Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes

Aniket Jalinder Yewale
aniketyewale29@gmail.com

STUDY OF BLOCKCHAIN-AS-A-SERVICE SYSTEMS WITH A CASE STUDY OF HYPERLEDGER FABRIC IMPLEMENTATION ON KUBERNETES

By

Aniket Yewale

Bachelor of Engineering - Computer Engineering
Savitribai Phule Pune University - India
2015

A thesis submitted in partial
fulfillment of the requirements for
the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
July 2018

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

July 20, 2018

This thesis prepared by

Aniket Yewale

entitled

Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Yoohwan Kim, Ph.D.
Examination Committee Chair

Laxmi Gewali, Ph.D.
Examination Committee Member

Wolfgang Bein, Ph.D.
Examination Committee Member

Sean Mulvenon, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

ABSTRACT

STUDY OF BLOCKCHAIN-AS-A-SERVICE SYSTEMS WITH A CASE STUDY OF HYPERLEDGER FABRIC IMPLEMENTATION ON KUBERNETES

By

Aniket Yewale

Dr. Yoohwan Kim, Examination Committee Chair

Associate Professor, Department of Computer Science

University of Nevada, Las Vegas

Blockchain is a shared, immutable, decentralized ledger to record the transaction history. Blockchain technology has changed the world, changed the way we do the business. It has transformed the commerce across every industry, which may be supply chain, IoT, financial services, banking, healthcare, agriculture and many more. It had introduced a new way of transactional applications that bring trust, security, transparency and accountability.

To develop any blockchain use case, the main task is to develop an environment for creating and deploying the application. In our case, we created an environment on IBM Cloud Kubernetes service using Kubernetes, a container orchestration tool and implemented Hyperledger Fabric network to create and deploy blockchain applications.

Implementing Hyperledger Fabric business blockchain network on IBM Cloud Kubernetes service provides several advantages. We can have multiple users work on the same setup. Moreover, this setup can be used and reused for many different blockchain applications as well as for deploying chaincodes and smart contracts. Fabric components can accomplish high availability by deploying on Kubernetes. We can execute several isolated Fabric instances on our Kubernetes platform as it supports multi-tenancy. This makes it easier for us to develop and test the blockchain applications. Hyperledger Fabric and Kubernetes, both if used together delivers a powerful and secure platform for processing blockchain transactions.

ACKNOWLEDGEMENTS

I would like to thank and express my gratitude to my research advisor Dr. Yoohwan Kim, for providing all the guidance, support and motivation throughout my graduate studies at University of Nevada, Las Vegas. He continuously encouraged me and gave me confidence to move in the right direction of this research.

I would like to extend my thanks to Dr. Laxmi Gewali, Dr. Wolfgang Bein and Dr. Sean Mulvenon for being part of my thesis committee and reviewing my thesis. I am grateful for all the support from Dr. Ajoy K Datta, who was always available for me and helped me whenever I needed his guidance.

I am grateful to Dr. Sean Mulvenon, Dean of Office of Research and Sponsored Projects from the College of Education at UNLV, where I worked as a graduate assistant. I would also like to thank ORSP team for their continuous support on this thesis.

I am thankful to Rizwan Patel, Senior Director at Caesars Entertainment where I worked as an intern under him in the Innovation team. I am grateful to him and the Innovation team for their continuous support and help on my thesis.

My deep sense of gratitude to my parents Jalinder Yewale and Shindu Yewale, my sister Minal who are my moral strength and always motivated me to learn. I would like to thank my relatives and friends in India for their constant support and backing me throughout my master’s program.

Finally, I would like to thank all my friends, seniors and juniors, UNLV alumnus who helped and supported me and made a memorable time at UNLV.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BLOCKCHAIN AND RELATED TERMINOLOGIES

1.1 Introduction to Blockchain

A blockchain is decentralized, digitized, distributed database of records or public ledger of all cryptocurrency transactions which have been executed and shared among the members who have participated[1]. It is an open database that keeps distributed ledger, which is basically deployed within peer-to-peer network. It consists of a list of records which are continuously increasing, called as blocks that includes transactions, recorded and added to it in sequential order[2]. So, the market participants can easily keep eye on the digital currency transactions. Each node which is connected to the network gets a copy of the blockchain that is automatically downloaded. These blocks are protected by cryptographic hashes and a consensus mechanism from tampering[2]. Once the information is entered in the blockchain, it cannot be erased. The blockchain consists verifiable record of every single transaction which has done until now[1]. It is a distributed database which removes the necessity of third party verification and doesn't require a central authority.

Blockchain is said to be the technology that will have same kind of impact as the Internet did with people's lives[3]. A blockchain is a decentralized, secure, shared, distributed, ledger enabling participants to exchange value, eliminating intermediaries. Let us see one by one;

- Decentralized: The transactions are verified by each node of a network rather than the central entity. This falls with the pattern of centralized consensus.

- Secure: Every transaction that's initiated and recorded in the Blockchain is digitally signed, guaranteed to be authentic and impervious to man-in-the-middle attack (fraud).
- Shared: All data written on the Blockchain is shared across all the participating parties, and herein, lies one of the key values of the Blockchain, the more parties that partake in a Blockchain the more the number of shared copies and less the likelihood of fraud.
- Distributed: Here, not only the copies are shared, but they are also distributed across the various entities, reducing the likelihood of data loss and increasing resiliency.
- Ledger – Finally, every transaction is recorded in a write once, read multiple paradigm – an immutable record of every transaction that's ever occurred, an out of the box, audit trail.

Blockchain consists a set of blocks, where each block includes hash of the preceding block, generating chain of blocks from genesis to current block. The first block in a blockchain is called as genesis block[4]. Every time a block is completed, a new block is generated. There are infinite number of blocks in the blockchain which are like links in the chain connected to each other in linear sequential order. Blocks have a set of transactions, where transaction is transfer of values among the entities that are broadcasted to network and are gathered in the blocks. These transactions are seen in blockchain and are mined by solo miners or pool miners into a block. These pool miners follow mining approach where miners contribute to block generation. The transaction records are added in the blockchain by the pool miners. This process is called as mining[4]. Figure 1 shows difference in traditional and blockchain transactions.

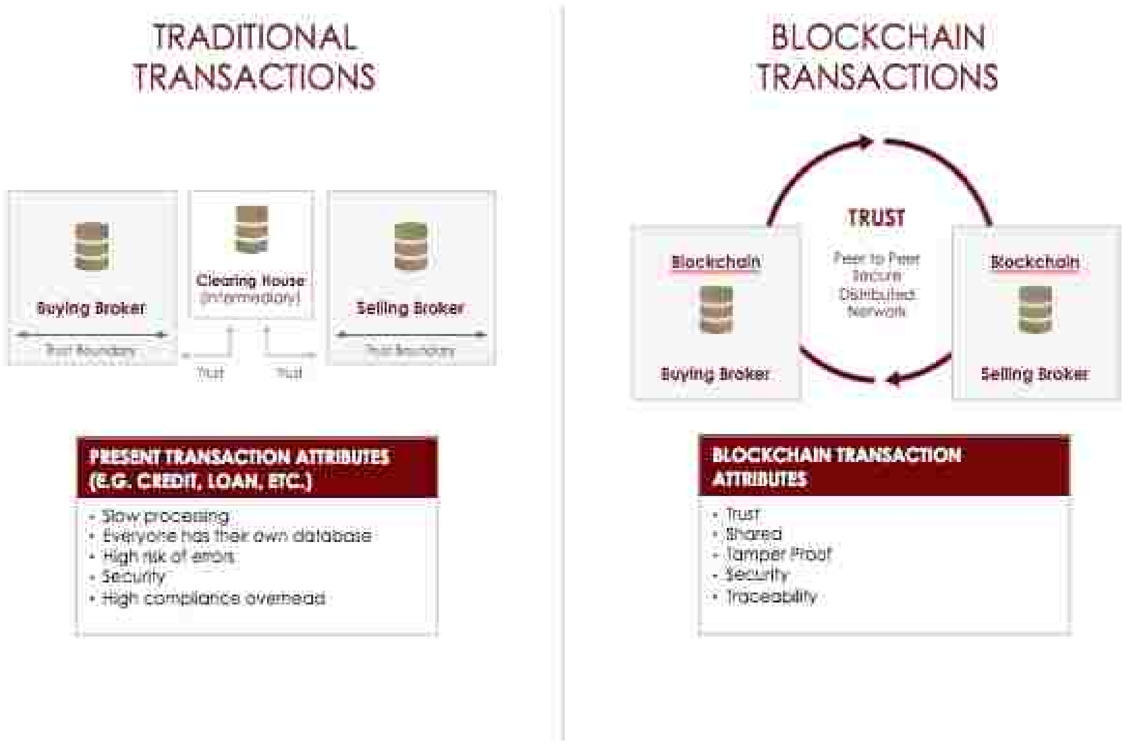


Figure 1 : Blockchain Transactions

Individual blocks should include a Proof of Work (PoW) to be legitimate in the blockchain. The other miners every time upon receiving a block, verifies PoW. The main aim of mining is to enable the nodes to achieve secure consensus in the system. When the miners validate a block, they are gifted with certain amount of transaction fee and resolved amount of newly generated coins[4].

As shown in the following figure 2, blockchain follows link list data structure. The blockchain in fact is represented as a singly linked list. Alike linked list, each block has a hash of previous block - which can be thought of as a pointer to previous block. In blockchain, every transaction in block is stored in a Merkle Tree. But the blockchain itself is not a tree.

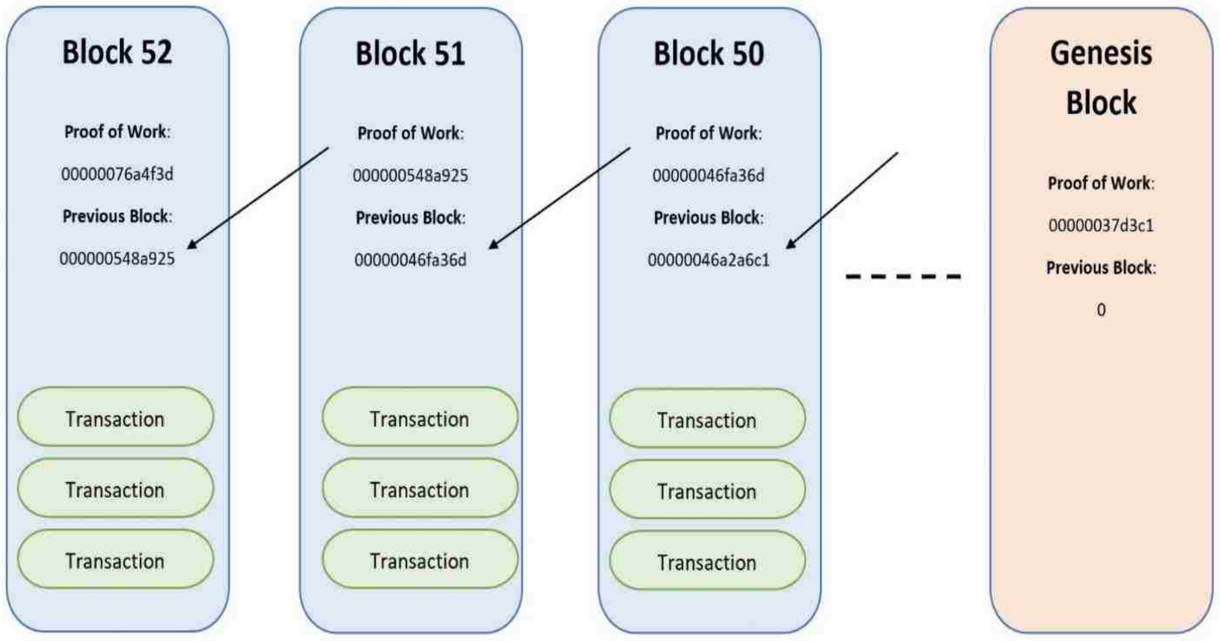


Figure 2 : Blockchain Structure

One of the major difference between a blockchain and a linked list is that you can't remove or add a block in the middle of the list/chain. The blockchain has total information about different user addresses as well as their balances straight away from the genesis block to the previously completed block. The blockchain was designed for the transactions unable to be deleted; i.e. immutable.

1.2 Key Concepts of Blockchain

We will explore individually on distributed ledger, smart contracts and consensus.

1.2.1 A Distributed Ledger

A distributed ledger which is also known as shared ledger, or distributed ledger technology (DLT) is a consensus of a shared, duplicated and synchronized digital data which is geographically broadcasted over number of websites, countries, or institutions. Distributed ledger is responsible for recording each transaction that occur on network. Distributed ledger is heart of blockchain network. A blockchain ledger is duplicated among multiple network participants, where each one of them contribute in its maintenance and hence called as decentralized.

Here, all the records are maintained in the blockchain network and duplicated among all the participants. So, it is called as distributed ledger. All nodes have copy and there is no central location where records are stored. There is no such central authority or regulatory that updates or maintains the blockchain. The ledger is maintained by all participating nodes using a consensus principle[7].

1.2.2 Smart Contracts

Smart Contract is computer program that handles the transfer of digital currencies between parties under some conditions. A smart contract is a set of computer instruction i.e. code that all the participants must agree. Smart contracts give ability to track and run the complex agreements between the parties without any human interference[4]. A smart contract is essentially business logic running on a blockchain[5].

Transaction chains may constitute self-executing service agreements, which are known as smart contracts[6]. Smart contracts are self-executing contracts consisting of agreement terms written in lines of code between buyer and seller. The agreements and corresponding code included exist across the blockchain network. According to blockchain context, smart contracts are nothing, but scripts present on blockchain. Since they are present on the chain, they have a specific unique address. A smart contract runs automatically and independently on each node in the network, depending upon the data present in triggering transaction[1].

Smart contracts are key process to encapsulate and secure information and keep this information simple across the network, Smart contracts can also be written to enable participants to run certain features of transactions automatically. Consider an example where smart contract may be used to specify an item's shipping cost that keeps on changing based upon its arrival. Depending on the agreed terms by parties as written to the ledger, when the item is received, the funds change automatically[7].

The blocks in a blockchain are written only on consensus, and one way to gain the agreement is through smart contract. Whenever a block is written into the blockchain, the code validates the data against the agreement, and on successful validation of data it is converted into a block and written into the blockchain. After this, all the participants in the network update their blockchain. The outcome of running the smart contract is usually writing of the block in the blockchain[7].

1.2.3 Consensus

Consensus is nothing but a general agreement. It is the process where ledger transactions are kept synchronized and updated along the network to allow that ledgers modify them only when transactions are validated by the correct participants; and when ledgers do the modification, they update and modify with same transactions and that too in the same order is called consensus[7].

Consensus is basic problem in distributed systems. It needs two or more agents to mutually agree on a given value. Now, some agents can be untrusted, unreliable. Hence, the consensus process should be dependent. Blockchains may use different consensus algorithms or mechanisms like Proof of Work (PoW), Proof of Stake (PoS), etc[4].

Proof of stake (PoS) is an alternative mechanism to PoW. PoS is built on the concept that only those can take part in the consensus process who have assets in the system. While PoW method makes miners to continuously execute hashing algorithms to approve the transactions, PoS asks users to show their possession of some amount of currency; meaning their stake in the currency[4].

1.3 Overview of Thesis

To manage and deploy the Hyperledger Fabric system, many people face problems as Hyperledger Fabric has lot of complexity in configuration. To make the Hyperledger Fabric's operation easier, we require some tools to control and handle distributed system of Hyperledger Fabric.

We have various ways to setup Hyperledger Fabric environment. Firstly, we can setup Hyperledger Fabric network locally. But there are lot of complexities and challenges involved in setting up Hyperledger Fabric locally. It is not simple to setup Hyperledger Fabric environment locally as compared to the other ways we have. Secondly, we can use IBM Blockchain Platform which is hosted on IBM Cloud for production purpose where we have starter and enterprise membership plan, which are bit costly. Lastly, we can setup Hyperledger Fabric network with the help of Kubernetes APIs on IBM Cloud Kubernetes Service.

