilities
    # V2_0 orderer capability ensures that orderers behave according
    # to v2.0 orderer capabilities. Orderers from
    # prior releases would behave in an incompatible way, and are therefore
    # not able to participate in channels at v2.0 orderer capability.
    # Prior to enabling V2.0 orderer capabilities, ensure that all
    # orderers on channel are at v2.0.0 or later.
    V2_0: true

  # Application capabilities apply only to the peer network, and may be safely
  # used with prior release orderers.
  # Set the value of the capability to true to require it.
  Application: &ApplicationCapabilities
    # V2_0 application capability ensures that peers behave according
    # to v2.0 application capabilities. Peers from
    # prior releases would behave in an incompatible way, and are therefore
    # not able to participate in channels at v2.0 application capability.
    # Prior to enabling V2.0 application capabilities, ensure that all
    # peers on channel are at v2.0.0 or later.
    V2_0: true

################################################################################
#
#   SECTION: Application
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
#
################################################################################
Application: &ApplicationDefaults

  # Organizations is the list of orgs which are defined as participants on
  # the application side of the network
  Organizations:

  # Policies defines the set of policies at this level of the config tree
  # For Application policies, their canonical path is
  #   /Channel/Application/<PolicyName>
  Policies:
    Readers:
      Type: ImplicitMeta
      Rule: "ANY Readers"
    Writers:
      Type: ImplicitMeta
      Rule: "ANY Writers"
    Admins:
      Type: ImplicitMeta
      Rule: "MAJORITY Admins"
    LifecycleEndorsement:
      Type: Signature
      Rule: "OR('${SIGNATURE::-1}')"
    Endorsement:
      Type: Signature
      Rule: "OR('${SIGNATURE::-1}')"
```

```
  Capabilities:
    <<: *ApplicationCapabilities
################################################################################
#
#   SECTION: Orderer
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters
#
################################################################################
Orderer: &OrdererDefaults

  # Orderer Type: The orderer implementation to start
  OrdererType: etcdraft

  EtcdRaft:
    Consenters:' >> configtx.yaml
    for ORDERER in ${ORDERERS}; do
     ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
     ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
     echo -n '
      - Host: '${ORDERER}'-'${ORDERER_ID}'.'${ORDERER_NS}'
        Port: '${ORDERER_GENERAL_LISTENPORT}'
        ClientTLSCert: '${ORDERER_TLS}/${ORDERER_FQDNS}/client/client.crt'
        ServerTLSCert: '${ORDERER_TLS}/${ORDERER_FQDNS}/tls/server.crt >> configtx.yaml
     done
echo -n '

    # Options to be specified for all the etcd/raft nodes. The values here
    # are the defaults for all new channels and can be modified on a
    # per-channel basis via configuration updates.
    Options:
      # TickInterval is the time interval between two Node.Tick invocations.
      #TickInterval: 500ms default
      TickInterval: 2500ms

      # ElectionTick is the number of Node.Tick invocations that must pass
      # between elections. That is, if a follower does not receive any
      # message from the leader of current term before ElectionTick has
      # elapsed, it will become candidate and start an election.
      # ElectionTick must be greater than HeartbeatTick.
      # ElectionTick: 10 default
      ElectionTick: 5

      # HeartbeatTick is the number of Node.Tick invocations that must
      # pass between heartbeats. That is, a leader sends heartbeat
      # messages to maintain its leadership every HeartbeatTick ticks.
      HeartbeatTick: 1

      # MaxInflightBlocks limits the max number of in-flight append messages
      # during optimistic replication phase.
      MaxInflightBlocks: 5

      # SnapshotIntervalSize defines number of bytes per which a snapshot is taken
      SnapshotIntervalSize: 16 MB

  # Batch Timeout: The amount of time to wait before creating a batch
  BatchTimeout: 2s

  # Batch Size: Controls the number of messages batched into a block
  BatchSize:

    # Max Message Count: The maximum number of messages to permit in a batch
    MaxMessageCount: 10

    # Absolute Max Bytes: The absolute maximum number of bytes allowed for
    # the serialized messages in a batch.
    AbsoluteMaxBytes: 99 MB

    # Preferred Max Bytes: The preferred maximum number of bytes allowed for
    # the serialized messages in a batch. A message larger than the preferred
    # max bytes will result in a batch larger than preferred max bytes.
```

```
    PreferredMaxBytes: 512 KB

  # Organizations is the list of orgs which are defined as participants on
  # the orderer side of the network
  Organizations:

  # Policies defines the set of policies at this level of the config tree
  # For Orderer policies, their canonical path is
  #   /Channel/Orderer/<PolicyName>
  Policies:
    Readers:
      Type: ImplicitMeta
      Rule: "ANY Readers"
    Writers:
      Type: ImplicitMeta
      Rule: "ANY Writers"
    Admins:
      Type: ImplicitMeta
      Rule: "MAJORITY Admins"
    # BlockValidation specifies what signatures must be included in the block
    # from the orderer for the peer to validate it.
    BlockValidation:
      Type: ImplicitMeta
      Rule: "ANY Writers"

################################################################################
#
#   CHANNEL
#
#   This section defines the values to encode into a config transaction or
#   genesis block for channel related parameters.
#
################################################################################
Channel: &ChannelDefaults
  # Policies defines the set of policies at this level of the config tree
  # For Channel policies, their canonical path is
  #   /Channel/<PolicyName>
  Policies:
    # Who may invoke the "Deliver" API
    Readers:
      Type: ImplicitMeta
      Rule: "ANY Readers"
    # Who may invoke the "Broadcast" API
    Writers:
      Type: ImplicitMeta
      Rule: "ANY Writers"
    # By default, who may modify elements at this config level
    Admins:
      Type: ImplicitMeta
      Rule: "MAJORITY Admins"

  # Capabilities describes the channel level capabilities, see the
  # dedicated Capabilities section elsewhere in this file for a full
  # description
  Capabilities:
    <<: *ChannelCapabilities

################################################################################
#
#   Profile
#
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
#
################################################################################
Profiles:

  # test network profile with application (not system) channel.' >> configtx.yaml
  echo -n '
  '${NUM_OF_ORGS}'OrgsApplicationGenesis:
    <<: *ChannelDefaults
    Orderer:
      <<: *OrdererDefaults
```

```
      Organizations:
        - *'${ORDERER_ID^}'Org
      Capabilities: *OrdererCapabilities
    Application:
      <<: *ApplicationDefaults
      Organizations:' >> configtx.yaml
      for (( idx=${#arr[@]}-1 ; idx>=0 ; idx-- )) ; do
       ORG=$(echo "${arr[idx]}")
       echo -n '
        - *'${ORG^} >> configtx.yaml
      done
      echo -n '
      Capabilities: *ApplicationCapabilities' >> configtx.yaml
)
 echo -e "${GREEN}✓ ${BLUE}Creating Channel ${CHANNEL_NAME} genesis block${NC}"
 export FABRIC_CFG_PATH=${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts
 configtxgen \
  -profile      ${NUM_OF_ORGS}OrgsApplicationGenesis \
  -channelID    ${CHANNEL_NAME} \
  -outputBlock  ${FABRIC_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb > /dev/null 2>&1 ||
true

 if [[ ! -f "${FABRIC_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
  echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
  exit 1
 fi
}
```

Chaincode Operations

scripts/chaincode.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
set -o errexit

set -o allexport
source .env
. scripts/spinner.sh
. scripts/exportPeerContext.sh
set +o allexport

function chaincode_command_group(){
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi

 COMMAND=$1
 shift

 if [ "${COMMAND}" == "deploy" ]; then
  echo -e "${GREEN}✓ ${BLUE}Deploying Chaincode to K8s${NC}"
  deployChaincode
  echo -e "${GREEN}✓ ${BLUE}Chaincode is Deployed${NC}"
 elif [ "${COMMAND}" == "install" ]; then
  echo -e "${GREEN}✓ ${BLUE}Installing Chaincode to ${ORG}.${DOMAIN} Peers${NC}"
  installChaincode
  echo -e "${GREEN}✓ ${BLUE}Chaincode is Installed to ${ORG}.${DOMAIN} Peers${NC}"
 elif [ "${COMMAND}" == "approve" ]; then
  echo -e "${GREEN}✓ ${BLUE}Approving Chaincode for Organization ${ORG}.${DOMAIN}${NC}"
  approveForMyOrg
  echo -e "${GREEN}✓ ${BLUE}Chaincode is Approved for Organization ${ORG}.${DOMAIN}${NC}"
 elif [ "${COMMAND}" == "commit" ]; then
  echo -e "${GREEN}✓ ${BLUE}Committing Chaincode for Organization ${ORG}.${DOMAIN}${NC}"
  commitChaincode
```

```
  echo -e "${GREEN}✓ ${BLUE}Chaincode is Committed by Organization ${ORG}.${DOMAIN}${NC}"
 else
  print_help
  exit 1
 fi
}

function commitChaincode() {
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi

 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 ORG_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORG_NAME}.'' '{print $2}')

  # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${CHECK_NAMESPACE}})
 for NAMESPACE in ${GET_NAMESPACES}; do
  if [[ $(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}') ==
*"orderer"* ]]; then
   ORDERER_NS=${NAMESPACE}
   ORDERER_ID=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
   ORDERER_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORDERER_ID})
   if [[ ! -d ${WORKING_DIR}/${ORDERER_DIRECTORY} ]]; then
       echo -e "${RED}X ${BLUE}Directory ${ORDERER_DIRECTORY} is Missing${NC}"
   fi
   ORDERER_DOMAIN=$(echo ${ORDERER_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
   ORDERERS=$(kubectl -n ${ORDERER_NS} get deployments | awk -F ''${ORDERER_ID}'-ca'
'{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
   for ORDERER in ${ORDERERS}; do
     ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
     ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
     ORDERER_FILE=${WORKING_DIR}/${ORDERER_DIRECTORY}/orderers/${ORDERER_FQDNS}

     (cd ${WORKING_DIR}/${ORG_DIRECTORY}
      if [[ ! -d "orderers" ]]; then
       echo -e "${RED}X ${BLUE}Orderer Certificates are Missing.${NC}"
       mkdir -p orderers > /dev/null 2>&1 || true
      fi
      (cd orderers
      if [[ ! -d "${ORDERER_FQDNS}/operations" ]]; then
       mkdir -p ${ORDERER_FQDNS}/operations > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/operations/* ${ORDERER_FQDNS}/operations > /dev/null 2>&1 ||
true
      fi
      if [[ ! -d "${ORDERER_FQDNS}/client" ]]; then
       mkdir -p ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/client/* ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
      fi
      )
     )
   done
```

```
   ORDERER_CFG_PATH=${WORKING_DIR}/${ORDERER_DIRECTORY}/channel-artifacts
   if [[ ! -f "${ORDERER_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
    echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
    exit 1
   fi
  fi
 done

 unset ORDERER_FQDNS

 for ORDERER in ${ORDERERS}; do
  ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
  ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${ORDERER_FQDNS} is UP${NC}"
  ORDERER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.key \
   https://operations.${ORDERER_FQDNS}:443/healthz | jq -r .status)
  if [[ "${ORDERER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} is UP${NC}"
   echo -e "${GREEN}✓ ${BLUE}Choosing ${ORDERER_FQDNS} to connect${NC}"
   break
  fi
 done

 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)

 # Remove Orderer's Namespace from the list
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${ORDERER_NS}})
 arr=()
 for NAMESPACE in ${GET_NAMESPACES}; do
  ORG_NAME=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
  arr=("${ORG_NAME}" "${arr[@]}")
  SIGNATURE+="'"${ORG_NAME^}MSP.member"'",
 done

 PEERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORG}'-ca' '{print $1}' | awk '{print
$1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep 'peer[[:digit:]]-'${ORG}'*' -o | uniq)
 for PEER in ${PEERS}; do
  PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
  PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${PEER_FQDNS} is UP${NC}"
  PEER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/client.key \
   https://operations.${PEER_FQDNS}:443/healthz | jq -r .status)
  if [[ "${PEER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Peer ${PEER_FQDNS} is UP${NC}"
   echo -e "${GREEN}✓ ${BLUE}Choosing ${PEER_FQDNS} to connect${NC}"
   break
  fi
 done

 setChaincodeId ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external

 echo -e "${GREEN}✓ ${BLUE}Checking if Chaincode has successfully installed on Peer
${PEER_FQDNS}${NC}"
 exportPeerContext ${ORG} ${WORKING_DIR}/${ORG_DIRECTORY} ${PEER}
 QUERYINSTALLED=$(peer lifecycle chaincode queryinstalled \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz \
  --peerAddresses ${PEER_FQDNS}:443 \
```

```
  --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
  --output json)
 PACKAGE_FOUND=false
 for len in $(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes | jq length); do
  if [[ "$(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes[$(($len-1))].package_id)"
!= "$(echo ${CHAINCODE_ID} | envsubst)" ]]; then
   PACKAGE_FOUND=false;
  else
   PACKAGE_FOUND=true
  fi
 done

 if [[ "${PACKAGE_FOUND}" = false ]]; then
  echo -e "${RED}X ${BLUE}Chaincode is not installed on Peer ${PEER_FQDNS}${NC}"
  exit 1
 fi

 echo -e "${GREEN}✓ ${BLUE}Checking Commit Readiness for ${CHAINCODE_NAME}${NC}"
 CHECKCOMMITREADINESS=$(peer lifecycle chaincode checkcommitreadiness \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --version ${CHAINCODE_VERSION} \
  --signature-policy "OR("${SIGNATURE::-1}")" \
  --sequence ${CHAINCODE_SEQUENCE} \
  --output json)
 PARTICIPANTS=$(echo ${CHECKCOMMITREADINESS} | jq -r '.approvals | keys | .[]' | wc -l)
 PARTICIPANTS_APPROVE=$(echo ${CHECKCOMMITREADINESS} | jq -r '.approvals | values | .[]' |
grep true | wc -l)
 PARTICIPANTS_DISAPROVE=$(echo ${CHECKCOMMITREADINESS} | jq -r '.approvals | values | .[]'
| grep false | wc -l)
 MAJORITY=$(echo $PARTICIPANTS | awk '{print int($1/2)+1}')
 if [[ $PARTICIPANTS_APPROVE -gt $PARTICIPANTS_DISAPROVE ]] && [[ $PARTICIPANTS_APPROVE -ge
$MAJORITY ]]; then
  echo -e "${GREEN}✓ ${BLUE}The majority of the participants has approved the chaincode
${CHAINCODE_NAME}${NC}"
  echo -e "${GREEN}✓ ${BLUE}Chaincode ${CHAINCODE_NAME} is eligible to be commited to the
channel ${CHANNEL_NAME}${NC}"
 else
  echo -e "${RED}X ${BLUE}The majority of the participants has not approved the chaincode
${CHAINCODE_NAME}${NC}"
  echo -e "${GREEN}✓ ${BLUE}Chaincode ${CHAINCODE_NAME} is not eligible to be commited to
the channel ${CHANNEL_NAME}${NC}"
  exit 1
 fi

 echo -e "${GREEN}✓ ${BLUE}Commiting Chaincode ${CHAINCODE_NAME} to channel
${CHANNEL_NAME}${NC}"
 COMMIT="peer lifecycle chaincode commit \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --version ${CHAINCODE_VERSION} \
  --signature-policy ""OR("${SIGNATURE::-1}")"" \
  --sequence ${CHAINCODE_SEQUENCE}"
  for PEER in ${PEERS}; do
   PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
   PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
   COMMIT+=" --peerAddresses ${PEER_FQDNS}:443" \
   COMMIT+=" --tlsRootCertFiles
${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt"
```

```
  done
 COMMIT+=" --waitForEvent"

 COMMITTED=${COMMIT}
 until ${COMMIT}
 do
  if [[ "${COMMITTED}" == *"VALID"* ]]; then
   echo -e "${GREEN}✓ ${BLUE}Chaincode ${CHAINCODE_NAME} is Committed${NC}"
   break;
  fi
 done

 echo -e "${GREEN}✓ ${BLUE}Verifying if Chaincode ${CHAINCODE_NAME} is Committed${NC}"
 QUERYCOMMITTED=$(peer lifecycle chaincode querycommitted \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --peerAddresses ${PEER_FQDNS}:443 \
  --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
  --output json)
 echo ${QUERYCOMMITTED} | jq
}

function approveForMyOrg() {
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi

 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 ORG_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORG_NAME}.'' '{print $2}')

  # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${CHECK_NAMESPACE}})
 for NAMESPACE in ${GET_NAMESPACES}; do
  if [[ $(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}') ==
*"orderer"* ]]; then
   ORDERER_NS=${NAMESPACE}
   ORDERER_ID=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
   ORDERER_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORDERER_ID})
   if [[ ! -d ${WORKING_DIR}/${ORDERER_DIRECTORY} ]]; then
        echo -e "${RED}X ${BLUE}Directory ${ORDERER_DIRECTORY} is Missing${NC}"
   fi
   ORDERER_DOMAIN=$(echo ${ORDERER_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
   ORDERERS=$(kubectl -n ${ORDERER_NS} get deployments | awk -F ''${ORDERER_ID}'-ca'
'{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
   for ORDERER in ${ORDERERS}; do
```

```
    ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
    ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
    ORDERER_FILE=${WORKING_DIR}/${ORDERER_DIRECTORY}/orderers/${ORDERER_FQDNS}

    (cd ${WORKING_DIR}/${ORG_DIRECTORY}
     if [[ ! -d "orderers" ]]; then
      echo -e "${RED}X ${BLUE}Orderer Certificates are Missing.${NC}"
      mkdir -p orderers > /dev/null 2>&1 || true
     fi
     (cd orderers
      if [[ ! -d "${ORDERER_FQDNS}/operations" ]]; then
       mkdir -p ${ORDERER_FQDNS}/operations > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/operations/* ${ORDERER_FQDNS}/operations > /dev/null 2>&1 ||
true
      fi
      if [[ ! -d "${ORDERER_FQDNS}/client" ]]; then
       mkdir -p ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/client/* ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
      fi
     )
    )
   done

   ORDERER_CFG_PATH=${WORKING_DIR}/${ORDERER_DIRECTORY}/channel-artifacts
   if [[ ! -f "${ORDERER_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
    echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
    exit 1
   fi
  fi
 done

 unset ORDERER_FQDNS

 for ORDERER in ${ORDERERS}; do
  ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
  ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${ORDERER_FQDNS} is UP${NC}"
  ORDERER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.key \
   https://operations.${ORDERER_FQDNS}:443/healthz | jq -r .status)
  if [[ "${ORDERER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} is UP${NC}"
   echo -e "${GREEN}✓ ${BLUE}Choosing ${ORDERER_FQDNS} to connect${NC}"
   break
  fi
 done

 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})\
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)

 # Remove Orderer's Namespace from the list
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${ORDERER_NS}})
 arr=()
 for NAMESPACE in ${GET_NAMESPACES}; do
  ORG_NAME=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
  arr=("${ORG_NAME}" "${arr[@]}")
  SIGNATURE+="'"${ORG_NAME^}MSP.member"'",
 done

 PEERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORG}'-ca' '{print $1}'| awk '{print
$1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep 'peer[[:digit:]]-'${ORG}'*' -o | uniq)
 for PEER in ${PEERS}; do
  PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
  PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${PEER_FQDNS} is UP${NC}"
  PEER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/operations/client.key \
   https://operations.${PEER_FQDNS}:443/healthz | jq -r .status)
```

```
  if [[ "${PEER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Peer ${PEER_FQDNS} is UP${NC}"
   echo -e "${GREEN}✓ ${BLUE}Choosing ${PEER_FQDNS} to connect${NC}"
   break
  fi
 done

 setChaincodeId ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external

 echo -e "${GREEN}✓ ${BLUE}Checking if Chaincode has successfully installed on Peer
${PEER_FQDNS}${NC}"
 exportPeerContext ${ORG} ${WORKING_DIR}/${ORG_DIRECTORY} ${PEER}
 QUERYINSTALLED=$(peer lifecycle chaincode queryinstalled \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz \
  --peerAddresses ${PEER_FQDNS}:443 \
  --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
  --output json)
 PACKAGE_FOUND=false
 for len in $(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes | jq length); do
  if [[ "$(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes[$(($len-1))].package_id)"
!= "$(echo ${CHAINCODE_ID} | envsubst)" ]]; then
   PACKAGE_FOUND=false;
  else
   PACKAGE_FOUND=true
  fi
 done

 if [[ "${PACKAGE_FOUND}" = false ]]; then
  echo -e "${RED}X ${BLUE}Chaincode is not installed on Peer ${PEER_FQDNS}${NC}"
  exit 1
 fi

 echo -e "${GREEN}✓ ${BLUE}Checking Commit Readiness for ${CHAINCODE_NAME}${NC}"
 CHECKCOMMITREADINESS=$(peer lifecycle chaincode checkcommitreadiness \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --version ${CHAINCODE_VERSION} \
  --signature-policy "OR("${SIGNATURE::-1}")" \
  --sequence ${CHAINCODE_SEQUENCE} \
  --output json)
 if [[ "$(echo ${CHECKCOMMITREADINESS} | jq -r .approvals.${ORG^}MSP)" = true ]]; then
  echo -e "${RED}X ${BLUE}Chaincode ${CHAINCODE_NAME} is already Approved from
${ORG}.${DOMAIN}${NC}"
  exit 1
 fi

 echo -e "${GREEN}✓ ${BLUE}Approving the Chaincode ${CHAINCODE_NAME}${NC}"
 APPROVE="peer lifecycle chaincode approveformyorg \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --package-id $(echo ${CHAINCODE_ID} | envsubst) \
  --version ${CHAINCODE_VERSION} \
```

```
  --signature-policy ""OR("${SIGNATURE::-1}")"" \
  --sequence ${CHAINCODE_SEQUENCE}"
  for PEER in ${PEERS}; do
   PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
   PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
   APPROVE+=" --peerAddresses ${PEER_FQDNS}:443" \
   APPROVE+=" --tlsRootCertFiles
${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt"
  done
 APPROVE+=" --waitForEvent"

 APPROVED=${APPROVE}
 until ${APPROVE}
 do
  if [[ "${APPROVED}" == *"VALID"* ]]; then
   echo -e "${GREEN}✓ ${BLUE}Chaincode ${CHAINCODE_NAME} is Approved${NC}"
   break;
  fi
 done

 echo -e "${GREEN}✓ ${BLUE}Verifying if Chaincode ${CHAINCODE_NAME} is Approved${NC}"
 QUERYAPPROVED=$(peer lifecycle chaincode queryapproved \
  --tls \
  --clientauth \
  --orderer ${ORDERER_FQDNS}:443 \
  --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
  --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
  --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
  --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
  --channelID ${CHANNEL_NAME} \
  --name ${CHAINCODE_NAME} \
  --sequence ${CHAINCODE_SEQUENCE} \
  --peerAddresses ${PEER_FQDNS}:443 \
  --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
  --output json)
 if [[ "$(echo ${QUERYAPPROVED} | jq -r .source.Type.LocalPackage.package_id)" != "$(echo
${CHAINCODE_ID} | envsubst)" ]]; then
  echo -e "${RED}X ${BLUE}Chaincode ${CHAINCODE_NAME} is not Approved${NC}"
  exit 1
 fi
}

function installChaincode() {
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi

 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 ORG_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORG_NAME}.'' '{print $2}')

 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${CHECK_NAMESPACE}})
```

```
 for NAMESPACE in ${GET_NAMESPACES}; do
  if [[ $(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}') ==
*"orderer"* ]]; then
   ORDERER_NS=${NAMESPACE}
   ORDERER_ID=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
   ORDERER_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORDERER_ID})
   if [[ ! -d ${WORKING_DIR}/${ORDERER_DIRECTORY} ]]; then
       echo -e "${RED}X ${BLUE}Directory ${ORDERER_DIRECTORY} is Missing${NC}"
   fi
   ORDERER_DOMAIN=$(echo ${ORDERER_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
   ORDERERS=$(kubectl -n ${ORDERER_NS} get deployments | awk -F ''${ORDERER_ID}'-ca'
'{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
   for ORDERER in ${ORDERERS}; do
     ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
     ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
     ORDERER_FILE=${WORKING_DIR}/${ORDERER_DIRECTORY}/orderers/${ORDERER_FQDNS}

     (cd ${WORKING_DIR}/${ORG_DIRECTORY}
      if [[ ! -d "orderers" ]]; then
       echo -e "${RED}X ${BLUE}Orderer Certificates are Missing.${NC}"
       mkdir -p orderers > /dev/null 2>&1 || true
      fi
      (cd orderers
       if [[ ! -d "${ORDERER_FQDNS}/operations" ]]; then
        mkdir -p ${ORDERER_FQDNS}/operations > /dev/null 2>&1 || true
        cp -a ${ORDERER_FILE}/operations/* ${ORDERER_FQDNS}/operations > /dev/null 2>&1 ||
true
       fi
       if [[ ! -d "${ORDERER_FQDNS}/client" ]]; then
        mkdir -p ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
        cp -a ${ORDERER_FILE}/client/* ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
       fi
      )
     )
   done

   ORDERER_CFG_PATH=${WORKING_DIR}/${ORDERER_DIRECTORY}/channel-artifacts
   if [[ ! -f "${ORDERER_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
    echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
    exit 1
   fi
  fi
 done

 unset ORDERER_FQDNS

 for ORDERER in ${ORDERERS}; do
  ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
  ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${ORDERER_FQDNS} is UP${NC}"
  ORDERER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.key \
   https://operations.${ORDERER_FQDNS}:443/healthz | jq -r .status)
  if [[ "${ORDERER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} is UP${NC}"
   echo -e "${GREEN}✓ ${BLUE}Choosing ${ORDERER_FQDNS} to connect${NC}"
   break
  fi
 done

 PEERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORG}'-ca' '{print $1}'| awk '{print
$1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep 'peer[[:digit:]]-'${ORG}'*' -o | uniq)
 for PEER in ${PEERS}; do
  PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
  PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
  exportPeerContext ${ORG} ${WORKING_DIR}/${ORG_DIRECTORY} ${PEER}
  echo -e "${GREEN}✓ ${BLUE}Calculating Package ID for Peer ${PEER_FQDNS}${NC}"
  CALCULATEPACKAGEID=$(peer lifecycle chaincode calculatepackageid \
   --tls \
   --clientauth \
   --orderer ${ORDERER_FQDNS}:443 \
```

```
    --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
    --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
    --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
    --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
    ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz --output
json)
  echo ${CALCULATEPACKAGEID} | jq
  PACKAGE_ID=$(echo ${CALCULATEPACKAGEID} | jq -r .package_id)

  echo -e "${GREEN}✓ ${BLUE}Checking if Chaincode has successfully installed in Peer
${PEER_FQDNS}${NC}"
  QUERYINSTALLED=$(peer lifecycle chaincode queryinstalled \
    --tls \
    --clientauth \
    --orderer ${ORDERER_FQDNS}:443 \
    --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
    --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
    --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
    --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
    ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz \
    --peerAddresses ${PEER_FQDNS}:443 \
    --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
    --output json)
  PACKAGE_FOUND=false
  for len in $(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes | jq length); do
    if [[ "$(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes[$(($len-1))].package_id)"
!= "${PACKAGE_ID}" ]]; then
      PACKAGE_FOUND=false;
    else
      PACKAGE_FOUND=true
    fi
  done

  if [[ "${PACKAGE_FOUND}" = true ]]; then
    echo -e "${RED}X ${BLUE}Chaincode is already installed on Peer ${PEER_FQDNS}${NC}"
    exit 1
  fi

  echo -e "${GREEN}✓ ${BLUE}Installing Chaincode to Peer ${PEER_FQDNS}${NC}"
  peer lifecycle chaincode install \
    --tls \
    --clientauth \
    --orderer ${ORDERER_FQDNS}:443 \
    --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
    --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
    --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
    --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
    ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz \
    --peerAddresses ${PEER_FQDNS}:443 \
    --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt

  echo -e "${GREEN}✓ ${BLUE}Checking if Chaincode has successfully installed in Peer
${PEER_FQDNS}${NC}"
  QUERYINSTALLED=$(peer lifecycle chaincode queryinstalled \
    --tls \
    --clientauth \
    --orderer ${ORDERER_FQDNS}:443 \
    --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
    --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
    --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
    --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
    ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/${CHAINCODE_NAME}.tar.gz \
    --peerAddresses ${PEER_FQDNS}:443 \
    --tlsRootCertFiles ${WORKING_DIR}/${ORG_DIRECTORY}/peers/${PEER_FQDNS}/client/ca.crt \
    --output json)
  PACKAGE_FOUND=false
  for len in $(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes | jq length); do
    if [[ "$(echo ${QUERYINSTALLED} | jq -r .installed_chaincodes[$(($len-1))].package_id)"
!= "${PACKAGE_ID}" ]]; then
      PACKAGE_FOUND=false;
    else
      PACKAGE_FOUND=true
    fi
```

```
  done

  if [[ "${PACKAGE_FOUND}" = false ]]; then
   echo -e "${RED}X ${BLUE}Chaincode is not installed in Peer ${PEER_FQDNS}${NC}"
   exit 1
  fi

 done
}

# Function is used to prepare the chaincode given only a GO file holding the business
logic.
function deployChaincode() {
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi
 # Copy cc to the org directory
 (cd ${WORKING_DIR}/${ORG}.${DOMAIN}
  if [ ! -d "peers" ]; then
   echo -e "${RED}X ${BLUE}Directory ${ORG}.${DOMAIN}/peers DOES NOT exists${NC}"
   echo -e "${RED}X ${BLUE}Organization ${ORG}.${DOMAIN} is not a member of any
channel${NC}"
   exit 1
  fi

  rm -rf chaincode-external > /dev/null 2>&1 || true
  mkdir -p chaincode-external > /dev/null 2>&1 || true

  (cd chaincode-external
   if [[ ${CHAINCODE_FILETYPE} != "go" ]]; then
    echo -e "${RED}X ${BLUE}Only Golang is supported at the moment${NC}"
    exit 1
   fi
   echo -e "${GREEN}✓ ${BLUE}Copying Chaincode into ${ORG}.${DOMAIN}${NC}"
   echo "$(cat
${CHAINCODE_EXTERNAL_TEMPLATE}/chaincode/${CHAINCODE_FILENAME}.${CHAINCODE_FILETYPE} |
envsubst)" >> ${CHAINCODE_NAME}.${CHAINCODE_FILETYPE}

   echo -e "${GREEN}✓ ${BLUE}Checking the Chaincode Package Name${NC}"
   PACKAGE_CHECK=$(cat ${CHAINCODE_NAME}.${CHAINCODE_FILETYPE} | grep package | awk -F
'package ' '{print $2}')
   if [[ ${PACKAGE_CHECK} != "main" ]]; then
    # CC Package Name must be 'main'
    echo -e "${RED}X ${BLUE}Chaincode Package Name must be 'main' not
'${PACKAGE_CHECK}'${NC}"
    exit 1
   fi

   echo -e "${GREEN}✓ ${BLUE}Creating Chaincode Dockerfile${NC}"
   cat ${CHAINCODE_EXTERNAL_TEMPLATE}/Dockerfile | envsubst > Dockerfile
   $(docker build -t ${CHAINCODE_NAME}:${CHAINCODE_VERSION} . --rm=true --force-rm
--iidfile Digest --no-cache > /dev/null 2>&1) & spinner "Building ${ORG}.${DOMAIN}
Dockerfile"
   if [[ ! $(cat Digest) ]]; then
    echo -e "${RED}X ${BLUE}Building ${ORG}.${DOMAIN} Dockerfile failed${NC}"
    exit 1
   fi
   echo -e "${GREEN}✓ ${BLUE}Container Digest: ${GREEN}$(cat Digest)${NC}"

   docker ps -a | grep Exit | cut -d ' ' -f 1 | xargs sudo docker rm > /dev/null 2>&1 ||
true
   docker images -a | grep none | awk '{ print $3; }' | xargs docker rmi -f - > /dev/null
2>&1 || true
   docker rmi -f golang:${GO_VERSION}-alpine${ALPINE_VERSION} > /dev/null 2>&1 || true

   echo -e "${GREEN}✓ ${BLUE}Publishing image to Minikube Local docker registry${NC}"
   minikube image load ${CHAINCODE_NAME}:${CHAINCODE_VERSION}

   docker rmi -f ${CHAINCODE_NAME}:${CHAINCODE_VERSION} > /dev/null 2>&1 || true

   export CHAINCODE_IMAGE=docker.io/library/${CHAINCODE_NAME}:${CHAINCODE_VERSION}
   if [ "${CHAINCODE_BUILDER}" == "k8s" ]; then
```

```
        packageAndDeployK8Schaincode
    elif [ "${CHAINCODE_BUILDER}" == "ccaas" ]; then
        packageAndDeployCCAASchaincode
    fi
  )
 )
}

function packageAndDeployK8Schaincode() {
 echo "TODO"
}

function packageAndDeployCCAASchaincode() {
 echo -e "${GREEN}✓ ${BLUE}Packaging CCAAS Chaincode ${CHAINCODE_NAME}${NC}"
 echo
 PEERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORG}'-ca' '{print $1}' | awk '{print
$1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep 'peer[[:digit:]]-'${ORG}'*' -o | uniq)
 for PEER in ${PEERS}; do
  # Add 4 spaces to match the YAML below
  ADD_PEER_COMMON_NAME+="        - ${PEER}-${CHAINCODE_NAME}\n"
 done
 ADD_PEER_COMMON_NAME=$(printf '%b\n' "${ADD_PEER_COMMON_NAME}" | awk /./)

cat <<EOF | envsubst | kubectl -n ${NS} apply -f -
---
# PEER CHAINCODE CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${ORG}-cc-client-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - ${CHAINCODE_NAME}-${ORG}
${ADD_PEER_COMMON_NAME}
  ipAddresses:
    - 127.0.0.1
  secretName: ${ORG}-cc-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
EOF
 sleep 5s & spinner "Deploying Chaincode Client Certificates"

 echo -e "${GREEN}✓ ${BLUE}Retrieving Chaincode Client Certificates${NC}"
 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG})
 mkdir -p ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external/tls > /dev/null 2>&1
 (cd ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external
  RETRIEVE_CC_TLS=$(kubectl -n ${NS} get secrets ${ORG}-cc-client-tls-cert -o json)
  echo "${RETRIEVE_CC_TLS}" | jq -r .data.\"tls.key\" | base64 -d > tls/client.key
  echo "${RETRIEVE_CC_TLS}" | jq -r .data.\"tls.crt\" | base64 -d > tls/client.crt
  cp ${INTERMEDIATE_CA_DIR}/tlsca/chain.tls.${ORG}.${DOMAIN}.crt tls/ca.crt

  export CORE_PEER_CHAINCODE_TLS_CLIENT_KEY=$(awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}'
tls/client.key)
  export CORE_PEER_CHAINCODE_TLS_CLIENT_CERT=$(awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}'
tls/client.crt)
  export CORE_PEER_CHAINCODE_TLS_ROOT_CERT=$(awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}'
tls/ca.crt)

  echo -e "${GREEN}✓ ${BLUE}Creating connection.json for ${ORG}.${DOMAIN} Peers${NC}"
  cat ${CHAINCODE_EXTERNAL_TEMPLATE}/connection.json | envsubst > connection.json

  echo -e "${GREEN}✓ ${BLUE}Creating metadata.json for ${ORG}.${DOMAIN} Peers${NC}"
  cat ${CHAINCODE_EXTERNAL_TEMPLATE}/metadata.json | envsubst > metadata.json

  echo -e "${GREEN}✓ ${BLUE}Creating ${CHAINCODE_NAME}.tar.gz for ${ORG}.${DOMAIN}
Peers${NC}"
```

```
  tar -zcf code.tar.gz connection.json
  tar -zcf ${CHAINCODE_NAME}.tar.gz code.tar.gz metadata.json
  rm code.tar.gz
 )

 unset CORE_PEER_CHAINCODE_TLS_CLIENT_KEY
 unset CORE_PEER_CHAINCODE_TLS_CLIENT_CERT
 unset CORE_PEER_CHAINCODE_TLS_ROOT_CERT

 echo -e "${GREEN}✓ ${BLUE}Exporting Package ID for ${ORG}.${DOMAIN} Peers${NC}"
 setChaincodeId ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external

 echo
 echo -e "${GREEN}✓ ${BLUE}Finding the Deployed Peers for ${ORG}.${DOMAIN}${NC}"
 PEERS_TLS=${WORKING_DIR}/${ORG_DIRECTORY}/peers
 for PEER in ${PEERS}; do
  PEER=$(echo ${PEER} | awk -F '-'${ORG} '{print $1}')
  PEER_FQDNS=${PEER}.${ORG}.${DOMAIN}
  mkdir -p ${PEERS_TLS}/${PEER_FQDNS}/chaincode > /dev/null 2>&1
  (cd ${WORKING_DIR}/${ORG_DIRECTORY}/chaincode-external
   cp -a tls/* ${PEERS_TLS}/${PEER_FQDNS}/chaincode

   echo -e "${GREEN}✓ ${BLUE}Launching $(cat <(echo ${PEER}.${ORG}.${DOMAIN} | envsubst) |
envsubst) Chaincode ConfigMap${NC}"
   export CHAINCODE_SERVER_ADDRESS=$(echo ${CHAINCODE_SERVER_ADDRESS} | sed
's,{{.PEER}},'\${PEER}',g')
   cat <(cat ${MANIFESTS_DIR}/chaincode/chaincode-env.yaml | envsubst) | envsubst | kubectl
-n ${NS} apply -f -
   sleep 5s & spinner "Waiting until $(cat <(echo ${PEER}.${ORG}.${DOMAIN} | envsubst) |
envsubst) Chaincode ConfigMap is Completed"
   echo -e "${GREEN}✓ ${BLUE}Launching $(cat <(echo ${PEER}.${ORG}.${DOMAIN} | envsubst) |
envsubst) Chaincode Manifest${NC}"
   cat <(cat ${MANIFESTS_DIR}/chaincode/chaincode.yaml | envsubst) | envsubst | kubectl -n
${NS} apply -f -
   kubectl -n ${NS} rollout status deploy/${PEER}-${ORG}-${CHAINCODE_NAME} > /dev/null 2>&1
\
        & spinner "Waiting until $(cat <(echo ${PEER}.${ORG}.${DOMAIN} | envsubst) |
envsubst) Chaincode Deployment is Completed"
  )
 done
}

function setChaincodeId() {
 local CC_PATH=$1
 if [[ ! -f "${CC_PATH}/${CHAINCODE_NAME}.tar.gz" ]]; then
  echo -e "${RED}X ${BLUE}The file ${CHAINCODE_NAME}.tar.gz is Missing${NC}"
  exit 1
 fi
 export PACKAGE_ID=$(shasum -a 256 ${CC_PATH}/${CHAINCODE_NAME}.tar.gz | tr -s ' ' | cut -d
' ' -f 1)
}
```

Channel Operations

scripts/channel.sh

```
#!/bin/bash
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
#TODO:: Pick one available orderer and check if is up

set -o errexit
set -o allexport
source .env
. scripts/exportPeerContext.sh
set +o allexport
```

```
function channel_command_group(){
 COMMAND=$1
 shift

 if [ "${COMMAND}" == "create" ]; then
  echo -e "${GREEN}✓ ${BLUE}Creating Channel ${CHANNEL_NAME}${NC}"
  joinOrderers
 elif [ "${COMMAND}" == "join" ]; then
  echo -e "${GREEN}✓ ${BLUE}Joinining ${ORG}.${DOMAIN} Peers to channel
${CHANNEL_NAME}${NC}"
  joinPeers
  echo -e "${GREEN}✓ ${BLUE}Channel ${CHANNEL_NAME} is Updated${NC}"
 elif [ "${COMMAND}" == "delete" ]; then
  echo -e "${GREEN}✓ ${BLUE}Unjoining Orderers from channel ${CHANNEL_NAME}${NC}"
  unjoinOrderers
  echo -e "${GREEN}✓ ${BLUE}Channel ${CHANNEL_NAME} is Deleted${NC}"
 else
  print_help
  exit 1
 fi
}

function joinOrderers(){
 if [ "${NODE_TYPE}" != "orderer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by Orderer${NC}"
  exit 1
 fi
 echo -e "${GREEN}✓ ${BLUE}Joining ${ORG}.${DOMAIN} Orderers to channel
${CHANNEL_NAME}${NC}"
 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi
 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 if [[ ${ORG_NAME} == *"orderer"* ]]; then
  ORDERER_ID=${ORG_NAME}
  ORDERER_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
  export FABRIC_CFG_PATH=${WORKING_DIR}/${ORDERER_ID}.${ORDERER_DOMAIN}/channel-artifacts
  if [[ ! -f "${FABRIC_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
   echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
   exit 1
  fi

  ORDERER_TLS=${WORKING_DIR}/${ORG_DIRECTORY}/orderers
  ORDERERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORDERER_ID}'-ca' '{print $1}' |
awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
  for ORDERER in ${ORDERERS}; do
   ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
   ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
   if [ ! -d "${ORDERER_TLS}/${ORDERER_FQDNS}" ]; then
        mkdir "${ORDERER_TLS}/${ORDERER_FQDNS}/admin" > /dev/null 2>&1
   fi
   (cd ${ORDERER_TLS}/${ORDERER_FQDNS}
    mkdir -p admin > /dev/null 2>&1
    echo -e "${GREEN}✓ ${BLUE}Retrieving ${ORDERER_FQDNS} Admin Certs${NC}"
    RETRIEVE_ADMIN_CERTS=$(kubectl -n ${NS} get secrets ${ORDERER}-admin-client-tls-cert -o
json)
    RETRIEVE_ADMIN_CERT=$(echo ${RETRIEVE_ADMIN_CERTS} | jq -r .data.\"tls.crt\" | base64
```

```
-d)
    echo "${RETRIEVE_ADMIN_CERT}" > admin/admin.crt
    RETRIEVE_ADMIN_KEY=$(echo ${RETRIEVE_ADMIN_CERTS} | jq -r .data.\"tls.key\" | base64
-d)
    echo "${RETRIEVE_ADMIN_KEY}" > admin/admin.key
    cp
${WORKING_DIR}/${ORG_DIRECTORY}/intermediateCA/tlsca/chain.tls.${ORDERER_ID}.${ORDERER_DOMA
IN}.crt admin/ca.crt
   )

   echo -e "${GREEN}✓ ${BLUE}Adding FQDNs entry to /etc/hosts${NC}"
   sudo sed -i '$ a\'$(minikube ip)' admin.'${ORDERER_FQDNS}'' /etc/hosts
   sudo sed -i '$ a\'$(minikube ip)' operations.'${ORDERER_FQDNS}'' /etc/hosts
   sudo sed -i '$ a\'$(minikube ip)' '${ORDERER_FQDNS}'' /etc/hosts

   echo -e "${GREEN}✓ ${BLUE}Joining orderer ${ORDERER_FQDNS} to channel
${CHANNEL_NAME}${NC}"
   JOIN_ORDERER=$(osnadmin channel join \
    --orderer-address admin.${ORDERER_FQDNS}:443 \
    --ca-file ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/ca.crt \
    --client-cert ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.crt \
    --client-key ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.key \
    --channelID ${CHANNEL_NAME} \
    --config-block ${FABRIC_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb)

   CHECK_ORDERER=$(osnadmin channel list \
    --orderer-address admin.${ORDERER_FQDNS}:443 \
    --ca-file ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/ca.crt \
    --client-cert ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.crt \
    --client-key ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.key \
    --channelID ${CHANNEL_NAME} | awk '{if(NR>1)print}')

   if [[ $(echo ${JOIN_ORDERER} | grep Status | awk '{printf $2}') == "201" ]]; then
    echo -e "${GREEN}✓ ${BLUE}Checking if orderer ${ORDERER_FQDNS} has joined the channel
${CHANNEL_NAME}${NC}"
    if [[ $(echo ${CHECK_ORDERER} | jq -r .status) == "active" ]]; then
     echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} has joined the channel
${CHANNEL_NAME}${NC}"
     echo ${CHECK_ORDERER} | jq
    else
     echo -e "${RED}X ${BLUE}Orderer ${ORDERER_FQDNS} has failed to join the channel
${CHANNEL_NAME}${NC}"
    fi
   elif [[ $(echo ${JOIN_ORDERER} | grep Status | awk '{printf $2}') == "405" ]]; then
    echo ${JOIN_ORDERER} | awk '{if(NR>1)print}' | jq
    continue
   fi
  done
 fi
 echo -e "${GREEN}✓ ${BLUE}Channel ${CHANNEL_NAME} is Ready${NC}"
}

function unjoinOrderers(){
 if [ "${NODE_TYPE}" != "orderer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by Orderer${NC}"
  exit 1
 fi
 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi
 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
```

```
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
 fi

 if [[ ${ORG_NAME} == *"orderer"* ]]; then
  ORDERER_ID=${ORG_NAME}
  ORDERER_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
  ORDERER_TLS=${WORKING_DIR}/${ORG_DIRECTORY}/orderers
  ORDERERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORDERER_ID}'-ca' '{print $1}' |
awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
  for ORDERER in ${ORDERERS}; do
   ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
   ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
   if [ ! -d "${ORDERER_TLS}/${ORDERER_FQDNS}" ]; then
    mkdir -p ${ORDERER_TLS}/${ORDERER_FQDNS}/admin > /dev/null 2>&1
   fi

   echo -e "${GREEN}✓ ${BLUE}Unjoining orderer ${ORDERER_FQDNS} from channel
${CHANNEL_NAME}${NC}"
   JOIN_ORDERER=$(osnadmin channel remove \
    --orderer-address admin.${ORDERER_FQDNS}:443 \
    --ca-file ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/ca.crt \
    --client-cert ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.crt \
    --client-key ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.key \
    --channelID ${CHANNEL_NAME})
   if [[ $(echo ${JOIN_ORDERER} | grep Status | awk '{printf $2}') == "204" ]]; then
    echo -e "${GREEN}✓ ${BLUE}Checking if orderer ${ORDERER_FQDNS} has unjoined the
channel ${CHANNEL_NAME}${NC}"
   fi

   echo -e "${GREEN}✓ ${BLUE}Checking if orderer ${ORDERER_FQDNS} has unjoined the channel
${CHANNEL_NAME}${NC}"
   CHECK_ORDERER=$(osnadmin channel list \
    --orderer-address admin.${ORDERER_FQDNS}:443 \
    --ca-file ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/ca.crt \
    --client-cert ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.crt \
    --client-key ${ORDERER_TLS}/${ORDERER_FQDNS}/admin/admin.key \
    --channelID ${CHANNEL_NAME} | awk '{if(NR>1)print}')
   if [[ $(echo ${CHECK_ORDERER} | jq -r .error) == "channel does not exist" ]]; then
    echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} has unjoined the channel
${CHANNEL_NAME}${NC}"
    echo ${CHECK_ORDERER} | jq
   else
    echo -e "${RED}X ${BLUE}Orderer ${ORDERER_FQDNS} has failed to unjoin the channel
${CHANNEL_NAME}${NC}"
   fi

  done
 fi
}

function joinPeers(){
 if [ "${NODE_TYPE}" != "peer" ]; then
  echo -e "${RED}X ${BLUE}This action must be performed only by an Organization${NC}"
  exit 1
 fi
 SUBSTRING="-${NFSv4_SUFFIX}"
 CHECK_NAMESPACE=$(kubectl get namespaces | grep ${NS} | awk -F ''${SUBSTRING}'' '{print
$1}' | awk '{print $1}' | uniq)
 if [[ ! $(echo ${CHECK_NAMESPACE}) ]]; then
  echo -e "${RED}X ${BLUE}Organization ${ORG_NAME}.${DOMAIN} is NOT Deployed${NC}"
  exit 1
 fi
 ORG_NAME=$(echo "${CHECK_NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
 if [[ "${ORG_NAME}" != "${ORG}" ]]; then
  echo -e "${RED}X ${BLUE}Wrong Organization Name is provided in .env${NC}"
  exit 1
 fi

 ORG_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORG_NAME})
 if [[ ! -d ${WORKING_DIR}/${ORG_DIRECTORY} ]]; then
  echo -e "${RED}X ${BLUE}Directory ${ORG_DIRECTORY} is Missing${NC}"
  exit 1
```

```
 fi

 ORG_DOMAIN=$(echo ${ORG_DIRECTORY} | awk -F ''${ORG_NAME}.'' '{print $2}')

 # Retrieve information from Namespaces (the format of namespace is fabric-network-${ORG})
 GET_NAMESPACES=$(kubectl get namespaces | grep ${NETWORK_NAME_PREFIX}* | awk -F
''${SUBSTRING}'' '{print $1}' | awk '{print $1}' | uniq)
 GET_NAMESPACES=$(echo ${GET_NAMESPACES[@]/${CHECK_NAMESPACE}})
 for NAMESPACE in ${GET_NAMESPACES}; do
  if [[ $(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}') ==
*"orderer"* ]]; then
   ORDERER_NS=${NAMESPACE}
   ORDERER_ID=$(echo "${NAMESPACE}" | awk -F ''${NETWORK_NAME_PREFIX}'' '{print $2}')
   ORDERER_DIRECTORY=$(ls ${WORKING_DIR}/ | grep ${ORDERER_ID})
   if [[ ! -d ${WORKING_DIR}/${ORDERER_DIRECTORY} ]]; then
        echo -e "${RED}X ${BLUE}Directory ${ORDERER_DIRECTORY} is Missing${NC}"
   fi
   ORDERER_DOMAIN=$(echo ${ORDERER_DIRECTORY} | awk -F ''${ORDERER_ID}.'' '{print $2}')
   ORDERERS=$(kubectl -n ${ORDERER_NS} get deployments | awk -F ''${ORDERER_ID}'-ca'
'{print $1}' | awk '{print $1}' | awk -F 'NAME' '{print $1}' | awk /./)
   for ORDERER in ${ORDERERS}; do
    ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
    ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
    ORDERER_FILE=${WORKING_DIR}/${ORDERER_DIRECTORY}/orderers/${ORDERER_FQDNS}

    (cd ${WORKING_DIR}/${ORG_DIRECTORY}
     if [[ ! -d "orderers" ]]; then
      echo -e "${RED}X ${BLUE}Orderer Certificates are Missing.${NC}"
      mkdir -p orderers > /dev/null 2>&1 || true
     fi
     (cd orderers
      if [[ ! -d "${ORDERER_FQDNS}/operations" ]]; then
       mkdir -p ${ORDERER_FQDNS}/operations > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/operations/* ${ORDERER_FQDNS}/operations > /dev/null 2>&1 ||
true
      fi
      if [[ ! -d "${ORDERER_FQDNS}/client" ]]; then
       mkdir -p ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
       cp -a ${ORDERER_FILE}/client/* ${ORDERER_FQDNS}/client > /dev/null 2>&1 || true
      fi
     )
    )

   done
   ORDERER_CFG_PATH=${WORKING_DIR}/${ORDERER_DIRECTORY}/channel-artifacts
   if [[ ! -f "${ORDERER_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb" ]]; then
    echo -e "${RED}X ${BLUE}Genesis Block ${CHANNEL_NAME}_genesis_block.pb is Missing${NC}"
    exit 1
   fi
  fi
 done

 unset ORDERER_FQDNS

 (cd ${WORKING_DIR}/${ORG_DIRECTORY}
  rm -rf peers channel-artifacts > /dev/null 2>&1 || true
  mkdir -p peers channel-artifacts > /dev/null 2>&1 || true
  cp ${ORDERER_CFG_PATH}/${CHANNEL_NAME}_genesis_block.pb
channel-artifacts/${CHANNEL_NAME}_genesis_block.pb > /dev/null 2>&1 || true
  mv config/*.yaml config/core.yaml > /dev/null 2>&1 || true
 )

 for ORDERER in ${ORDERERS}; do
  ORDERER=$(echo ${ORDERER} | awk -F '-'${ORDERER_ID} '{print $1}')
  ORDERER_FQDNS=${ORDERER}.${ORDERER_ID}.${ORDERER_DOMAIN}
  echo -e "${GREEN}✓ ${BLUE}Checking if ${ORDERER_FQDNS} is UP${NC}"
  ORDERER_STATUS=$(curl -s \
   --cacert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/ca.crt \
   --cert ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.crt \
   --key ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/operations/client.key \
   https://operations.${ORDERER_FQDNS}:443/healthz | jq -r .status)
  if [[ "${ORDERER_STATUS}" == "OK" ]]; then
   echo -e "${GREEN}✓ ${BLUE}Orderer ${ORDERER_FQDNS} is UP${NC}"
```

```
    echo -e "${GREEN}✓ ${BLUE}Choosing ${ORDERER_FQDNS} to connect${NC}"
    break
   fi
  done

  echo -e "${GREEN}✓ ${BLUE}Joining ${ORG_NAME}.${ORG_DOMAIN} Peers${NC}"
  PEER_TLS=${WORKING_DIR}/${ORG_DIRECTORY}/peers
  PEERS=$(kubectl -n ${NS} get deployments | awk -F ''${ORG_NAME}'-ca' '{print $1}' | awk
'{print $1}' | awk -F 'NAME' '{print $1}' | awk /./ | grep 'peer[[:digit:]]-'${ORG_NAME}'*'
-o | uniq)
  for PEER in ${PEERS}; do
   PEER=$(echo ${PEER} | awk -F '-'${ORG_NAME} '{print $1}')
   PEER_FQDNS=${PEER}.${ORG_NAME}.${ORG_DOMAIN}
   mkdir -p ${PEER_TLS}/${PEER_FQDNS}/client > /dev/null 2>&1 || true
   mkdir -p ${PEER_TLS}/${PEER_FQDNS}/operations > /dev/null 2>&1 || true

   echo -e "${GREEN}✓ ${BLUE}Retrieving ${PEER_FQDNS} Client Certs${NC}"
   (cd ${PEER_TLS}/${PEER_FQDNS}/client
    RETRIEVE_CLIENT_TLS=$(kubectl -n ${NS} get secrets ${PEER}-tls-cert -o json)
    echo "${RETRIEVE_CLIENT_TLS}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
    echo "${RETRIEVE_CLIENT_TLS}" | jq -r .data.\"tls.key\" | base64 -d > client.key
   )

   echo -e "${GREEN}✓ ${BLUE}Retrieving ${PEER_FQDNS} Client Operations Certs${NC}"
   (cd ${PEER_TLS}/${PEER_FQDNS}/operations
    RETRIEVE_OPERATIONS_TLS=$(kubectl -n ${NS} get secrets
${PEER}-operations-client-tls-cert -o json)
    echo "${RETRIEVE_OPERATIONS_TLS}" | jq -r .data.\"tls.crt\" | base64 -d > client.crt
    echo "${RETRIEVE_OPERATIONS_TLS}" | jq -r .data.\"tls.key\" | base64 -d > client.key
   )

   echo ${PEER_TLS}/${PEER_FQDNS}/client/ca.crt ${PEER_TLS}/${PEER_FQDNS}/operations/ca.crt
| xargs -n 1 cp
${WORKING_DIR}/${ORG_DIRECTORY}/intermediateCA/tlsca/chain.tls.${ORG_NAME}.${ORG_DOMAIN}.cr
t

   echo -e "${GREEN}✓ ${BLUE}Adding FQDNs entry to /etc/hosts${NC}"
   sudo sed -i '$ a\'$(minikube ip)' '${PEER_FQDNS}'' /etc/hosts
   sudo sed -i '$ a\'$(minikube ip)' operations.'${PEER_FQDNS}'' /etc/hosts

   echo -e "${GREEN}✓ ${BLUE}Joining peer ${PEER_FQDNS} to channel ${CHANNEL_NAME}${NC}"
   exportPeerContext ${ORG_NAME} ${WORKING_DIR}/${ORG_DIRECTORY} ${PEER}
   JOIN_PEER=$(peer channel join \
    --tls \
    --clientauth \
    --orderer ${ORDERER_FQDNS}:443 \
    --ordererTLSHostnameOverride ${ORDERER_FQDNS} \
    --certfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.crt \
    --keyfile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/client.key \
    --cafile ${WORKING_DIR}/${ORG_DIRECTORY}/orderers/${ORDERER_FQDNS}/client/ca.crt \
    -b ${WORKING_DIR}/${ORG_DIRECTORY}/channel-artifacts/${CHANNEL_NAME}_genesis_block.pb)
   echo -e "${GREEN}✓ ${BLUE}Checking if peer ${PEER_FQDNS} has joined the channel
${CHANNEL_NAME}${NC}"
   CHECK_PEER=$(peer channel list)
   GET_CHANNEL_INFO=$(peer channel getinfo -c ${CHANNEL_NAME} | awk -F 'Blockchain info: '
'{print $2}')
   if [[ $(echo "${CHECK_PEER}" | xargs | grep -o ${CHANNEL_NAME}) ]]; then
    echo -e "${GREEN}✓ ${BLUE}Peer ${PEER_FQDNS} has joined the channel
${CHANNEL_NAME}${NC}"
    echo ${GET_CHANNEL_INFO} | jq
   fi
  done
}
```