1.4 Proposed Solution

Setting up Hyperledger Fabric network using Kubernetes APIs on IBM Cloud Kubernetes Service is easier way to create Hyperledger Fabric network. It provides us with a free cluster that has 2

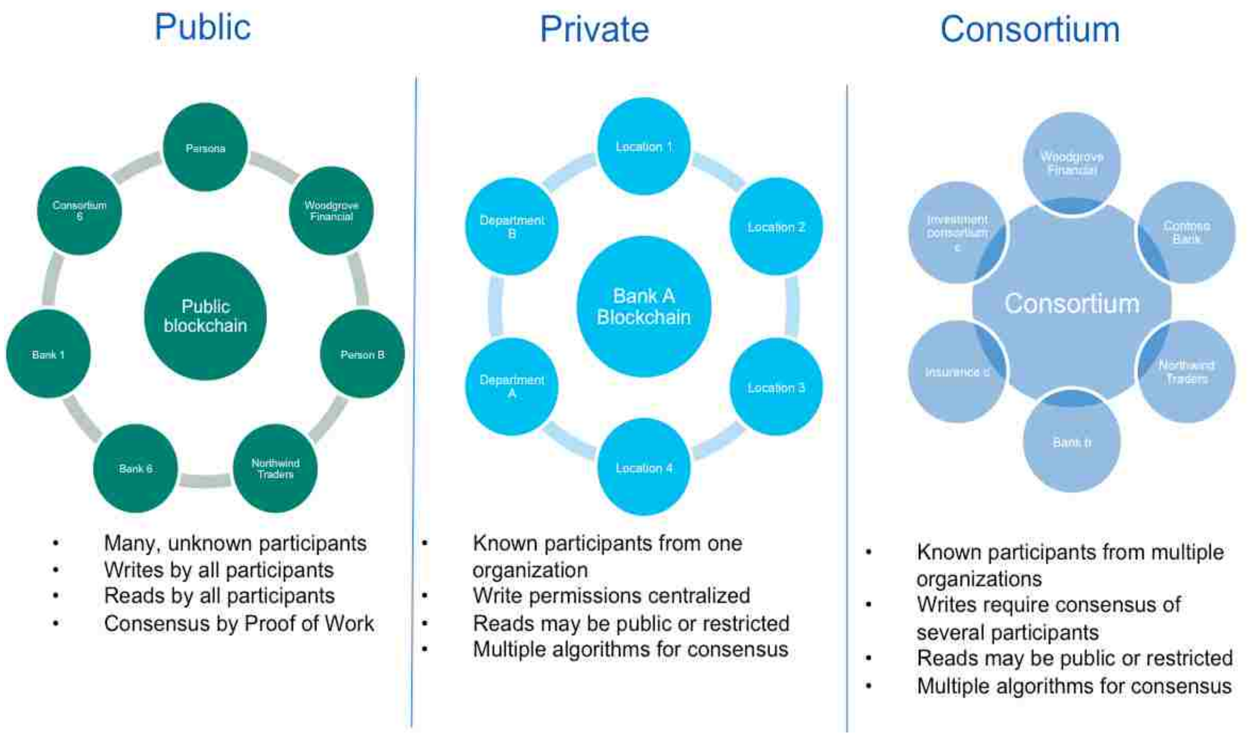
CPUs, 4 GB memory, and a worker node. Kubernetes is ideal for several reasons, because by deploying the network on Kubernetes, fabric components can achieve high availability. We can execute several isolated Fabric instances on our Kubernetes platform. This makes easier for us to develop and test the blockchain applications. So, this method is better, and we can setup our Hyperledger Fabric network in a simple way on Kubernetes.

Here, Hyperledger Fabric's bits are developed into container images. Fabric's chaincode also forces container to execute in the sandbox. Fabric system contains components executing in several containers. Moreover, Kubernetes has become one of the dominant platform for automating scaling, deployment, including management of containerized applications. So, both Hyperledger Fabric and Kubernetes naturally fit and go hand in hand with each other.

CHAPTER 2

BLOCKCHAIN TYPES AND PROTOCOLS

Blockchains can be deployed in different models – public, private and consortium. These are similar to the public, private, hybrid cloud deployment models. Private institutions like banks became aware that they can use the basic concept of blockchain as a DLT and generate a permissioned blockchain (private or federated), which includes validator, who is the member of consortium[9]. Figure 3 shows difference in public, private and consortium blockchains.



- Many, unknown participants
- Writes by all participants
- Reads by all participants
- Consensus by Proof of Work

- Known participants from one organization
- Write permissions centralized
- Reads may be public or restricted
- Multiple algorithms for consensus

- Known participants from multiple organizations
- Writes require consensus of several participants
- Reads may be public or restricted
- Multiple algorithms for consensus

Figure 3 : Public, Private & Consortium Blockchain

2.1 Blockchain Types

2.1.1 Public Blockchain

In public blockchain, anyone may read, send transactions and can take part in consensus process. These blockchains are considered as fully decentralized. Public blockchains are protected from cryptoeconomics – fusion of economic incentives and cryptographic verification using mechanisms like PoW(Bitcoin) or PoS (Ethereum)[8]. They are open source and not permissioned.

In public blockchain, the ledger or transaction record is being shared between the nodes in distributed peer-to-peer network and that results in a transaction record that is immutable[6]. Here, anyone without permission can participate. Moreover, anyone may be able to download the code and start executing public node on their local device and can send transactions via the network. These transactions are transparent, but anonymous[9].

Examples: Bitcoin, Ethereum, Litecoin, Dash, Monero, etc.[9].

2.1.2 Private Blockchain

Private blockchain is blockchain in which write permissions are given to one organization whereas read permissions can be restricted or public to some participants[9]. It is similar to private cloud, Here, the execution times (transaction times) are much faster on a private blockchain as compared to public blockchain. Private blockchains takes edge over blockchain technology by establishing participants and groups who could internally validate the transactions. Whenever it is about scalability and state compliance about regulatory issues and data privacy rules, private blockchains have their own use case. They come with some security advantages as well as disadvantages[9].

In private blockchain network, participants are whitelisted and are limited to stringent contractual obligations in order to work properly. So, this is the reason where an efficient consensus protocol like Practical Byzantine Fault Tolerance (PBFT) can be used[10].

Examples: Hyperledger Fabric, MONAX, Multichain

However, there are some similarities between public and private blockchain. Both public and private blockchains have decentralized peer-to-peer network. Every participant has a replica of shared append-only ledger with digitally signed transactions. Both has the copies in synchronization via a protocol which is called as consensus. Although some participants are faulty or malicious, both public & private blockchains offers some promise on ledger's immutability[8].

2.1.3 Consortium Blockchain

These are also called as federated blockchains. It operates under leadership of a group. In this model, we have multiple participants across organizations engaged in reading and writing on the blockchain. Akin to the private blockchain, given the participants are known, the consensus algorithms can be more relaxed as compared to the ones that govern the public blockchain.

Consortium blockchains offers at most transaction privacy and are faster. They have higher scalability. They are used widely in banking sector. The pre-selected set of nodes manages the consensus process. Consider an example with a consortium of fifteen financial institutions, where every institution handles a node. Out of that ten institutions should sign each block for the block to be viable. The read blockchain, the rights could be restricted or public to the participants[9].

Example: Corda, R3 (Banks), B3i (Insurance), EWF (Energy)[9].

2.2 Permissioned vs Permissionless blockchain

Let’s compare Permissioned blockchain and Permissionless blockchain:

Table 1: Comparison of Permissioned blockchain and Permissionless blockchain

Characteristics	Permissioned blockchain	Permissionless blockchain
Maintenance	One or small group of pre-selected & permissioned entities	Anyone who wants to
Storage	Central Servers	Massively distributed
Censorship Resistance	No	Yes
Need to use token	No	Yes
Production of underlying data	Permissioned group of people (e.g. Customers of a bank)	Anyone who wants to
Underlying mining model	Don’t use computing power-based mining. Uses consensus algorithms like RAFT or Paxos	Bitcoin uses Proof-of-Work (PoW). Ethereum is using Casper Implementation Proof-of-Stake (PoS) to reach consensus.
Need to trust central entities to secure it	Yes	No
Database Access	A small group pf pre-selected & permissioned entites	Anyone who wants to
Transaction Costs	Less	More
Speed	Less	More
Examples	Hyperledger, R3 Corda	Bitcoin, Ethereum

2.3 Major Blockchain Protocols

2.3.1 Bitcoin

Bitcoins are nothing but a virtual currency which is also called as cryptocurrency. Bitcoin allows users to perform non-reversible transactions. Bitcoin allows transactions without presence of any third party. Bitcoin helps to reduces credit cost in minor transactions. Moreover, it also reduces transaction fees and prevents double-spending.

Bitcoin includes technologies like public-key cryptography, digital signature, hash, P2P and Proof of Work and Proof of Work which allows users to make bitcoins. This has created a mechanism that stops data falsification and payment duplication. Along with this, it has also blocks malicious users as it’s critical for operating system such as for electronic money which doesn’t have a central authority[11].

2.3.2 Ethereum

Ethereum is an open-source, public and blockchain oriented protocol which includes functionality of smart contracts. These contracts are used to keep registries of debts, markets and to transfer funds. This protocol has decentralized virtual machine known as Ethereum Virtual Machine (EVM). By making use of global network of public nodes and token called as ether (also known as gas), EVM executes Turning-complete scripts. This gas is used to prevent spam on the networks. It is also used to allocate the resources in proper amount to the incentive given by the request. Ethereum has its native cryptocurrency called Ether & Ethereum Wallet that can carry ERC-20 assets. Ethereum enables developers to build and create decentralized applications. So, it is a protocol for smart contracts, decentralized applications and decentralized independent organizations along with bunch of functioning applications created on it[11].

2.3.3 Ripple Consensus Network

Ripple platform has been designed on open-source distributed consensus ledger, and local currency referred as XRP. It is developed for digital asset exchanges, banks, payment providers, and corporate who want to send money globally. Ripple allows quick, safe, secure and free financial transactions around the globe without chargeback. Ripple platform supports tokens which are used to represent cryptocurrency, fiat currency, commodity or other value units such as mobile minutes, frequent flier miles etc.[11].

2.3.4 Hyperledger

Hyperledger is an open source blockchain platform which started by Linux Foundation in 2015. It was started to support and assist the blockchain-based distributed ledgers. This protocol concentrates on the ledgers developed in supporting international, technological & supply chain businesses, catering leading, financial and business transactions. The primary objective here is to improve performance and reliability aspects. This project focuses on collaborative effort to bring people together from various industries & business to develop and advance the blockchain technology. This is done by providing a modular framework which supports various components for different use which contains a range of blockchain consisting of consensus models and their own storage, contracts and identity and services for access control[11].

Hyperledger includes of leaders from various fields like IoT, banking, finance, supply chain, manufacturing and technology. Hyperledger already has number of projects under it.

Some of the features of Hyperledger platform include transactions endorsement policies, Python support, highly confidential channel for private information sharing[11].

2.3.5 R3's Corda

Corda which is developed by Company R3 is distributed ledger open source protocol which is used to supervise, record, and sync the financial agreements with the controlled financial institutions. It was created by and for world's financial institutions but applicable in multiple industries. It is beneficial to the blockchain technology, but with no design choices it makes many protocols not suited for the banking scenarios. Corda eliminates the unnecessary global sharing of data. Corda's design came up because of big analysis and prototyping with team members[11].

In Corda, consensus is accomplished at individual deals and not at the system level. It directly allows regulatory and supervisory observer nodes. Corda doesn't have any of native cryptocurrency[11].

Corda is unified system which provides certain limits for transaction propagation to risen up privacy and performance. Corda has got rich smart-contract language. Transactions are sent only to appropriate parties which further are signed off by distributed notary service which improves privacy, as all transactions are not broadcasted to all participants, as well as performance, as all participants don't want to see all transactions[12].

2.3.6 Symbiont Distributed ledger

Symbiont distributed ledger protocol begun as SDK for Assembly. Assembly is permitted distributed ledger part of Symbiont's smart contracts system. Assembly is the first distributed ledger relevant for institutional finance. It is Byzantine fault-tolerant distributed ledger, which is high performing and highly secure. In a local multi-node network, it can process sustained eighty thousand transactions per second. Additionally, it allows cost-saving and sharing of business logic and market data[11].

Let’s compare Ethereum, Hyperledger Fabric and R3 Corda protocols:

Table 2: Comparison of blockchain protocols

Characteristics	**Ethereum**	**Hyperledger Fabric**	**R3 Corda**
Governance	Ethereum developers	Linux	R3
Platform description	Generic blockchain platform	Modular blockchain platform	Specialized distributed ledger platform for financial industry
Programming Language	Solidity	Go, Java	Kotlin, Java
Smart Contracts	Not legally bounded	Not legally bounded	Legally bounded
Consensus Algorithm	Proof-of-Work (PoW). Casper Implementation Proof-of-Stake (PoS)	Practical Byzantine Fault Tolerance (PBFT)	Notary nodes can run several consensus algorithms
Mode of Operation	Permissionless, public or private	Permissioned, private	Permissioned, private
Currency	Ether. Tokens via smart contract	None. Tokens via chaincode	None
Privacy	Existing privacy issue	Not prevalent	Not prevalent
Scalability	Existing privacy issue	Not prevalent	Not prevalent

2.4 Hashgraph

Hashgraph is a new consensus; an alternative to the blockchain. Hashgraph uses gossip protocol. Using gossip protocol, the nodes efficiently and rapidly exchange data with other nodes in the community. This then automatically builds a hashgraph data structure using "gossip about gossip" protocol. This data structure is cryptographically secure. It consists of the history of communication in a community. Using this as an input, nodes execute virtual-voting consensus algorithm as other nodes. The community reaches consensus on the order and timestamp without any further communication over the internet. Each event is digitally signed by its creator.

Every node in Hashgraph sends signed information known as events to its randomly chosen neighbors upon new transactions and the transactions received from other nodes. These neighbors will combine the received events with the information received from other nodes into a new event. Then this information is sent onto other randomly chosen neighbors. This process continues until all of the nodes are aware of the information created or received at the start. Each part of new information can reach each node in the network speedily due to the rapid convergence property of the gossip protocol. This gossip protocol is based on direct acyclic graph.

Every node maintains a graph representing sequences of forwarders for each transaction. All the nodes have the same view of all transactions and their witnesses. Then by making virtual voting, every node can find if a transaction is valid depending upon whether it has over two-thirds of nodes in the network as witnesses. The hashgraph data structure is based on the Byzantine setting, where the assumption is that less than a third of nodes are Byzantine[13].

The hashgraph data structure and consensus algorithm provides a new platform for distributed consensus. The motto of a distributed consensus algorithm is to enable a community of users to agree on the order in which certain are generated transactions when no single member is trusted by all. Hence, it is a system for creating and building trust when individuals do not rely and trust each other. Hashgraph accomplishes this in a fundamentally new way[13]

A blockchain is like a tree that is continuously prunes as it grows. Now this pruning is necessary to keep the branches from growing out of control. In hashgraph, rather than pruning new growth, it is woven back into the body. In blockchain & hashgraph, any of the member can generate a transaction, which ultimately will be put in a container i.e. block and will be propagated all over the community. In blockchain, those blocks are meant to be a single long chain. Consider, if two blocks are generated by two miners at the same time, the community will finally choose to go with one, and get rid of the other one. In hashgraph, every container is used and considered without discarding them. All the branches will continue to survive forever and ultimately grow back together into a single whole which is more efficient[13].

Additionally, if new containers arrive early then blockchain fails, because new branches sprout faster compared to what they can be pruned. Hence, blockchain needs PoW or some other mechanism to artificially retard the development. In hashgraph, nothing is discarded. There is no harm in the structure developing rapidly. Each member can generate transactions and containers when they need. Ultimately, as the hashgraph doesn't need pruning and therefore is simpler, it enables mathematical guarantees which are stronger, like Byzantine agreement and fairness. Distributed databases like Paxos are Byzantine, but not fair. Blockchain is neither Byzantine nor fair. Hashgraph is both Byzantine and fair[13].

The hashgraph algorithm achieves to be fast, secure, fair, reliable, trusted, Byzantine, ACID compliant, efficient, cost-effective, timestamped, and DoS resistant[13].

CHAPTER 3

BLOCKCHAIN-AS-A-SERVICE

Blockchain-as-a-Service (BaaS) is a service which enables customers to use cloud-based solutions to develop, build, host and use their own blockchain apps, smart contracts and functions on the blockchain. To maintain the infrastructure agile, seamless and operational, the cloud-based service provider will manage and control all the required tasks and activities. It's working is pretty similar and based on the concept of Software-as-a-Service (SaaS) model. It is a development which is interesting in blockchain ecosystem which is indirectly helping the blockchain adoption in the businesses.

The BaaS model is widely used that enables the business personals in their respective business ventures to take the profit of Blockchain technology without any sort of investment in the development. It is simple for them to make blockchain based applications and smart contracts using BaaS model. If you pay for BaaS, you pay a company to set up blockchain connected nodes on your behalf. A BaaS provider would deal with the confusing back end for you or your business.

This makes blockchain a which offers a software just like Gmail, where instead of setting up your own infrastructure and e-mail servers, you logon and can access your emails. In the same way, BaaS lets you to consume blockchain as a service without the time to deployment and ongoing management like that in traditional deployment.

3.1 Pros & Cons of Blockchain-as-a-Service (BaaS)

Let's see the advantages of BaaS Service Model:

- **Cost**: It has low-cost access to the technology. Moreover, as BaaS is consumed as a service, there is no big upfront cost as it has self-owned kit. Also, the costs can be predicted as costs will basically depend upon monthly billing model which can be projected.
- **Easy set-up**: BaaS takes away the burden of the initial setup and ongoing infrastructure maintenance.
- **Scalability & Compatibility**: It offers companies huge scalability along with good compatibility. It can be accessed from anywhere.
- **Elastic**: The cloud based hosting and monthly OPEX costing model enables the projects to be quickly scaled up or down to meet the needs.
- **Accuracy of deployments**: Since most of the BaaS deployments are template-driven, it allows them to deploy precisely and accurately in repeatable fashion.
- **Risk**: The OPEX model and speed of deployment enabled the companies to rapidly test whether a blockchain is suitable for them without making a big upfront investment.
- **Resilient and secure infrastructure**: It allows cloud providers to pool the resources and have data centers with huge capabilities than the standard company. Moreover, it offers more data security.

Now, let’s see the disadvantages of BaaS Service Model:

- **Centralization**: A main thing about of blockchain is that they have decentralized infrastructure. By hosting it in a single cloud provider, it generates a degree of centralization. So, BaaS involves adding some centralization to the blockchain, which is not ideal.
- **Limited control**: As the infrastructure is managed by the service provider, there is limited flexibility in the backend infrastructure. So, the customer can just control the blockchain.
- **Lock-in**: You are putting all the trust in the cloud provider without owning or controlling the infrastructure. Additionally, the organizations can find it complex to migrate their services from one vendor to the another.
- **Restrictions and Policy**: Some of the organizations can be managed and controlled by internal policy or legal requirements that needs all data which is placed on the company property.