Create Local MSP Operations
scripts/createLocalMSP.sh

```
#!/bin/bash
#
```

```
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
# Function to Retrieve secrets
function getSecret(){
 local SECRET_NAME=$1
 kubectl -n ${NS} get secrets fabric-${ORG}-ca-env-secrets -o json | jq -r
.data.${SECRET_NAME} | base64 -d
}

# Function to Fabric CA Server Admin to Host
function enrollAdmin(){
 local CA=$1
 local MSPDIR=$2
 local ADMIN_CREDS=$3
 local CA_HOST=$4
 local CA_PORT=443
 set -x
 fabric-ca-client enroll \
  --caname ${ORG}-${CA} \
  --mspdir ${MSPDIR} \
  --csr.names C=GR,ST=Athens,L=Athens,O=${ORG}.${DOMAIN} \
  --csr.keyrequest.algo ecdsa \
  --csr.keyrequest.size 384 \
  --csr.hosts admin \
  -m admin \
  --url https://${ADMIN_CREDS}@${CA_HOST}:${CA_PORT} \
  --tls.certfiles ${FABRIC_CA_CLIENT_TLS_CERTFILES}/ca.crt \
  --tls.client.certfile ${FABRIC_CA_CLIENT_TLS_CERTFILES}/client.crt \
  --tls.client.keyfile ${FABRIC_CA_CLIENT_TLS_CERTFILES}/client.key
 set +x
}

#Function to Register New Identities against Fabric CA Server from the OS Host
function registerIdentity(){
 local MSPDIR=$1
 local USERNAME=$2
 local PASSWORD=$3
 local NODE_TYPE=$4
 local CA_HOST=$5
 local CA_PORT=443
 set -x
 fabric-ca-client register \
  --caname ${ORG}-ca \
  --mspdir ${MSPDIR} \
  --id.name ${USERNAME} \
  --id.secret ${PASSWORD} \
  --id.type ${NODE_TYPE} \
  --id.attrs "hf.Registrar.Roles=${NODE_TYPE}" \
  --url https://${CA_HOST}:${CA_PORT} \
  --tls.certfiles ${FABRIC_CA_CLIENT_TLS_CERTFILES}/ca.crt \
  --tls.client.certfile ${FABRIC_CA_CLIENT_TLS_CERTFILES}/client.crt \
  --tls.client.keyfile ${FABRIC_CA_CLIENT_TLS_CERTFILES}/client.key
 set +x
}
```

Organization Namespace Manifest

organization/manifest/ns.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
apiVersion: v1
kind: Namespace
```

```
metadata:
  name: ${NS}
```

Organization Persistent Volume Claim Manifest

organization/manifest/org-pvc.yaml

```
#
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: fabric-${ORG}-nfsv4-pv
spec:
  capacity:
    storage: ${STORAGE_RESOURCES}
  accessModes:
    - ReadWriteMany
  storageClassName: ${STORAGE_CLASS_NAME}
  persistentVolumeReclaimPolicy: Retain
  nfs: # URL or IP of NFS Server
    server: ${NFSv4_SERVER_URL}
    path: /
    readOnly: false
  mountOptions:
    - hard
    - nfsvers=4.2
    - port=${NFSv4_PORT}
  claimRef:
    name: fabric-${ORG}
    namespace: ${NS}
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fabric-${ORG}
  namespace: ${NS}
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ${STORAGE_CLASS_NAME}
  resources:
    requests:
      storage: ${STORAGE_RESOURCES}
  volumeName: fabric-${ORG}-nfsv4-pv
```

Organization Root CA OpenSSL Configuration

organization/config/ca/rootCA/

TLSCA	IdentityCA
tlsca/openssl_root-tls.cnf	identity/openssl_root-identity.cnf

```
# OpenSSL root CA configuration file.
# Copy to `/root/ca/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir = ${OPENSSL_TLSCA_CONFIG}
certs = ${OPENSSL_TLSCA_CONFIG}/certs
crl_dir = ${OPENSSL_TLSCA_CONFIG}/crl
new_certs_dir = ${OPENSSL_TLSCA_CONFIG}/newcerts
database = ${OPENSSL_TLSCA_CONFIG}/index.txt
serial = ${OPENSSL_TLSCA_CONFIG}/serial
RANDFILE = ${OPENSSL_TLSCA_CONFIG}/private/.rand

# The root key and root certificate.
private_key =
${OPENSSL_TLSCA_CONFIG}/private/rca.tls.${ORG}.${DOMAIN}
.key
certificate =
${OPENSSL_TLSCA_CONFIG}/certs/rca.tls.${ORG}.${DOMAIN}.c
rt

# For certificate revocation lists.
crlnumber = ${OPENSSL_TLSCA_CONFIG}/crlnumber
crl =
${OPENSSL_TLSCA_CONFIG}/crl/rca.tls.${ORG}.${DOMAIN}.crl
crl_extensions = crl_ext
default_crl_days = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md            = sha512
name_opt              = ca_default
cert_opt              = ca_default
default_days          = 375
preserve              = no
policy                = policy_strict

[ policy_strict ]
# The root CA should only sign intermediate certificates
that match.
# See the POLICY FORMAT section of `man ca`.
countryName           = match
stateOrProvinceName   = match
organizationName      = match
organizationalUnitName  = optional
commonName            = supplied
emailAddress          = optional

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range
of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName           = optional
stateOrProvinceName   = optional
localityName          = optional
organizationName      = optional
organizationalUnitName  = optional
commonName            = supplied
emailAddress          = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits = 4096
distinguished_name = req_distinguished_name
string_mask = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha512

# Extension to add when the -x509 option is used.
x509_extensions = v3_ca

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName = Locality Name
0.organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName = Common Name
emailAddress = Email Address

[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign,
keyCertSign
subjectAltName = @alt_names

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man
x509v3_config`).
subjectKeyIdentifier = hash
```

```
# OpenSSL root CA configuration file.
# Copy to `/root/ca/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir = ${OPENSSL_IDENTITY_CONFIG}
certs = ${OPENSSL_IDENTITY_CONFIG}/certs
crl_dir = ${OPENSSL_IDENTITY_CONFIG}/crl
new_certs_dir = ${OPENSSL_IDENTITY_CONFIG}/newcerts
database = ${OPENSSL_IDENTITY_CONFIG}/index.txt
serial = ${OPENSSL_IDENTITY_CONFIG}/serial
RANDFILE = ${OPENSSL_IDENTITY_CONFIG}/private/.rand

# The root key and root certificate.
private_key =
${OPENSSL_IDENTITY_CONFIG}/private/rca.identity.${ORG}.${DOMAI
N}.key
certificate =
${OPENSSL_IDENTITY_CONFIG}/certs/rca.identity.${ORG}.${DOMAIN}
.crt

# For certificate revocation lists.
crlnumber = ${OPENSSL_IDENTITY_CONFIG}/crlnumber
crl =
${OPENSSL_IDENTITY_CONFIG}/crl/rca.identity.${ORG}.${DOMAIN}.c
rl
crl_extensions = crl_ext
default_crl_days  = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md            = sha512

name_opt              = ca_default
cert_opt              = ca_default
default_days          = 375
preserve              = no
policy                = policy_strict

[ policy_strict ]
# The root CA should only sign intermediate certificates that
match.
# See the POLICY FORMAT section of `man ca`.
countryName           = match
stateOrProvinceName   = match
organizationName      = match
organizationalUnitName  = optional
commonName            = supplied
emailAddress          = optional

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of
certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName           = optional
stateOrProvinceName   = optional
localityName          = optional
organizationName      = optional
organizationalUnitName  = optional
commonName            = supplied
emailAddress          = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits = 4096
distinguished_name  = req_distinguished_name
string_mask = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha512

# Extension to add when the -x509 option is used.
x509_extensions = v3_ca

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName = Locality Name
0.organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName = Common Name
emailAddress = Email Address

[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
subjectAltName = @alt_names

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man
x509v3_config`).
```

```
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign,
keyCertSign
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = rca.tls.${ORG}.${DOMAIN}

[ usr_cert ]
# Extensions for client certificates (`man
x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature,
keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (`man
x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generate Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier = keyid:always

[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

```
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = rca.identity.${ORG}.${DOMAIN}

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature,
keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generate Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

Organization Intermediate TLSCA Issuer Manifest

organization/config/ca/intermediateCA/ica-tls-issuer.yaml

```
#
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ${ORG}-ica-tls-issuer
  namespace: ${NS}
spec:
  ca:
    secretName: ${ORG}-intermediate-tlsca-key-pair
```

Organization Fabric CA Server Config

Identity

organization/config/ca/fabric-ca/fabric-ca-server-config.yaml

```
#############################################################################
#   This is a configuration file for the fabric-ca-server command.
#
#   COMMAND LINE ARGUMENTS AND ENVIRONMENT VARIABLES
#   ------------------------------------------------
#   Each configuration element can be overridden via command line
#   arguments or environment variables.  The precedence for determining
```

```
#   the value of each element is as follows:
#   1) command line argument
#       Examples:
#       a) --port 443
#       To set the listening port
#       b) --ca.keyfile ../mykey.pem
#       To set the "keyfile" element in the "ca" section below;
#       note the "." separator character.
#   2) environment variable
#       Examples:
#       a) FABRIC_CA_SERVER_PORT=443
#       To set the listening port
#       b) FABRIC_CA_SERVER_CA_KEYFILE="../mykey.pem"
#       To set the "keyfile" element in the "ca" section below;
#       note the "_" separator character.
#   3) configuration file
#   4) default value (if there is one)
#       All default values are shown beside each element below.
#
#   FILE NAME ELEMENTS
#   ------------------
#   The value of all fields whose name ends with "file" or "files" are
#   name or names of other files.
#   For example, see "tls.certfile" and "tls.clientauth.certfiles".
#   The value of each of these fields can be a simple filename, a
#   relative path, or an absolute path.  If the value is not an
#   absolute path, it is interpretted as being relative to the location
#   of this configuration file.
#
###############################################################################
# Version of config file
version: ${FABRIC_CA_VERSION}

# Servers listening port (default: 7054)
port: ${FABRIC_CA_SERVER_PORT}

# Cross-Origin Resource Sharing (CORS)
cors:
  enabled: false
  origins:
    - "*"

# Enables debug logging (default: false)
debug: ${FABRIC_CA_SERVER_DEBUG}

# Size limit of an acceptable CRL in bytes (default: 512000)
crlsizelimit: 512000
###############################################################################
#  TLS section for the servers listening port
#
#  The following types are supported for client authentication: NoClientCert,
#  RequestClientCert, RequireAnyClientCert, VerifyClientCertIfGiven,
#  and RequireAndVerifyClientCert.
#
#  Certfiles is a list of root certificate authorities that the server uses
#  when verifying client certificates.
###############################################################################
tls:
  # Enable TLS (default: false)
  enabled: ${FABRIC_CA_SERVER_TLS_ENABLED}
  # TLS for the servers listening port
  certfile: ${FABRIC_CA_SERVER_TLS_CERTFILE}
  keyfile: ${FABRIC_CA_SERVER_TLS_KEYFILE}
  clientauth:
    type: ${FABRIC_CA_SERVER_TLS_CLIENTAUTH_TYPE}
    certfiles: ${FABRIC_CA_SERVER_TLS_CLIENTAUTH_CERTFILES}
###############################################################################
#  The CA section contains information related to the Certificate Authority
#  including the name of the CA, which should be unique for all members
#  of a blockchain network.  It also includes the key and certificate files
#  used when issuing enrollment certificates (ECerts) and transaction
#  certificates (TCerts).
#  The chainfile (if it exists) contains the certificate chain which
```

```
#  should be trusted for this CA, where the 1st in the chain is always the
#  root CA certificate.
#############################################################################
ca:
  # Name of this CA
  name: ${FABRIC_CA_SERVER_CA_NAME}
  # Key file (is only used to import a private key into BCCSP)
  keyfile: ${FABRIC_CA_SERVER_CA_KEYFILE}
  # Certificate file (default: ca-cert.pem)
  certfile: ${FABRIC_CA_SERVER_CA_CERTFILE}
  # Chain file
  chainfile: ${FABRIC_CA_SERVER_CA_CHAINFILE}
#############################################################################
#  The gencrl REST endpoint is used to generate a CRL that contains revoked
#  certificates. This section contains configuration options that are used
#  during gencrl request processing.
#############################################################################
crl:
  # Specifies expiration for the generated CRL. The number of hours
  # specified by this property is added to the UTC time, the resulting time
  # is used to set the "Next Update" date of the CRL.
  expiry: 24h
#############################################################################
#  The registry section controls how the fabric-ca-server does two things:
#  1) authenticates enrollment requests which contain a username and password
#        (also known as an enrollment ID and secret).
#  2) once authenticated, retrieves the identity attribute names and
#        values which the fabric-ca-server optionally puts into TCerts
#        which it issues for transacting on the Hyperledger Fabric blockchain.
#        These attributes are useful for making access control decisions in
#        chaincode.
#  There are two main configuration options:
#  1) The fabric-ca-server is the registry.
#        This is true if "ldap.enabled" in the ldap section below is false.
#  2) An LDAP server is the registry, in which case the fabric-ca-server
#        calls the LDAP server to perform these tasks.
#        This is true if "ldap.enabled" in the ldap section below is true,
#        which means this "registry" section is ignored.
#############################################################################
registry:
  # Maximum number of times a password/secret can be reused for enrollment
  # (default: -1, which means there is no limit)
  maxenrollments: -1

  # Contains identity information which is used when LDAP is disabled
  identities:
     - name: ${FABRIC_CA_SERVER_USERNAME}
       pass: ${FABRIC_CA_SERVER_PASSWORD}
       type: client
       affiliation: ""
       attrs:
          hf.Registrar.Roles: "*"
          hf.Registrar.DelegateRoles: "*"
          hf.Revoker: true
          hf.IntermediateCA: true
          hf.GenCRL: true
          hf.Registrar.Attributes: "*"
          hf.AffiliationMgr: true

#############################################################################
#  Database section
#  Supported types are: "sqlite3", "postgres", and "mysql".
#  The datasource value depends on the type.
#  If the type is "sqlite3", the datasource value is a file name to use
#  as the database store.  Since "sqlite3" is an embedded database, it
#  may not be used if you want to run the fabric-ca-server in a cluster.
#  To run the fabric-ca-server in a cluster, you must choose "postgres"
#  or "mysql".
#############################################################################
db:
  type: ${FABRIC_CA_SERVER_DB_TYPE}
  datasource: ${FABRIC_CA_SERVER_DB_DATASOURCE}
  tls:
```

```
    enabled: ${FABRIC_CA_SERVER_DB_TLS_ENABLED}
    certfiles: ${FABRIC_CA_SERVER_DB_TLS_CERTFILES}
    client:
      certfile: ${FABRIC_CA_SERVER_DB_TLS_CLIENT_CERTFILE}
      keyfile: ${FABRIC_CA_SERVER_DB_TLS_CLIENT_KEYFILE}

###########################################################################
#  LDAP section
#  If LDAP is enabled, the fabric-ca-server calls LDAP to:
#  1) authenticate enrollment ID and secret (i.e. username and password)
#       for enrollment requests;
#  2) To retrieve identity attributes
###########################################################################
ldap:
   # Enables or disables the LDAP client (default: false)
   # If this is set to true, the "registry" section is ignored.
   enabled: false
   # The URL of the LDAP server
   url: ldap://<adminDN>:<adminPassword>@<host>:<port>/<base>
   # TLS configuration for the client connection to the LDAP server
   tls:
      certfiles:
      client:
      certfile:
      keyfile:
   # Attribute related configuration for mapping from LDAP entries to Fabric CA attributes
   attribute:
      # 'names' is an array of strings containing the LDAP attribute names which are
      # requested from the LDAP server for an LDAP identity's entry
      names: ['uid','member']
      # The 'converters' section is used to convert an LDAP entry to the value of
      # a fabric CA attribute.
      # For example, the following converts an LDAP 'uid' attribute
      # whose value begins with 'revoker' to a fabric CA attribute
      # named "hf.Revoker" with a value of "true" (because the boolean expression
      # evaluates to true).
        #       converters:
        #       - name: hf.Revoker
        #       value: attr("uid") =~ "revoker*"
        converters:
           - name:
             value:
        # The 'maps' section contains named maps which may be referenced by the 'map'
        # function in the 'converters' section to map LDAP responses to arbitrary values.
        # For example, assume a user has an LDAP attribute named 'member' which has
multiple
        # values which are each a distinguished name (i.e. a DN). For simplicity, assume
the
        # values of the 'member' attribute are 'dn1', 'dn2', and 'dn3'.
        # Further assume the following configuration.
        #       converters:
        #       - name: hf.Registrar.Roles
        #       value: map(attr("member"),"groups")
        #       maps:
        #       groups:
        #       - name: dn1
        #               value: peer
        #       - name: dn2
        #               value: client
        # The value of the user's 'hf.Registrar.Roles' attribute is then computed to be
        # "peer,client,dn3".  This is because the value of 'attr("member")' is
        # "dn1,dn2,dn3", and the call to 'map' with a 2nd argument of
        # "group" replaces "dn1" with "peer" and "dn2" with "client".
        maps:
           groups:
              - name:
                value:
###########################################################################
# Affiliations section. Fabric CA server can be bootstrapped with the
# affiliations specified in this section. Affiliations are specified as maps.
# For example:
#   businessunit1:
#       department1:
```

```
#       - team1
#   businessunit2:
#       - department2
#       - department3
#
# Affiliations are hierarchical in nature. In the above example,
# department1 (used as businessunit1.department1) is the child of businessunit1.
# team1 (used as businessunit1.department1.team1) is the child of department1.
# department2 (used as businessunit2.department2) and department3
(businessunit2.department3)
# are children of businessunit2.
# Note: Affiliations are case sensitive except for the non-leaf affiliations
# (like businessunit1, department1, businessunit2) that are specified in the configuration
file,
# which are always stored in lower case.
#############################################################################
affiliations:
   ${ORG}:
        - department1
        - department2
#############################################################################
#  Signing section
#
#  The "default" subsection is used to sign enrollment certificates;
#  the default expiration ("expiry" field) is "8760h", which is 1 year in hours.
#
#  The "ca" profile subsection is used to sign intermediate CA certificates;
#  the default expiration ("expiry" field) is "43800h" which is 5 years in hours.
#  Note that "isca" is true, meaning that it issues a CA certificate.
#  A maxpathlen of 0 means that the intermediate CA cannot issue other
#  intermediate CA certificates, though it can still issue end entity certificates.
#  (See RFC 5280, section 4.2.1.9)
#
#  The "tls" profile subsection is used to sign TLS certificate requests;
#  the default expiration ("expiry" field) is "8760h", which is 1 year in hours.
#############################################################################
signing:
    default:
      usage:
        - digital signature
      expiry: 8760h
      backdate: 1s
    profiles:
      ca:
        usage:
           - cert sign
           - crl sign
        expiry: 43800h
        caconstraint:
          isca: true
          maxpathlen: 0
      tls:
        usage:
           - signing
           - key encipherment
           - server auth
           - client auth
           - key agreement
        expiry: 8760h

#############################################################################
#  Certificate Signing Request (CSR) section.
#  This controls the creation of the root CA certificate.
#  The expiration for the root CA certificate is configured with the
#  "ca.expiry" field below, whose default value is "131400h" which is
#  15 years in hours.
#  The pathlength field is used to limit CA certificate hierarchy as described
#  in section 4.2.1.9 of RFC 5280.
#  Examples:
#  1) No pathlength value means no limit is requested.
#  2) pathlength == 1 means a limit of 1 is requested which is the default for
#       a root CA.  This means the root CA can issue intermediate CA certificates,
#       but these intermediate CAs may not in turn issue other CA certificates
```

```
#       though they can still issue end entity certificates.
#  3) pathlength == 0 means a limit of 0 is requested;
#       this is the default for an intermediate CA, which means it can not issue
#       CA certificates though it can still issue end entity certificates.
###########################################################################
csr:
   cn: fabric-ca-server
   names:
       - C: US
         ST: "North Carolina"
         L:
         O: Hyperledger
         OU: Fabric
   hosts:
    - localhost
    - 127.0.0.1
    - ${ORG}-ca
    - ${ORG}-ca.${NS}.svc.cluster.local
    - ica.${ORG}.${DOMAIN}
   ca:
     expiry: 131400h
     pathlength: 1
###########################################################################
# BCCSP (BlockChain Crypto Service Provider) section is used to select which
# crypto library implementation to use
###########################################################################
bccsp:
    default: SW
    sw:
        hash: SHA2
        security: 256
        filekeystore:
            # The directory used for the software file-based keystore
             keystore: msp/keystore

###########################################################################
# Multi CA section
#
# Each Fabric CA server contains one CA by default.  This section is used
# to configure multiple CAs in a single server.
#
# 1) --cacount <number-of-CAs>
# Automatically generate <number-of-CAs> non-default CAs.  The names of these
# additional CAs are "ca1", "ca2", ... "caN", where "N" is <number-of-CAs>
# This is particularly useful in a development environment to quickly set up
# multiple CAs. Note that, this config option is not applicable to intermediate CA server
# i.e., Fabric CA server that is started with intermediate.parentserver.url config
# option (-u command line option)
#
# 2) --cafiles <CA-config-files>
# For each CA config file in the list, generate a separate signing CA.  Each CA
# config file in this list MAY contain all of the same elements as are found in
# the server config file except port, debug, and tls sections.
#
# Examples:
# fabric-ca-server start -b admin:adminpw --cacount 2
#
# fabric-ca-server start -b admin:adminpw --cafiles ca/ca1/fabric-ca-server-config.yaml
# --cafiles ca/ca2/fabric-ca-server-config.yaml
#
###########################################################################
cacount:

cafiles:
- ${FABRIC_CA_SERVER_HOME}/tlsca/fabric-ca-server-config.yaml

###########################################################################
# Intermediate CA section
#
# The relationship between servers and CAs is as follows:
#   1) A single server process may contain or function as one or more CAs.
#       This is configured by the "Multi CA section" above.
#   2) Each CA is either a root CA or an intermediate CA.
```

```
#   3) Each intermediate CA has a parent CA which is either a root CA or another
intermediate CA.
#
# This section pertains to configuration of #2 and #3.
# If the "intermediate.parentserver.url" property is set,
# then this is an intermediate CA with the specified parent
# CA.
#
# parentserver section
#       url - The URL of the parent server
#       caname - Name of the CA to enroll within the server
#
# enrollment section used to enroll intermediate CA with parent CA
#       profile - Name of the signing profile to use in issuing the certificate
#       label - Label to use in HSM operations
#
# tls section for secure socket connection
#   certfiles - PEM-encoded list of trusted root certificate files
#   client:
#       certfile - PEM-encoded certificate file for when client authentication
#       is enabled on server
#       keyfile - PEM-encoded key file for when client authentication
#       is enabled on server
###############################################################################
intermediate:
  parentserver:
    url:
    caname:

  enrollment:
    hosts:
    profile:
    label:

  tls:
    certfiles:
    client:
      certfile:
      keyfile:

###############################################################################
#
#       Operations section
#
###############################################################################
operations:
    # host and port for the operations server
    listenAddress:
    # TLS configuration for the operations endpoint
    tls:
        # TLS enabled
        enabled:
        # path to PEM encoded server certificate for the operations server
        cert:
            file:
        # path to PEM encoded server key for the operations server
        key:
            file:
        # require client certificate authentication to access all resources
        clientAuthRequired:
        # paths to PEM encoded ca certificates to trust for client authentication
        clientRootCAs:
            files: []

###############################################################################
#
#       Metrics section
#
###############################################################################
metrics:
    # statsd, prometheus, or disabled
    provider: disabled
    # statsd configuration
```

```
    statsd:
        # network type: tcp or udp
        network: udp
        # statsd server address
        address: 127.0.0.1:8125
        # the interval at which locally cached counters and gauges are pushsed
        # to statsd; timings are pushed immediately
        writeInterval: 10s
        # prefix is prepended to all emitted statsd merics
        prefix: server
```

TLSCA

organization/config/ca/fabric-ca/tlsca/fabric-ca-server-config.yaml

```
############################################################################
#   This is a configuration file for the fabric-ca-server command.
#
#   COMMAND LINE ARGUMENTS AND ENVIRONMENT VARIABLES
#   ------------------------------------------------
#   Each configuration element can be overridden via command line
#   arguments or environment variables.  The precedence for determining
#   the value of each element is as follows:
#   1) command line argument
#       Examples:
#       a) --port 443
#       To set the listening port
#       b) --ca.keyfile ../mykey.pem
#       To set the "keyfile" element in the "ca" section below;
#       note the "." separator character.
#   2) environment variable
#       Examples:
#       a) FABRIC_CA_SERVER_PORT=443
#       To set the listening port
#       b) FABRIC_CA_SERVER_CA_KEYFILE="../mykey.pem"
#       To set the "keyfile" element in the "ca" section below;
#       note the "_" separator character.
#   3) configuration file
#   4) default value (if there is one)
#       All default values are shown beside each element below.
#
#   FILE NAME ELEMENTS
#   ------------------
#   The value of all fields whose name ends with "file" or "files" are
#   name or names of other files.
#   For example, see "tls.certfile" and "tls.clientauth.certfiles".
#   The value of each of these fields can be a simple filename, a
#   relative path, or an absolute path.  If the value is not an
#   absolute path, it is interpretted as being relative to the location
#   of this configuration file.
#
############################################################################

# Version of config file
version: ${FABRIC_CA_VERSION}

# Servers listening port (default: 7054)
port: ${FABRIC_CA_SERVER_PORT}

# Cross-Origin Resource Sharing (CORS)
cors:
  enabled: false
  origins:
        - "*"

# Enables debug logging (default: false)
debug: ${FABRIC_CA_SERVER_DEBUG}

# Size limit of an acceptable CRL in bytes (default: 512000)
crlsizelimit: 512000

############################################################################
```

```
#  The CA section contains information related to the Certificate Authority
#  including the name of the CA, which should be unique for all members
#  of a blockchain network.  It also includes the key and certificate files
#  used when issuing enrollment certificates (ECerts) and transaction
#  certificates (TCerts).
#  The chainfile (if it exists) contains the certificate chain which
#  should be trusted for this CA, where the 1st in the chain is always the
#  root CA certificate.
############################################################################
ca:
  # Name of this CA
  name: ${FABRIC_TLSCA_SERVER_CA_NAME}
  # Key file (is only used to import a private key into BCCSP)
  keyfile: ${FABRIC_TLSCA_SERVER_CA_KEYFILE}
  # Certificate file (default: ca-cert.pem)
  certfile: ${FABRIC_TLSCA_SERVER_CA_CERTFILE}
  # Chain file
  chainfile: ${FABRIC_TLSCA_SERVER_CA_CHAINFILE}