3.2 Blockchain-as-a-Service (BaaS) Providers

To democratize blockchain technology, several key companies have introduced their own platforms offering blockchain-as-a-Service (BaaS).

3.2.1 Amazon AWS BaaS

Amazon, the world’s largest & leading cloud provider of PaaS, IaaS and SaaS services that joined hands with Digital Currency Group (DCG) to make BaaS environment offers via AWS platform. The goal of the BaaS environment is to enable DCG affiliated blockchain providers to work in a blockchain environment which is secure with its clients to help in development. With this, it’s simpler to access, manage and control financial institutes, enterprise technology companies, and insurance firms which describes the basic need that makes it easy to enable BaaS. Insurance companies, enterprise technology, financial institutions will help developers on the project to learn what the real business requirements are for blockchain to allow them to push BaaS development.

3.2.2 Microsoft Azure BaaS

Microsoft follows an open source Ethereum Blockchain for the BaaS development. This is an interesting way to get access to blockchain technology which supports in business innovation. Microsoft Azure’s BaaS service delivers reasonable and quick access to blockchain technology. For developers and businesses, it provides a better environment to ease innovation of new processes.

By using Microsoft Azure’s networking compute and storage services present all over the globe and with a bunch of user inputs via Azure portal, clients in less time can supply blockchain network topology which is fully configured. Microsoft has automated these time-consuming pieces to enable clients to concentrate to build application and scenarios.

3.2.3 IBM’s Hyperledger BaaS system

IBM has their BaaS service which is based on the Hyperledger BaaS system. This platform is fully designed particularly to fasten development and handle operations in the network. For creating blockchain solutions, it gives environment for developers and businesses. With this platform, you can build an application, govern and manage the network. Its main goal is to digitize transaction workflow via shared ledger and to make a secure blockchain network.

The IBM Blockchain Platform is completely integrated blockchain platform developed to quicken the governing, development multi-institution business network. IBM's Enterprise Membership Plan consist of a highly available certificate authority, entry to the network's transaction ordering service, network peer executed in highly secure environment detached from another member's environment.

3.2.4 Oracle Blockchain Cloud Service

This platform is based on Hyperledger Fabric which enables users to contribute to its development and provides a clear view of the roadmap. There is no cloud vendor lock-in as the blockchain network can be extended to on premise and other clouds based on Hyperledger. Its easy to expose the function through API which permits developers to make application integration. Oracle also offers out-of-the-box connectors and sample smart contracts for other Oracle SaaS applications.

With this BaaS service, customer can create new income streams, increase business pace, make cost reductions and reduce the risk rate by providing security. To make transactions simple and secure to the trusted network, security is extended to the ERP, SaaS and supply chain applications. They offer privacy, security and reduced transaction cost. It shares real-time information with reliable and trusted network across Oracle SCM & ERP Cloud and Netsuite Suite Cloud Platform.

3.2.5 Huawei BaaS platform

The Huawei BaaS is a Hyperledger-powered BaaS which allows companies to generate smart contracts using distributed ledger network. This platform is based on Hyperledger 1.0 and Kubernetes. This platform can be used for bunch of applications consisting of digital assets, supply chain, finance and traceability, crowdfunding notarization, etc. The platform enables clients to create smart contract applications which focus on tokenized securities, supply chain, assets and public services like ID verification & financial auditing.

Huawei's blockchain service consists of three layers; cloud, pipe and devices, with a combination of software and hardware protection, homomorphic encryption and zero-knowledge proofs to have one of the most reliable and secure systems. This service is delivered through the Huawei Cloud, ensuring lower costs and always accessible to clients.

CHAPTER 4

INTRODUCTION TO HYPERLEDGER

Hyperledger supports a series of business blockchain technologies such as smart contract engines, distributed ledger frameworks, sample applications, client libraries, utility libraries, graphical interfaces, etc. The Hyperledger by the Linux Foundation is an umbrella project under which open source blockchain approaches and tools are developed collaboratively. Figure 4 represents Hyperledger modular umbrella approach.

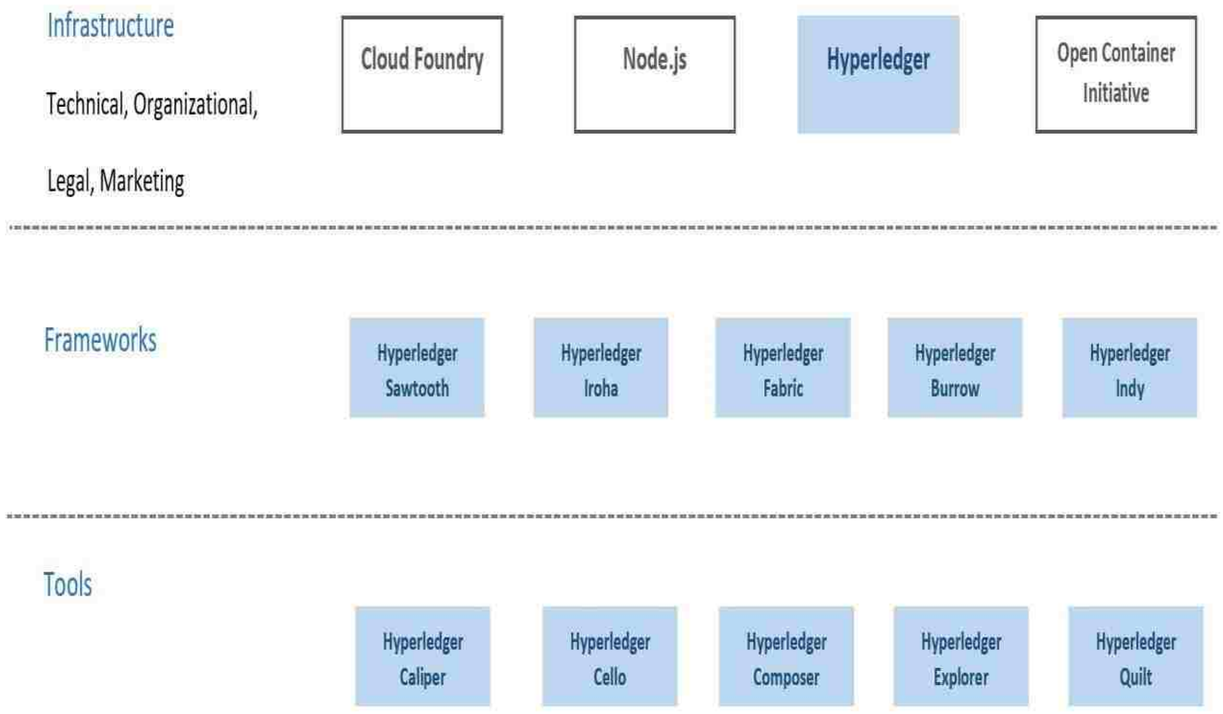


Figure 4 : Hyperledger Modular Umbrella Approach

It features innovators in IoT, banking, finance, banking, supply chains, manufacturing and technology. It is a platform to produce tangible business results where we can create open, enterprise-grade and standardized distributed ledger blockchain frameworks and code bases. There have been several improvements as certain industry partners provided a helping hand which included Microsoft's Coco platform, Enterprise Ethereum Alliance (EEA), Cisco's blockchain IoT protocol initiative[6].

4.1 Hyperledger Frameworks

Hyperledger business blockchain frameworks are utilized to develop enterprise blockchains for the purpose of consortium of organizations. They are completely different than public ledgers like Bitcoin blockchain and Ethereum. The Hyperledger frameworks includes smart contracts so that transaction requests can be processed, consensus algorithm to agree changes in the ledger, append-only distributed ledger and privacy of transactions via permissioned access.

Following are the different Hyperledger Frameworks:

4.1.1 Hyperledger Sawtooth

Hyperledger Sawtooth contributed by Intel is a modular platform to develop, build, deploy and execute distributed ledgers. It has a various consensus algorithm depending on size of network. One of them is Proof of Elapsed Time (PoET), which gives scalability and aims large distributed validator populations along with little utilization of resources. This framework is designed and developed for versatility. Moreover, it provides support for both permissioned and permissionless deployments[14].

Most important features that Hyperledger Sawtooth provides is security, scalability & greater autonomy for each participant in the Blockchain network.

4.1.2 Hyperledger Iroha

Hyperledger Iroha is a business blockchain framework from the contribution of NTT Data, Hitachi, Soramitsu, Colu developed to be simple for integration into infrastructural projects which needs DLT. This framework emphasizes mobile application development along with the client libraries for iOS and Android, different from other Hyperledger frameworks. This framework is inspired from Hyperledger Fabric framework. It provides a development environment for C++ developers contributing to Hyperledger. Hyperledger Iroha provides a new, simple domain-driven C++ design and providing consensus algorithm YAC[14].

4.1.3 Hyperledger Fabric

Hyperledger Fabric was first proposal for codebase, with fusion of the recent work done by Digital Asset Holdings, IBM's OpenBlockchain and Blockstream's libconsensus. This is one of the foundation frameworks for building blockchain applications or solution. It gives modular architecture that enables components like membership services and consensus to be plug-and-play which is the most striking feature. This framework is revolutionary in enabling entities without passing information through a central authority and by making confidential transactions. This is achieved by different channels that execute within the network. Hyperledger Fabric supports permissioned deployments.

At the core of Hyperledger Fabric, there is a container technology which is used to provide smart contracts known as chaincode that contains application logic of system. Some main features of Hyperledger Fabric: includes, pluggable Consensus (ordering service) and, membership service provider services, private channels for sharing confidential information and support for CouchDB for storing world state.

4.1.4 Hyperledger Burrow

Hyperledger Burrow is a permissionable smart contract machine which gives a modular blockchain client along with permissioned smart contract interpreter[5]. It is the only existing apache-licensed EVM implementation. It has various components serving different purpose. The Smart contract application engine facilitates integration of complex business logic. The Gateway delivers interfaces for systems integration and user interfaces. The Consensus Engine is used for maintaining the networking stack between the nodes & ordering transactions. Lastly, the Application Blockchain Interface (ABCI) is the component that gives interface specification for the smart contract application engine and consensus engine to connect[14].

4.1.5 Hyperledger Indy

Hyperledger Indy is a distributed ledger, built with the intent of decentralized identity. It goal is to accomplish this by issuing tools, artifacts, libraries, and reusable components independent of any ledger by generating and making use of autonomous digital identities present on blockchains for the purpose of interoperability across any DLT that supports them. With DLT, Hyperledger Indy puts people, not the organizations in charge of decisions about their own privacy and disclosure, which is striking feature[14].

4.2 Hyperledger Tools

The Hyperledger tools or modules are nothing but auxiliary software used to maintain and deploy the blockchains. They are also used to check, examine the data on the ledgers. Additionally, they are used to design, prototype, and extend blockchain networks.
Let’s see the different Hyperledger tools:

4.2.1 Hyperledger Caliper

Hyperledger Caliper is a blockchain benchmark tool. With a set of predefined use cases, it enables users to compute performance of a particular blockchain implementation. Hyperledger Caliper will generate reports including several performance indicators including transaction latency, TPS (transactions/second), resource utilization etc. Caliper results are utilized by different Hyperledger projects as they develop their frameworks, and for the reference in providing support for the choice of a implementing blockchain which is appropriate for user’s specific requirements.

4.2.2 Hyperledger Cello

Hyperledger Cellos a toolkit for creating Blockchain-as-a-service (Baas) platform to provision a customizable Blockchain networks in easier and faster way. It helps in both development & deployment by aiming to create/provision a customizable Blockchain network on-demand. To manage the blockchain network, it provides dashboard for checking the system status, adjusting the chain numbers, scale resources etc. If you are looking to create and manage one or more Blockchain network, Hyperledger Cello could be the tool of choice. It helps to keep track of system status, adjusting chain numbers, scale resources etc. via the dashboards. It also helps in managing the lifecycle of blockchains[14].

4.2.3 Hyperledger Composer

Hyperledger Composer provides set of tools to develop and build blockchain business networks, accelerate the growth of smart contracts and deploying them across distributed ledger. These tools allow to model business blockchain network, create REST APIs for interacting with the blockchain network and create a skeleton angular application. The advantages of Hyperledger Composer are: reduced risk, faster creation of blockchain applications & greater flexibility. Moreover, the UI-based interface makes it easier for developers and business owner to generate smart contracts.

4.2.4 Hyperledger Explorer

Hyperledger Explorer is used to invoke, view, deploy, query blocks, transactions, , chain codes, network information along with other connected information stored in the ledger. It is a web application for viewing operations on the blockchain network. It is the first blockchain explorer for permissioned ledgers, enabling them to discover distributed ledger projects generated by Hyperledger's members internally without bargaining about privacy[14].

4.2.5 Hyperledger Quilt

Hyperledger Quilt is used for making transactions across distributed and non-distributed ledgers. Hyperledger Quilt provides interoperability among ledger systems by implementing Interledger Protocol (ILP), which is a payment protocol built to transfer value across different blockchain networks. In simple words, a user registered with one blockchain network can easily and securely transfer the value to another user registered with another blockchain network[14].

4.3 Consensus in Hyperledger

Consensus is the process in which a network of nodes gives an assurance for ordering of transactions and approves block of transactions[5]. It may be implemented by several ways such as lottery-based algorithms including PoW and Proof of Elapsed Time (PoET). It can also be implemented by using voting-based methods like Paxos and RBFT i.e. Redundant Byzantine Fault Tolerance. These approaches aim various network requirements and fault tolerance models[15]. Consensus must provide the following main functionality:

- According to endorsement and consensus policies, it should confirm the correctness of every transaction in suggested block,
- Concur on correctness and order and so on running results
- Depends and interacts on smart-contract layer for validating correctness of ordered set of transactions contained in block[15][5].

There are several ways to accomplish the consensus. Figure 5 is a generalized view of Hyperledger consensus process flow. Various Hyperledger frameworks may require executing these steps in a different manner. Hyperledger business blockchain frameworks achieve consensus by carrying out two different activities; viz, ordering of transactions and approving transactions.

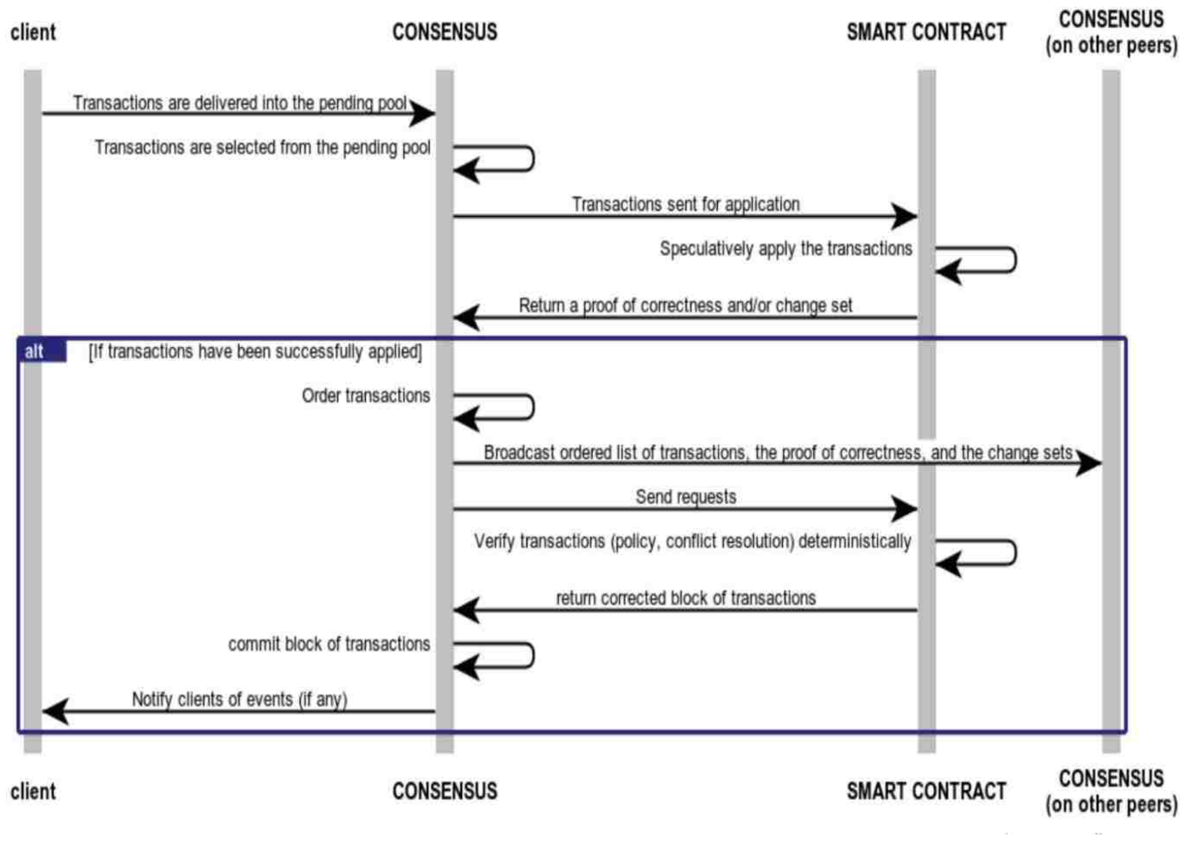

client
CONSENSUS
SMART CONTRACT
CONSENSUS (on other peers)
Transactions are delivered into the pending pool
Transactions are selected from the pending pool
Transactions sent for application
Speculatively apply the transactions
Return a proof of correctness and/or change set
alt
[If transactions have been successfully applied]
Order transactions
Broadcast ordered list of transactions, the proof of correctness, and the change sets
Send requests
Verify transactions (policy, conflict resolution) deterministically
return corrected block of transactions
commit block of transactions
Notify clients of events (if any)
client
CONSENSUS
SMART CONTRACT
CONSENSUS (on other peers)

Figure 5 : Generalized Hyperledger Consensus Process Flow

Getting transactions from client application is the very first step of consensus process flow. Consensus is dependent upon ordering service for ordering transactions which can be executed in various ways. Transactions are forwarded via interface to ordering service. Consensus is dependent on the smart contract layer to approve the transactions. This is because, it includes business logic which makes a transaction legitimate. Smart contract layer approves every transaction by making sure that they comply with the rules & policy and contract defined for transaction. Here, the invalid transactions are discarded and removed from the block. The consensus layer makes use of the communication layer to communicate with client and other peers on the network.

There are various consensus algorithms utilized around Hyperledger frameworks. RBFT in Hyperledger Indy, Sumeragi in Hyperledger Iroha, Apache Kafka in Hyperledger Fabric use a voting-based approach for consensus, whereas PoET in Hyperledger Sawtooth makes use of lottery-based approach for consensus[5].

CHAPTER 5

HYPERLEDGER FABRIC

Hyperledger Fabric is a blockchain framework implementation of a distributed ledger platform to execute the smart contracts. It is one of the Hyperledger projects which is been hosted by the Linux Foundation[16]. It was developed with the intention of foundation to develop applications with modular architecture. Hyperledger Fabric enables components, such as membership services and consensus, to be plug-and-play.

To host smart contracts, Hyperledger Fabric uses container technology known as chaincode. Chaincode has application logic of system[3]. It is one of the most stable permissioned, enterprise-ready blockchain development platform right now. For Hyperledger Fabric, all the queries are executed using RESTful APIs[3]. As the framework provides IBM Blockchain Platform, Hyperledger Fabric is supporting businesses with trust, accountability and transparency.

It is an open-source and executes smart contracts that are user-defined and along with powerful security it supports identity features [16]. The distributed ledger protocol of Hyperledger fabric is executed by the peers. The fabric has two types of peers: A validating peer and non-validating peer.

- Validating peer: It is node on the network which is responsible for executing the consensus, maintaining ledger and validating the transaction.

- Non-validating peer: It is a node which works as a proxy for interaction between the clients to the validating peers. A non-validating peer doesn’t run the transactions but has the ability to validate them[1].

The Hyperledger Fabric is a private and permissioned blockchain platform developed for business use; meaning, unlike, in permissionless or public network systems that permits participation of unknown identities in the network. The members can enroll via Membership Service Provider (MSP). Moreover, it possesses capability to generate channels with ability to enable group of participants to generate separate ledger of transactions[17].

5.1 Hyperledger Fabric Reference Architecture

The Hyperledger fabric reference architecture can serve as a guideline to build permissioned distributed ledgers. This reference architecture consists of two main components: Hyperledger Services and Hyperledger APIs, SDKs and CLI. The following figure 6 represents Hyperledger Fabric reference architecture.

Hyperledger services provides various services such as smart contract services, identity services, blockchain services and policy services. On the other hand, Hyperledger APIs, SDKs and CLIs provide an interface into blockchain services via appropriate application programming interfaces, software development kits, or command line interfaces. Additionally, an event stream which is basically a gRPC channel runs across all services. It can receive and send events. Events are either pre-defined or custom. Validating peers or chaincode can emit events to which external application can respond or listen to.

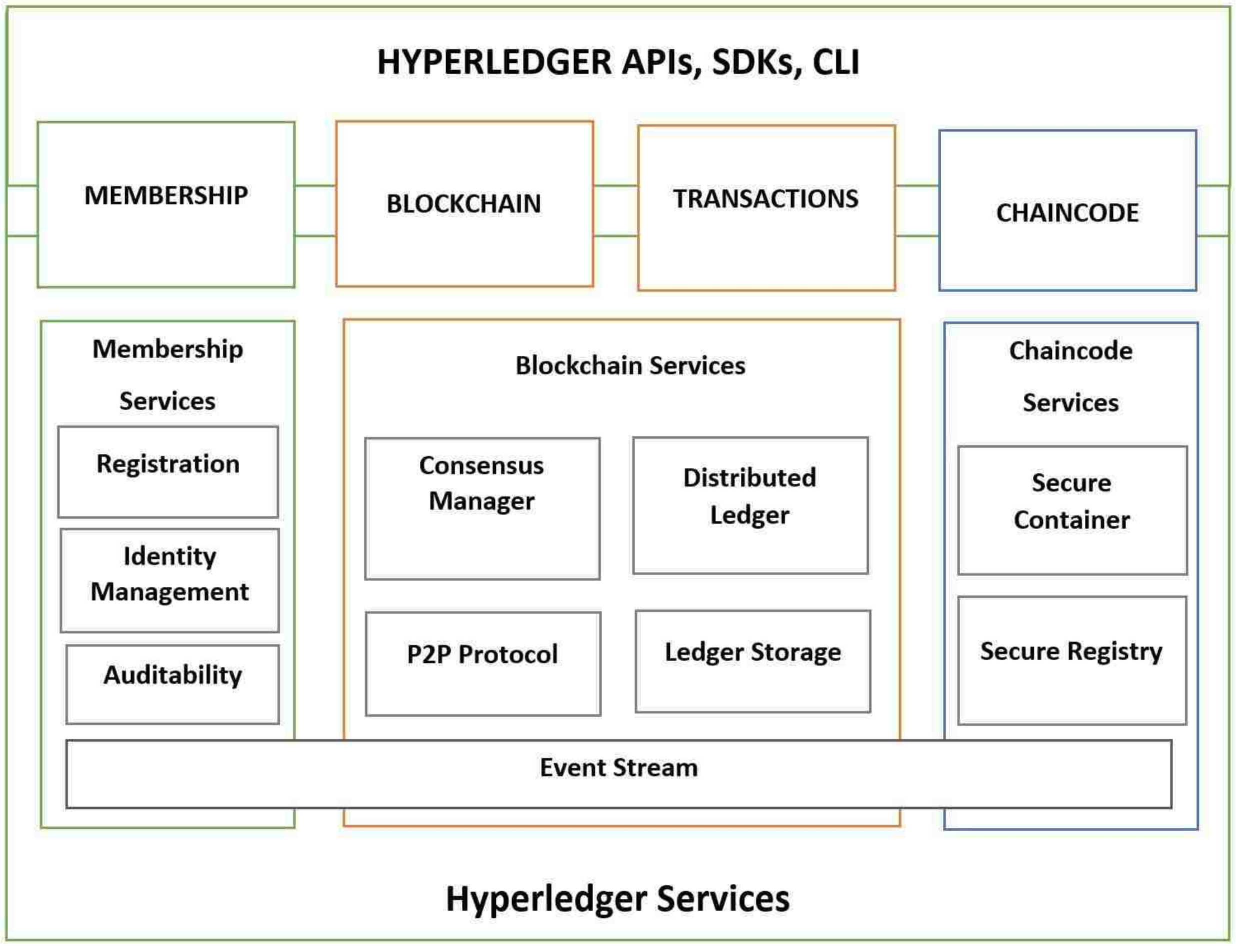


Figure 6 : Hyperledger Fabric Reference Architecture

Let us see the modules of the above architecture:

5.1.1 Membership services

This module is permissioning module which acts like a vehicle to create root of trust while creating network and plays an important role in managing the identity of members. Membership services are nothing but a certificate authority, utilized elements of the public key infrastructure (PKI) for management, key distribution and to establish federated trust as network expands. This module gives a specialized digital certificate authority for providing certificates to members which belongs to the blockchain network and controls the cryptographic functions of Hyperledger Fabric[18].

5.1.2 Transactions

A transaction is nothing but a request to the blockchain to run a function on the ledger. Chaincode implements this function. Cryptography makes sure that there is integrity of transactions by linking the transaction to previous blocks. Along with this, it also ensures transactional integrity, Every channel in Hyperledger Fabric is its own blockchain[18]. Smart contract or Chaincode services

Chaincode is piece of code which is stored on the ledger and is part of transaction. A smart contract in Hyperledger Fabric is a program, called chaincode[5]. Chaincode executes transactions that could change the world state. The program logic is written in Go or JavaScript and executed in secure Docker containers. The transaction modifies the data, scoped by chaincode on the channel from which it controls[18].

Chaincode initializes and further controls the ledger state via transactions sent by the applications. Chaincode basically handles the business logic that members in network have agreed to. The state created by a chaincode cannot be accessed by the another chaincode directly. But, with suitable permission, chaincode in the same network can allow another chaincode to access its state[5].

We can use these chaincodes to design decentralized applications and business contracts. Also, they can be used to define and manage the assets[5].

5.1.3 Events

Events are generated by approving chaincodes and peers on the network that applications can listen and take action. Events may be pre-defined events or custom events produced by chaincode. Events are used by event adapters, which may send events using vehicles like Kafka or WebHooks. Fabric-committing peers generate an event stream to publish events to the registered listeners[18].

5.1.4 Consensus

Consensus is the main thing of blockchain system. It ensures a trust system. Consensus service enables digitally signed transactions to be proposed and validated by network members. Consensus is pluggable and linked tightly to the endorse-order-validation model in Hyperledger Fabric. The consensus system is represented by ordering services in Hyperledger Fabric. The ordering service combines several transactions into blocks and produces a hash-chained sequence of blocks which includes transactions.

5.1.5 Ledger

Another module is distributed encrypted ledger which gives the capability to query and write data around the distributed ledgers. We have are two types in this, one; Level DB which supports composite key queries, keyed queries, and key range queries and the other is Couch DB which supports composite key queries, keyed queries, key range queries and additionally full data rich queries.

5.1.6 Client SDK

Client SDK allows the generation of applications that invoke and deploy the transactions on the top of a shared ledger. The Hyperledger Fabric Reference Architecture provides support to the both Node.js and Java SDK. Software developer kit is a programming kit just like a set of tools that gives developers an environment of libraries to write and test the chaincode applications.

5.2 Hyperledger Fabric Architecture with hierarchy of components

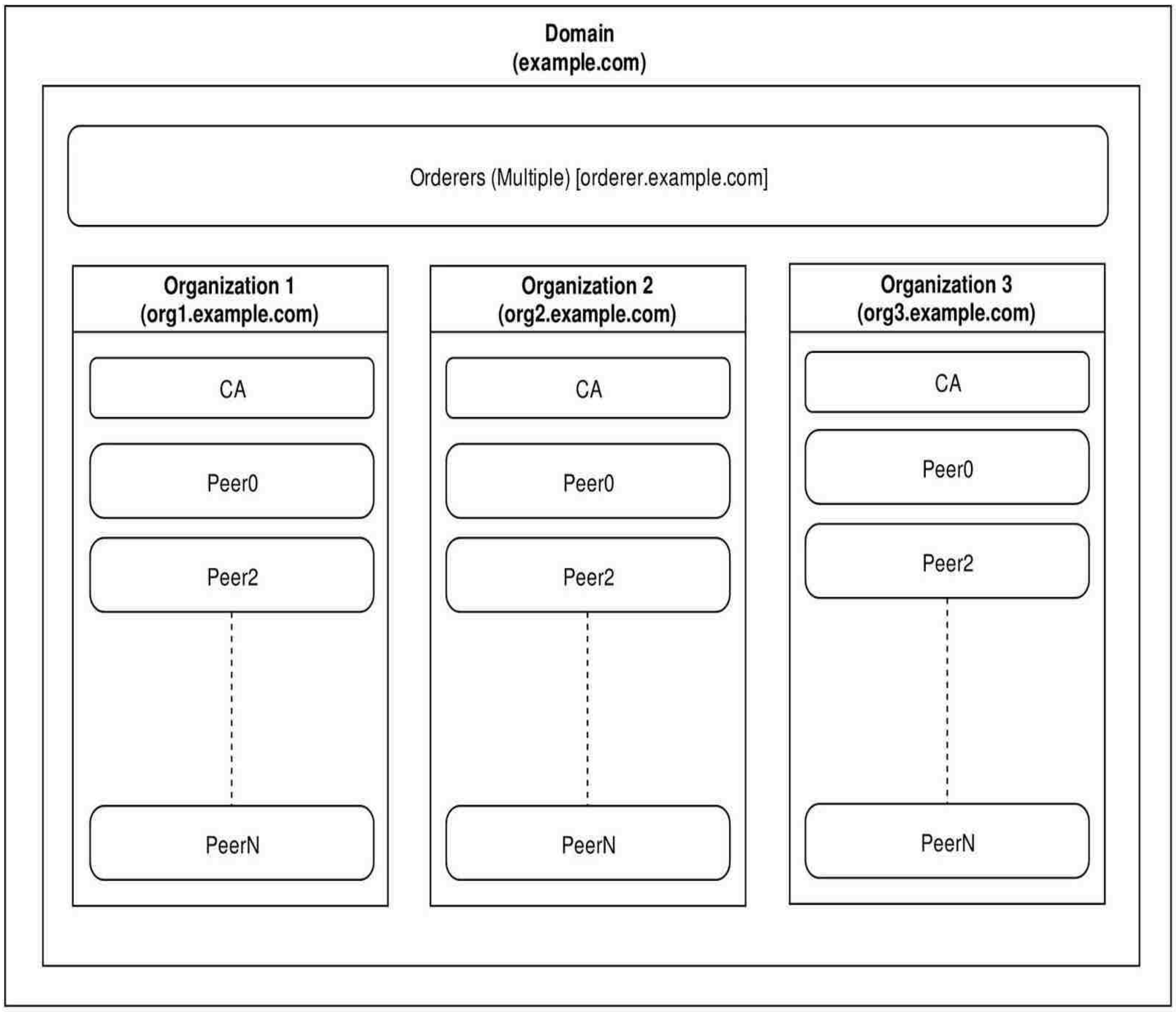


Figure 7 : Hyperledger Fabric Architecture with hierarchy of each of the components

Figure 7 represents Hyperledger Fabric architecture with hierarchy of each of the components. Let's see the Hyperledger Fabric Architecture with hierarchy of each of the components:

- Domain: Domain is top-level namespace. If we are developing a network for a supply chain, then generally the project or domain name is used as the Hyperledger Fabric's domain.
- Orderers: Orderers are just below the domain and can have multiple instances. Orderers guarantees that all peers in the network have completed a transaction. When a transaction is completed by a peer, the orderer is made aware of the new transaction and then it forwards and commits this block to all the neighbouring peers. Orderers does not depend on one organization. But to decrease the failure rates, it is good to have multiple orderers.
- Peers: Peers are nodes connected to the clients, which are responsible for committing transactions. Every peer in the database has its own replica of transactions. An organization consists of more than one peer. To avoid loss of data, though we should have multiple peers in an orderer, but having more than 3 or 4 peers might only outcome higher latency rates.
- Organizations: Organizations are nothing but containers for peers and particular certificate authorities (CA). Each organization has got its own list of peers and CA. Most probably, organizations are used to physically separate the blockchain network, where each organization using the product can join the network by setting up physical machines.

- Certificate Authorities: The certificate authority is used for generating users certificates and validating ownership in network. Each certificate authority is connected with an organization[19].

5.3 Key features of Hyperledger Fabric for enterprise blockchain

- Assets: Hyperledger fabric allows exchanging monetary value on the network[17].
- Chaincode: Partitioned from transaction ordering, limiting the needed levels of trust and validation along node types & optimizing network scalability and performance[17].
- Ledger Features: Encodes all transaction history for every channel and contains SQL-like query ability privacy through[17].
- Channels: It enables multi-lateral transactions with high confidentiality and privacy, thus promising security.
- Security & Membership Services: Participants in permissioned membership have idea that authorized regulators and auditors can detect & trace all the transactions.
- Consensus: It allows network starters to make choice of consensus mechanism that best denotes relationship existing between participants[17].

5.4 Advantages of Hyperledger Fabric architecture

- Modular Architecture: Hyperledger Fabric architecture being modular is the biggest advantage as it encourages developers to create pluggable components into its architecture. Due to its robust architecture, this kind of modularity is enabled that looks into the future of blockchain technology. This is good when someone wants to get things in into the system, for example; custom identity management system for the users to use the blockchain platform built on top of Hyperledger Fabric.

- Scalability: The endorser nodes are responsible for specific chaincode which are orthogonal to orderers, the system may behave better if the same nodes does these functions. This occurs when various chaincodes define disjoint endorsers, that involves partitioning of chaincodes between the endorsers which enables execution of parallel chaincode.
- Confidentiality: This architecture provides chaincode deployment which provides confidentiality requirements wrt content and state updates of its transactions.
- On-Demand Data Retrieval: Channels allows data partitioning. This allows us to protect the data & privacy. It is beneficial for those finacial companies who are willing to adopt blockchain, express deep concerns on their competitors who observe their data. Now-a-days, even banks and companies with good cryptography are not safe from hackers. So, with channels on Hyperledger Fabric, you can only display the data which you need to and store the sensitive data in data partitions.
- Built for Permissioned Blockchains: Hyperledger Fabric allows for all the entities to have known identities. Permissioned blockchains are the ones that the finance companies need, considering the data protection in particular. For example, in case of a mortgage company using blockchain, mortgage is not something that is publicly exposed. This calls for the parties to identify themselves in the network to verify authenticity.

- Level of Trust: In this architecture, the aim is to reduce the layers of trust & number of transaction verification, so that transactions can be faster and smoother. The wat Hyperledger Fabric handles the transaction is completely different than others.
- Community Support: The community that is building Hyperledger Fabric is super enthusiastic, aiming to be the biggest contributor to this blockchain platform With mainstream companies like IBM, Toyota and many other corporates adopting Hyperledger Fabric in production, the community and its support is rising[7][20].