############################################################################
#  The gencrl REST endpoint is used to generate a CRL that contains revoked
#  certificates. This section contains configuration options that are used
#  during gencrl request processing.
############################################################################
crl:
  # Specifies expiration for the generated CRL. The number of hours
  # specified by this property is added to the UTC time, the resulting time
  # is used to set the "Next Update" date of the CRL.
  expiry: 24h

############################################################################
#  The registry section controls how the fabric-ca-server does two things:
#  1) authenticates enrollment requests which contain a username and password
#       (also known as an enrollment ID and secret).
#  2) once authenticated, retrieves the identity attribute names and
#       values which the fabric-ca-server optionally puts into TCerts
#       which it issues for transacting on the Hyperledger Fabric blockchain.
#       These attributes are useful for making access control decisions in
#       chaincode.
#  There are two main configuration options:
#  1) The fabric-ca-server is the registry.
#       This is true if "ldap.enabled" in the ldap section below is false.
#  2) An LDAP server is the registry, in which case the fabric-ca-server
#       calls the LDAP server to perform these tasks.
#       This is true if "ldap.enabled" in the ldap section below is true,
#       which means this "registry" section is ignored.
############################################################################
registry:
  # Maximum number of times a password/secret can be reused for enrollment
  # (default: -1, which means there is no limit)
  maxenrollments: -1

  # Contains identity information which is used when LDAP is disabled
  identities:
     - name: ${FABRIC_TLSCA_SERVER_USERNAME}
       pass: ${FABRIC_TLSCA_SERVER_PASSWORD}
       type: client
       affiliation: ""
       attrs:
          hf.Registrar.Roles: "*"
          hf.Registrar.DelegateRoles: "*"
          hf.Revoker: true
          hf.IntermediateCA: true
          hf.GenCRL: true
          hf.Registrar.Attributes: "*"
          hf.AffiliationMgr: true

############################################################################
#  Database section
#  Supported types are: "sqlite3", "postgres", and "mysql".
#  The datasource value depends on the type.
#  If the type is "sqlite3", the datasource value is a file name to use
#  as the database store.  Since "sqlite3" is an embedded database, it
```

```
#  may not be used if you want to run the fabric-ca-server in a cluster.
#  To run the fabric-ca-server in a cluster, you must choose "postgres"
#  or "mysql".
###############################################################################
db:
  type: ${FABRIC_TLSCA_SERVER_DB_TYPE}
  datasource: ${FABRIC_TLSCA_SERVER_DB_DATASOURCE}
  tls:
    enabled: ${FABRIC_TLSCA_SERVER_DB_TLS_ENABLED}
    certfiles: ${FABRIC_TLSCA_SERVER_DB_TLS_CERTFILES}
    client:
      certfile: ${FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_CERTFILE}
      keyfile: ${FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_KEYFILE}

###############################################################################
#  LDAP section
#  If LDAP is enabled, the fabric-ca-server calls LDAP to:
#  1) authenticate enrollment ID and secret (i.e. username and password)
#       for enrollment requests;
#  2) To retrieve identity attributes
###############################################################################
ldap:
   # Enables or disables the LDAP client (default: false)
   # If this is set to true, the "registry" section is ignored.
   enabled: false
   # The URL of the LDAP server
   url: ldap://<adminDN>:<adminPassword>@<host>:<port>/<base>
   # TLS configuration for the client connection to the LDAP server
   tls:
      certfiles:
      client:
         certfile:
         keyfile:
   # Attribute related configuration for mapping from LDAP entries to Fabric CA attributes
   attribute:
      # 'names' is an array of strings containing the LDAP attribute names which are
      # requested from the LDAP server for an LDAP identity's entry
      names: ['uid','member']
      # The 'converters' section is used to convert an LDAP entry to the value of
      # a fabric CA attribute.
      # For example, the following converts an LDAP 'uid' attribute
      # whose value begins with 'revoker' to a fabric CA attribute
      # named "hf.Revoker" with a value of "true" (because the boolean expression
      # evaluates to true).

      #    converters:
      #       - name: hf.Revoker
      #         value: attr("uid") =~ "revoker*"
      converters:
         - name:
           value:
      # The 'maps' section contains named maps which may be referenced by the 'map'
      # function in the 'converters' section to map LDAP responses to arbitrary values.
      # For example, assume a user has an LDAP attribute named 'member' which has multiple
      # values which are each a distinguished name (i.e. a DN). For simplicity, assume the
      # values of the 'member' attribute are 'dn1', 'dn2', and 'dn3'.
      # Further assume the following configuration.
      #     converters:
      #         - name: hf.Registrar.Roles
      #           value: map(attr("member"),"groups")
      #     maps:
      #        groups:
      #            - name: dn1
      #              value: peer
      #            - name: dn2
      #              value: client
      # The value of the user's 'hf.Registrar.Roles' attribute is then computed to be
      # "peer,client,dn3".  This is because the value of 'attr("member")' is
      # "dn1,dn2,dn3", and the call to 'map' with a 2nd argument of
      # "group" replaces "dn1" with "peer" and "dn2" with "client".
      maps:
         groups:
            - name:
```

```
            value:

###############################################################################
# Affiliations section. Fabric CA server can be bootstrapped with the
# affiliations specified in this section. Affiliations are specified as maps.
# For example:
#   businessunit1:
#       department1:
#       - team1
#   businessunit2:
#       - department2
#       - department3
#
# Affiliations are hierarchical in nature. In the above example,
# department1 (used as businessunit1.department1) is the child of businessunit1.
# team1 (used as businessunit1.department1.team1) is the child of department1.
# department2 (used as businessunit2.department2) and department3
(businessunit2.department3)
# are children of businessunit2.
# Note: Affiliations are case sensitive except for the non-leaf affiliations
# (like businessunit1, department1, businessunit2) that are specified in the configuration
file,
# which are always stored in lower case.
###############################################################################
affiliations:
   ${ORG}:
        - department1
        - department2

###############################################################################
#  Signing section
#
#  The "default" subsection is used to sign enrollment certificates;
#  the default expiration ("expiry" field) is "8760h", which is 1 year in hours.
#
#  The "ca" profile subsection is used to sign intermediate CA certificates;
#  the default expiration ("expiry" field) is "43800h" which is 5 years in hours.
#  Note that "isca" is true, meaning that it issues a CA certificate.
#  A maxpathlen of 0 means that the intermediate CA cannot issue other
#  intermediate CA certificates, though it can still issue end entity certificates.
#  (See RFC 5280, section 4.2.1.9)
#
#  The "tls" profile subsection is used to sign TLS certificate requests;
#  the default expiration ("expiry" field) is "8760h", which is 1 year in hours.
###############################################################################
signing:
    default:
      usage:
        - digital signature
      expiry: 8760h
      backdate: 1s
    profiles:
      ca:
        usage:
          - cert sign
          - crl sign
        expiry: 43800h
        caconstraint:
          isca: true
          maxpathlen: 0
      tls:
        usage:
          - signing
          - key encipherment
          - server auth
          - client auth
          - key agreement
        expiry: 8760h

###############################################################################
#  Certificate Signing Request (CSR) section.
#  This controls the creation of the root CA certificate.
#  The expiration for the root CA certificate is configured with the
```

```
#  "ca.expiry" field below, whose default value is "131400h" which is
#  15 years in hours.
#  The pathlength field is used to limit CA certificate hierarchy as described
#  in section 4.2.1.9 of RFC 5280.
#  Examples:
#  1) No pathlength value means no limit is requested.
#  2) pathlength == 1 means a limit of 1 is requested which is the default for
#       a root CA.  This means the root CA can issue intermediate CA certificates,
#       but these intermediate CAs may not in turn issue other CA certificates
#       though they can still issue end entity certificates.
#  3) pathlength == 0 means a limit of 0 is requested;
#       this is the default for an intermediate CA, which means it can not issue
#       CA certificates though it can still issue end entity certificates.
###############################################################################
csr:
   cn: fabric-ca-server
   names:
      - C: US
        ST: "North Carolina"
        L:
        O: Hyperledger
        OU: Fabric
   hosts:
    - localhost
    - 127.0.0.1
    - ${ORG}-ca
    - ${ORG}-ca.${NS}.svc.cluster.local
    - ica.${ORG}.${DOMAIN}
   ca:
     expiry: 131400h
     pathlength: 1
###############################################################################
# BCCSP (BlockChain Crypto Service Provider) section is used to select which
# crypto library implementation to use
###############################################################################
bccsp:
    default: SW
    sw:
        hash: SHA2
        security: 256
        filekeystore:
            # The directory used for the software file-based keystore
            keystore: msp/keystore

###############################################################################
# Intermediate CA section
#
# The relationship between servers and CAs is as follows:
#   1) A single server process may contain or function as one or more CAs.
#       This is configured by the "Multi CA section" above.
#   2) Each CA is either a root CA or an intermediate CA.
#   3) Each intermediate CA has a parent CA which is either a root CA or another
intermediate CA.
#
# This section pertains to configuration of #2 and #3.
# If the "intermediate.parentserver.url" property is set,
# then this is an intermediate CA with the specified parent
# CA.
#
# parentserver section
#       url - The URL of the parent server
#       caname - Name of the CA to enroll within the server
#
# enrollment section used to enroll intermediate CA with parent CA
#       profile - Name of the signing profile to use in issuing the certificate
#       label - Label to use in HSM operations
#
# tls section for secure socket connection
#   certfiles - PEM-encoded list of trusted root certificate files
#   client:
#       certfile - PEM-encoded certificate file for when client authentication
#       is enabled on server
#       keyfile - PEM-encoded key file for when client authentication
```

```
#       is enabled on server
################################################################################
intermediate:
  parentserver:
    url:
    caname:

  enrollment:
    hosts:
    profile:
    label:

  tls:
    certfiles:
    client:
      certfile:
      keyfile:
```

Organization Fabric CA Server ConfigMap Manifest

organization/manifests/fabric-ca/org-ca-env.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: fabric-${ORG}-ca-env
data:
  # NAMESPACE
  NS: ${NS}
  # ORGANIZATION NAME
  ORG: ${ORG}
  # DOMAIN NAME
  DOMAIN: ${DOMAIN}
  FABRIC_CA_CLIENT_HOME: ${FABRIC_CA_CLIENT_HOME}
  FABRIC_CA_SERVER_HOME: ${FABRIC_CA_SERVER_HOME}
  FABRIC_CA_CLIENT_CERTS: ${FABRIC_CA_CLIENT_CERTS}
  FABRIC_CA_SERVER_CERTS: ${FABRIC_CA_SERVER_CERTS}
  FABRIC_CA_VERSION: ${FABRIC_CA_VERSION}
  FABRIC_CA_SERVER_PORT: "${FABRIC_CA_SERVER_PORT}"
  FABRIC_CA_SERVER_DEBUG: "${FABRIC_CA_SERVER_DEBUG}"
  # CA
  FABRIC_CA_SERVER_TLS_ENABLED: "${FABRIC_CA_SERVER_TLS_ENABLED}"
  FABRIC_CA_SERVER_TLS_CERTFILE: ${FABRIC_CA_SERVER_TLS_CERTFILE}
  FABRIC_CA_SERVER_TLS_KEYFILE: ${FABRIC_CA_SERVER_TLS_KEYFILE}
  FABRIC_CA_SERVER_TLS_CLIENTAUTH_TYPE: ${FABRIC_CA_SERVER_TLS_CLIENTAUTH_TYPE}
  FABRIC_CA_SERVER_TLS_CLIENTAUTH_CERTFILES: ${FABRIC_CA_SERVER_TLS_CLIENTAUTH_CERTFILES}
  FABRIC_CA_SERVER_CA_NAME: ${FABRIC_CA_SERVER_CA_NAME}
  FABRIC_CA_SERVER_CA_KEYFILE: ${FABRIC_CA_SERVER_CA_KEYFILE}
  FABRIC_CA_SERVER_CA_CERTFILE: ${FABRIC_CA_SERVER_CA_CERTFILE}
  FABRIC_CA_SERVER_CA_CHAINFILE: ${FABRIC_CA_SERVER_CA_CHAINFILE}
  FABRIC_CA_SERVER_DB_TYPE: ${FABRIC_CA_SERVER_DB_TYPE}
  FABRIC_CA_SERVER_DB_DATASOURCE: ${FABRIC_CA_SERVER_DB_DATASOURCE}
  FABRIC_CA_SERVER_DB_TLS_ENABLED: "${FABRIC_CA_SERVER_DB_TLS_ENABLED}"
  FABRIC_CA_SERVER_DB_TLS_CERTFILES: ${FABRIC_CA_SERVER_DB_TLS_CERTFILES}
  FABRIC_CA_SERVER_DB_TLS_CLIENT_CERTFILE: ${FABRIC_CA_SERVER_DB_TLS_CLIENT_CERTFILE}
  FABRIC_CA_SERVER_DB_TLS_CLIENT_KEYFILE: ${FABRIC_CA_SERVER_DB_TLS_CLIENT_KEYFILE}
  FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS: ${FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS}
  FABRIC_CA_SERVER_OPERATIONS_TLS_ENABLED: "${FABRIC_CA_SERVER_OPERATIONS_TLS_ENABLED}"
  FABRIC_CA_SERVER_OPERATIONS_TLS_KEY_FILE: ${FABRIC_CA_SERVER_OPERATIONS_TLS_KEY_FILE}
  FABRIC_CA_SERVER_OPERATIONS_TLS_CERT_FILE: ${FABRIC_CA_SERVER_OPERATIONS_TLS_CERT_FILE}
  FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES:
${FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES}
  FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTAUTHREQUIRED:
"${FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTAUTHREQUIRED}"
  FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTROOTCAS_FILES:
"${FABRIC_CA_SERVER_OPERATIONS_TLS_CLIENTROOTCAS_FILES}"
  # TLSCA
  FABRIC_TLSCA_SERVER_CA_NAME: ${FABRIC_TLSCA_SERVER_CA_NAME}
  FABRIC_TLSCA_SERVER_CA_KEYFILE: ${FABRIC_TLSCA_SERVER_CA_KEYFILE}
  FABRIC_TLSCA_SERVER_CA_CERTFILE: ${FABRIC_TLSCA_SERVER_CA_CERTFILE}
```

```
  FABRIC_TLSCA_SERVER_CA_CHAINFILE: ${FABRIC_TLSCA_SERVER_CA_CHAINFILE}
  FABRIC_TLSCA_SERVER_DB_TYPE: ${FABRIC_TLSCA_SERVER_DB_TYPE}
  FABRIC_TLSCA_SERVER_DB_DATASOURCE: ${FABRIC_TLSCA_SERVER_DB_DATASOURCE}
  FABRIC_TLSCA_SERVER_DB_TLS_ENABLED: "${FABRIC_TLSCA_SERVER_DB_TLS_ENABLED}"
  FABRIC_TLSCA_SERVER_DB_TLS_CERTFILES: ${FABRIC_TLSCA_SERVER_DB_TLS_CERTFILES}
  FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_CERTFILE: ${FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_CERTFILE}
  FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_KEYFILE: ${FABRIC_TLSCA_SERVER_DB_TLS_CLIENT_KEYFILE}
```

Organization Fabric CA Server Deployment Manifest

organization/config/ca/intermediateCA/ica-tls-issuer.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
# FABRIC CA SERVER TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${ORG}-fabric-ca-server-tls-cert
spec:
  isCA: false
  subject:
    organizations:
      - ${ORG}.${DOMAIN}
  privateKey:
    algorithm: ECDSA
    size: 384
  duration: 2160h0m0s # 90d
  renewBefore: 360h0m0s # 15d
  commonName: ica.${ORG}.${DOMAIN}
  dnsNames:
    - localhost
    - ${ORG}-ca
    - ${ORG}-ca.${NS}.svc.cluster.local
    - ica.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${ORG}-fabric-ca-server-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# FABRIC CA CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${ORG}-fabric-ca-client-tls-cert
spec:
  isCA: false
  subject:
    organizations:
      - ${ORG}.${DOMAIN}
  privateKey:
    algorithm: ECDSA
    size: 384
  duration: 2160h0m0s # 90d
  renewBefore: 360h0m0s # 15d
  commonName: ica.${ORG}.${DOMAIN}
  dnsNames:
    - localhost
    - ${ORG}-ca
    - ${ORG}-ca.${NS}.svc.cluster.local
    - ica.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
```

```
  secretName: ${ORG}-fabric-ca-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# FABRIC CA OPERATION TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${ORG}-fabric-ca-operations-server-tls-cert
spec:
  isCA: false
  subject:
    organizations:
      - ${ORG}.${DOMAIN}
  privateKey:
    algorithm: ECDSA
    size: 384
  duration: 2160h0m0s # 90d
  renewBefore: 360h0m0s # 15d
  commonName: ica.${ORG}.${DOMAIN}
  dnsNames:
    - localhost
    - operations.ica.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${ORG}-fabric-ca-operations-server-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# FABRIC CA OPERATION CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${ORG}-fabric-ca-operations-client-tls-cert
spec:
  isCA: false
  subject:
    organizations:
      - ${ORG}.${DOMAIN}
  privateKey:
    algorithm: ECDSA
    size: 384
  duration: 2160h0m0s # 90d
  renewBefore: 360h0m0s # 15d
  commonName: ica.${ORG}.${DOMAIN}
  dnsNames:
    - localhost
    - operations.ica.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${ORG}-fabric-ca-operations-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# FABRIC CA SERVER
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ${ORG}-ca
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ${ORG}-ca
  template:
    metadata:
      labels:
        app: ${ORG}-ca
    spec:
      hostAliases:
```

```
      - ip: 127.0.0.1
        hostnames:
        - ica.${ORG}.${DOMAIN}
        - operations.ica.${ORG}.${DOMAIN}
      initContainers:
        - name: ${ORG}-ca-init
          image: ${FABRIC_CA_ALPINE_ENVSUBST}
          imagePullPolicy: Never
          command: ["/bin/sh"]
          args:
            - -c
            - >-
              mkdir -p ${ORG_DIR}/ca ${ORG_DIR}/tlsca ${ORG_DIR}/${NODE_TYPE}s
${ORG_DIR}/users > /dev/null 2>&1 &&
              mkdir -p ${ORG_DIR}/msp/admincerts ${ORG_DIR}/msp/intermediatecerts
${ORG_DIR}/msp/cacerts ${ORG_DIR}/msp/tlscacerts ${ORG_DIR}/msp/tlsintermediatecerts
${ADMIN_DIR}/msp/admincerts > /dev/null 2>&1 &&
              mkdir -p ${ADMIN_DIR}/msp/tlsintermediatecerts ${ADMIN_DIR}/msp/tlscacerts >
/dev/null 2>&1 &&
              mkdir -p ${FABRIC_CA_SERVER_HOME}/ca ${FABRIC_CA_SERVER_HOME}/ca/certs >
/dev/null 2>&1 &&
              mkdir -p ${FABRIC_CA_SERVER_HOME}/tlsca ${FABRIC_CA_SERVER_HOME}/tlsca/certs
> /dev/null 2>&1 &&
              mkdir -p ${FABRIC_CA_SERVER_OPERATIONS_CERTS}/server > /dev/null 2>&1 &&
              mkdir -p ${FABRIC_CA_SERVER_CERTS} > /dev/null 2>&1 &&
              mkdir -p ${FABRIC_CA_CLIENT_CERTS} > /dev/null 2>&1 &&
              cat ${FABRIC_EMPTY_DIR}/ca/fabric-ca-server-config.yaml | envsubst >
${FABRIC_CA_SERVER_HOME}/ca/fabric-ca-server-config.yaml &&
              cat ${FABRIC_EMPTY_DIR}/tlsca/fabric-ca-server-config.yaml | envsubst >
${FABRIC_CA_SERVER_HOME}/tlsca/fabric-ca-server-config.yaml &&
              (cd ${FABRIC_EMPTY_DIR}/ca/certs
               echo ${ORG_DIR}/ca ${FABRIC_CA_SERVER_CA_CHAINFILE} | xargs -n 1 cp
chain.identity.${ORG}.${DOMAIN}.crt
               cp ica.identity.${ORG}.${DOMAIN}.key ${FABRIC_CA_SERVER_CA_KEYFILE}
               awk "/-----BEGIN CERTIFICATE-----/{i++}i==1"
chain.identity.${ORG}.${DOMAIN}.crt > ${FABRIC_CA_SERVER_CA_CERTFILE}
              ) &&
               echo ${ORG_DIR}/tlsca ${FABRIC_TLSCA_SERVER_CA_CHAINFILE} \
                ${FABRIC_TLSCA_SERVER_CA_CERTFILE} \
                ${FABRIC_CA_SERVER_TLS_CLIENTAUTH_CERTFILES} \
                ${FABRIC_CA_CLIENT_CERTS}/ca.crt \
                ${FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES} \
                | xargs -n 1 cp
${FABRIC_EMPTY_DIR}/tlsca/certs/chain.tls.${ORG}.${DOMAIN}.crt &&
              (cd ${FABRIC_EMPTY_DIR}/tlsca/certs
               cp ica.tls.${ORG}.${DOMAIN}.key ${FABRIC_TLSCA_SERVER_CA_KEYFILE}
               awk "/-----BEGIN CERTIFICATE-----/{i++}i==1" chain.tls.${ORG}.${DOMAIN}.crt
> ${FABRIC_TLSCA_SERVER_CA_CERTFILE}
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/server
               cp server.key ${FABRIC_CA_SERVER_TLS_KEYFILE}
               cp server.crt ${FABRIC_CA_SERVER_TLS_CERTFILE}
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/client
               cp client.key ${FABRIC_CA_CLIENT_CERTS}/client.key
               cp client.crt ${FABRIC_CA_CLIENT_CERTS}/client.crt
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/operations/server
               cp server.key ${FABRIC_CA_SERVER_OPERATIONS_TLS_KEY_FILE}
               cp server.crt ${FABRIC_CA_SERVER_OPERATIONS_TLS_CERT_FILE}
              )
          envFrom:
          - configMapRef:
              name: fabric-${ORG}-ca-env
          - secretRef:
              name: fabric-${ORG}-ca-env-secrets
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
              readOnly: false
            - name: fabric-${ORG}-emptydir
              mountPath: ${FABRIC_EMPTY_DIR}
            - name: fabric-server-ca-config
```

```
            mountPath: ${FABRIC_EMPTY_DIR}/ca/fabric-ca-server-config.yaml
            subPath: fabric-ca-server-config.yaml
            readOnly: true
          - name: fabric-server-tlsca-config
            mountPath: ${FABRIC_EMPTY_DIR}/tlsca/fabric-ca-server-config.yaml
            subPath: fabric-ca-server-config.yaml
            readOnly: true
          - name: tls-server-cert-volume
            mountPath: ${FABRIC_EMPTY_DIR}/tls/server
            readOnly: true
          - name: tls-client-cert-volume
            mountPath: ${FABRIC_EMPTY_DIR}/tls/client
            readOnly: true
          - name: intermediate-tls-certs
            mountPath: ${FABRIC_EMPTY_DIR}/tlsca/certs
            readOnly: true
          - name: intermediate-identity-certs
            mountPath: ${FABRIC_EMPTY_DIR}/ca/certs
            readOnly: true
          - name: operations-server-tls-cert
            mountPath: ${FABRIC_EMPTY_DIR}/operations/server
            readOnly: true
      containers:
        - name: ${ORG}-ca
          image: ${FABRIC_CONTAINER_REGISTRY}/fabric-ca:${FABRIC_CA_VERSION}
          imagePullPolicy: Never
          command: ["/bin/sh"]
          args:
            - -c
            - >-
              (cd ${FABRIC_CA_SERVER_HOME}/ca
               fabric-ca-server start
              )
          env:
          - name: FABRIC_CA_SERVER_HOME
            value: "${FABRIC_CA_SERVER_HOME}/ca"
          envFrom:
          - configMapRef:
            name: fabric-${ORG}-ca-env
          - secretRef:
            name: fabric-${ORG}-ca-env-secrets
          ports:
            - containerPort: ${FABRIC_CA_SERVER_PORT}
            - containerPort: ${FABRIC_CA_SERVER_OPERATIONS_PORT}
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
              readOnly: false
          readinessProbe:
            exec:
              command:
              - /bin/sh
              - -c
              - >-
                if [[ $(curl -s --cacert ${FABRIC_CA_SERVER_OPERATIONS_TLS_ROOTCAS_FILES}
https://operations.ica.${ORG}.${DOMAIN}:${FABRIC_CA_SERVER_OPERATIONS_PORT}/healthz | jq -r
.status) != "OK" ]]; then
                  exit 1;
                fi
            initialDelaySeconds: 5
            failureThreshold: 5
            periodSeconds: 2
      volumes:
        - name: fabric-volume
          persistentVolumeClaim:
            claimName: fabric-${ORG}
        - name: fabric-${ORG}-emptydir
          emptyDir: {}
        - name: fabric-server-ca-config
          configMap:
            name: fabric-ca-server-config
        - name: fabric-server-tlsca-config
          configMap:
```

```
            name: fabric-tlsca-server-config
        - name: tls-server-cert-volume
          secret:
            secretName: ${ORG}-fabric-ca-server-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: server.crt
            - key: tls.key
              path: server.key
        - name: tls-client-cert-volume
          secret:
            secretName: ${ORG}-fabric-ca-client-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: client.crt
            - key: tls.key
              path: client.key
        - name: intermediate-tls-certs
          secret:
            secretName: ${ORG}-intermediate-tlsca-key-pair
            items:
            - key: tls.crt
              path: chain.tls.${ORG}.${DOMAIN}.crt
            - key: tls.key
              path: ica.tls.${ORG}.${DOMAIN}.key
        - name: intermediate-identity-certs
          secret:
            secretName: ${ORG}-intermediate-identity-key-pair
            items:
            - key: tls.crt
              path: chain.identity.${ORG}.${DOMAIN}.crt
            - key: tls.key
              path: ica.identity.${ORG}.${DOMAIN}.key
        - name: operations-server-tls-cert
          secret:
            secretName: ${ORG}-fabric-ca-operations-server-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: server.crt
            - key: tls.key
              path: server.key
---
# FABRIC CA SERVER SERVICE
---
apiVersion: v1
kind: Service
metadata:
  name: ${ORG}-ca
spec:
  ports:
    - name: general
      port: ${FABRIC_CA_SERVER_PORT}
      targetPort: ${FABRIC_CA_SERVER_PORT}
      protocol: TCP
    - name: operations
      port: ${FABRIC_CA_SERVER_OPERATIONS_PORT}
      targetPort: ${FABRIC_CA_SERVER_OPERATIONS_PORT}
      protocol: TCP
  selector:
    app: ${ORG}-ca
---
# FABRIC CA SERVER INGRESS
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: ${NS}
```

```
  annotations:
    nginx.ingress.kubernetes.io/proxy-connect-timeout: 60s
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  labels:
    app: ${ORG}-ca
  name: ${ORG}-ca
spec:
  ingressClassName: nginx
  rules:
    - host: ica.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${ORG}-ca
                port:
                  name: general
            path: /
            pathType: ImplementationSpecific
    - host: operations.ica.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${ORG}-ca
                port:
                  name: operations
            path: /
            pathType: ImplementationSpecific
  tls:
    - hosts:
        - ica.${ORG}.${DOMAIN}
    - hosts:
        - operations.ica.${ORG}.${DOMAIN}
```

Organization Peer Gateway Service Manifest

organization/manifests/peer/peer-gateway.yaml