5.5 Use cases of Hyperledger Fabric

5.5.1 Business-to-business (B2B) contract

This technology can be applied to automate business contracts in a trusted way.

5.5.2 Asset depository

Assets can be dematerialized on a blockchain network which can allow all the stakeholders of an asset type to have access to each asset without going through the middlemen. Currently, assets can be tracked in many ledgers, which must restore. Hyperledger Fabric replaces these multiple ledgers with a single decentralized ledger by providing transparency and removing intermediaries.

5.5.3 Loyalty

A loyalty rewards platform can be securely built on top of blockchain i.e. Hyperledger Fabric and smart contracts technology.

5.5.4 Distributed storage

Distributed storage which is required to increase trust between the parties.

5.6 Consensus in Hyperledger Fabric

As shown in figure 8, consensus in Hyperledger Fabric has three phases, viz; Endorsement, Ordering, and Validation.

• Endorsement: It is operated by policy in which participants authenticate a transaction.

• Ordering: This phase takes the authenticated transactions and agrees to the order to be completed to ledger.

• Validation: It gets block of ordered transactions and approves correctness of results, consisting of double-spending and verifying of endorsement policy.

Hyperledger Fabric supports pluggable consensus service for each of these three phases. Applications may plugin various endorsement, ordering, and validation models based on their needs. Specifically, ordering service API enables plugging in BFT-based agreement algorithms. The ordering service API has generally two operations: broadcast and deliver.

• broadcast(blob): To circulate all over the channel, the client calls this to send a random message blob. It is known as request(blob) in context of BFT when request is sent to service.

• deliver (seqno, prevhash, blob): To send message blob with defined non-negative integer sequence number (seqno) and hash of most previously sent blob (prevhash), the ordering service calls this on peer. Meaning, it is the output event from ordering service. deliver() is also known as commit() in BFT systems. And in pub-sub systems it is known as notify() [15][5].

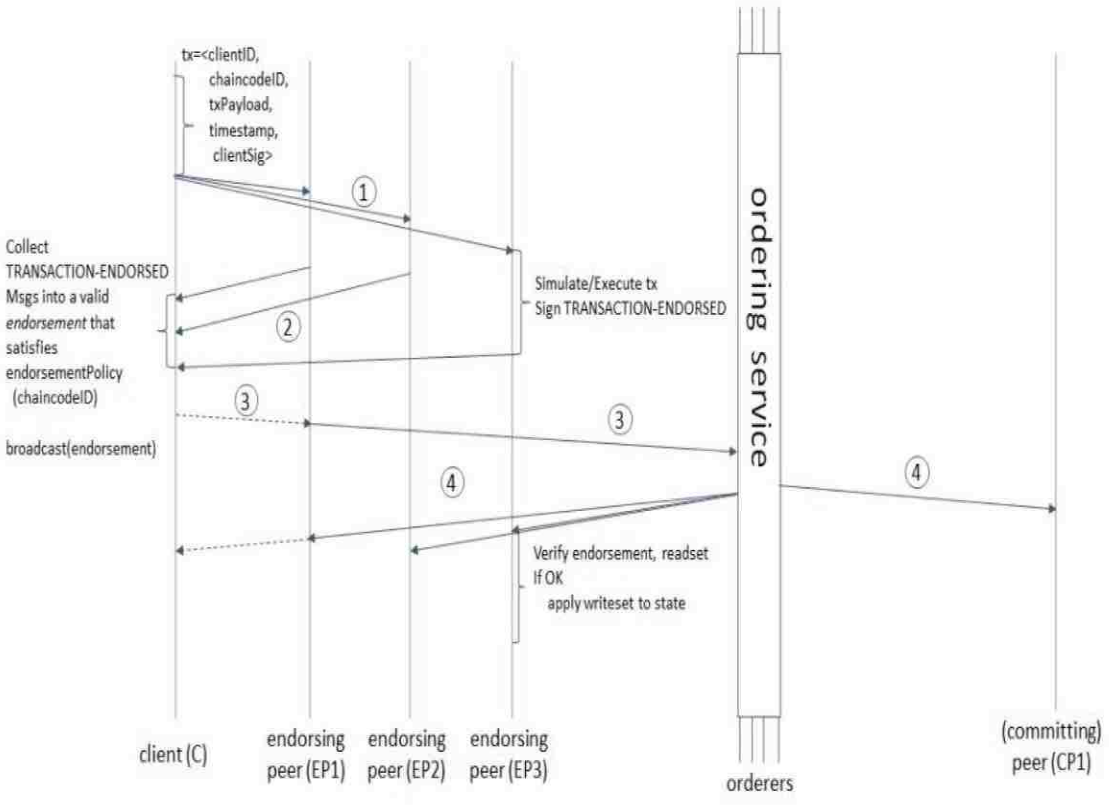


Figure 8 : Hyperledger Fabric Protocol

CHAPTER 6

KUBERNETES

Kubernetes is portable, extensible open-source container-orchestration system to provision, manage, and scale applications. It manages scheduling and workloads depending on user-defined parameters. It combines the containers that makes application into logical units for discovery and simple management. Kubernetes allows us to handle the life cycle of containerized applications in a cluster of nodes. The key pattern that Kubernetes follows is declarative model.

6.1 History of Kubernetes

Kubernetes was originally developed and designed by Google and now it is maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes is developed and build by the community, with the purpose of management requirements and addressing container scaling. When Kubernetes was in its born state, the community contributors took advantage of their knowledge of creating and executing internal tools like Borg and Omega. With the rise of CNCF, the community adopted Open Governance for Kubernetes. IBM which is a founding member of CNCF also contributes to CNCF's cloud-native projects along with Google, Microsoft, Amazon and Red Hat. Kubernetes works with a series of container tools, including Docker.

6.2 Features of Kubernetes

- Storage orchestration
- Automatic binpacking
- Self-healing

- Batch execution
- Horizontal scaling
- Service discovery & Load balancing
- Secret and configuration management
- Automated rollouts and rollbacks

6.3 Common terms in Kubernetes

- **Kubernetes Cluster**: Cluster is a group of one or several bare-metal servers or virtual machines known as nodes, which provides the resources the Kubernetes uses to execute one or multiple applications.

  In Kubernetes Engine, a container cluster contains at least one cluster master and several worker machines called nodes. A container cluster is basis of Kubernetes Engine.
- **Kubernetes Master**: Master is machine which handles Kubernetes nodes. It is the source of all task assignments.
- **Kubernetes Nodes**: Kubernetes nodes are also known as worker machines that does the requested given tasks. The Kubernetes master manages all of them.
- **Containers**: A container is a process or group of processes that are executed in isolation. Containers gives isolation like virtual machines, except given by OS & at process level. Basically, containers explicitly execute only a single process as they have no need for the standard system services.
- **Kubernetes Pods**: Pod is fundamental building block of Kubernetes. It is simplest and smallest unit in the Kubernetes object model that may be generated or deployed. A Pod

represents a process that is executing on cluster. Pods are groups of volumes and containers co-located on same host.

- **Kubernetes Jobs**: A job generates one or several pods and makes sure that the defined number of them successfully end.
- **Kubernetes Deployment:** A deployment is a Kubernetes resource where you specify your containers and other Kubernetes resources that are required to run your app, such as persistent storage, services, or annotations.
- **Kubernetes Services:** Kubernetes service groups a set of pods and provides network connection to these pods for other services in the cluster without exposing the actual private IP address of each pod.
- **Kubernetes Persistent Volumes (PV)**: Persistent Volumes are a way for users to claim durable storage such as NFS file storage.
- **Replication Controller**: Replication Controller manages pods scheduling across the cluster. It handles how many similar replicas of a pod must be executing on the cluster.
- **Service**: Service act as load balancers and ambassadors for other containers, giving them exposure to the external world.
- **Kubelet**: This service is executed on nodes and reads container manifests and makes sure that the stated containers have begun and are executing.
- **Kubectl**: Kubectl is command line configuration tool for Kubernetes.

6.4 Kubernetes Architecture

At primary is the datastore of Kubernetes which is known as etcd. The data store stores declarative model as objects. For example, if you want five instances of a container, then data store stores this request. After viewing this information change, it is sent to the controllers to take some action on it. Controllers after getting the request, react to the model and act to achieve desired state.

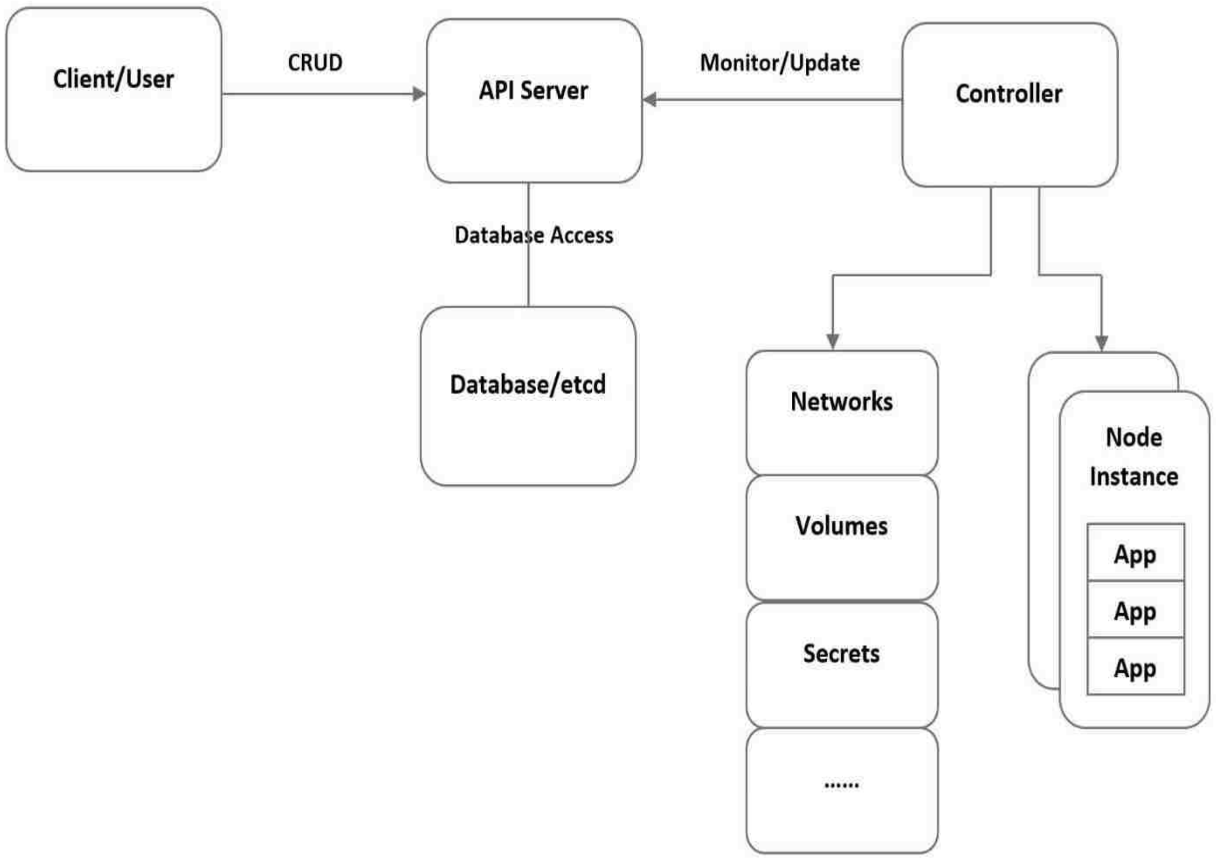


Figure 9 : Kubernetes Architecture

As shown in the figure 9, an API server is a simple HTTP server handling CRUD operation i.e. create/read/update/delete on the data store. The Controller sees the change and acts on it. Controllers play an important part in instantiating the actual resource which is represented by any of the Kubernetes resource. The application needs these actual resources to enable it to run successfully.

6.5 Kubernetes Resource Model

Kubernetes doesn't have concept of an application, whereas it has simple building blocks that are needed to compose. Kubernetes is cloud native platform where the internal resource model is the same as the end user resource model. In Kubernetes infrastructure, every resource is monitored and processed by controller. While defining an application, it contains collection of these resources read by controllers for building applications actual backing instances.

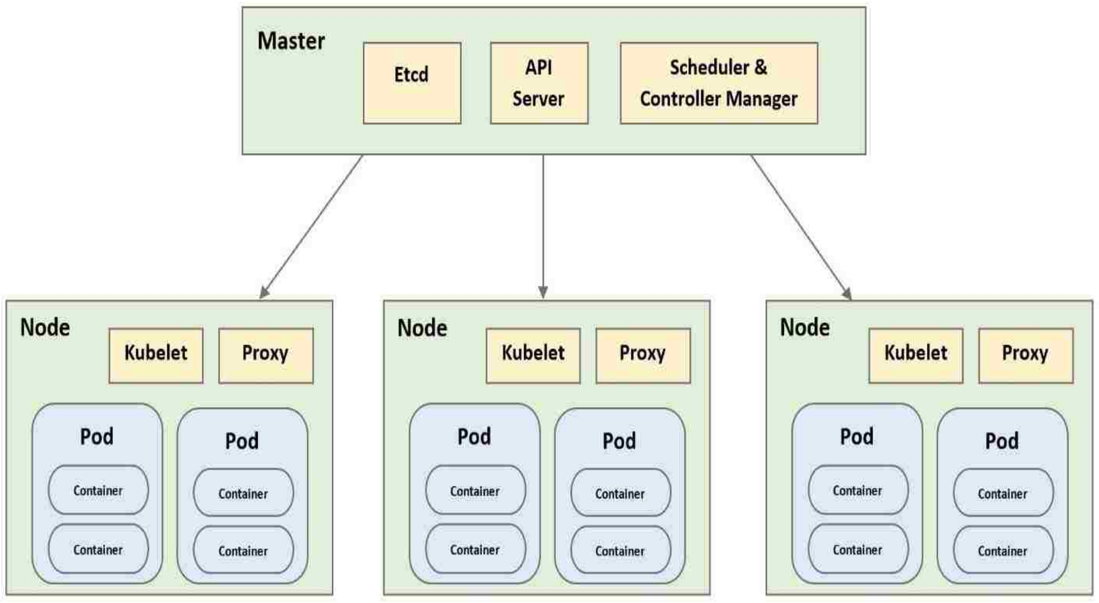


Figure 10 : Kubernetes Resource Model

Figure 10 represents Kubernetes resource model. The Kubernetes resource model aims to be: simple - for common cases; extensible - to be compatible with future development; regular - with few special cases; and precise - to promote pod portability.

A Kubernetes resource is can be requested, allocated or consumed by a pod or container. When resources on the node are distributed to the pod, they are not to be distributed further until that pod is eradicated or exits. The Kubernetes schedulers should ensure that the sum of the resources allocated, which may be requested and granted, to its pods never exceeds the usable capacity of the node.

6.6 Kubernetes application deployment workflow

As shown in figure 11, it shows how applications are deployed in the Kubernetes environment.

1. The user deploys new application using kubectl CLI. After that, kubectl then delivers the request to API server.
2. After receiving the request from kubectl, it is stored in the data store i.e. etcd. After writing the request to data store, the API server is done with the request.
3. The viewers see the resource changes and forward notifications to the Controller to act on the changes.
4. The Controller detects the new application and generates new pods to match the required no. of instances. The changes in the stored model are used for creating or deleting pods.
5. The Scheduler assigns new pods to a node and decides whether to execute pods on specific nodes in the cluster. The Scheduler changes the model with node information.
6. The kubelet on the node detects the pod and deploys the containers which it requested through the container runtime. Each node checks the storage to watch what pods it is assigned to execute. The node then performs needed actions on the resources which are assigned to it like creating or deleting the pods.

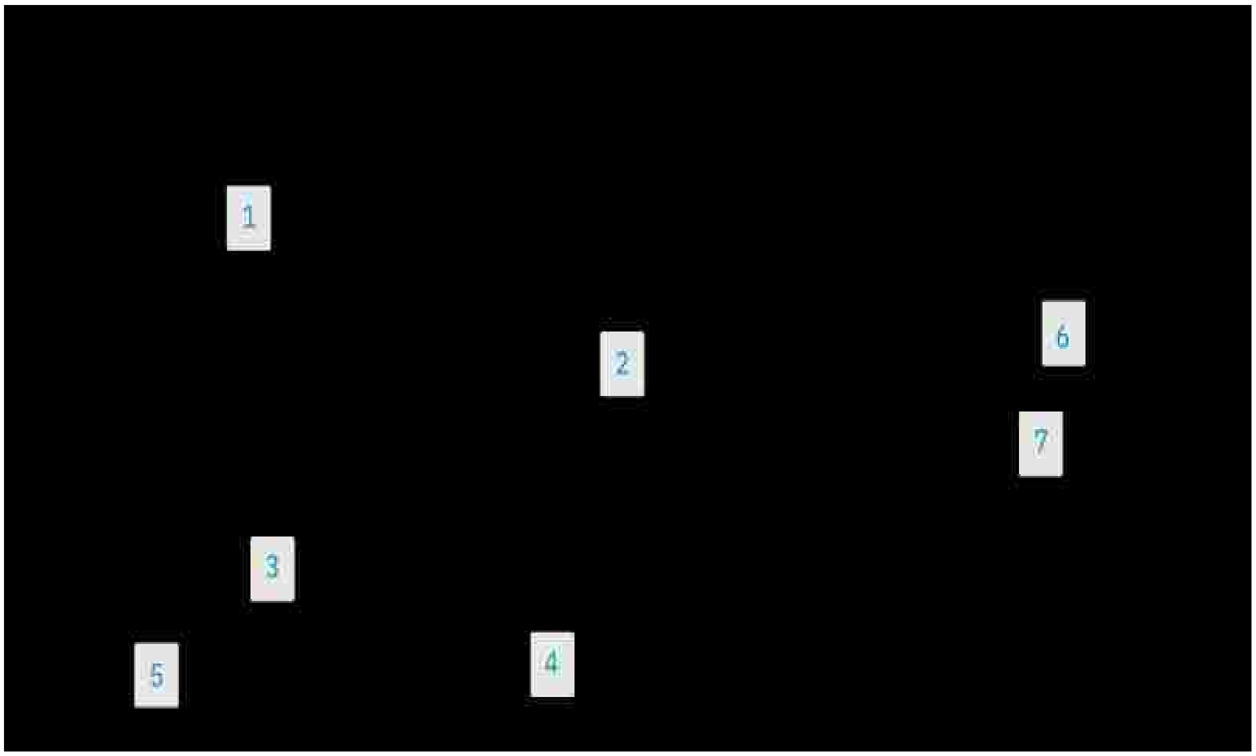


Figure 11 : Kubernetes application deployment workflow (IBM developerWorks Courses)

7. Kubeproxy then manages and controls the network traffic for the pods which consists of load balancing and service discovery. Kubeproxy is responsible for communication and interaction between the pods.