```
---
# PEER GATEWAY SERVICE
---
apiVersion: v1
kind: Service
metadata:
  name: ${ORG}-gateway
spec:
  ports:
    - name: grpc
      port: ${PEER_GRPC_PORT}
      protocol: TCP
  selector:
    org: ${ORG}
---
# PEER INGRESS
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/proxy-connect-timeout: 60s
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  labels:
    app: ${ORG}-gateway
  name: ${ORG}-gateway
spec:
  ingressClassName: nginx
  rules:
    - host: gateway.${ORG}.${DOMAIN}
      http:
```

```
        paths:
          - backend:
              service:
                name: ${ORG}-gateway
                port:
                name: grpc
            path: /
            pathType: ImplementationSpecific
  tls:
    - hosts:
        - gateway.${ORG}.${DOMAIN}
```

Organization Peer Config

organization/config/peer.yaml

https://github.com/hyperledger/fabric/blob/v2.4.6/sampleconfig/core.yaml

Organization Peer ConfigMap Manifest

organization/manifests/peer/peer-env.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ${NODE_NAME}-env
data:
  FABRIC_CFG_PATH: ${FABRIC_CFG_PATH}
  FABRIC_LOGGING_SPEC: "${FABRIC_LOGGING_SPEC}"
  CORE_PEER_ID: ${CORE_PEER_ID}
  CORE_PEER_LISTENADDRESS: ${CORE_PEER_LISTENADDRESS}
  CORE_PEER_ADDRESS: ${CORE_PEER_ADDRESS}
  CORE_PEER_ADDRESSAUTODETECT: "${CORE_PEER_ADDRESSAUTODETECT}"
  CORE_PEER_LOCALMSPID: ${CORE_PEER_LOCALMSPID}
  CORE_PEER_MSPCONFIGPATH: ${CORE_PEER_MSPCONFIGPATH}
  CORE_PEER_GOSSIP_ORGLEADER: "${CORE_PEER_GOSSIP_ORGLEADER}"
  CORE_PEER_GOSSIP_USELEADERELECTION: "${CORE_PEER_GOSSIP_USELEADERELECTION}"
  CORE_PEER_GOSSIP_STATE_ENABLED: "${CORE_PEER_GOSSIP_STATE_ENABLED}"
  CORE_PEER_DELIVERYCLIENT_BLOCKGOSSIPENABLED:
"${CORE_PEER_DELIVERYCLIENT_BLOCKGOSSIPENABLED}"
  CORE_PEER_TLS_ENABLED: "${CORE_PEER_TLS_ENABLED}"
  CORE_PEER_TLS_KEY_FILE: ${CORE_PEER_TLS_KEY_FILE}
  CORE_PEER_TLS_CERT_FILE: ${CORE_PEER_TLS_CERT_FILE}
  CORE_PEER_TLS_ROOTCERT_FILE: ${CORE_PEER_TLS_ROOTCERT_FILE}
  CORE_PEER_TLS_CLIENTAUTHREQUIRED: "${CORE_PEER_TLS_CLIENTAUTHREQUIRED}"
  CORE_PEER_TLS_CLIENTKEY_FILE: ${CORE_PEER_TLS_CLIENTKEY_FILE}
  CORE_PEER_TLS_CLIENTCERT_FILE: ${CORE_PEER_TLS_CLIENTCERT_FILE}
  CORE_PEER_TLS_CLIENTROOTCAS_FILES: ${CORE_PEER_TLS_CLIENTROOTCAS_FILES}
  CORE_OPERATIONS_LISTENADDRESS: ${CORE_OPERATIONS_LISTENADDRESS}
  CORE_OPERATIONS_TLS_ENABLED: "${CORE_OPERATIONS_TLS_ENABLED}"
  CORE_OPERATIONS_TLS_KEY_FILE: ${CORE_OPERATIONS_TLS_KEY_FILE}
  CORE_OPERATIONS_TLS_CERT_FILE: ${CORE_OPERATIONS_TLS_CERT_FILE}
  CORE_OPERATIONS_TLS_ROOTCAS_FILES: ${CORE_OPERATIONS_TLS_ROOTCAS_FILES}
  CORE_OPERATIONS_TLS_CLIENTAUTHREQUIRED: "${CORE_OPERATIONS_TLS_CLIENTAUTHREQUIRED}"
  CORE_OPERATIONS_TLS_CLIENTROOTCAS_FILES: "${CORE_OPERATIONS_TLS_CLIENTROOTCAS_FILES}"
  CORE_PEER_CHAINCODEADDRESS: ${CORE_PEER_CHAINCODEADDRESS}
  CORE_PEER_CHAINCODELISTENADDRESS: ${CORE_PEER_CHAINCODELISTENADDRESS}
  CORE_PEER_FILESYSTEMPATH: ${CORE_PEER_FILESYSTEMPATH}
  CORE_LEDGER_SNAPSHOTS_ROOTDIR: ${CORE_LEDGER_SNAPSHOTS_ROOTDIR}
  CORE_LEDGER_STATE_STATEDATABASE: ${CORE_LEDGER_STATE_STATEDATABASE}
  CORE_LEDGER_STATE_COUCHDBCONFIG_MAXRETRIESONSTARTUP:
"${CORE_LEDGER_STATE_COUCHDBCONFIG_MAXRETRIESONSTARTUP}"
  CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS:
${CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS}
  CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG: "{\"PEER\":\"${NODE_NAME}\"}"
  CORE_PEER_GATEWAY_ENABLED: "${CORE_PEER_GATEWAY_ENABLED}"
```

Organization Peer Deployment Manifest

organization/manifests/peer/peer.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
# PEER CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - ${NODE_NAME}-${ORG}
    - ${NODE_NAME}-${ORG}.${NS}
    - ${NODE_NAME}-${ORG}.${NS}.svc.cluster.local
    - ${NODE_NAME}.${ORG}.${DOMAIN}
    - gateway.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# PEER OPERATIONS TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-operations-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - operations.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-operations-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# PEER OPERATIONS CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-operations-client-tls-cert
  namespace: ${NS}
```

```
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - operations.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-operations-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# PEER DEPLOYMENT
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ${NODE_NAME}-${ORG}
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ${NODE_NAME}-${ORG}
  template:
    metadata:
      labels:
        app: ${NODE_NAME}-${ORG}
        org: ${ORG}
    spec:
      hostAliases:
      - ip: 127.0.0.1
        hostnames:
        - operations.${NODE_NAME}.${ORG}.${DOMAIN}
      initContainers:
        - name: ${NODE_NAME}-${ORG}-init
          image: alpine:${ALPINE_VERSION}
          imagePullPolicy: IfNotPresent
          command: ["/bin/sh"]
          args:
            - -c
            - >-
              mkdir -p ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client &&
              mkdir -p ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations &&
              mkdir -p ${FABRIC_CFG_PATH} &&
              (cd ${FABRIC_EMPTY_DIR}/config
               cp peer.yaml ${FABRIC_CFG_PATH}/core.yaml
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/client
               echo "$(awk "/-----BEGIN CERTIFICATE-----/{i++}i==2" client.crt; cat
ca.crt)" | tee ${CORE_PEER_TLS_CLIENTROOTCAS_FILES} | tee
${CORE_OPERATIONS_TLS_ROOTCAS_FILES}
               cp client.key ${CORE_PEER_TLS_CLIENTKEY_FILE}
               cp client.crt ${CORE_PEER_TLS_CLIENTCERT_FILE}
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/operations
               cp server.key ${CORE_OPERATIONS_TLS_KEY_FILE}
               cp server.crt ${CORE_OPERATIONS_TLS_CERT_FILE}
              )
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
```

```
            - name: fabric-config
              mountPath: ${FABRIC_EMPTY_DIR}/config
            - name: client-tls-cert
              mountPath: ${FABRIC_EMPTY_DIR}/tls/client
              readOnly: true
            - name: operations-tls-cert
              mountPath: ${FABRIC_EMPTY_DIR}/tls/operations
              readOnly: true
      containers:
        - name: ${NODE_NAME}-${ORG}
          image: ${FABRIC_CONTAINER_REGISTRY}/fabric-peer:${FABRIC_VERSION}
          imagePullPolicy: Never
          envFrom:
            - configMapRef:
                name: ${NODE_NAME}-env
          env:
            - name: CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME
              valueFrom:
                secretKeyRef:
                  name: ${NODE_TYPE}-couchdb-env-secrets
                  key: COUCHDB_USER
            - name: CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: ${NODE_TYPE}-couchdb-env-secrets
                  key: COUCHDB_PASSWORD
          ports:
            - containerPort: ${PEER_GRPC_PORT}
            - containerPort: ${PEER_CHAINCODE_PORT}
            - containerPort: ${PEER_OPERATIONS_PORT}
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
          readinessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - >-
                  if [[ $(curl -s --cacert ${CORE_OPERATIONS_TLS_ROOTCAS_FILES} --key
${CORE_OPERATIONS_TLS_KEY_FILE} --cert ${CORE_OPERATIONS_TLS_CERT_FILE}
https://operations.${NODE_NAME}.${ORG}.${DOMAIN}:${PEER_OPERATIONS_PORT}/healthz | jq -r
.status) != "OK" ]]; then
                   exit 1;
                  fi
            initialDelaySeconds: 5
            failureThreshold: 5
            periodSeconds: 2
        - name: couchdb
          image: ${COUCHDB_IMAGE}:${COUCHDB_VERSION}
          imagePullPolicy: IfNotPresent
          envFrom:
            - secretRef:
                name: ${NODE_TYPE}-couchdb-env-secrets
          ports:
              - containerPort: ${COUCHDB_PORT}
      volumes:
        - name: fabric-volume
          persistentVolumeClaim:
            claimName: fabric-${ORG}
        - name: fabric-config
          configMap:
            name: ${NODE_TYPE}-config
```

```
        - name: client-tls-cert
          secret:
            secretName: ${NODE_NAME}-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: client.crt
            - key: tls.key
              path: client.key
        - name: operations-tls-cert
          secret:
            secretName: ${NODE_NAME}-operations-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: server.crt
            - key: tls.key
              path: server.key
---
# PEER SERVICE
---
apiVersion: v1
kind: Service
metadata:
  name: ${NODE_NAME}-${ORG}
spec:
  ports:
    - name: grpc
      port: ${PEER_GRPC_PORT}
      protocol: TCP
    - name: chaincode
      port: ${PEER_CHAINCODE_PORT}
      protocol: TCP
    - name: operations
      port: ${PEER_OPERATIONS_PORT}
      protocol: TCP
  selector:
    app: ${NODE_NAME}-${ORG}
---
# PEER INGRESS
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/proxy-connect-timeout: 60s
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  labels:
    app: ${NODE_NAME}-${ORG}
  name: ${NODE_NAME}-${ORG}
spec:
  ingressClassName: nginx
  rules:
    - host: ${NODE_NAME}.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${NODE_NAME}-${ORG}
                port:
                  name: grpc
```

```
            path: /
            pathType: ImplementationSpecific
        - host: operations.${NODE_NAME}.${ORG}.${DOMAIN}
          http:
            paths:
              - backend:
                  service:
                    name: ${NODE_NAME}-${ORG}
                    port:
                      name: operations
        path: /
        pathType: ImplementationSpecific
  tls:
    - hosts:
      - ${NODE_NAME}.${ORG}.${DOMAIN}
    - hosts:
      - operations.${NODE_NAME}.${ORG}.${DOMAIN}
```

Organization Orderer Config

organization/config/orderer.yaml

https://github.com/hyperledger/fabric/blob/v2.4.6/sampleconfig/orderer.yaml

Organization Orderer ConfigMap Manifest

organization/manifests/orderer/orderer-env.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ${NODE_NAME}-env
data:
 FABRIC_CFG_PATH: ${FABRIC_CFG_PATH}
 FABRIC_LOGGING_SPEC: "${FABRIC_LOGGING_SPEC}"
 ORDERER_CHANNELPARTICIPATION_ENABLED: "${ORDERER_CHANNELPARTICIPATION_ENABLED}"
 ORDERER_GENERAL_LISTENADDRESS: ${ORDERER_GENERAL_LISTENADDRESS}
 ORDERER_GENERAL_LISTENPORT: "${ORDERER_GENERAL_LISTENPORT}"
 ORDERER_GENERAL_LOCALMSPID: ${ORDERER_GENERAL_LOCALMSPID}
 ORDERER_GENERAL_LOCALMSPDIR: ${ORDERER_GENERAL_LOCALMSPDIR}
 ORDERER_GENERAL_BOOTSTRAPMETHOD: ${ORDERER_GENERAL_BOOTSTRAPMETHOD}
 ORDERER_GENERAL_TLS_ENABLED: "${ORDERER_GENERAL_TLS_ENABLED}"
 ORDERER_GENERAL_TLS_PRIVATEKEY: ${ORDERER_GENERAL_TLS_PRIVATEKEY}
 ORDERER_GENERAL_TLS_CERTIFICATE: ${ORDERER_GENERAL_TLS_CERTIFICATE}
 ORDERER_GENERAL_TLS_ROOTCAS: "${ORDERER_GENERAL_TLS_ROOTCAS}"
 ORDERER_GENERAL_TLS_CLIENTAUTHREQUIRED: "${ORDERER_GENERAL_TLS_CLIENTAUTHREQUIRED}"
 ORDERER_GENERAL_TLS_CLIENTROOTCAS: "${ORDERER_GENERAL_TLS_CLIENTROOTCAS}"
 ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE: ${ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE}
 ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY: ${ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY}
 ORDERER_GENERAL_CLUSTER_ROOTCAS: ${ORDERER_GENERAL_CLUSTER_ROOTCAS}
 ORDERER_GENERAL_CLUSTER_LISTENPORT: ${ORDERER_GENERAL_CLUSTER_LISTENPORT}
 ORDERER_GENERAL_CLUSTER_LISTENADDRESS: ${ORDERER_GENERAL_CLUSTER_LISTENADDRESS}
 ORDERER_GENERAL_CLUSTER_SERVERCERTIFICATE: ${ORDERER_GENERAL_CLUSTER_SERVERCERTIFICATE}
 ORDERER_GENERAL_CLUSTER_SERVERPRIVATEKEY: ${ORDERER_GENERAL_CLUSTER_SERVERPRIVATEKEY}
 ORDERER_ADMIN_LISTENADDRESS: ${ORDERER_ADMIN_LISTENADDRESS}
 ORDERER_ADMIN_TLS_ENABLED: "${ORDERER_ADMIN_TLS_ENABLED}"
 ORDERER_ADMIN_TLS_PRIVATEKEY: ${ORDERER_ADMIN_TLS_PRIVATEKEY}
 ORDERER_ADMIN_TLS_CERTIFICATE: ${ORDERER_ADMIN_TLS_CERTIFICATE}
 ORDERER_ADMIN_TLS_ROOTCAS: ${ORDERER_ADMIN_TLS_ROOTCAS}
 ORDERER_ADMIN_TLS_CLIENTAUTHREQUIRED: "${ORDERER_ADMIN_TLS_CLIENTAUTHREQUIRED}"
 ORDERER_ADMIN_TLS_CLIENTROOTCAS: "${ORDERER_ADMIN_TLS_CLIENTROOTCAS}"
 ORDERER_FILELEDGER_LOCATION: ${ORDERER_FILELEDGER_LOCATION}
 ORDERER_CONSENSUS_WALDIR: ${ORDERER_CONSENSUS_WALDIR}
 ORDERER_CONSENSUS_SNAPDIR: ${ORDERER_CONSENSUS_SNAPDIR}
 ORDERER_OPERATIONS_LISTENADDRESS: ${ORDERER_OPERATIONS_LISTENADDRESS}
```

```
ORDERER_OPERATIONS_TLS_ENABLED: "${ORDERER_OPERATIONS_TLS_ENABLED}"
ORDERER_OPERATIONS_TLS_PRIVATEKEY: ${ORDERER_OPERATIONS_TLS_PRIVATEKEY}
ORDERER_OPERATIONS_TLS_CERTIFICATE: ${ORDERER_OPERATIONS_TLS_CERTIFICATE}
ORDERER_OPERATIONS_TLS_ROOTCAS: ${ORDERER_OPERATIONS_TLS_ROOTCAS}
ORDERER_OPERATIONS_TLS_CLIENTAUTHREQUIRED: "${ORDERER_OPERATIONS_TLS_CLIENTAUTHREQUIRED}"
ORDERER_OPERATIONS_TLS_CLIENTROOTCAS: "${ORDERER_OPERATIONS_TLS_CLIENTROOTCAS}"
```

Organization Orderer Deployment Manifest

organization/manifests/orderer/orderer.yaml

```
#
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
---
# ORDERER CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - ${NODE_NAME}-${ORG}
    - ${NODE_NAME}-${ORG}.${NS}
    - ${NODE_NAME}-${ORG}.${NS}.svc.cluster.local
    - ${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# ORDERER ADMIN SERVER TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-admin-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - admin.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-admin-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
```

```
# ORDERER ADMIN CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-admin-client-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - admin.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-admin-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# ORDERER OPERATIONS TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-operations-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - operations.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-operations-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
# ORDERER OPERATIONS CLIENT TLS
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ${NODE_NAME}-operations-client-tls-cert
  namespace: ${NS}
spec:
  isCA: false
  privateKey:
    algorithm: ECDSA
    size: 384
  dnsNames:
    - localhost
    - operations.${NODE_NAME}.${ORG}.${DOMAIN}
  ipAddresses:
    - 127.0.0.1
  secretName: ${NODE_NAME}-operations-client-tls-cert
  issuerRef:
    name: ${ORG}-ica-tls-issuer
---
```

```
# ORDERER DEPLOYMENT
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ${NODE_NAME}-${ORG}
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ${NODE_NAME}-${ORG}
  template:
    metadata:
      labels:
        app: ${NODE_NAME}-${ORG}
    spec:
      hostAliases:
      - ip: 127.0.0.1
        hostnames:
        - operations.${NODE_NAME}.${ORG}.${DOMAIN}
      initContainers:
        - name: ${NODE_NAME}-${ORG}-init
          image: alpine:${ALPINE_VERSION}
          imagePullPolicy: IfNotPresent
          command: ["/bin/sh"]
          args:
            - -c
            - >-
              mkdir -p ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client &&
              mkdir -p ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/admin &&
              mkdir -p ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/operations &&
              (cd ${FABRIC_EMPTY_DIR}/tls/client
               echo "$(awk "/-----BEGIN CERTIFICATE-----/{i++}i==2" client.crt; cat
ca.crt)" | tee ${PEER_DIR}/${NODE_NAME}.${ORG}.${DOMAIN}/client/ca.crt | tee
${ORDERER_ADMIN_TLS_ROOTCAS} | tee ${ORDERER_OPERATIONS_TLS_ROOTCAS}
               cp client.key ${ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY}
               cp client.crt ${ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE}
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/admin
               cp admin.key ${ORDERER_ADMIN_TLS_PRIVATEKEY}
               cp admin.crt ${ORDERER_ADMIN_TLS_CERTIFICATE}
              ) &&
              (cd ${FABRIC_EMPTY_DIR}/tls/operations
               cp operations.key ${ORDERER_OPERATIONS_TLS_PRIVATEKEY}
               cp operations.crt ${ORDERER_OPERATIONS_TLS_CERTIFICATE}
              )
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
            - name: fabric-config
              mountPath: ${FABRIC_CFG_PATH}
            - name: client-tls-cert
              mountPath: ${FABRIC_EMPTY_DIR}/tls/client
              readOnly: true
            - name: admin-tls-cert
              mountPath: ${FABRIC_EMPTY_DIR}/tls/admin
              readOnly: true
            - name: operations-tls-cert
              mountPath: ${FABRIC_EMPTY_DIR}/tls/operations
              readOnly: true
      containers:
        - name: ${NODE_NAME}-${ORG}
          image: ${FABRIC_CONTAINER_REGISTRY}/fabric-orderer:${FABRIC_VERSION}
```

```
          imagePullPolicy: Never
          envFrom:
            - configMapRef:
                name: ${NODE_NAME}-env
          ports:
            - containerPort: ${ORDERER_GENERAL_LISTENPORT}
            - containerPort: ${ORDERER_OPERATIONS_PORT}
            - containerPort: ${ORDERER_ADMIN_PORT}
          volumeMounts:
            - name: fabric-volume
              mountPath: ${SHARE_DIRECTORY}
          readinessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - >-
                 if [[ $(curl -s --cacert ${ORDERER_OPERATIONS_TLS_ROOTCAS} --key
${ORDERER_OPERATIONS_TLS_PRIVATEKEY} --cert ${ORDERER_OPERATIONS_TLS_CERTIFICATE}
https://operations.${NODE_NAME}.${ORG}.${DOMAIN}:${ORDERER_OPERATIONS_PORT}/healthz | jq -r
.status) != "OK" ]]; then
                  exit 1;
                 fi
            initialDelaySeconds: 5
            failureThreshold: 5
            periodSeconds: 2
      volumes:
        - name: fabric-volume
          persistentVolumeClaim:
            claimName: fabric-${ORG}
        - name: fabric-config
          configMap:
            name: ${NODE_TYPE}-config
        - name: client-tls-cert
          secret:
            secretName: ${NODE_NAME}-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: client.crt
            - key: tls.key
              path: client.key
        - name: admin-tls-cert
          secret:
            secretName: ${NODE_NAME}-admin-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: admin.crt
            - key: tls.key
              path: admin.key
        - name: operations-tls-cert
          secret:
            secretName: ${NODE_NAME}-operations-tls-cert
            items:
            - key: ca.crt
              path: ca.crt
            - key: tls.crt
              path: operations.crt
            - key: tls.key
              path: operations.key
```

```
---
# ORDERER SERVICE
---
apiVersion: v1
kind: Service
metadata:
  name: ${NODE_NAME}-${ORG}
spec:
  ports:
    - name: general
      port: ${ORDERER_GENERAL_LISTENPORT}
      protocol: TCP
    - name: operations
      port: ${ORDERER_OPERATIONS_PORT}
      protocol: TCP
    - name: admin
      port: ${ORDERER_ADMIN_PORT}
      protocol: TCP
  selector:
    app: ${NODE_NAME}-${ORG}

---
# ORDERER INGRESS
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/proxy-connect-timeout: 60s
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  labels:
    app: ${NODE_NAME}-${ORG}
  name: ${NODE_NAME}-${ORG}
spec:
  ingressClassName: nginx
  rules:
    - host: ${NODE_NAME}.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${NODE_NAME}-${ORG}
                port:
                name: general
            path: /
            pathType: ImplementationSpecific
    - host: operations.${NODE_NAME}.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${NODE_NAME}-${ORG}
                port:
                name: operations
            path: /
            pathType: ImplementationSpecific
    - host: admin.${NODE_NAME}.${ORG}.${DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${NODE_NAME}-${ORG}
                port:
```

```
                name: admin
            path: /
            pathType: ImplementationSpecific
  tls:
    - hosts:
      - ${NODE_NAME}.${ORG}.${DOMAIN}
    - hosts:
      - operations.${NODE_NAME}.${ORG}.${DOMAIN}
    - hosts:
      - admin.${NODE_NAME}.${ORG}.${DOMAIN}
```

Appendix C

Note: The code presented in this Appendix is a result of the research project ERATOSTHENES and intellectual property of INLECOM INNOVATION Non-Profit Organization, which should be specifically referenced in any use of this appendix.

This Appendix is dedicated to demonstrate the codes that were used for the creation and deployment of the gRPC Bidirectional Server.

Proto Compiler Installation Guide

```
// Install protobuf compiler.
sudo apt install protobuf-compiler
sudo apt install golang-goprotobuf-dev

// Verify the version of the protobuf compiler. Ensure that the compiler has version 3+
protoc --version

// Create the necessary project structure.
export GO111MODULE=on
cd
mkdir -p "<Path to the Project>/grpc-go"
cd "<Path to the Project>/grpc-go"
go mod init fabric-vdr

// Install grpc
sudo $(which go) get -u -x google.golang.org/grpc@latest
// Install protoc-gen-go.
sudo $(which go) get -u -x google.golang.org/protobuf/cmd/protoc-gen-go@latest

// (Optional - Binary Provided)
// Download the latest protoc-gen-go-grpc binary, extract it and place it within the
project folder:
cd "<Path to the Project>/grpc-go"
wget
https://github.com/grpc/grpc-go/releases/download/cmd%2Fprotoc-gen-go-grpc%2Fv1.2.0/protoc-
gen-go-grpc.v1.2.0.linux.amd64.tar.gz
sudo tar -xvf protoc-gen-go-grpc.v1.2.0.linux.amd64.tar.gz
rm protoc-gen-go-grpc.v1.2.0.linux.amd64.tar.gz

// Create a Proto3 file. Folder name and .proto file must possess the same name.
cd "<Path to the Project>/grpc-go"
mkdir grpcFabricVdrAPI
touch grpcFabricVdrAPI.proto

// Define the messages and the services in the grpcFabricVdrAPI.proto file.
// (Described in the 4.4.1 Section)

// Execute the following command to create the grpc artifacts.
cd "<Path to the Project>/grpc-go"
protoc grpcFabricVdrAPI.proto --plugin=protoc-gen-go-grpc/protoc-gen-go-grpc
--go_out=grpcFabricVdrAPI/ --go_opt=paths=source_relative --go-grpc_out=grpcFabricVdrAPI/
--go-grpc_opt=paths=source_relative --proto_path=${PWD}/grpcFabricVdrAPI
```

Project Filesystem Structure

```
.
├── CA #👉 CA directory (Omitted for brevity)
├── client
│   ├── client.go #👉 Actual gRPC client, used for the VDR integration with HLA
│   └── unitTest.go #👉 VDR gRPC server, table driven Unit-Tests based on VDR gRPC client
├── go.mod #👉 Dependency Requirements
├── go.sum
├── grpcFabricVdrAPI
│   ├── grpcFabricVdrAPI_grpc.pb.go #👉 Grpc Methods Protobuf
│   ├── grpcFabricVdrAPI.pb.go      #👉 Messages Protobuf (Omitted due to GOB)
│   └── grpcFabricVdrAPI.proto      #👉 Actual Proto File
├── keys.sh #👉 Script that creates CA and Client - Server TLS
├── packages
│   ├── codec
│   │   └── codec.go #👉 GOB Encoder/Decoder
│   ├── functions
│   │   └── functions.go #👉 Helper Functions
│   └── structs
│       └── fabricVdrStruct.go #👉 Custom Messages
├── protoc-gen-go-grpc
│   ├── LICENSE
│   ├── protoc-gen-go-grpc
│   └── README.md
├── server
│   └── server.go  #👉 Actual gRPC Server
└── TLS
    ├── client #👉 Client TLS directory (Omitted for brevity)
    └── server #👉 Server TLS directory (Omitted for brevity)

16 directories, 32 files
```

GRPC Protobuf

grpcFabricVdrAPI_grpc.pb.go

```go
// Code generated by protoc-gen-go-grpc. DO NOT EDIT.
// versions:
// - protoc-gen-go-grpc v1.2.0
// - protoc             v3.12.4
// source: grpcFabricVdrAPI.proto

package grpcFabricVdrAPI

import (
    context "context"
    grpc "google.golang.org/grpc"
    codes "google.golang.org/grpc/codes"
    status "google.golang.org/grpc/status"
)

// This is a compile-time assertion to ensure that this generated file
// is compatible with the grpc package it is being compiled against.
// Requires gRPC-Go v1.32.0 or later.
const _ = grpc.SupportPackageIsVersion7

// GrpcFabricVdrEndpointsClient is the client API for GrpcFabricVdrEndpoints service.
//
// For semantics around ctx use and closing/ending streaming RPCs, please refer to
https://pkg.go.dev/google.golang.org/grpc/?tab=doc#ClientConn.NewStream.
type GrpcFabricVdrEndpointsClient interface {
 // Services Endpoint
 Services(ctx context.Context, opts ...grpc.CallOption)
(GrpcFabricVdrEndpoints_ServicesClient, error)
}

type grpcFabricVdrEndpointsClient struct {
    cc grpc.ClientConnInterface
```

```
}

func NewGrpcFabricVdrEndpointsClient(cc grpc.ClientConnInterface)
GrpcFabricVdrEndpointsClient {
    return &grpcFabricVdrEndpointsClient{cc}
}

func (c *grpcFabricVdrEndpointsClient) Services(ctx context.Context, opts
...grpc.CallOption) (GrpcFabricVdrEndpoints_ServicesClient, error) {
    stream, err := c.cc.NewStream(ctx, &GrpcFabricVdrEndpoints_ServiceDesc.Streams[0],
"/grpcFabricVdrAPI.GrpcFabricVdrEndpoints/Services", opts...)
    if err != nil {
        return nil, err
    }
    x := &grpcFabricVdrEndpointsServicesClient{stream}
    return x, nil
}

type GrpcFabricVdrEndpoints_ServicesClient interface {
    Send(*StreamingRawBytes) error
    Recv() (*StreamingRawBytes, error)
    grpc.ClientStream
}

type grpcFabricVdrEndpointsServicesClient struct {
    grpc.ClientStream
}

func (x *grpcFabricVdrEndpointsServicesClient) Send(m *StreamingRawBytes) error {
    return x.ClientStream.SendMsg(m)
}

func (x *grpcFabricVdrEndpointsServicesClient) Recv() (*StreamingRawBytes, error) {
    m := new(StreamingRawBytes)
    if err := x.ClientStream.RecvMsg(m); err != nil {
        return nil, err
    }
    return m, nil
}

// GrpcFabricVdrEndpointsServer is the server API for GrpcFabricVdrEndpoints service.
// All implementations must embed UnimplementedGrpcFabricVdrEndpointsServer
// for forward compatibility
type GrpcFabricVdrEndpointsServer interface {
    // Services Endpoint
    Services(GrpcFabricVdrEndpoints_ServicesServer) error
    mustEmbedUnimplementedGrpcFabricVdrEndpointsServer()
}

// UnimplementedGrpcFabricVdrEndpointsServer must be embedded to have forward compatible
implementations.
type UnimplementedGrpcFabricVdrEndpointsServer struct {
}

func (UnimplementedGrpcFabricVdrEndpointsServer)
Services(GrpcFabricVdrEndpoints_ServicesServer) error {
    return status.Errorf(codes.Unimplemented, "method Services not implemented")
}
func (UnimplementedGrpcFabricVdrEndpointsServer)
mustEmbedUnimplementedGrpcFabricVdrEndpointsServer() {
}

// UnsafeGrpcFabricVdrEndpointsServer may be embedded to opt out of forward compatibility
for this service.
// Use of this interface is not recommended, as added methods to
GrpcFabricVdrEndpointsServer will
// result in compilation errors.
type UnsafeGrpcFabricVdrEndpointsServer interface {
    mustEmbedUnimplementedGrpcFabricVdrEndpointsServer()
}

func RegisterGrpcFabricVdrEndpointsServer(s grpc.ServiceRegistrar, srv
GrpcFabricVdrEndpointsServer) {
```

```
    s.RegisterService(&GrpcFabricVdrEndpoints_ServiceDesc, srv)
}

func _GrpcFabricVdrEndpoints_Services_Handler(srv interface{}, stream grpc.ServerStream)
error {
    return
srv.(GrpcFabricVdrEndpointsServer).Services(&grpcFabricVdrEndpointsServicesServer{stream})
}

type GrpcFabricVdrEndpoints_ServicesServer interface {
    Send(*StreamingRawBytes) error
    Recv() (*StreamingRawBytes, error)
    grpc.ServerStream
}

type grpcFabricVdrEndpointsServicesServer struct {
    grpc.ServerStream
}

func (x *grpcFabricVdrEndpointsServicesServer) Send(m *StreamingRawBytes) error {
    return x.ServerStream.SendMsg(m)
}

func (x *grpcFabricVdrEndpointsServicesServer) Recv() (*StreamingRawBytes, error) {
    m := new(StreamingRawBytes)
    if err := x.ServerStream.RecvMsg(m); err != nil {
        return nil, err
    }
    return m, nil
}

// GrpcFabricVdrEndpoints_ServiceDesc is the grpc.ServiceDesc for GrpcFabricVdrEndpoints
service.
// It's only intended for direct use with grpc.RegisterService,
// and not to be introspected or modified (even as a copy)
var GrpcFabricVdrEndpoints_ServiceDesc = grpc.ServiceDesc{
    ServiceName: "grpcFabricVdrAPI.GrpcFabricVdrEndpoints",
    HandlerType: (*GrpcFabricVdrEndpointsServer)(nil),
    Methods:     []grpc.MethodDesc{},
    Streams: []grpc.StreamDesc{
        {
                StreamName:    "Services",
                Handler:       _GrpcFabricVdrEndpoints_Services_Handler,
                ServerStreams: true,
                ClientStreams: true,
        },
    },
    Metadata: "grpcFabricVdrAPI.proto",
}
```

Dependency Requirements

go.mod

```
module fabric-vdr

go 1.15
```

FabricVdr Data Structures

fabricVdrStruct.go

```
package request

import (
    status "google.golang.org/genproto/googleapis/rpc/status"     // GRPC status
    did "github.com/hyperledger/aries-framework-go/pkg/doc/did"   // Aries Framework GO DID
```

```
)

type StreamingRawBytes struct {
    Subscribe           *Subscribe           `json:"subscribe,omitempty"`
    SubscribeEvent      *SubscribeEvent      `json:"subscribeEvent,omitempty"`
    CreateDoc           *CreateDoc           `json:"createDoc,omitempty"`
    CreateDocEvent      *CreateDocEvent      `json:"createDocEvent,omitempty"`
    ResolveDID          *ResolveDID          `json:"resolveDID,omitempty"`
    ResolveDIDEvent     *ResolveDIDEvent     `json:"resolveDIDEvent,omitempty"`
    Error               *status.Status       `json:"error,omitempty"`
}

/* Subscribe Request */
type Subscribe struct {
    ClientId string
}

type SubscribeEvent struct{
    Event string
}

/* CreateDoc Request */
type CreateDoc struct {
    Doc *did.Doc
}

type CreateDocEvent struct{
    DocResolution *did.DocResolution
}

/* ResolveDID Request */
type ResolveDID struct {
    DocId string
}

type ResolveDIDEvent struct {
    DocResolution *did.DocResolution
}
```

Certificates Generation Script

key.sh

Copy the result of `openssl version` and paste it instead of:
`OpenSSL 3.0.5 5 Jul 2022 (Library: OpenSSL 3.0.5 5 Jul 2022)`

```
#!/bin/bash
set -u
set -e

CN=${1-'localhost'}

currentDir=${PWD}

function checking() {
 echo -e "${PASS}[*] Searching whether OpenSSL is installed${NC}"
 set +e
 OpenSSLPath=`which openssl`
 set -e
 OPENSSL=
 if [[ ! -z "$OpenSSLPath" ]]; then
  echo -e "${PASS}[*] OpenSSL Found${NC}"
  echo -e "${PASS}[*] Checking the OpenSSL Version${NC}"
  if [[ `$OpenSSLPath version` == "OpenSSL 3.0.5 5 Jul 2022 (Library: OpenSSL 3.0.5 5 Jul
2022)" ]]; then
   echo -e "${PASS}[*] OpenSSL has the latest $($OpenSSLPath version) version${NC}"
   OPENSSL=`dirname $OpenSSLPath`
  else
   echo -e "${ERROR}[x] OPENSSL specific version is required. Exited...${NC}"
```

```
   exit
  fi
 else
  echo -e "${ERROR}[x] OPENSSL is not installed. Exited...${NC}"
  exit
 fi
}

function CA() {
 COMMON_NAME=$1
 echo -e "${PASS}[*] Generating Certificate Authority (CA)${NC}"
 DDIR="${currentDir}"
 mkdir -p ${DDIR}/CA
 pushd ${DDIR}/CA
  echo "FabricVdrCApassword" > password.enc
  $OPENSSL/openssl genpkey -out CAKey.pem -algorithm EC -pkeyopt ec_paramgen_curve:P-521
-aes256 -pass file:password.enc
  $OPENSSL/openssl req -new -x509 -nodes -key CAKey.pem -sha256 -days 1024 -out CAcert.pem
--subj "/C=GR/ST=Athens/L=Athens/O=FabricVdrCA/OU=FabricVdrGrpcAPICA/CN=${COMMON_NAME}"
-addext "subjectAltName = DNS:${COMMON_NAME}" -passin file:password.enc
 popd
}

function TLS() {
 TLSDIR=$1
 NODE=$2
 COMMON_NAME=$3

 #TODO:: add input for type of EC key and certificate subject options
 CADIR="${currentDir}/CA"
 mkdir -p ${TLSDIR}/server ${TLSDIR}/client
 cp ${CADIR}/CAcert.pem ${TLSDIR}/server
 cp ${CADIR}/CAcert.pem ${TLSDIR}/client

 echo -e "${PASS}[*] Generating TLS Certificates for ${NODE} Server${NC}"
 pushd ${TLSDIR}/server
  echo "${NODE}ServerTLSPassword" > password.enc
  echo "subjectAltName = DNS:${COMMON_NAME}" > server.ext
  $OPENSSL/openssl genpkey -out serverKey.pem -algorithm EC -pkeyopt
ec_paramgen_curve:P-521 -aes256 -pass file:password.enc
  $OPENSSL/openssl req -new -key serverKey.pem -sha256 -out server.csr --subj
"/C=GR/ST=Athens/L=Athens/O=FabricVdr/OU=FabricVdrGrpcServer/CN=${COMMON_NAME}" -passin
file:password.enc
  $OPENSSL/openssl x509 -req -days 1024 -in server.csr -CA ${TLSDIR}/server/CAcert.pem
-CAkey ${CADIR}/CAKey.pem -CAcreateserial -out serverCert.pem -extfile server.ext -passin
file:${CADIR}/password.enc
  $OPENSSL/openssl pkcs8 -topk8 -inform pem -in serverKey.pem -passin file:password.enc
-outform pem -nocrypt -out serverUnencryptedKey.pem
  rm *.csr
  echo -e "${PASS}[*] Generating TLS Certificates for ${NODE} Server is Finished${NC}"
 popd

 echo -e "${PASS}[*] Generating TLS Certificates for ${NODE} Client${NC}"
 pushd ${TLSDIR}/client
  echo "${NODE}ClientTLSPassword" > password.enc
  echo "subjectAltName = DNS:${COMMON_NAME}" > client.ext
  $OPENSSL/openssl genpkey -out clientKey.pem -algorithm EC -pkeyopt
ec_paramgen_curve:P-521 -aes256 -pass file:password.enc
  $OPENSSL/openssl req -new -key clientKey.pem -sha256 -out client.csr --subj
"/C=GR/ST=Athens/L=Athens/O=FabricVdr/OU=FabricVdrGrpcClient/CN=${COMMON_NAME}" -passin
file:password.enc
  $OPENSSL/openssl x509 -req -days 1024 -in client.csr -CA ${TLSDIR}/client/CAcert.pem
-CAkey ${CADIR}/CAKey.pem -CAcreateserial -out clientCert.pem -extfile client.ext -passin
file:${CADIR}/password.enc
  $OPENSSL/openssl pkcs8 -topk8 -inform pem -in clientKey.pem -passin file:password.enc
-outform pem -nocrypt -out clientUnencryptedKey.pem
  rm *.csr
  echo -e "${PASS}[*] Generating TLS Certificates for ${NODE} Client is Finished${NC}"
 popd
}

PASS="\033[0;32m"
ERROR="\033[0;31m"
```

```
WARNING="\033[0;33m"
NC="\033[0m" # No Color

checking
CA ${CN}
tlsDir="${PWD}/TLS"
TLS ${tlsDir} "FabricVdrgRPC" ${CN}
```

Helper Functions

functions.go

```
package functions

import (
    "crypto/x509"
    "crypto/tls"
    "io/ioutil"
    "path"
    "time"
    "fmt"

    // Gateway
    "github.com/hyperledger/fabric-gateway/pkg/client"
    "github.com/hyperledger/fabric-gateway/pkg/identity"
    // GRPC
    "google.golang.org/grpc"
    "google.golang.org/grpc/credentials"
)

// NewGrpcConnection creates a new gRPC connection with the Peer as a client
func NewGrpcConnection(certFile, keyFile, caFile, gatewayPeer string) (*grpc.ClientConn,
error) {
  // Load certificate of the CA who signed client's certificate
  pemClientCA, err := ioutil.ReadFile(caFile)
  if err != nil {
   return nil, fmt.Errorf("Failed to read CA certificate: %s\n", err)
  }

  certPool := x509.NewCertPool()
  if !certPool.AppendCertsFromPEM(pemClientCA) {
   return nil, fmt.Errorf("Failed to add CA certificate to the pool: %s\n", err)
  }

  // Load server's certificate and private key
  clientCert, err := tls.LoadX509KeyPair(certFile, keyFile)
  if err != nil {
   return nil, fmt.Errorf("Failed to load X509 Key Pair: %s\n", err)
  }

  // Create the credentials and return it
  config := &tls.Config{
   ServerName:   gatewayPeer,
    Certificates: []tls.Certificate{clientCert},
    RootCAs:  certPool,
  }
  creds := credentials.NewTLS(config)

  var opts []grpc.DialOption
  opts = append(opts, grpc.WithTransportCredentials(creds))

  peerEndpoint := gatewayPeer + ":443"
  connection, err := grpc.Dial(peerEndpoint, opts...)
  if err != nil {
   return nil, fmt.Errorf("Connection Failed: %s\n", err)
  }
  return connection, nil
}

func NewGateway(mspID, adminCert, adminKeystore string, clientConnection *grpc.ClientConn)
```

```
(*client.Gateway, error) {
 id, err := newIdentity(mspID, adminCert)
 if err != nil {
   return nil, fmt.Errorf("Failed to create client's identity: %s\n", err)
 }

 sign, err := newSign(adminKeystore)
 if err != nil {
  return nil, fmt.Errorf("Failed to generate client's digital signature: %s\n", err)
 }

 gateway, err := client.Connect(
  id,
  client.WithSign(sign),
  client.WithClientConnection(clientConnection),
  client.WithEvaluateTimeout(5*time.Second),
  client.WithEndorseTimeout(10*time.Second),
  client.WithSubmitTimeout(5*time.Second),
  client.WithCommitStatusTimeout(30*time.Second),
 )
 if err != nil {
  return nil, fmt.Errorf("Failed to connect to gateway: %s\n", err)
 }
 return gateway, nil
}

// newIdentity creates a client identity for this Gateway connection using an X.509
certificate.
func newIdentity(mspID, certPath string) (*identity.X509Identity, error) {
 certificatePEM, err := ioutil.ReadFile(certPath)
 if err != nil {
  return nil, fmt.Errorf("Failed to read certificate file: %s\n", err)
 }

 certificate, err := identity.CertificateFromPEM(certificatePEM)
 if err != nil {
  return nil, fmt.Errorf("Failed to create X.509 certificate from PEM certificate: %s\n",
err)
 }

 id, err := identity.NewX509Identity(mspID, certificate)
 if err != nil {
  return nil, fmt.Errorf("Failed to create a new Identity: %s\n", err)
 }
 return id, nil
}

// newSign creates a function that generates a digital signature from a message digest
using a private key.
func newSign(keyPath string) (identity.Sign, error) {
 files, err := ioutil.ReadDir(keyPath)
 if err != nil {
  return nil, fmt.Errorf("Failed to read private key directory: %s\n", err)
 }

 privateKeyPEM, err := ioutil.ReadFile(path.Join(keyPath, files[0].Name()))
 if err != nil {
  return nil, fmt.Errorf("Failed to read private key file: %s\n", err)
 }

 privateKey, err := identity.PrivateKeyFromPEM(privateKeyPEM)
 if err != nil {
  return nil, fmt.Errorf("Failed to create a private key from PEM key: %s\n", err)
 }

 sign, err := identity.NewPrivateKeySign(privateKey)
 if err != nil {
  return nil, fmt.Errorf("Failed to create a private key from PEM key: %s\n", err)
 }
 return sign, nil
}
```