6.7 Network Topology in Kubernetes

The network topology is shown as in the figure 12. Kubernetes contains one or multiple master and worker nodes. There is CMD client to throw deployment commands. For configuration files and other data, we use NFS file system. We have indicated the physical network with blue lines. All the nodes are connected by a physical network. Kubernetes network model allows all pods to directly interact with one another. Flannel is a Kubernetes CNI addon. With Flannel, it is simple to generate an overlay network. Kubernetes links all pods to Flannel network enabling containers of those pods to interact with one another, which is shown by red lines. We specify IP address range of the kube_dns and flannel network in add-on configuration file. We have to ensure that the IP address of kube_dns is in defined address range. For e.g., here in the figure, Flannel network is 10.0.0.1/16 whereas kube_dns address is 10.0.0.10.

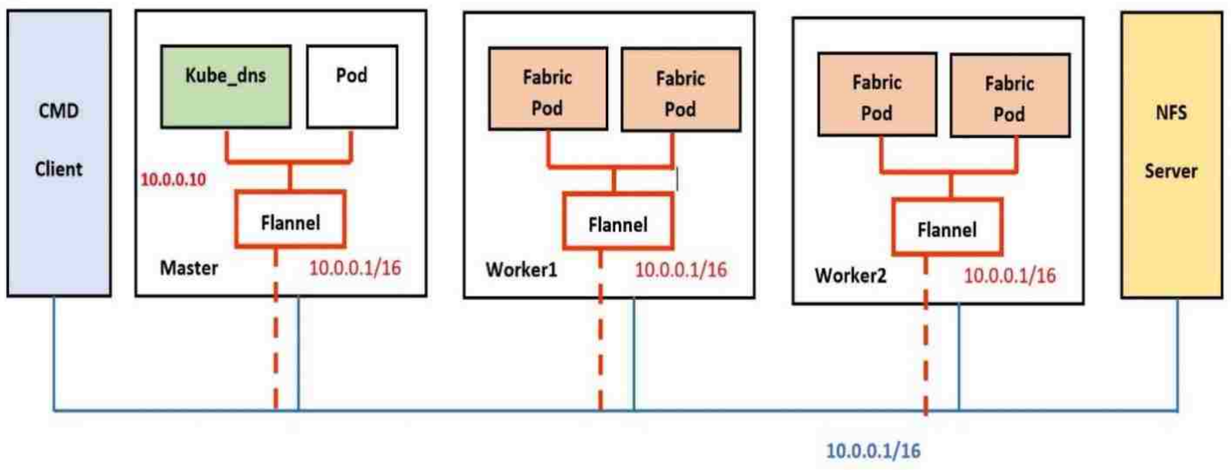


Figure 12 : Kubernetes Network Topology

6.8 Container Orchestration Tools

When container use grows with organizations deploying them on large scale, then the requirement for tools to handle these containers over the infrastructure also rises up. We have various container orchestration tools in the market offering various set of features.

Container orchestration tools may be widely described as allowing an enterprise-level framework for combining and controlling the containers at scale. Such tools can be used to manage several containers as one entity to serve the purpose of availability, scaling, and networking.

6.8.1 Amazon Elastic Container Service (Amazon ECS)

The Amazon EC2 Container Service (ECS) provides support for Docker containers and lets us execute applications on managed cluster of Amazon EC2 instances.

Pros: Highest capacity for scale; deeply integrated with the AWS ecosystem

Cons: AWS only; limited container discovery options. Amazon ECS seems to be falling behind the other main players. Since ECS is a tool which is not cloud agnostic.

6.8.2 Docker Swarm

Docker Swarm has native clustering feature for Docker containers, which enables us to modify a set of Docker engines into a single, virtual Docker engine.

Pros: Already deployed with Docker; simplest configuration; integrates with Docker-Compose

Cons: Limited cloud integration; one ELB per cluster.

6.8.3 Mesosphere Marathon

Marathon is a container orchestration framework for Apache Mesos that is built to establish the applications that are long-running. It provides important functions for executing applications in a clustered environment.

Pros: Mesosphere DC/OS is the existing connection to big-data applications. It has found its way into organizations before containers to control big data frameworks like Hadoop, Spark, etc.

Cons: We can run Kubernetes on top of DC/OS and schedule containers with it instead of using Marathon. Moreover, Marathon aggregates APIs and provides a relatively small amount of API resources, whereas Kubernetes provides a larger variety of resources and is based on label selectors. Also, about the size of the community and offering support; Kubernetes has almost ten times the commits and GitHub stars as compared to Marathon.

6.8.4 Azure Container Service (ACS)

ACS allows us to make a cluster of virtual machines which then act as container hosts with master machines which are used to control the application containers.

6.9 Why Kubernetes?

Kubernetes provides us with orchestration and management abilities which are needed to deploy containers. With Kubernetes orchestration we can build application services that span, schedule and scale several containers across the cluster. Additionally, it can control the health of those containers. It's easy to achieve high availability with Kubernetes. Fabric is built into container images, plus; Kubernetes is useful in orchestration, scaling, and managing containers. Kubernetes supports multi-tenancy, with which we can develop and test blockchain applications. To provide a comprehensive container infrastructure, Kubernetes requires integration with security, storage, networking, telemetry, security and some other services.

The main benefit of using Kubernetes is that it provides us the platform to schedule and execute the containers on clusters of physical or virtual machines. In production environments, it provides support to fully implement and depend on container-based infrastructure. With Kubernetes we can arrange containers across multiple hosts and make good hardware use to increase resources required to execute the enterprise apps. With this, we can automate and manage deployments and updates of the application.

Table 3: Comparison of Container Orchestration Tools

Characteristics	**Kubernetes**	**Docker Swarm**	**Amazon ECS**
Node Support	Supports up to 5000 nodes	Supports 2000 plus nodes	Supports upto 1000 nodes
Container limit	Limited to 3,00,000 containers	Limited to 95,000 containers	Limited to 5,00,000 containers
Optimized for	Optimized for single large cluster	Optimized for multiple smaller clusters per SDLC	Optimized for one cluster per SDLC
Replacement of Worker Nodes	Lost minion nodes are automatically replaced	Lost worker nodes automatically replaced	Worker nodes are easily added, replaced or removed
Health Check	Health checks are of two types; liveness and readiness	Docker Swarm health checks are limited to services.	ECS gives health checks using CloudWatch.
Load Balancing	Pods are given exposure via service which is used as load-balancer in cluster. For load balancing, we use ingress.	Load Balancer limitations can come with specific SSL or DNS requirements	Best-in breed container autoscaling & native ALB support
Management Tier Failure	Master nodes must maintain a quorum & failed master tier will cause most services to fail	Manager nodes must maintain a quorum, but failed manager will continue to run services	No single point of failure in managed control plane

Deployment	May be deployed on private clouds, public clouds, on-premises.	Deployment is easier and Swarm mode is present in Docker Engine.	Approved within Amazon. For deployment, ECS is not publicly available outside Amazon.
Community	Biggest community among container orchestration tools. More than 1200 contributors whereas 50,000 commits.	Has small community. Consists of over 160 contributors and 3,000 commits.	15 contributors and 200 commits.

CHAPTER 7

DEPLOYING HYPERLEDGER FABRIC ON IBM CLOUD USING KUBERNETES

Many people face problems to manage and deploy the Hyperledger Fabric system, as Hyperledger Fabric has lot of complexity in configuration. To simplify Hyperledger Fabric's operation, we require some tools to control distributed system of Hyperledger Fabric.

For developing any blockchain use case, firstly we will have to develop an environment for Hyperledger Fabric to create and deploy the application. In the environment setup, we will create a small blockchain network running Hyperledger Fabric. A Hyperledger Fabric network can be set up in multiple ways:

- Setting up local Hyperledger Fabric network using Docker Compose
- Using IBM Blockchain Platform, a flexible software-as-a-service offering hosted on IBM Cloud
- Kubernetes APIs on IBM Cloud

Kubernetes looks ideal for several reasons, because by deploying the network on Kubernetes, fabric components can accomplish high availability. It has got a feature known as replicator to monitor the pods which are running and automatically brings up the crashed ones. Moreover, Kubernetes supports multi-tenancy which is an important property. We can execute several isolated Fabric instances on our Kubernetes platform. This makes easier for us to develop and test the blockchain applications. If both used together, Hyperledger Fabric and Kubernetes offer a powerful, secure platform for processing blockchain transactions.

7.1 IBM Cloud Kubernetes Service

In our case, we are going to implement Hyperledger Fabric environment using Kubernetes to deploy blockchain applications using this IBM Cloud Kubernetes Service only. IBM Cloud Kubernetes Service was previously known as IBM Cloud Container Service.

IBM Cloud Kubernetes Service provides powerful tools by fusing Docker containers, Kubernetes technology, intuitive user experience and built-in security for quick delivery of applications that can be bound together with cloud services that are related to IoT, IBM Watson, DevOps and data analytics. It also delivers isolation for automating deployment, scaling, operation and monitoring of containerized apps in a cluster of compute hosts.

IBM Cloud Kubernetes Service delivers self-healing, service discovery & load balancing, intelligent scheduling, automated rollouts & rollbacks, horizontal scaling and secret & configuration management. The Kubernetes service also provides advanced capabilities around container security and isolation policies, cluster management, ability to design and implement our own cluster, and integrated operational tools for consistency in deployment.

7.1.1 Features of IBM Cloud Kubernetes Service

- Develop and build cloud-native apps
- Modernize and extend existing apps
- Secure the stack
- Accelerate DevOps pipeline

- Multi-cloud and multivendor portability
- Manage and control the microservices
- Great flexibility
- A Kubernetes-centric approach

7.1.2 High availability for IBM Cloud Kubernetes Service

High availability is a main discipline in IT environment infrastructure to continue the apps from going and executing; say, even if the site fails fully or partially. We use Kubernetes and IBM Cloud Kubernetes Service features to provide high availability to the cluster and protect the app from downtime when any component in the cluster fails.

The main aim of high availability is to remove potential points of failures in IT infrastructure. Consider a case where by adding redundancy and by setting up failover mechanisms, you can be prepared for failure of one system. We will get high availability on different levels in IT infrastructure and within different components of the cluster. The level of availability is based on several factors like business requirements, Service Level Agreements with customers, etc.

7.1.3 Managing apps in containers and clusters on IBM Cloud

We can manage and control the containers with IBM Cloud Kubernetes Service using Kubernetes clusters. To manage a cloud infrastructure, we use Kubernetes as an orchestration tool to scale, load-balance and monitor the containers. IBM Cloud Kubernetes Service gives a native Kubernetes experience which is simple, reliable and secure. Using the containers, we can isolate the ecosystem to execute any application on any host OS.

Besides this, containers can wrap code, system tools and libraries, runtimes that can be installed and executed on a server. Containers are just like virtual machines but differ in architecture. Images that execute on virtual machines have a full replica of the guest OS which consists of necessary libraries and binaries. Images that execute on the containers share the OS kernel on the host. The Docker Engine builds images on the containers. The docker engine is a lightweight container runtime that can run on almost any OS. You can run a container anywhere that a Docker Engine can be installed; on bare metal servers, clouds, and even inside a VM. We can relocate the containers from one environment to the other without any need to recode the application.

7.2 Mapping Fabric Components to Kubernetes Pods

Fabric is distributed system consisting of several nodes, where nodes may held by various entities. As shown in the figure 13, every organization has its own set of nodes. The orderers form a public consensus service.

Now, to deploy Hyperledger Fabric network on Kubernetes, firstly for deployment, we must turn all the components into pods and further utilize namespace to divide the organizations. Now in Kubernetes, we use namespace for separating cluster resources among multiple users. In the case of Hyperledger Fabric, for organizations to have their dedicated resource, they can be mapped into namespaces. Peers of every organization could be differentiated by domain name after this mapping. Later, we can divide different organizations by setting the network policy.

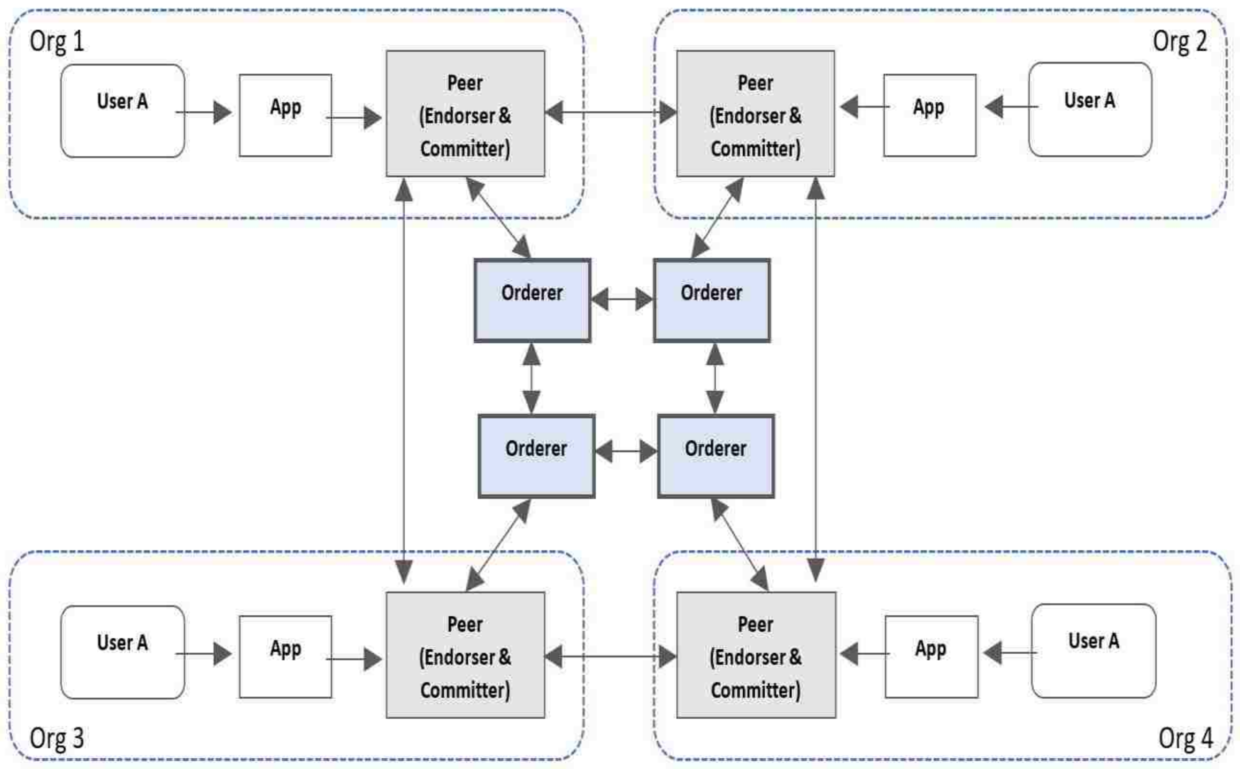


Figure 13 : Hyperledger Fabric Deployment Model

We need to make configuration files of its components like peers and orderers. We have tool for automating the creation of these configuration files, which are located in shared file system like NFS. After launching the pods of Fabric, we organize various subsets of configuration files into pods; as they will have configuration same as the organization. We use Persistent Volume (PV) and Persistent Volume Claim (PVC) to mount files or directories into a pod in Kubernetes[21]. For resource isolation, we make PVs and PVCs for every organization in fabric. In the NFS server, every organization can its own directory. We then specify Persistent Volume Claim after Persistent Volume is created, so that Fabric nodes can use Persistent Volume to retrieve equivalent files and directories[21].

When all Fabric's components are kept into Kubernetes pods, we must think upon the network connectivity among these pods[21]. In Kubernetes, every pod has an internal IP address. When pod is restarted, its IP address also changes. So, it is required to generate services for pods in Kubernetes, so they can communicate and interact among themselves via service name.

As an example, Hyperledger Fabric's peer0 of organization org1 is mapped to pod named peer0 under namespace org1. The service binding corresponding to it must be named peer0.org1, where peer0 denotes name of the service and org1 denotes namespace of the service. Other pods can then connect to the peer0 of org1 by service name peer0.org1, which is peer0's hostname[21].

7.3 Setting up business network on Hyperledger Fabric

Hosting the Hyperledger Fabric network on IBM Cloud has many advantages. If hosted, multiple users can work on the same setup and then the setup can be used for different blockchain applications. Thereafter, the setup can be reused.

The IBM Cloud Kubernetes Service is combination of Docker and Kubernetes to provide powerful tools to automate scaling deployment, operation, and monitoring of containerized applications over a cluster of independent compute hosts by using the Kubernetes APIs. The cloud-hosted Hyperledger Fabric network can be used for simpler collaboration among the team members.

The pattern of Hyperledger Fabric network consists of four organizations, each having one peer node, and an independent ordering service. Let's see what exactly peer is, orderer and channel.

- Peer: Peer is a node on the network which maintains the state of the ledger and manages chaincodes. The peers which may participate in a network can be of any number. A peer can be anything such as, an endorser or committer. Peers form a peer-to-peer gossip network.
- Orderer: Orderer manages a pluggable trust engine which performs the ordering of transactions.
- Channel: Channel is a data-partitioning mechanism which limits transaction visibility only to stakeholders. The consensus takes place within a channel only and is done by the members of the channel.

7.4 Process flow for deploying blockchain network on Kubernetes

Figure 14 represents the architecture and process flow of how blockchain network is deployed on Kubernetes. Firstly, we setup & login into IBM Cloud Developer Tools CLI which called as bx and initialize the IBM Cloud Kubernetes Service plugin. Then, set context for Kubernetes cluster using CLI and download Kubernetes configuration files. Once we export KUBECONFIG environment variable, then we can execute kubectl commands from CLI.

Now when we are set to run kubectl commands, this pattern provides scripts which automatically deploy the business network. We will have to run that script to deploy Hyperledger network on Kubernetes cluster. This script in turn will create persistent volume. It will copy the data into the persistent volume and will generate all the required network artifacts, channel artifacts etc. Now it will create peers and channels, and then join all peers on a channel. Then, it will install chaincode and will instantiate chaincode on channel. In this way, then we can access

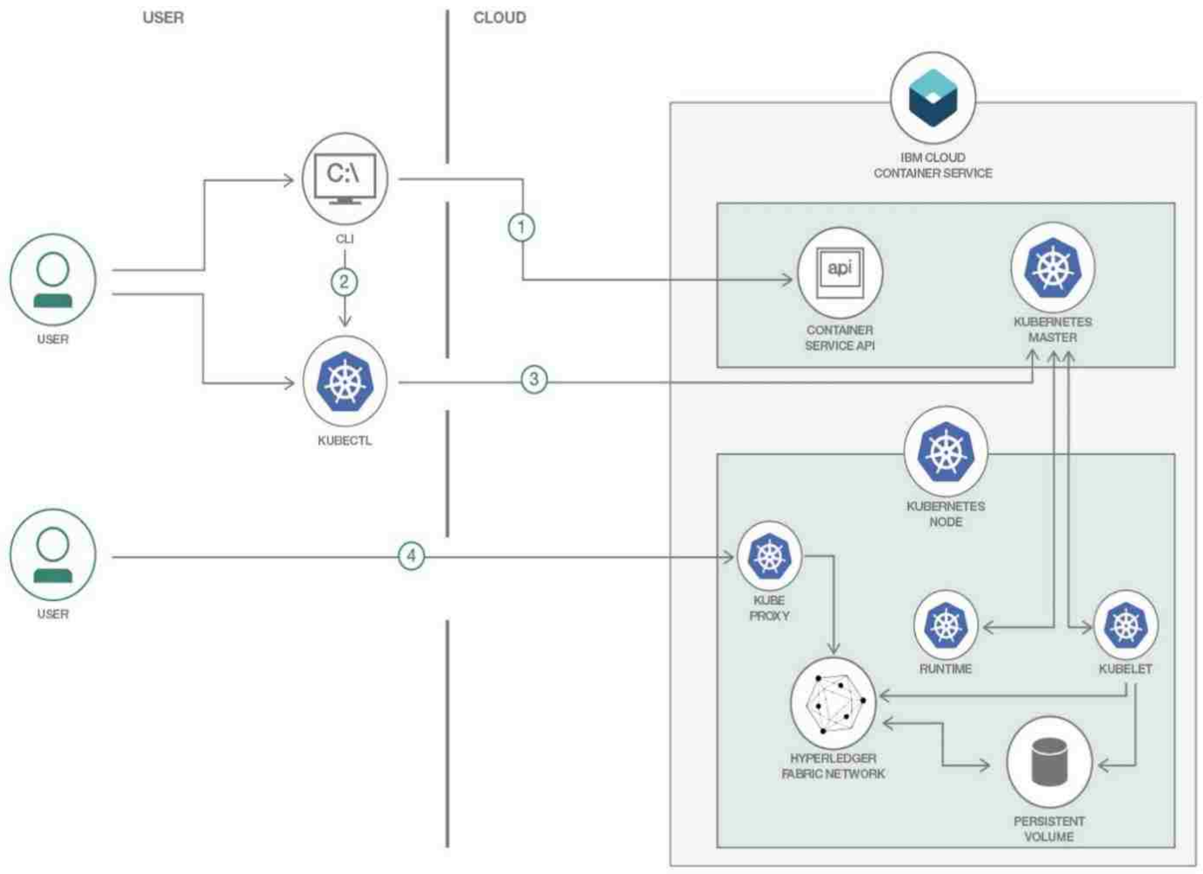


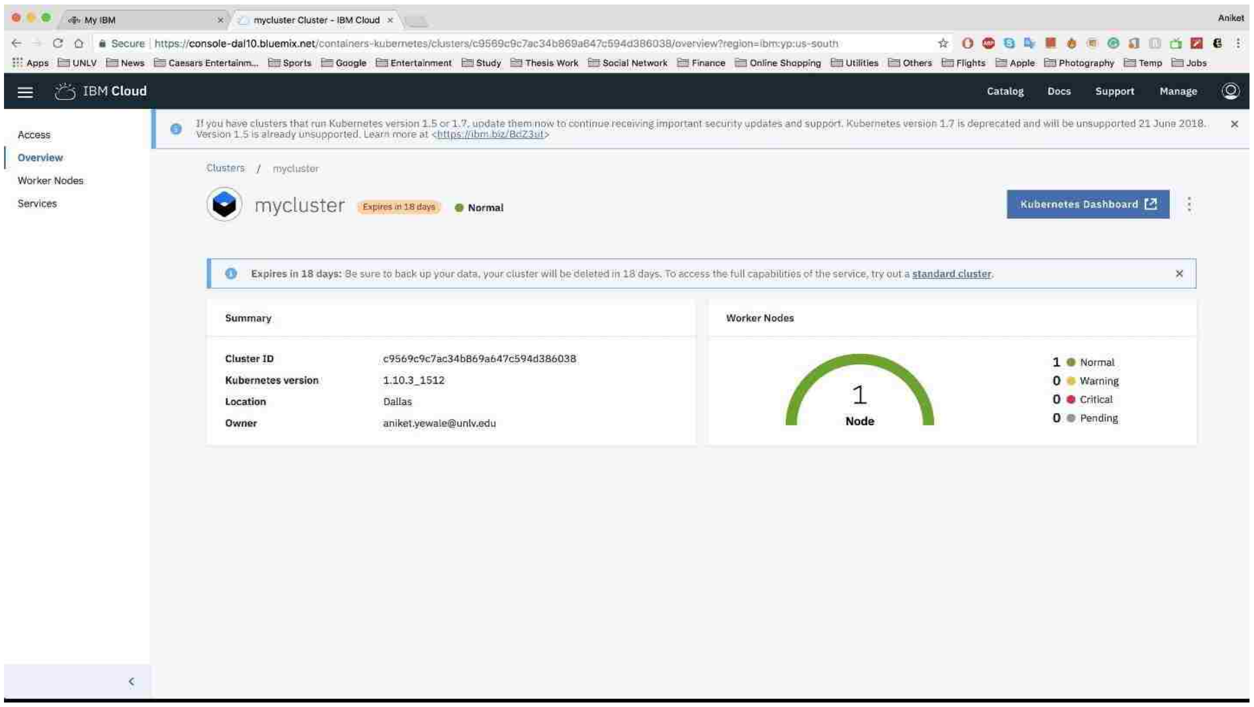
Figure 14 : Architecture of deploying blockchain network on Kubernetes

7.5 Steps involved during blockchain network setup on Kubernetes

Let's see the detailed steps and process involved in deploying the blockchain network on Hyperledger Fabric using Kubernetes APIs on the IBM Cloud Kubernetes Service.

7.5.1 Creating Kubernetes Cluster on IBM Cloud

As shown in figure 15, we created a free cluster on IBM Cloud Container Service which comes with 2 CPUs, 4 GB memory, and 1 worker node. It takes around 15-20 minutes for the cluster to be set up and in working state. Figure 16 represents worker node created along with the cluster.



If you have clusters that run Kubernetes version 1.5 or 1.7, update them now to continue receiving important security updates and support. Kubernetes version 1.7 is deprecated and will be unsupported 21 June 2018. Version 1.5 is already unsupported. Learn more at <https://ibm.biz/BdZ3ut>

Expires in 18 days: Be sure to back up your data, your cluster will be deleted in 18 days. To access the full capabilities of the service, try out a standard cluster.

Summary

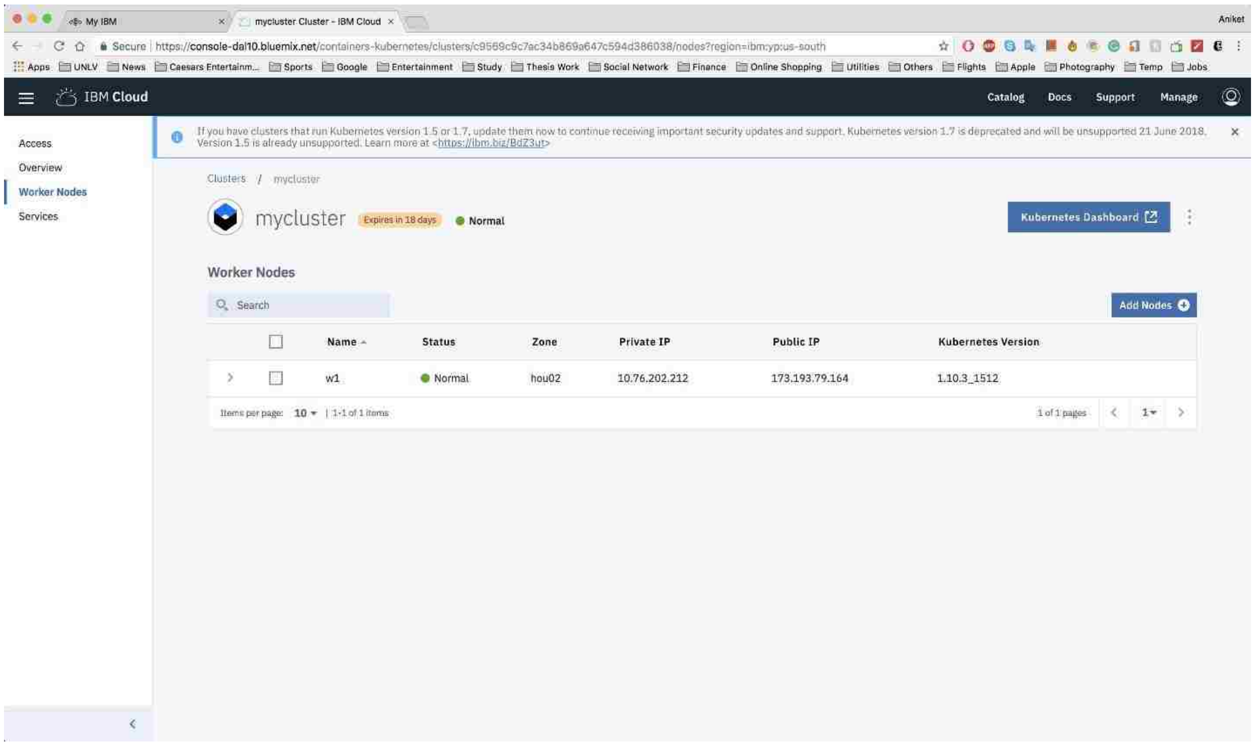
Cluster ID	c9569c9c7ac34b869a647c594d386038
Kubernetes version	1.10.3_1512
Location	Dallas
Owner	aniket.yewale@unlv.edu

Worker Nodes

1 Node

1	● Normal
0	● Warning
0	● Critical
0	● Pending

Figure 15 : Cluster 'mycluster' generated using IBM Cloud Container Service



If you have clusters that run Kubernetes version 1.5 or 1.7, update them now to continue receiving important security updates and support. Kubernetes version 1.7 is deprecated and will be unsupported 21 June 2018. Version 1.5 is already unsupported. Learn more at <https://ibm.biz/BdZ3ut>

Worker Nodes

Name	Status	Zone	Private IP	Public IP	Kubernetes Version
w1	● Normal	hou02	10.76.202.212	173.193.79.164	1.10.3_1512

Items per page: 10 | 1-1 of 1 items   1 of 1 pages

Figure 16 : Lone Worker Node w1 created along with Cluster

7.5.2 Setting up CLIs

We firstly install IBM Cloud CLI. This is done by executing commands using the Bluemix CLI which is bx. Similarly, we then install Kubernetes CLI which is done by running commands using the Kubernetes CLI which is kubectl. Then we install the container service plugin.

7.5.3 Gaining access to Kubernetes Cluster

After logging in IBM Cloud account, we target the IBM Cloud Container Service region in which you want to work. Then we will have to set the context for the cluster in the CLI by setting the environment variable and downloading the Kubernetes configuration files.

7.5.4 Deploy Hyperledger Fabric Network into Kubernetes Cluster

To deploy the blockchain network we will have to firstly decide on the network topology needed. Network topology such as number of organizations, number of peers/organization and the ordering service. Then we will have to set up the blockchain network using this pattern. After setting up the network, we can start developing blockchain applications on the deployed network.

Following is the process involved while deploying blockchain network on Kubernetes. This blockchain network is set up on Kubernetes using shell script which involves following steps.

1. Creating Persistant Volume
2. Copy the required files (configtx.yaml, crypto-config.yaml, sample chaincode etc.) into persistent volume
3. Generate Network artifacts using configtx.yaml and crypto-config.yaml

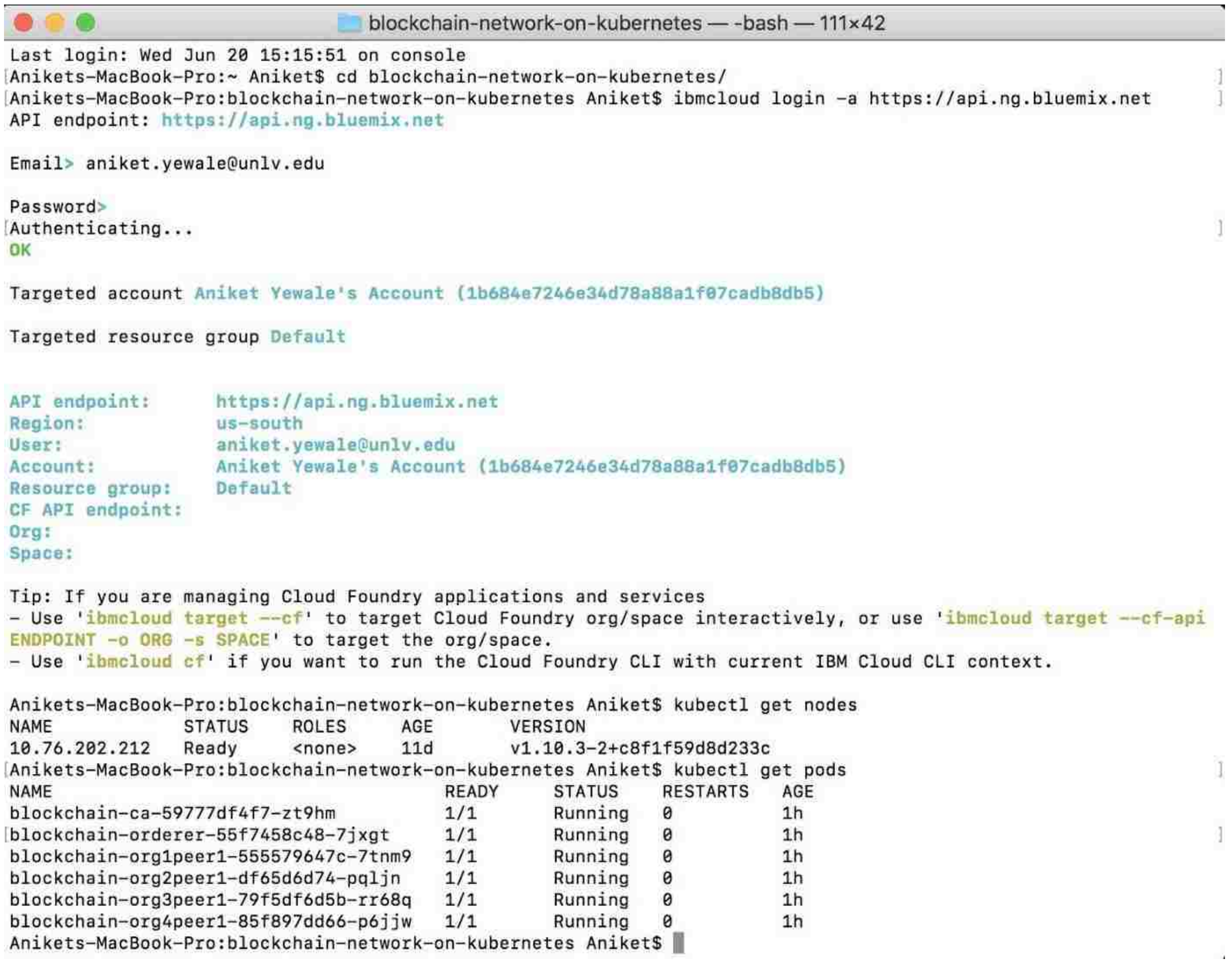
4. Create services for all peers, ca, orderer
5. Create peers, ca, orderer using Kubernetes Deployments
6. Generate channel artifacts using configtx.yaml and then create channel
7. Join all peers on a channel
8. Install chaincode on each peer
9. Instantiate chaincode on channel

'configFiles' contains Kubernetes configuration files. 'artifacts' contains the network configuration files '*.sh' scripts is to deploy and delete the network.