GRPC Server

server.go

```
package main

import (
 "path/filepath"
 "encoding/gob"
 "crypto/x509"
 "crypto/tls"
 "io/ioutil"
 "context"
 "runtime"
 "reflect"
 "strconv"
 "strings"
 "errors"
 "bytes"
 "sync"
 "flag"
 "time"
 "fmt"
 "net"
 "log"
 "io"
 "os"

 "github.com/davecgh/go-spew/spew"

 // GRPC
 "google.golang.org/grpc"
 "google.golang.org/grpc/codes"
 "google.golang.org/grpc/status"
 "google.golang.org/grpc/credentials"
 "google.golang.org/grpc/keepalive"

 // Aries Framework GO
 did "github.com/hyperledger/aries-framework-go/pkg/doc/did"

 // Fabric Gateway
 "github.com/hyperledger/fabric-gateway/pkg/client"

 // GRPC Bidirectional functions
 grpcFabricVDR "fabric-vdr/grpcFabricVdrAPI"

 // Custom Packages
 fabricVdrSDK "fabric-vdr/packages/functions"
 fabricVDR "fabric-vdr/packages/structs"
 gobCodec "fabric-vdr/packages/codec"
)

const (
 mspID          = "Org1MSP"
 org            = "org1.example.com"
 orgPath        = "../../../multihost_k8s/organization/" + org
 gatewayPeer    = "peer0." + org
 certPath       = orgPath + "/peers/" + gatewayPeer + "/client/client.crt"
 keyPath        = orgPath + "/peers/" + gatewayPeer + "/client/client.key"
 caCertPath     = orgPath + "/peers/" + gatewayPeer + "/client/ca.crt"
 adminCert      = orgPath + "/users/Admin@" + org + "/msp/signcerts/cert.pem"
 adminKeystore  = orgPath + "/users/Admin@" + org + "/msp/keystore"
 channelName    = "mychannel" // Channel Name
 chaincodeName  = "fabricvdr" // Chaincode Name
)

var (
 TLS        = flag.Bool("tls", false, "Connection uses TLS if true, else plain TCP")
 certFile   = flag.String("certFile", "", "The TLS cert file")
 keyFile    = flag.String("keyFile", "", "The TLS key file")
 caFile     = flag.String("caFile", "", "The CA file")
 host       = flag.String("host", "localhost", "The server's host")
```

```
 port       = flag.Int("port", 4001, "The server's port")

 // Keepalive EnforcementPolicy
 kaep = keepalive.EnforcementPolicy{
  MinTime:              5 * time.Second, // If a client pings more than once every 5
seconds, terminate the connection
  PermitWithoutStream: true,             // Allow pings even when there are no active
streams
 }

 // Keepalive ServerParameters
 kasp = keepalive.ServerParameters{
  MaxConnectionIdle:     1 * time.Hour, // If a client is idle for 1 Hour, send a GOAWAY
  MaxConnectionAge:      0, // Long-lived streams
  MaxConnectionAgeGrace: 0,  // Long-lived streams
  Time:                  5 * time.Second,  // Ping the client if it is idle for 5 seconds
to ensure the connection is still active
  Timeout:               1 * time.Second,  // Wait 1 second for the ping ack before
assuming the connection is dead
 }

 Reset  = "\033[0m"   // No Color
 Red    = "\033[31m"  // Red Color
 Green  = "\033[32m"  // Green Color

 // Logger
 buf bytes.Buffer //Initialiaze a buffer
 // INFO
 info = func(req, msg string) {
  logger := log.New(&buf, "[" + Green + "INFO" + Reset + "] [" + req + "] --> ", log.Ldate
| log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Green + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }
 // ERROR
 warn = func(req, msg string) {
  logger := log.New(&buf, "[" + Red + "WARN" + Reset + "] [" + req + "] --> ", log.Ldate |
log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Red + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }

 // Connection to the Channel
 network  *client.Network
 // Connection to the Contract
 contract *client.Contract
)

type sub struct {
 // stream refers to the bidirectional stream of the RPC
 stream   grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer
 // finished is used to signal closure of a client subscribing goroutine
 finished chan<- bool
}

type server struct {
 grpcFabricVDR.GrpcFabricVdrEndpointsServer
 // SubscriptionClientId is a HashMap that uses clientId as Key and
 // grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer (stream) as Value
 // (stream is the server side of the RPC stream)
 subscriptionClientId map[string]sub
 // SubscriptionStream is a reverse HashMap of subscriptionClientId hashMap that
 // uses grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer (stream) as Key
 // and clientId as Value
 // (stream is the server side of the RPC stream)
 subscriptionStream map[grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer]string
 // Protect array from concurrent access
 mu sync.Mutex
 // UnsuccessfulEvents is a HashMap that uses clientId as Key and
 // []*grpcFabricVDR.StreamingResponse as Value, that stores events
 // of Disconnected Clients for future Delivery
```

```
 unsuccessfulEvents map[string][]*fabricVDR.StreamingRawBytes
 // DocResolutionEvents is a HashMap that uses DID as Key and *did.DocResolution as Value.
 // It is used as a pool in order to avoid to pass huge amount of data between chaincode
 // events and gRPC server
 docResolutionEvents map[string]*did.DocResolution
}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////          Services Endpoint       ////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
func (s *server) Services(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer) error
{
 for {
  in := new(fabricVDR.StreamingRawBytes)
  err := stream.RecvMsg(in)
  if err == io.EOF {
   //Client has closed the connection
   return nil
  }

  if err != nil {
   //Unable to read stream from client
   return err
  }

  spew.Dump(in)

  // Checking the type of the incoming request.
  switch {
  case in.Subscribe != nil:
   info("services", "Subscribe Request")
   clientId := in.Subscribe.ClientId
   s.subscribe(stream, clientId)
  case in.CreateDoc != nil:
   info("services", "Create DID DOC Request")
   s.createDoc(stream, in.CreateDoc.Doc)
  case in.ResolveDID != nil:
   info("services", "Resolve DID Request")
   s.resolveDID(stream, in.ResolveDID.DocId)
  default:
   warn("services", "Received Unknown GRPC Request")
  }
 }
 return nil
}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////          Subscribe Endpoint      ////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
func (s *server) subscribe(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer,
clientId string) error {
 info("subscribe", "Subscribe a new client")

 // Check if clientId is empty
 if clientId == "" {
  msg := "ClientID should not be empty"
  warn("subscribe", msg)
  s.disseminateError(stream, msg)
  return nil
 }

 //Check if it is already registered
 var isExists bool
 if _, isExists = s.subscriptionClientId[clientId]; isExists {
  msg := "Client " + clientId + " is already Subscripted"
  warn("subscribe", msg)
  s.disseminateError(stream, msg)
  return nil
 }
```

```
var messages []*fabricVDR.StreamingRawBytes

// Make a finished channel before subscription
fin := make(chan bool)

if !isExists && clientId != "" {
 info("subscribe", "Received subscribe request from ID: " + clientId)

 // Save the subscriber stream according to the given client ID
 s.subscriptionClientId[clientId] = sub{stream: stream, finished: fin}
 // Save the clientId according to the given stream
 s.subscriptionStream[stream] = clientId

 // Formulate SubscribeEvent struct
 var msg *fabricVDR.StreamingRawBytes
 msg = &fabricVDR.StreamingRawBytes{
  SubscribeEvent: &fabricVDR.SubscribeEvent{
       Event : "Client Subscribed",
  },
 }

 messages = append(messages, msg)

 // Upon subscription, check whether there are available
 // events designated to the newly created clientId
 // Lock to protect the s.unsuccessfulEvents HashMap
 // from appending new records when searching for events
 s.mu.Lock()
 info("subscribe", "Sending Missed Events for clientId: " + clientId)
 // Fetch the unsuccessfulEvents
 events, found := s.unsuccessfulEvents[clientId]
 if !found {
  info("subscribe", "All Events Sent for clientId: " + clientId)
 }
 messages = append(messages, events...)
 // After append to messages, then delete the Key
 delete(s.unsuccessfulEvents, clientId)
 s.mu.Unlock()
}

for i, message := range messages {
 if err := stream.SendMsg(message); err != nil {
  warn("subscribe", "Failed to stream Event: " + err.Error())
  // If error occured during stream, then re-append into the unsuccessfulEvents HashMap
  s.mu.Lock()
  s.unsuccessfulEvents[clientId] = messages[i:]
  s.mu.Unlock()
  // Unsubscribe client
  s.unsubscribe(stream)
 }
}

// Spawn a goroutine that keeps watching for client disconnects
go func(){
 ctx := stream.Context()
 // Keep this scope alive because once this scope exits - the stream is closed
 for {
  select {
   case <-fin:
    warn("subscribe", "Closing stream for client ID: " + clientId)
    s.unsubscribe(stream)
    return
   case <- ctx.Done():
    warn("subscribe", "Client ID " + clientId + " has disconnected")
    s.unsubscribe(stream)
    return
  }
 }
}()
return nil
}
```

```
// Disseminate error events from endpoints to clients.
func (s *server) disseminateError (x interface{}, errMsg string) {
 var stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer
 var clientId string
 switch x.(type) {
  case string:
   clientId = x.(string)
   stream = s.findStreamFromClientId(clientId)
  case grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer:
   clientId = s.findClientIdFromStream(stream)
   stream = x.(grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer)
  default:
   warn("disseminateError", "Unknown input type")
 }

 // Form Error Message
 var msg *fabricVDR.StreamingRawBytes
 msg = &fabricVDR.StreamingRawBytes{
  Error: status.New(codes.InvalidArgument, errMsg).Proto(),
 }

 // If stream is nil, store it as unsuccessful
 if stream == nil || (reflect.ValueOf(stream).Kind() == reflect.Ptr &&
reflect.ValueOf(stream).IsNil()) {
  s.mu.Lock()
  events, _ := s.unsuccessfulEvents[clientId]
  s.unsuccessfulEvents[clientId] = append(events, msg)
  s.mu.Unlock()
  return
 }

 // Send message through the stream
 if err := stream.SendMsg(msg); err != nil {
  warn("disseminateError", "Failed to disseminate Error: " + err.Error())
  s.mu.Lock()
  events, _ := s.unsuccessfulEvents[clientId]
  s.unsuccessfulEvents[clientId] = append(events, msg)
  s.mu.Unlock()
  // Unsubscribe client
  s.unsubscribe(stream)
 }
}

// Find the clientId using the stream
func (s *server) findStreamFromClientId(clientId string)
grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer {
 //Load the clientId if exists
 if sub, isExists := s.subscriptionClientId[clientId]; isExists {
  return sub.stream
 }
 return nil
}

// Find the clientId using the stream
func (s *server) findClientIdFromStream(stream
grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer) string {
 // Find which clientId has this stream
 if clientId, isExists := s.subscriptionStream[stream]; isExists {
  return clientId
 }
 return ""
}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////       Unsubscribe Endpoint (INTERNALLY USED ONLY)      /////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
// Unsubscribe is used when a client cuts ties with the server.
func (s *server) unsubscribe(x interface{}) error {
 var clientId string
 var stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer
 // Find the type of the input
```

```
 switch x.(type) {
  case string:
   clientId = x.(string)
   stream = s.findStreamFromClientId(x.(string))
  case grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer:
   clientId =
s.findClientIdFromStream(x.(grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer))
   stream = x.(grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer)
  default:
   warn("unsubscribe", "Unknown input type")
 }
 delete(s.subscriptionClientId, clientId)
 delete(s.subscriptionStream, stream)
 return nil
}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////   Create DID DOC   //////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
func (s *server) createDoc(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer,
didDoc *did.Doc) error {
 info("createDoc", "Create a new DID DOC")

 // Find the id of the caller.
 caller := s.findClientIdFromStream(stream)
 info("createDoc", "Received Create DidDoc request from ID: " + caller)

 // Formulate Doc Resolution struct
 var docResolution *did.DocResolution
 docResolution = &did.DocResolution{
  Context: didDoc.Context,
  DIDDocument: didDoc,
  DocumentMetadata: &did.DocumentMetadata{},
 }

 // Extract the docId
 docId := string(docResolution.DIDDocument.ID)
 // Store it in the docResolutionEvents Hashmap
 s.mu.Lock()
 s.docResolutionEvents[docId] = docResolution
 s.mu.Unlock()

 // GOB Encoder to convert Struct to Bytes
 var b bytes.Buffer
 enc := gob.NewEncoder(&b)
 err := enc.Encode(docResolution)
 if err != nil {
  warn("createDoc", "Error Converting Struct to Bytes: " + err.Error())
  return nil
 }

 // Create transient map
 transient := make(map[string][]byte)
 transient["DID"] = []byte(docId)
 transient["docResolution"] = b.Bytes()

 // Submit Txn
 _, err = contract.Submit(
  "createDoc",
  client.WithArguments(string(caller)),
  client.WithTransient(transient),
  client.WithEndorsingOrganizations(mspID),
 )

 if err != nil {
  msg := "Failed to submit transaction: "
  var errMsg string
  switch err := err.(type) {
  case *client.EndorseError:
   errMsg = fmt.Errorf("Transaction %s failed to endrose with gRPC status %v: %w",
err.TransactionID, status.Code(err), err).Error()
```

```
  case *client.SubmitError:
   errMsg = fmt.Errorf("Transaction %s failed to submit to the orderer with gRPC status %v:
%w", err.TransactionID, status.Code(err), err).Error()
  case *client.CommitStatusError:
   if errors.Is(err, context.DeadlineExceeded) {
    errMsg = fmt.Errorf("Timeout waiting for transaction %s commit status: %w",
err.TransactionID, err).Error()
   } else {
    errMsg = fmt.Errorf("Transaction %s failed to obtain commit status with gRPC status %v:
%w", err.TransactionID, status.Code(err), err).Error()
   }
  case *client.CommitError:
   errMsg = fmt.Errorf("Transaction %s failed to commit with status %d: %w",
err.TransactionID, int32(err.Code), err).Error()
  default:
   errMsg = "Unknown Error: " + err.Error()
  }

  msg = msg + errMsg
  warn("createDoc", msg)
  // In case of a error delete it from the Hashmap
  s.mu.Lock()
  delete(s.docResolutionEvents, docId)
  s.mu.Unlock()
  s.disseminateError(stream, msg)
  return nil
 }

 return nil
}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////        Resolve DID     //////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
func (s *server) resolveDID(stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer,
docId string) error {
 info("resolveDID", "Resolve a DID")

 // Find the id of the caller.
 caller := s.findClientIdFromStream(stream)
 info("resolveDID", "Received Resolve DID request from ID: " + caller)

 // Check is docId is empty
 if len(docId) == 0 {
  msg := "DocId should not be empty."
  warn("resolveDID", msg)
  s.disseminateError(stream, msg)
  return nil
 }

 // Evaluate Transaction
 result, err := contract.EvaluateTransaction("resolveDid", string(docId))
 if err != nil {
  msg := "Failed to evaluate transaction: "
  var errMsg string
  if errors.Is(err, context.DeadlineExceeded) {
   errMsg = fmt.Errorf("Timeout: %w", err).Error()
  } else {
   errMsg = fmt.Errorf("Evaluation failed due to gRPC status %v: %w", status.Code(err),
err).Error()
  }
  msg = msg + errMsg
  warn("resolveDID", msg)
  s.disseminateError(stream, msg)
  return nil
 }

 // GOB Decoder to convert Bytes to Struct
 var docResolution *did.DocResolution
 r := bytes.NewReader(result)
 dec := gob.NewDecoder(r)
```

```
 err = dec.Decode(&docResolution)
 if err != nil {
  warn("resolveDID", "Error Converting Bytes to Struct: " + err.Error())
  return nil
 }

 // Disseminate Event to Device
 var msg *fabricVDR.StreamingRawBytes
 msg = &fabricVDR.StreamingRawBytes{
  ResolveDIDEvent: &fabricVDR.ResolveDIDEvent{
   DocResolution: docResolution,
  },
 }

 if err := stream.SendMsg(msg); err != nil {
  warn("resolveDID", "Failed to stream DID Doc: " + err.Error())
  // Unsubscribe client
  s.unsubscribe(stream)
  return nil
 }
 return nil
}

//LoadServerTLSCredentials loads the certificates in case of TLS and creates the server's
certificate pool.
func loadServerTLSCredentials(certFile, keyFile, caFile string)
(credentials.TransportCredentials, error) {
 // Load certificate of the CA who signed client's certificate
 pemClientCA, err := ioutil.ReadFile(caFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to read CA certificate")
 }

 certPool := x509.NewCertPool()
 if !certPool.AppendCertsFromPEM(pemClientCA) {
  return nil, fmt.Errorf("Failed to add CA certificate to the certificate pool")
 }

 // Load server's certificate and private key
 serverCert, err := tls.LoadX509KeyPair(certFile, keyFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to load X509 Key Pair.")
 }

 // Create the credentials and return it
 config := &tls.Config{
  Certificates: []tls.Certificate{serverCert},
  ClientAuth:   tls.RequireAndVerifyClientCert,
  ClientCAs:    certPool,
 }
 return credentials.NewTLS(config), nil
}

//Path finds the absolute path of the current directory and joins the remaining path.
func path(rel string) (string, error) {
 currentPath, err := os.Getwd()
 if err != nil {
  return "", fmt.Errorf("Failed to retrieve current directory: %v", err)
 }

 if filepath.IsAbs(rel) {
  isExists, err := exists(rel)
  if err != nil {
   return "", err
  }
  if !isExists {
   return "", fmt.Errorf("File does not exists.")
  }
  return rel, nil
 }

 formNewPath := filepath.Join(currentPath, rel)
 isExists, err := exists(formNewPath)
```

```
 if err != nil {
  return "", err
 }
 if !isExists {
  return "", fmt.Errorf("File does not exists.")
 }
 return formNewPath, nil
}

// Checks if a path exists
func exists(name string) (bool, error) {
 _, err := os.Stat(name)
 if err == nil {
  return true, nil
 }
 if errors.Is(err, os.ErrNotExist) {
  return false, nil
 }
 return false, err
}

// Goroutine that listens for createDoc chaincode events
func (s *server) listenCreateDocEvents(wg *sync.WaitGroup, maxProcs int) {
 defer wg.Done()
 waitc := make(chan struct{})

 ctx := context.Background()

 go func(){
  done := make(chan bool, maxProcs) // Done Channel
  errc := make(chan error) // Error Channel
  var err error

  // Register for chaincode events
  events, err := network.ChaincodeEvents(ctx, chaincodeName)
  if err != nil {
   warn("createDocEvents", "Failed creating the Gateway: " + err.Error())
   errc <- err
  }

  // Start Listening
  for {
   select {
    case err := <-errc:
     warn("createDocEvents", "Fail to connect to Hyperledger Fabric Gateway: " +
err.Error())
     return
    case <-done:
     warn("createDocEvents", "Goroutine execution failed")
     return
    case chaincodeEvent, ok := <-events:
     if ok {
      ccPayload := string(chaincodeEvent.Payload)
      info("createDocEvents", "Received Chaincode Event Payload: {" + ccPayload + "}")
      payload := strings.Split(ccPayload, ",")
      deviceId := strings.TrimSpace(strings.Split(strings.Trim(payload[0],"{}"), "=")[1])
      docID := strings.TrimSpace(strings.Split(strings.Trim(payload[1],"{}"), "=")[1])
      // Disseminate DocResolution back to the intended device.
      s.disseminateEvents(docID, deviceId)
     }
   }
  }
 }()
 <-waitc
}

// Disseminate events from the blockchain back to the agents
func (s *server) disseminateEvents(docId, deviceId string) error {
 info("createDocEvents", "Disseminate Events ...")

 s.mu.Lock()
 docResolution, found := s.docResolutionEvents[docId]
 if !found {
```

```
  warn("createDocEvents", "No DocResolution Found.")
  return nil
 }
 delete(s.docResolutionEvents, docId)

 // Disseminate Event to Device
 var msg *fabricVDR.StreamingRawBytes
 msg = &fabricVDR.StreamingRawBytes{
  CreateDocEvent: &fabricVDR.CreateDocEvent{
   DocResolution: docResolution,
  },
 }

 // Check if the client is already registered
 stream := s.findStreamFromClientId(deviceId)
 if stream == nil {
  events, _ := s.unsuccessfulEvents[deviceId]
  s.unsuccessfulEvents[deviceId] = append(events, msg)
  return nil
 }
 s.mu.Unlock()

 // Send data over the gRPC stream to the client
 if err := stream.SendMsg(msg); err != nil {
  warn("createDocEvents", "Failed to stream Event: " + err.Error())

  s.mu.Lock()
  events, _ := s.unsuccessfulEvents[deviceId]
  s.unsuccessfulEvents[deviceId] = append(events, msg)
  s.mu.Unlock()

  info("createDocEvents", "Unsubscribe client: " + deviceId)
  delete(s.subscriptionClientId, deviceId)
  delete(s.subscriptionStream, stream)
  return nil
 }
 return nil
}

func main() {
 os.Setenv("DISCOVERY_AS_LOCALHOST", "false")
 flag.Parse()

 // Remove Color in Windows
 if runtime.GOOS == "windows" {
  Reset = ""
  Red = ""
  Green = ""
 }

 // Register GOB
 if !gobCodec.Register() {
  warn("main", "GOB Register map[string]interface Failed.")
 }

 // Find the number of CPUs
 maxProcs := runtime.NumCPU()
 info("main", "Number of CPUs: " + strconv.Itoa(maxProcs))
 runtime.GOMAXPROCS(maxProcs)

 // Connect to Fabric
 conn, err := fabricVdrSDK.NewGrpcConnection(certPath, keyPath, caCertPath, gatewayPeer)
 if err != nil {
  warn("main", "Failed connecting to Fabric: " + err.Error())
  os.Exit(1)
 }
 defer conn.Close()

 // Connect to the Gateway
 gateway, err := fabricVdrSDK.NewGateway(mspID, adminCert, adminKeystore, conn)
 if err != nil {
  warn("main", "Failed creating the Gateway: " + err.Error())
  os.Exit(1)
```

```
}
defer gateway.Close()

// Connect to the Channel
network  = gateway.GetNetwork(channelName)
// Connect to the Contract
contract = network.GetContract(chaincodeName)

// Server Initialization
s := &server{
 subscriptionClientId: make(map[string]sub),
 subscriptionStream: make(map[grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesServer]string),
 unsuccessfulEvents: make(map[string][]*fabricVDR.StreamingRawBytes),
 docResolutionEvents: make(map[string]*did.DocResolution),
}

var wg sync.WaitGroup
wg.Add(1)

// Goroutine that connects to the Hyperledger Fabric Gateway and listens for events
go s.listenCreateDocEvents(&wg, maxProcs)

// Start a TCP listener
lis, err := net.Listen("tcp", fmt.Sprintf("%s:%d", *host, *port))
if err != nil {
 warn("main", "Failed to listen: " + err.Error())
 os.Exit(1)
}

var GRPC string
GRPC = "GRPC"

// Grpc Server Options
var opts []grpc.ServerOption
opts = []grpc.ServerOption{
 grpc.KeepaliveEnforcementPolicy(kaep), // Keepalive EnforcementPolicy
 grpc.KeepaliveParams(kasp),            // Keepalive ServerParameters
 grpc.CustomCodec(&gobCodec.Codec{}),   // Custom Codec
}

// Enable TLS
if *TLS {
 if *caFile == "" {
  *caFile, err = path("CA/CAcert.pem")
  if err != nil  {
       warn("main", "CA File: " + err.Error())
       os.Exit(1)
  }
 }
 if *certFile == "" {
  *certFile, err = path("TLS/server/serverCert.pem")
  if err != nil  {
       warn("main", "Cert File: " + err.Error())
       os.Exit(1)
  }
 }
 if *keyFile == "" {
  *keyFile, err = path("TLS/server/serverUnencryptedKey.pem")
  if err != nil  {
       warn("main", "Key File: " + err.Error())
       os.Exit(1)
  }
 }
 creds, err := loadServerTLSCredentials(*certFile, *keyFile, *caFile)
 if err != nil {
  warn("main", "Failed to Generate Credentials: " + err.Error())
  os.Exit(1)
 }
 opts = append(opts, grpc.Creds(creds))
 GRPC = "GRPCS"
}

info("main", "Starting " + GRPC + " server on Host " + *host + " and Port " +
```

```
strconv.Itoa(*port))
 // Create a new GRPC server
 grpcServer := grpc.NewServer(opts...)
 // Register Services
 grpcFabricVDR.RegisterGrpcFabricVdrEndpointsServer(grpcServer, s)
 // Serve gRPC requests
 grpcServer.Serve(lis)
}
```

GRPC Server Unit Tests

unitTest.go

```
package main

import (
 "path/filepath"
 "encoding/hex"
 "crypto/rand"
 "crypto/x509"
 "crypto/tls"
 "io/ioutil"
 "unicode"
 "runtime"
 "context"
 "strings"
 "strconv"
 "errors"
 "bytes"
 "time"
 "flag"
 "sync"
 "fmt"
 "log"
 "io"
 "os"
 // GRPC
 "google.golang.org/grpc"
 "google.golang.org/grpc/credentials"
 "google.golang.org/grpc/keepalive"
 "google.golang.org/grpc/codes"
 "google.golang.org/grpc/status"
 // Aries Framework GO
 did "github.com/hyperledger/aries-framework-go/pkg/doc/did"
 // GRPC Bidirectional functions
 grpcFabricVDR "fabric-vdr/grpcFabricVdrAPI"
 //Custom Packages
 fabricVDR "fabric-vdr/packages/structs"
 gobCodec "fabric-vdr/packages/codec"
)

var (
 TLS            = flag.Bool("tls", false, "Connection uses TLS if true, else plain TCP")
 certFile       = flag.String("certFile", "", "The TLS cert file")
 keyFile        = flag.String("keyFile", "", "The TLS key file")
 caFile         = flag.String("caFile", "", "The CA file")
 serverAddr     = flag.String("serverAddr", "localhost:4001", "The server address in the
format of host:port")
 serverHostOverride = flag.String("serverHostOverride", "localhost", "The server name used
to verify the hostname returned by the TLS handshake")

 clientId = flag.String("clientID", "", "Declare the ID of the device.")

 // Keepalive ClientParameters
 kacp = keepalive.ClientParameters{
  Time: 10 * time.Second, // Send pings every 10 seconds if there is no activity
  Timeout: time.Second,  // Wait 1 sec for ping ack before considering the connection dead
  PermitWithoutStream: true, // Send pings even without active streams
 }
```

```
 Reset  = "\033[0m"   // No Color
 Red    = "\033[31m"  // Red Color
 Green  = "\033[32m"  // Green Color

 // Logger
 buf bytes.Buffer //Initialiaze a buffer
 // INFO
 info = func(req, msg string) {
  logger := log.New(&buf, "[" + Green + "INFO" + Reset + "] [" + req + "] --> ", log.Ldate
| log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Green + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }
 // ERROR
 warn = func(req, msg string) {
  logger := log.New(&buf, "[" + Red + "WARN" + Reset + "] [" + req + "] --> ", log.Ldate |
log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Red + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }

 tests map[string]map[string]unitTests
 testGlobal string
 scenarioGlobal string
 done = make(chan bool)
 pass = make(chan string)
 fail = make(chan string)
 failGlobal = false

 val, _ = randomHex(20)
 successDID = "did:fabric:" + val
)

type unitTests struct {
 in *fabricVDR.StreamingRawBytes
 expected expectation
}

type expectation struct {
 out *fabricVDR.StreamingRawBytes
 err error
}

func receiver(wg *sync.WaitGroup, stream
grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesClient, rch chan <-
*fabricVDR.StreamingRawBytes){
 defer wg.Done()
 waitc := make(chan struct{})

 //Reading Data from the server
 go func() {
  for {
   if stream == nil {
    // Stream Failed. Try again TODO
    warn("receiver", "Stream Failed")
    os.Exit(1)
    close(waitc)
    return
   }
   // Receivind Data from the server
   in := new(fabricVDR.StreamingRawBytes)
   err := stream.RecvMsg(in)
   if err == io.EOF {
    // Reading is done.
    close(waitc)
    return
   }

   if err != nil && err != io.EOF {
    warn("receiver", "Failed to Receive Message : " + err.Error())
    stream = nil
```

```
    time.Sleep(5 * time.Second)
    //Retry on failure
    continue
   }

   var req *fabricVDR.StreamingRawBytes
   wantTest := tests[testGlobal][scenarioGlobal]
   switch testGlobal {
    case "Subscribe":
     switch scenarioGlobal {
      case "Success":
       want := wantTest.expected.out.SubscribeEvent.Event
       got := in.SubscribeEvent.Event
       checkResponse(testGlobal, scenarioGlobal, got, want)
      case "ClientID Empty":
       want := wantTest.expected.out.Error.String()
       got := in.Error.String()
       checkResponse(testGlobal, scenarioGlobal, got, want)
      case "ClientID Already Exists":
       want := wantTest.expected.out.Error.String()
       got := in.Error.String()
       checkResponse(testGlobal, scenarioGlobal, got, want)
     }
    case "CreateDoc":
     switch scenarioGlobal {
      case "Empty DocId":
       want := wantTest.expected.out.Error.String()
       got := in.Error.String()
       checkResponse(testGlobal, scenarioGlobal, got, want)
      case "Success":
       want := wantTest.expected.out.CreateDocEvent.DocResolution.DIDDocument.ID
       got := in.CreateDocEvent.DocResolution.DIDDocument.ID
       checkResponse(testGlobal, scenarioGlobal, got, want)
     }
    case "ResolveDID":
     switch scenarioGlobal {
      case "Empty DocId":
       want := wantTest.expected.out.Error.String()
       got := in.Error.String()
       checkResponse(testGlobal, scenarioGlobal, got, want)
      case "Success":
       want := wantTest.expected.out.ResolveDIDEvent.DocResolution.DIDDocument.ID
       got := in.ResolveDIDEvent.DocResolution.DIDDocument.ID
       checkResponse(testGlobal, scenarioGlobal, got, want)
     }
    default:
     warn("receiver", "Received Unknown GRPC Response...")
   }
   if req == nil {continue;}
   rch <- req
  }
  close(rch)
 }()
 <-waitc
}

func sender(wg *sync.WaitGroup, stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesClient,
sch <-chan *fabricVDR.StreamingRawBytes) {
 defer wg.Done()
 for{
  select {
   case x, ok := <-sch:
    if ok {
     data := x
     if err := stream.SendMsg(data); err != nil {
      warn("sender", "Failed to send Request: " + err.Error())
     }
    } else {
     info("sender", "Closing Stream!")
     if err := stream.CloseSend(); err != nil {
      warn("sender", "Failed to close stream: " + err.Error())
     }
     wg.Done()
```

```
      return
     }
    }
   }
}

func loadClientTLSCredentials(certFile, keyFile, caFile string)
(credentials.TransportCredentials, error) {
 // Load certificate of the CA who signed client's certificate
 pemClientCA, err := ioutil.ReadFile(caFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to read CA certificate")
 }

 certPool := x509.NewCertPool()
 if !certPool.AppendCertsFromPEM(pemClientCA) {
  return nil, fmt.Errorf("Failed to add CA certificate to the pool")
 }

 // Load server's certificate and private key
 clientCert, err := tls.LoadX509KeyPair(certFile, keyFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to load X509 Key Pair.")
 }

 // Create the credentials and return it
 config := &tls.Config{
  ServerName: *serverHostOverride,
  Certificates: []tls.Certificate{clientCert},
  RootCAs:      certPool,
 }
 return credentials.NewTLS(config), nil
}

//Path finds the absolute path of the current directory and joins the remaining path.
func path(rel string) (string, error) {
 currentPath, err := os.Getwd()
 if err != nil {
  return "", fmt.Errorf("Failed to retrieve current directory: %v", err)
 }

 if filepath.IsAbs(rel) {
  isExists, err := exists(rel)
  if err != nil {
   return "", err
  }
  if !isExists {
   return "", fmt.Errorf("File does not exists.")
  }
  return rel, nil
 }

 formNewPath := filepath.Join(currentPath, rel)
 isExists, err := exists(formNewPath)
 if err != nil {
  return "", err
 }
 if !isExists {
  return "", fmt.Errorf("File does not exists.")
 }
 return formNewPath, nil
}

func exists(name string) (bool, error) {
 _, err := os.Stat(name)
 if err == nil {
  return true, nil
 }
 if errors.Is(err, os.ErrNotExist) {
  return false, nil
 }
 return false, err
}
```

```
func randomHex(n int) (string, error) {
 bytes := make([]byte, n)
 if _, err := rand.Read(bytes); err != nil {
  return "", err
 }
 return hex.EncodeToString(bytes), nil
}

func toUpper(str string) string {
 strRune := []rune(str)
 strRune[0] = unicode.ToUpper(strRune[0])
 return string(strRune)
}

func checkResponse(test, scenario, got, want string) {
 done <- true
 showTest := toUpper(test)
 showScenario := strings.Join(strings.Split(toUpper(scenario), " "), "_")
 if !strings.Contains(got, want) {
  fail <- fmt.Sprintf("  --- FAIL: Test" + showTest + "/" + showScenario + "\n")
  fail <- fmt.Sprintf("          Expected : %q\n          Got      : %q\n", want, got)
 } else {
  pass <- fmt.Sprintf("  --- PASS: Test" + showTest + "/" + showScenario + "\n")
 }
}

func didDocument(docId string) *did.Doc {
 didDoc := []byte(`{
  "@context": ["https://w3id.org/did/v0.11"],
  "id": "` + docId + `",
  "publicKey": [
    {
     "id": "did:example:123456789abcdefghi#keys-1",
     "type": "Secp256k1VerificationKey2018",
     "owner": "did:example:123456789abcdefghi",
     "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
  ],
  "authentication": [
   {
    "type": "Secp256k1VerificationKey2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
   }
  ],
  "service": [
   {
    "id": "did:example:123456789abcdefghi#inbox",
    "type": "SocialWebInboxService",
    "serviceEndpoint": "https://social.example.com/83hfh37dj",
    "spamCost": {
     "amount": "0.50",
     "currency": "USD"
    }
   }
  ],
  "created": "2002-10-10T17:00:00Z",
  "updated": "2002-10-10T17:00:00Z"
}`)
 doc, _ := did.ParseDocument(didDoc)
 return doc
}

func main() {
 // Remove Color in Windows
 if runtime.GOOS == "windows" {
  Reset = ""
  Red = ""
  Green = ""
 }

 flag.Parse()
 if !gobCodec.Register() {
```

```
 warn("main", "GOB Register map[string]interface")
}

maxProcs := runtime.NumCPU()
info("main", "Number of CPUs: " + strconv.Itoa(maxProcs))
runtime.GOMAXPROCS(maxProcs)

if *clientId == "" {
 warn("main", "Client ID must be declared before the GRPC Client starts.")
 os.Exit(1)
}

var opts []grpc.DialOption
opts = []grpc.DialOption{
 grpc.WithKeepaliveParams(kacp),        // Keepalive Client Parameters
 grpc.WithInitialConnWindowSize(256*1024),
 grpc.WithCodec(&gobCodec.Codec{}),     // Custom Codec
}

var err error
var GRPC string

if *TLS {
 if *caFile == "" {
  *caFile, err = path("CA/CAcert.pem")
  if err != nil  {
        warn("main", "CA File: " + err.Error())
        os.Exit(1)
  }
 }
 if *certFile == "" {
  *certFile, err = path("TLS/client/clientCert.pem")
  if err != nil  {
        warn("main", "Cert File: " + err.Error())
        os.Exit(1)
  }
 }
 if *keyFile == "" {
  *keyFile, err = path("TLS/client/clientUnencryptedKey.pem")
  if err != nil  {
        warn("main", "Key File: " + err.Error())
        os.Exit(1)
  }
 }
 creds, err := loadClientTLSCredentials(*certFile, *keyFile, *caFile)
 if err != nil {
  warn("main", "Failed to Generate Credentials: " + err.Error())
  os.Exit(1)
 }
 opts = append(opts, grpc.WithTransportCredentials(creds))
 GRPC = "GRPCS"
} else {
 opts = append(opts, grpc.WithInsecure())
 GRPC = "GRPC"
}

info("main", "Starting " + GRPC + " client on host " + *serverAddr)

conn, err := grpc.Dial(*serverAddr, opts...)
if err != nil {
 warn("main", "Fail to Dial: " + err.Error())
 os.Exit(1)
}
defer conn.Close()

client := grpcFabricVDR.NewGrpcFabricVdrEndpointsClient(conn)

info("main", "Client " + *clientId + " is connected")

stream, err := client.Services(context.Background())
if err != nil {
 warn("main", *clientId + ".Services(_): " + err.Error())
 os.Exit(1)
```

```
}

var wg sync.WaitGroup
rchannel := make(chan *fabricVDR.StreamingRawBytes, maxProcs)
schannel := make(chan *fabricVDR.StreamingRawBytes, maxProcs)
wg.Add(2)

// Magic happens here
go func() {
 go receiver(&wg, stream, rchannel)
 go sender(&wg, stream, schannel)
 for {
  select {
   case dataToSend , ok := <-rchannel:
    if ok {
     schannel <- dataToSend
    }
   case msg := <-pass:
    fmt.Printf(msg)
   case msg := <-fail:
    failGlobal = true
    fmt.Printf(msg)
  }
 }
}()

tests = map[string]map[string]unitTests{
 "Subscribe": {
  "ClientID Empty": {
   in: &fabricVDR.StreamingRawBytes{
    Subscribe: &fabricVDR.Subscribe{
     ClientId: "",
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     Error: status.New(codes.InvalidArgument, "ClientID should not be empty").Proto(),
    },
   },
  },
  "Success": {
   in: &fabricVDR.StreamingRawBytes{
    Subscribe: &fabricVDR.Subscribe{
     ClientId: *clientId,
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     SubscribeEvent: &fabricVDR.SubscribeEvent{
      Event : "Client Subscribed",
     },
    },
   },
  },
  "ClientID Already Exists": {
   in: &fabricVDR.StreamingRawBytes{
    Subscribe: &fabricVDR.Subscribe{
     ClientId: *clientId,
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     Error: status.New(codes.InvalidArgument, "Client " + *clientId + " is already
Subscripted").Proto(),
    },
   },
  },
 },
 "CreateDoc": {
  "Empty DocId": {
   in: &fabricVDR.StreamingRawBytes{
    CreateDoc: &fabricVDR.CreateDoc{
     Doc: didDocument(""),
```

```
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     Error: status.New(codes.InvalidArgument, "DocId should not be empty.").Proto(),
    },
   },
  },
  "Success": {
   in: &fabricVDR.StreamingRawBytes{
    CreateDoc: &fabricVDR.CreateDoc{
     Doc: didDocument(successDID),
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     CreateDocEvent: &fabricVDR.CreateDocEvent{
      DocResolution: &did.DocResolution{
       Context: didDocument(successDID).Context,
       DIDDocument: didDocument(successDID),
       DocumentMetadata: &did.DocumentMetadata{},
      },
     },
    },
   },
  },
 },
 "ResolveDID": {
  "Empty DocId": {
   in: &fabricVDR.StreamingRawBytes{
    ResolveDID: &fabricVDR.ResolveDID{
     DocId: "",
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     Error: status.New(codes.InvalidArgument, "DocId should not be empty.").Proto(),
    },
   },
  },
  "Success": {
   in: &fabricVDR.StreamingRawBytes{
    ResolveDID: &fabricVDR.ResolveDID{
     DocId: successDID,
    },
   },
   expected: expectation{
    out: &fabricVDR.StreamingRawBytes{
     ResolveDIDEvent: &fabricVDR.ResolveDIDEvent{
      DocResolution: &did.DocResolution{DIDDocument: &did.Doc{ID: successDID}},
     },
    },
   },
  },
 },
}

var priorityList = []string{"Subscribe", "CreateDoc", "ResolveDID"}
var scenarioList = map[string][]string{
 "Subscribe": []string{"ClientID Empty", "Success", "ClientID Already Exists"},
 "CreateDoc": []string{"Empty DocId", "Success"},
 "ResolveDID": []string{"Empty DocId", "Success"},
}

for _, test := range priorityList {
 showTest := toUpper(test)
 fmt.Printf("=== RUN   Test" + showTest + "\n")
 testGlobal = test
 scenarios := tests[test]
 for _, scenario := range scenarioList[test] {
  showScenario := strings.Join(strings.Split(toUpper(scenario), " "), "_")
  fmt.Printf("=== RUN   Test" + showTest + "/" + showScenario + "\n")
 }
```

```
  fmt.Printf("--- PASS: Test" + showTest + "\n")
  for _, scenario := range scenarioList[test] {
   scenarioGlobal = scenario
   schannel <- scenarios[scenario].in
   select {
    case ok := <-done:
     if ok {
      continue;
     }
   }
  }
 }
 if failGlobal {fmt.Printf("FAIL\n")} else {fmt.Printf("PASS\n")}
}
```

Appendix D

Note: The code presented in this Appendix is a result of the research project ERATOSTHENES and intellectual property of INLECOM INNOVATION Non-Profit Organization, which should be specifically referenced in any use of this appendix.

This Appendix is dedicated to demonstrate the codes that were used for the creation, packaging and deployment of the VDR gRPC Client package, as well as the integration with Hyperledger Aries Framework GO.

Project Filesystem Structure

```
.
├── creator.go #👉 Implements the Create(did *did.Doc, opts ...DIDMethodOption) function
├── go.mod #👉 Dependency Requirements
├── go.sum
├── grpcFabricVdrAPI
│   ├── grpcFabricVdrAPI_grpc.pb.go #👉 Grpc Methods Protobuf, same as here
│   ├── grpcFabricVdrAPI.pb.go #👉 Messages Protobuf (Omitted due to GOB)
│   └── grpcFabricVdrAPI.proto #👉 Actual Proto File, as denoted here, in Protofile
├── packages
│   ├── codec
│   │   └── codec.go #👉 GOB Encoder/Decoder, as denoted here, in Custom Codec (codec.go)
│   ├── cryptoutil
│   │   ├── legacy_utils.go
│   │   ├── utils.go
│   │   └── utils_test.go
│   └── request
│       └── fabricVdrStruct.go #👉 Custom Messages, same as here
├── resolver.go #👉 Implements the Read(did string, opts ...DIDMethodOption) function
└── vdr.go #👉 Implements among others, the New() function which contains the gRPC VDR
Client source code

10 directories, 16 files
```

Dependency Requirements

go.mod

```
module fabricvdr

go 1.17

require (
    github.com/btcsuite/btcutil v1.0.3-0.20201208143702-a53e38424cce
    github.com/davecgh/go-spew v1.1.1
    github.com/golang/protobuf v1.5.2
    github.com/hyperledger/aries-framework-go v0.1.8
```

```
    github.com/hyperledger/fabric-sdk-go v1.0.0
    github.com/stretchr/testify v1.8.0
    github.com/teserakt-io/golang-ed25519 v0.0.0-20210104091850-3888c087a4c8
    golang.org/x/crypto v0.0.0-20220622213112-05595931fe9d
    google.golang.org/genproto v0.0.0-20220713161829-9c7dac0a6568
    google.golang.org/grpc v1.48.0
)

require (
    github.com/Knetic/govaluate v3.0.0+incompatible // indirect
    github.com/beorn7/perks v1.0.1 // indirect
    github.com/btcsuite/btcd v0.22.0-beta // indirect
    github.com/cloudflare/cfssl v1.4.1 // indirect
    github.com/fsnotify/fsnotify v1.4.7 // indirect
    github.com/go-kit/kit v0.8.0 // indirect
    github.com/go-logfmt/logfmt v0.4.0 // indirect
    github.com/golang/mock v1.4.4 // indirect
    github.com/google/certificate-transparency-go v1.0.21 // indirect
    github.com/google/uuid v1.1.2 // indirect
    github.com/hashicorp/hcl v1.0.0 // indirect
    github.com/hyperledger/aries-framework-go/spi v0.0.0-20220322085443-50e8f9bd208b //
indirect
    github.com/hyperledger/fabric-config v0.0.5 // indirect
    github.com/hyperledger/fabric-lib-go v1.0.0 // indirect
    github.com/hyperledger/fabric-protos-go v0.0.0-20200707132912-fee30f3ccd23 // indirect
    github.com/kilic/bls12-381 v0.1.1-0.20210503002446-7b7597926c69 // indirect
    github.com/kr/logfmt v0.0.0-20140226030751-b84e30acd515 // indirect
    github.com/magiconair/properties v1.8.1 // indirect
    github.com/matttproud/golang_protobuf_extensions v1.0.1 // indirect
    github.com/mitchellh/mapstructure v1.3.2 // indirect
    github.com/mr-tron/base58 v1.1.3 // indirect
    github.com/multiformats/go-base32 v0.0.3 // indirect
    github.com/multiformats/go-multibase v0.0.1 // indirect
    github.com/pelletier/go-toml v1.8.0 // indirect
    github.com/piprate/json-gold v0.4.1-0.20210813112359-33b90c4ca86c // indirect
    github.com/pkg/errors v0.9.1 // indirect
    github.com/pmezard/go-difflib v1.0.0 // indirect
    github.com/pquerna/cachecontrol v0.0.0-20180517163645-1555304b9b35 // indirect
    github.com/prometheus/client_golang v1.1.0 // indirect
    github.com/prometheus/client_model v0.0.0-20190812154241-14fe0d1b01d4 // indirect
    github.com/prometheus/common v0.6.0 // indirect
    github.com/prometheus/procfs v0.0.3 // indirect
    github.com/spf13/afero v1.3.1 // indirect
    github.com/spf13/cast v1.3.1 // indirect
    github.com/spf13/jwalterweatherman v1.1.0 // indirect
    github.com/spf13/pflag v1.0.5 // indirect
    github.com/spf13/viper v1.1.1 // indirect
    github.com/square/go-jose/v3 v3.0.0-20200630053402-0a67ce9b0693 // indirect
    github.com/weppos/publicsuffix-go v0.5.0 // indirect
    github.com/xeipuuv/gojsonpointer v0.0.0-20190905194746-02993c407bfb // indirect
    github.com/xeipuuv/gojsonreference v0.0.0-20180127040603-bd5ef7bd5415 // indirect
    github.com/xeipuuv/gojsonschema v1.2.0 // indirect
    github.com/zmap/zcrypto v0.0.0-20190729165852-9051775e6a2e // indirect
    github.com/zmap/zlint v0.0.0-20190806154020-fd021b4cfbeb // indirect
    golang.org/x/net v0.0.0-20211112202133-69e39bad7dc2 // indirect
    golang.org/x/sys v0.0.0-20210630005230-0f9fa26af87c // indirect
    golang.org/x/text v0.3.6 // indirect
    google.golang.org/protobuf v1.28.0 // indirect
    gopkg.in/yaml.v2 v2.3.0 // indirect
    gopkg.in/yaml.v3 v3.0.1 // indirect
)
```

VDR gRPC Client

vdr.go

```
/*
Copyright SecureKey Technologies Inc. All Rights Reserved.
SPDX-License-Identifier: Apache-2.0
*/
```

```
package fabricvdr

import (
 "path/filepath"
 "crypto/x509"
 "crypto/tls"
 "io/ioutil"
 "context"
 "runtime"
 "strconv"
 "errors"
 "bytes"
 "fmt"
 "log"
 "io"
 "os"
 "sync"
 "time"
 // Spew
 "github.com/davecgh/go-spew/spew"

 // GRPC
 "google.golang.org/grpc"
 "google.golang.org/grpc/credentials"
 "google.golang.org/grpc/keepalive"

 // GRPC Bidirectional functions
 grpcFabricVDR "fabricvdr/grpcFabricVdrAPI"

 // Custom functions and structs
 gobCodec "fabricvdr/packages/codec"
 fabricVDR "fabricvdr/packages/request"

 // Aries Framework GO
 diddoc "github.com/hyperledger/aries-framework-go/pkg/doc/did"
 vdrapi "github.com/hyperledger/aries-framework-go/pkg/framework/aries/api/vdr"
)

var (
 // Keepalive ClientParameters
 kacp = keepalive.ClientParameters{
  Time: 10 * time.Second, // Send pings every 10 seconds if there is no activity
  Timeout: time.Second,  // Wait 1 sec for ping ack before considering the connection dead
  PermitWithoutStream: true, // Send pings even without active streams
 }

 Reset  = "\033[0m"
 Red    = "\033[31m"
 Green  = "\033[32m"

 // Logger
 buf bytes.Buffer //Initialize a buffer
 // INFO
 info = func(req, msg string) {
  logger := log.New(&buf, "[" + Green + "INFO" + Reset + "] [" + req + "] --> ", log.Ldate
| log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Green + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }
 // ERROR
 warn = func(req, msg string) {
  logger := log.New(&buf, "[" + Red + "WARN" + Reset + "] [" + req + "] --> ", log.Ldate |
log.Ltime | log.Lshortfile | log.Lmsgprefix)
  logger.Output(2, Red + msg + Reset)
  fmt.Print(&buf)
  buf.Reset()
 }

 // Declare Receiver Channel and Sender Channel
 maxProcs = runtime.NumCPU() // Find the number of cpus the system has.
 schannel = make(chan *fabricVDR.StreamingRawBytes, maxProcs)
 creator = make(map[string](chan *diddoc.DocResolution))
```

```
 resolver = make(map[string](chan *diddoc.DocResolution))
)

const (
 // DIDMethod did method.
 DIDMethod = "fabricvdr"
 // EncryptionKey encryption key.
 EncryptionKey = "encryptionKey"
 // KeyType option to create a new kms key for DIDDocs with empty VerificationMethod.
 KeyType = "keyType"
)

// Function that loads an environmental variable
func getenvStr(key string) (string, error) {
 v := os.Getenv(key)
 if v == "" {
  return v, fmt.Errorf("Environment variable is empty")
 }
 return v, nil
}

// Function that loads an environmental variable and convert it to Boolean
func getenvBool(key string) (bool, error) {
 s, err := getenvStr(key)
 if err != nil {
  return false, err
 }
 v, err := strconv.ParseBool(s)
 if err != nil {
  return false, err
 }
 return v, nil
}

// Goroutine that receives messages from the bidirectional channel
func receiver(wg *sync.WaitGroup, stream
grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesClient) {
 defer wg.Done()

 waitc := make(chan struct{})
 //Reading Data from the server
 go func() {
  for {
   if stream == nil {
    // Stream Failed. Try again TODO
    warn("receiver", "Stream Failed")
    os.Exit(1)
    close(waitc)
    return
   }

   // Receiving Data from the server
   in := new(fabricVDR.StreamingRawBytes)
   err := stream.RecvMsg(in)

   if err == io.EOF {
    // Reading is done.
    close(waitc)
    return
   }

   if err != nil && err != io.EOF {
    warn("receiver", "Failed to Receive Message : " + err.Error())
    stream = nil
    time.Sleep(5 * time.Second)
    //Retry on failure
    continue
   }

   switch {
   case in.SubscribeEvent != nil:
    subscriptionEvent := in.SubscribeEvent.Event
    info("subscriptionEvent", subscriptionEvent)
```

```
    case in.CreateDocEvent != nil:
     createDocEvent := in.CreateDocEvent.DocResolution
     info("createDocEvent", "A CreateDoc Event arrived.")
     creator[createDocEvent.DIDDocument.ID] <- createDocEvent
    case in.ResolveDIDEvent != nil:
     resolveDIDEvent := in.ResolveDIDEvent.DocResolution
     info("resolveDIDEvent", "A ResolveDID Event arrived.")
     resolver[resolveDIDEvent.DIDDocument.ID] <- resolveDIDEvent
    case in.Error != nil:
     err := in.Error
     spew.Dump(err)
    default:
     warn("receiver", "Received Unknown GRPC Response...")
    }
   }
  }()
  <-waitc
}

// Goroutine that sends messages to the bidirectional channel
func sender(wg *sync.WaitGroup, stream grpcFabricVDR.GrpcFabricVdrEndpoints_ServicesClient,
sch <-chan *fabricVDR.StreamingRawBytes) {
 defer wg.Done()
 for {
  select {
   case x, ok := <-sch:
    if ok {
     data := x
     if err := stream.SendMsg(data); err != nil {
      warn("sender", "Failed to send Request: " + err.Error())
     }
    } else {
     info("sender", "Closing Stream!")
     if err := stream.CloseSend(); err != nil {
      warn("sender", "Failed to close stream: " + err.Error())
     }
     wg.Done()
     return
    }
   }
  }
}

// Function that loads client TLS Certificates
func loadClientTLSCredentials(certFile, keyFile, caFile string, serverHostOverride string)
(credentials.TransportCredentials, error) {
 // Load certificate of the CA who signed client's certificate
 pemClientCA, err := ioutil.ReadFile(caFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to read CA certificate")
 }

 certPool := x509.NewCertPool()
 if !certPool.AppendCertsFromPEM(pemClientCA) {
  return nil, fmt.Errorf("Failed to add CA certificate to the pool")
 }

 // Load server's certificate and private key
 clientCert, err := tls.LoadX509KeyPair(certFile, keyFile)
 if err != nil {
  return nil, fmt.Errorf("Failed to load X509 Key Pair.")
 }

 // Create the credentials and return it
 config := &tls.Config{
  ServerName:   serverHostOverride,
  Certificates: []tls.Certificate{clientCert},
  RootCAs:      certPool,
 }
 return credentials.NewTLS(config), nil
}

//Path finds the absolute path of the current directory and joins the remaining path.
```

```
func path(rel string) (string, error) {
 currentPath, err := os.Getwd()
 if err != nil {
  return "", fmt.Errorf("Failed to retrieve current directory: %v", err)
 }

 if filepath.IsAbs(rel) {
  isExists, err := exists(rel)
  if err != nil {
   return "", err
  }
  if !isExists {
   return "", fmt.Errorf("File does not exists.")
  }
  return rel, nil
 }

 formNewPath := filepath.Join(currentPath, rel)
 isExists, err := exists(formNewPath)
 if err != nil {
  return "", err
 }
 if !isExists {
  return "", fmt.Errorf("File does not exists.")
 }
 return formNewPath, nil
}

// Function that determines whether a path exists or not
func exists(name string) (bool, error) {
 _, err := os.Stat(name)
 if err == nil {
  return true, nil
 }
 if errors.Is(err, os.ErrNotExist) {
  return false, nil
 }
 return false, err
}

// VDR implements did:key method support.
type VDR struct{}

// New returns a new instance of VDR that works with did:fabric method.
func New(vdr *VDR){
 var err error

 serverAddress, err := getenvStr("GRPC_SERVER_ADDRESS")
 if err != nil {
  warn("new", "GRPC_SERVER_ADDRESS EnvVar not found : " + err.Error())
 }

 serverPort, err := getenvStr("GRPC_SERVER_PORT")
 if err != nil {
  warn("new", "GRPC_SERVER_PORT EnvVar not found : " + err.Error())
 }

 var serverAddr string
 serverAddr = fmt.Sprintf("%s:%s", serverAddress, serverPort)

 serverHostOverride, err := getenvStr("GRPC_SERVER_HOST_OVERRIDE")
 if err != nil {
  warn("new", "GRPC_SERVER_HOST_OVERRIDE EnvVar not found : " + err.Error())
 }

 TLS, err := getenvBool("GRPC_TLS_ENABLED")
 if err != nil {
  warn("new", "GRPC_TLS_ENABLED EnvVar not found : " + err.Error())
 }

 certFile, err := getenvStr("GRPC_TLS_CERT_FILE")
 if err != nil {
  warn("new", "GRPC_TLS_CERT_FILE EnvVar not found : " + err.Error())
```

```
}

keyFile, err := getenvStr("GRPC_TLS_KEY_FILE")
if err != nil {
 warn("new", "GRPC_TLS_KEY_FILE EnvVar not found : " + err.Error())
}

caFile, err := getenvStr("GRPC_TLS_CA_FILE")
if err != nil {
 warn("new", "CAGRPC_TLS_CA_FILE EnvVar not found : " + err.Error())
}

clientId, err := getenvStr("GRPC_CLIENT_ID")
if err != nil {
 warn("new", "GRPC_CLIENT_ID EnvVar not found : " + err.Error())
}

if !gobCodec.Register() {
 warn("new", "GOB Register map[string]interface")
}

info("new", "Number of CPUs: " + strconv.Itoa(maxProcs))
runtime.GOMAXPROCS(maxProcs)

if clientId == "" {
 warn("new", "Client ID must be declared before the GRPC Client starts.")
 os.Exit(1)
}

var opts []grpc.DialOption
opts = []grpc.DialOption{
 grpc.WithKeepaliveParams(kacp), // Keepalive Client Parameters
 grpc.WithInitialConnWindowSize(256 * 1024),
 grpc.WithCodec(&gobCodec.Codec{}), // Custom Codec
}

var GRPC string

if TLS {
 if caFile == "" {
  caFile, err = path("CA/CAcert.pem")
  if err != nil {
       warn("new", "CA File: " + err.Error())
       os.Exit(1)
  }
 }
 if certFile == "" {
  certFile, err = path("TLS/client/clientCert.pem")
  if err != nil {
       warn("new", "Cert File: " + err.Error())
       os.Exit(1)
  }
 }
 if keyFile == "" {
  keyFile, err = path("TLS/client/clientUnencryptedKey.pem")
  if err != nil {
       warn("new", "Key File: " + err.Error())
       os.Exit(1)
  }
 }
 creds, err := loadClientTLSCredentials(certFile, keyFile, caFile, serverHostOverride)
 if err != nil {
  warn("new", "Failed to Generate Credentials: " + err.Error())
  os.Exit(1)
 }
 opts = append(opts, grpc.WithTransportCredentials(creds))
 GRPC = "GRPCS"
} else {
 opts = append(opts, grpc.WithInsecure())
 GRPC = "GRPC"
}

info("new", "Starting " + GRPC + " client on host " + serverAddr)
```

```
 conn, err := grpc.Dial(serverAddr, opts...)
 if err != nil {
  warn("new", "Fail to Dial: " + err.Error())
  os.Exit(1)
 }
 defer conn.Close()

 client := grpcFabricVDR.NewGrpcFabricVdrEndpointsClient(conn)

 info("new", "Client " + clientId + " is connected")

 stream, err := client.Services(context.Background())
 if err != nil {
  warn("new", clientId + ".Services(_): " + err.Error())
  os.Exit(1)
 }

 var wg sync.WaitGroup
 wg.Add(2)

 // Magic happens here
 go func() {
  go receiver(&wg, stream)
  go sender(&wg, stream, schannel)
 }()

 // Create the gRPC StreamingRawBytes request
 var req *fabricVDR.StreamingRawBytes
 req = &fabricVDR.StreamingRawBytes{
  Subscribe: &fabricVDR.Subscribe{
   ClientId: clientId,
  },
 }
 schannel <- req
 wg.Wait()
}

// Accept accepts did:key method.
func (v *VDR) Accept(method string) bool {
 return method == DIDMethod
}

// Close frees resources being maintained by VDR.
func (v *VDR) Close() error {
 return nil
}

// Update did doc.
func (v *VDR) Update(didDoc *diddoc.Doc, opts ...vdrapi.DIDMethodOption) error {
 return fmt.Errorf("Not Supported")
}

// Deactivate did doc.
func (v *VDR) Deactivate(didID string, opts ...vdrapi.DIDMethodOption) error {
 return fmt.Errorf("Not Supported")
}
```

VDR gRPC Client Create

creator.go

```
/*
Copyright SecureKey Technologies Inc. All Rights Reserved.

SPDX-License-Identifier: Apache-2.0
*/

package fabricvdr
```

```
import (
 "fmt"
 "time"

 "github.com/davecgh/go-spew/spew"

 fabricVDR "fabricvdr/packages/request"

 "github.com/hyperledger/aries-framework-go/pkg/doc/did"
 vdrapi "github.com/hyperledger/aries-framework-go/pkg/framework/aries/api/vdr"
)

const (
 ed25519VerificationKey2018 = "Ed25519VerificationKey2018"
)

// Create builds a new DID Doc.
func (v *VDR) Create(didDoc *did.Doc, opts ...vdrapi.DIDMethodOption) (*did.DocResolution,
error) {
 createDIDOpts := &vdrapi.DIDMethodOpts{Values: make(map[string]interface{})}

 // Apply options
 for _, opt := range opts {
  opt(createDIDOpts)
 }

 // Parse DID
 parsedDID, err := did.Parse(didDoc.ID)
 if err != nil {
  return nil, fmt.Errorf("Fail to Parse DID")
 }

 if parsedDID.Scheme != "did" {
  return nil, fmt.Errorf("Invalid Scheme Detected")
 }

 if parsedDID.Method != DIDMethod {
  return nil, fmt.Errorf("Invalid Method Name Detected: %s", parsedDID.MethodSpecificID)
 }

 didDocument := didDoc

 var req *fabricVDR.StreamingRawBytes
 req = &fabricVDR.StreamingRawBytes{
  CreateDoc: &fabricVDR.CreateDoc{
   Doc: didDocument,
  },
 }
 schannel <- req

 // Consume *did.DocResolution Events from Blockchain
 creator[didDocument.ID] = make(chan *did.DocResolution, maxProcs)
 var docResolution *did.DocResolution
 select {
  case docResolution = <-creator[didDocument.ID]:
   spew.Dump(docResolution)
   delete(creator, didDocument.ID)
   return docResolution, nil
  case <-time.After(time.Second * 20):
   return nil, fmt.Errorf("No Doc Resolution Event sent")
 }
 return nil, nil
}
```

VDR gRPC Client Read

resolver.go

```
package fabricvdr
```

```
import (
 "fmt"
 "time"

 fabricVDR "fabricvdr/packages/request"
 "github.com/davecgh/go-spew/spew"

 diddoc "github.com/hyperledger/aries-framework-go/pkg/doc/did"
 vdrapi "github.com/hyperledger/aries-framework-go/pkg/framework/aries/api/vdr"
)

const (
 schemaV1 = "https://w3id.org/did/v1"
 keyType  = "Ed25519VerificationKey2018"
)

func (v *VDR) Read(did string, opts ...vdrapi.DIDMethodOption) (*diddoc.DocResolution,
error) {

 // Parse DID
 parsedDID, err := diddoc.Parse(did)
 if err != nil {
  return nil, fmt.Errorf("Error while parsing DID, failed in resolver: (%w)", err)
 }

 if parsedDID.Method != DIDMethod {
  return nil, fmt.Errorf("Invalid Method Name Detected: %s", parsedDID.MethodSpecificID)
 }

 resOpts := &vdrapi.DIDMethodOpts{}

 for _, opt := range opts {
  opt(resOpts)
 }

 var req *fabricVDR.StreamingRawBytes
 req = &fabricVDR.StreamingRawBytes{
  ResolveDID: &fabricVDR.ResolveDID{
   DocId: did, //We sent the whole did for now
  },
 }
 schannel <- req

 // Consume *did.DocResolution Events from Blockchain
 resolver[did] = make(chan *diddoc.DocResolution, maxProcs)
 var docResolution *diddoc.DocResolution
 select {
  case docResolution = <-resolver[did]:
   spew.Dump(docResolution)
   delete(resolver, did)
   return docResolution, nil
  case <-time.After(time.Second * 20):
   return nil, fmt.Errorf("No Doc Resolution Event sent")
 }
 return nil, nil
}
```

Cryptoutil
utils.go
This file is required only during building. It is not used by the VDR gRPC Client

```
/*
Copyright SecureKey Technologies Inc. All Rights Reserved.

SPDX-License-Identifier: Apache-2.0
```

```
*/

package cryptoutil

import (
    "crypto/ed25519"
    "encoding/binary"
    "errors"
    "fmt"

    "github.com/teserakt-io/golang-ed25519/extra25519"
    chacha "golang.org/x/crypto/chacha20poly1305"
    "golang.org/x/crypto/curve25519"
)

// DeriveECDHX25519 does X25519 ECDH using fromPrivKey and toPubKey.
func DeriveECDHX25519(fromPrivKey, toPubKey *[chacha.KeySize]byte) ([]byte, error) {
    if fromPrivKey == nil || toPubKey == nil {
        return nil, errors.New("deriveECDHX25519: invalid key")
    }

    // do ScalarMult of the sender's private key with the recipient key to get a derived Z
point (ECDH)
    z, err := curve25519.X25519(fromPrivKey[:], toPubKey[:])
    if err != nil {
        return nil, fmt.Errorf("deriveECDHX25519: %w", err)
    }

    return z, nil
}

// LengthPrefix array with a bigEndian uint32 value of array's length.
func LengthPrefix(array []byte) []byte {
    const prefixLen = 4

    arrInfo := make([]byte, prefixLen+len(array))
    binary.BigEndian.PutUint32(arrInfo, uint32(len(array)))
    copy(arrInfo[prefixLen:], array)

    return arrInfo
}

// Curve25519KeySize number of bytes in a Curve25519 public or private key.
const Curve25519KeySize = 32

// NonceSize size of a nonce used by Box encryption (Xchacha20Poly1305).
const NonceSize = 24

// PublicEd25519toCurve25519 takes an Ed25519 public key and provides the corresponding
Curve25519 public key
// This function wraps PublicKeyToCurve25519 from Adam Langley's ed25519 repo:
https://github.com/agl/ed25519 now
// moved to https://github.com/teserakt-io/golang-ed25519
func PublicEd25519toCurve25519(pub []byte) ([]byte, error) {
    if len(pub) == 0 {
        return nil, errors.New("public key is nil")
    }

    if len(pub) != ed25519.PublicKeySize {
        return nil, fmt.Errorf("%d-byte key size is invalid", len(pub))
    }

    pkOut := new([Curve25519KeySize]byte)
```

```
    pKIn := new([Curve25519KeySize]byte)
    copy(pKIn[:], pub)

    success := extra25519.PublicKeyToCurve25519(pkOut, pKIn)
    if !success {
        return nil, errors.New("error converting public key")
    }

    return pkOut[:], nil
}

// SecretEd25519toCurve25519 converts a secret key from Ed25519 to curve25519 format
// This function wraps PrivateKeyToCurve25519 from Adam Langley's ed25519 repo:
https://github.com/agl/ed25519 now
// moved to https://github.com/teserakt-io/golang-ed25519
func SecretEd25519toCurve25519(priv []byte) ([]byte, error) {
    if len(priv) == 0 {
        return nil, errors.New("private key is nil")
    }

    sKIn := new([ed25519.PrivateKeySize]byte)
    copy(sKIn[:], priv)

    sKOut := new([Curve25519KeySize]byte)
    extra25519.PrivateKeyToCurve25519(sKOut, sKIn)

    return sKOut[:], nil
}
```

Cryptoutil Legacy

legacy_utils.go

This file is required only during building. It is not used by the VDR gRPC Client

```
/*
Copyright SecureKey Technologies Inc. All Rights Reserved.

SPDX-License-Identifier: Apache-2.0
*/

package cryptoutil

import "golang.org/x/crypto/blake2b"

// Nonce makes a nonce using blake2b, to match the format expected by libsodium.
func Nonce(pub1, pub2 []byte) (*[NonceSize]byte, error) {
    var nonce [NonceSize]byte
    // generate an equivalent nonce to libsodium's (see link above)
    nonceWriter, err := blake2b.New(NonceSize, nil)
    if err != nil {
        return nil, err
    }

    _, err = nonceWriter.Write(pub1)
    if err != nil {
        return nil, err
    }

    _, err = nonceWriter.Write(pub2)
```

```
    if err != nil {
        return nil, err
    }

    nonceOut := nonceWriter.Sum(nil)
    copy(nonce[:], nonceOut)

    return &nonce, nil
}
```