Now, if there is any change in network topology, we need to modify the configuration files (.yaml files) located in artifacts and configFiles directory. After these changes are done, we can execute the shell script by using the command $./setup_blockchainNetwork.sh, to deploy hyperledger fabric network.

7.5.5 Testing the deployed network

After successful execution of shell script, we need to the test the network by checking the status of pods. The shell script joins all peers on one channel, install chaincode on all peers and instantiate chaincode on a channel. It means we can execute an invoke/query command on any peer and the response should be same on all peers. For running a query against any peer, we need to get into a bash shell of a peer, execute the query and exit from the peer container. Figure 17 shows the current status of pods by executing '$kubectel get pods' command.



```
Last login: Wed Jun 20 15:15:51 on console
Anikets-MacBook-Pro:~ Aniket$ cd blockchain-network-on-kubernetes/
Anikets-MacBook-Pro:blockchain-network-on-kubernetes Aniket$ ibmcloud login -a https://api.ng.bluemix.net
API endpoint: https://api.ng.bluemix.net

Email> aniket.yewale@unlv.edu

Password>
Authenticating...
OK

Targeted account Aniket Yewale's Account (1b684e7246e34d78a88a1f07cadb8db5)

Targeted resource group Default


API endpoint:      https://api.ng.bluemix.net
Region:            us-south
User:              aniket.yewale@unlv.edu
Account:           Aniket Yewale's Account (1b684e7246e34d78a88a1f07cadb8db5)
Resource group:    Default
CF API endpoint:
Org:
Space:

Tip: If you are managing Cloud Foundry applications and services
- Use 'ibmcloud target --cf' to target Cloud Foundry org/space interactively, or use 'ibmcloud target --cf-api
ENDPOINT -o ORG -s SPACE' to target the org/space.
- Use 'ibmcloud cf' if you want to run the Cloud Foundry CLI with current IBM Cloud CLI context.

Anikets-MacBook-Pro:blockchain-network-on-kubernetes Aniket$ kubectl get nodes
NAME            STATUS    ROLES     AGE       VERSION
10.76.202.212   Ready     <none>    11d       v1.10.3-2+c8f1f59d8d233c
Anikets-MacBook-Pro:blockchain-network-on-kubernetes Aniket$ kubectl get pods
NAME                                      READY     STATUS    RESTARTS   AGE
blockchain-ca-59777df4f7-zt9hm            1/1       Running   0          1h
blockchain-orderer-55f7458c48-7jxgt       1/1       Running   0          1h
blockchain-org1peer1-555579647c-7tnm9     1/1       Running   0          1h
blockchain-org2peer1-df65d6d74-pqljn      1/1       Running   0          1h
blockchain-org3peer1-79f5df6d5b-rr68q     1/1       Running   0          1h
blockchain-org4peer1-85f897dd66-p6jjw     1/1       Running   0          1h
Anikets-MacBook-Pro:blockchain-network-on-kubernetes Aniket$
```

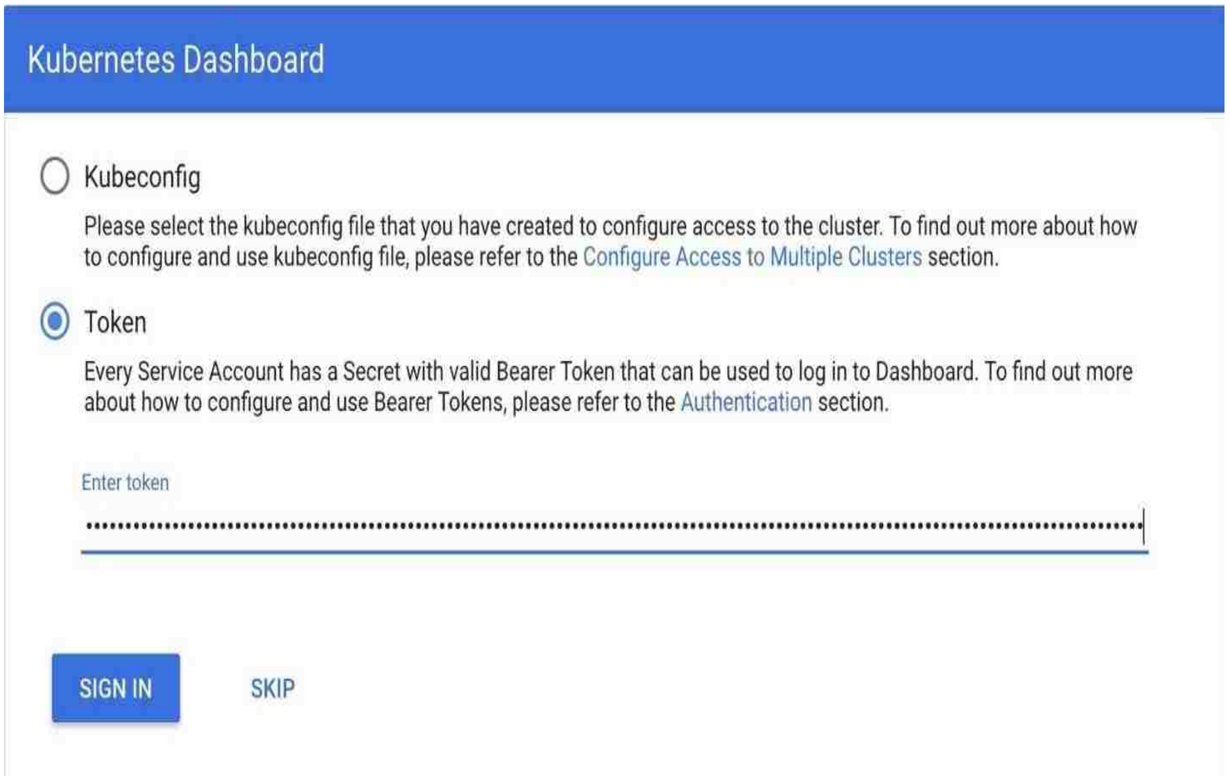
Figure 17 : Checking Status of Pods

7.5.6 Viewing the Kubernetes Dashboard

We need to get the token using the following command to authenticate and access Kubernetes dashboard.

$ kubectl config view -o jsonpath='{.users[0].user.auth-provider.config.id-token}'

After copying the token, we need to run $ kubectl proxy command to launch Kubernetes dashboard with the default port 8001 as shown in the figure 18.


Kubernetes Dashboard

Kubeconfig [✓ Token]

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

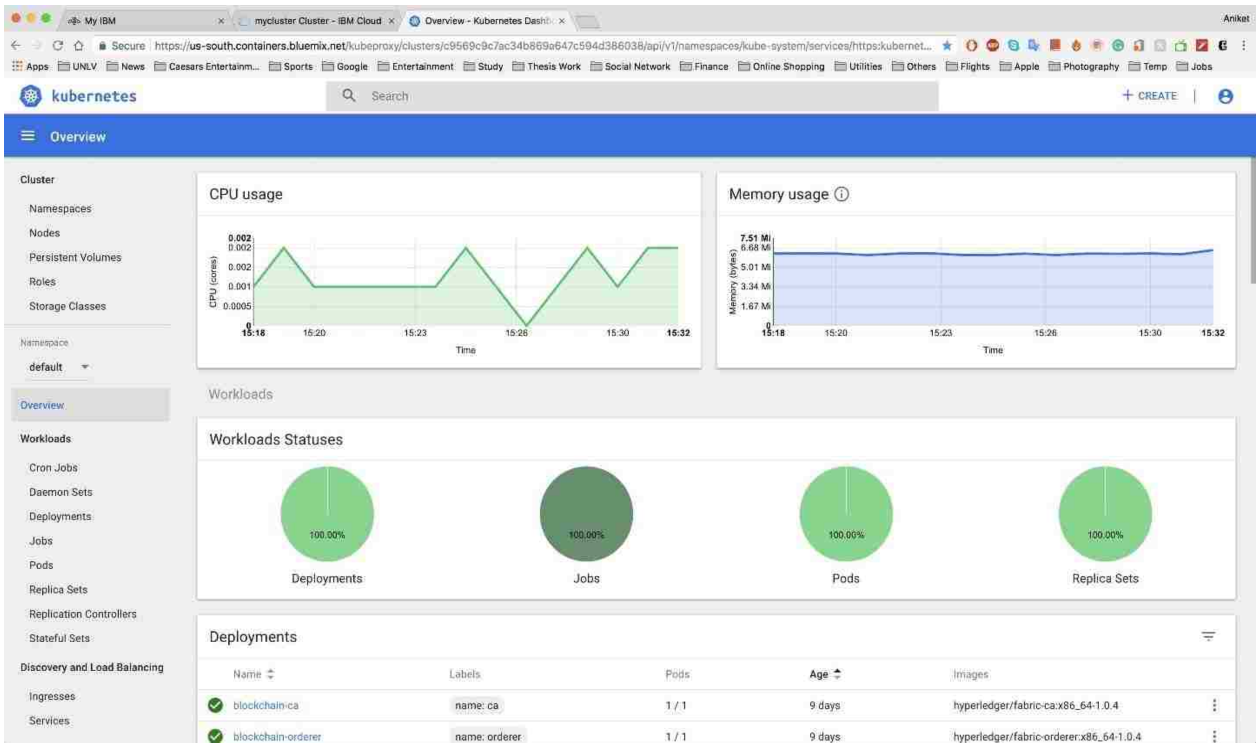
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Enter token

Figure 18 : Kubernetes Authentication Dashboard

The Hyperledger Fabric network is deployed and ready to use. Now, on this network we can develop our blockchain applications for this deployed network.

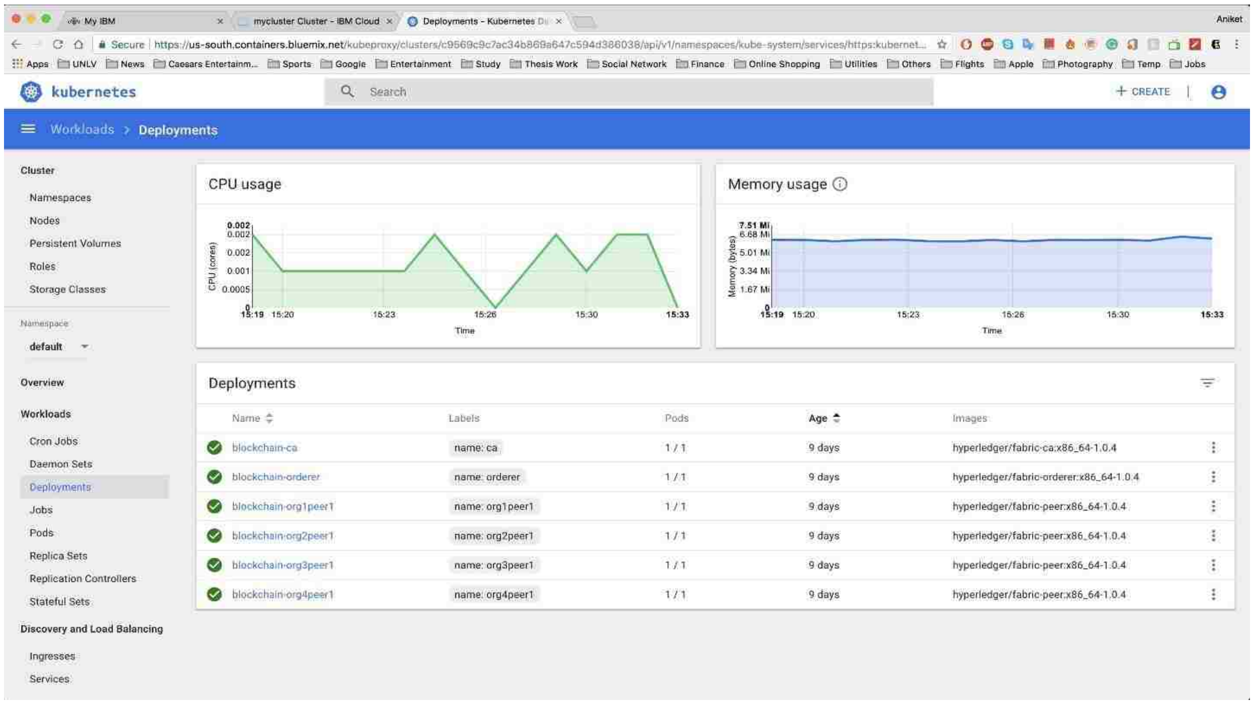
Let’s have a look at the resources on Kubernetes Dashboard. Figure 19 is Kubernetes dashboard overview. Figure 20 represents deployments on the Kubernetes dashboard. Figure 21 represents pods on the Kubernetes dashboard. Figure 22 shows one of the peers on the dashboard. Figure 23 represents jobs, figure 24 shows persistent volume claims on the Kubernetes dashboard. Figure 25 represents nodes and figure 26 represents namespaces on the Kubernetes dashboard.



Deployments

Name	Labels	Pods	Age	Images
blockchain-ca	name: ca	1 / 1	9 days	hyperledger/fabric-ca:x86_64-1.0.4
blockchain-orderer	name: orderer	1 / 1	9 days	hyperledger/fabric-orderer:x86_64-1.0.4

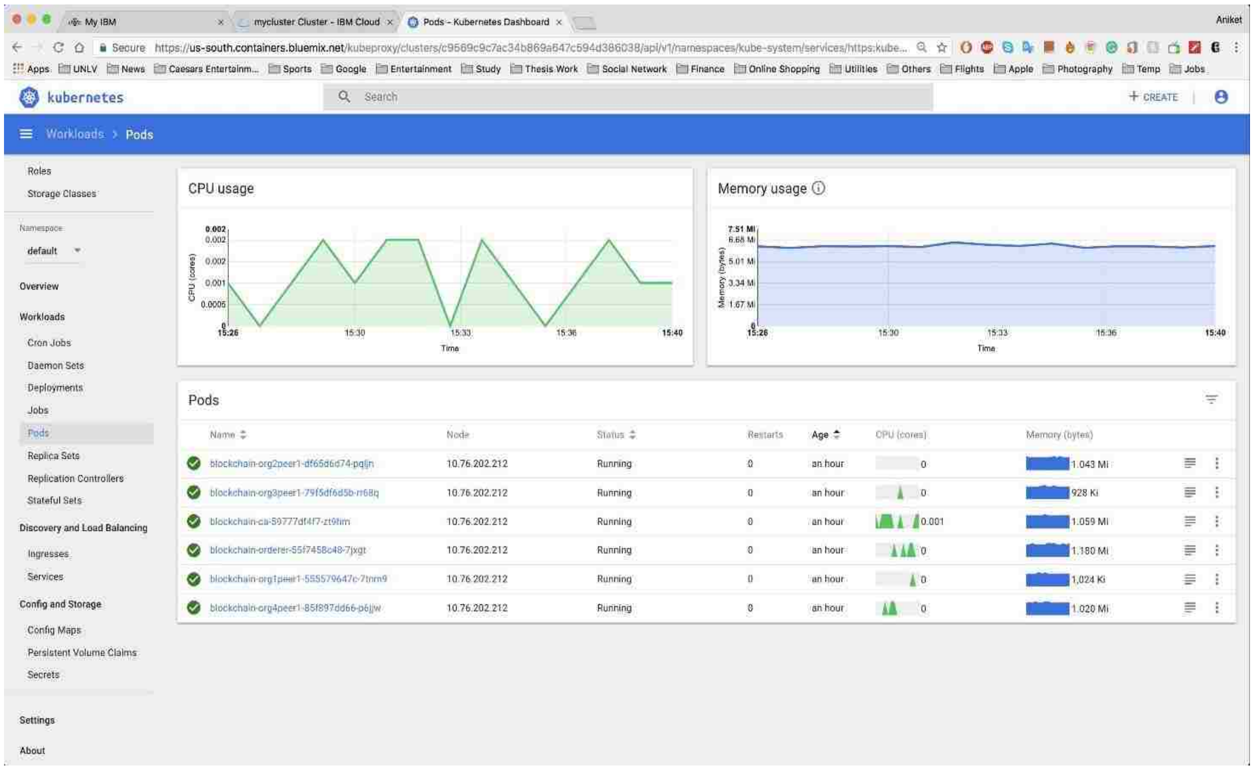
Figure 19 : Kubernetes Dashboard Overview



Deployments

Name	Labels	Pods	Age	Images
blockchain-ca	name: ca	1 / 1	9 days	hyperledger/fabric-ca:x86_64-1.0.4
blockchain-orderer	name: orderer	1 / 1	9 days	hyperledger/fabric-orderer:x86_64-1.0.4
blockchain-org1peer1	name: org1peer1	1 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4
blockchain-org2peer1	name: org2peer1	1 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4
blockchain-org3peer1	name: org3peer1	1 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4
blockchain-org4peer1	name: org4peer1	1 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4

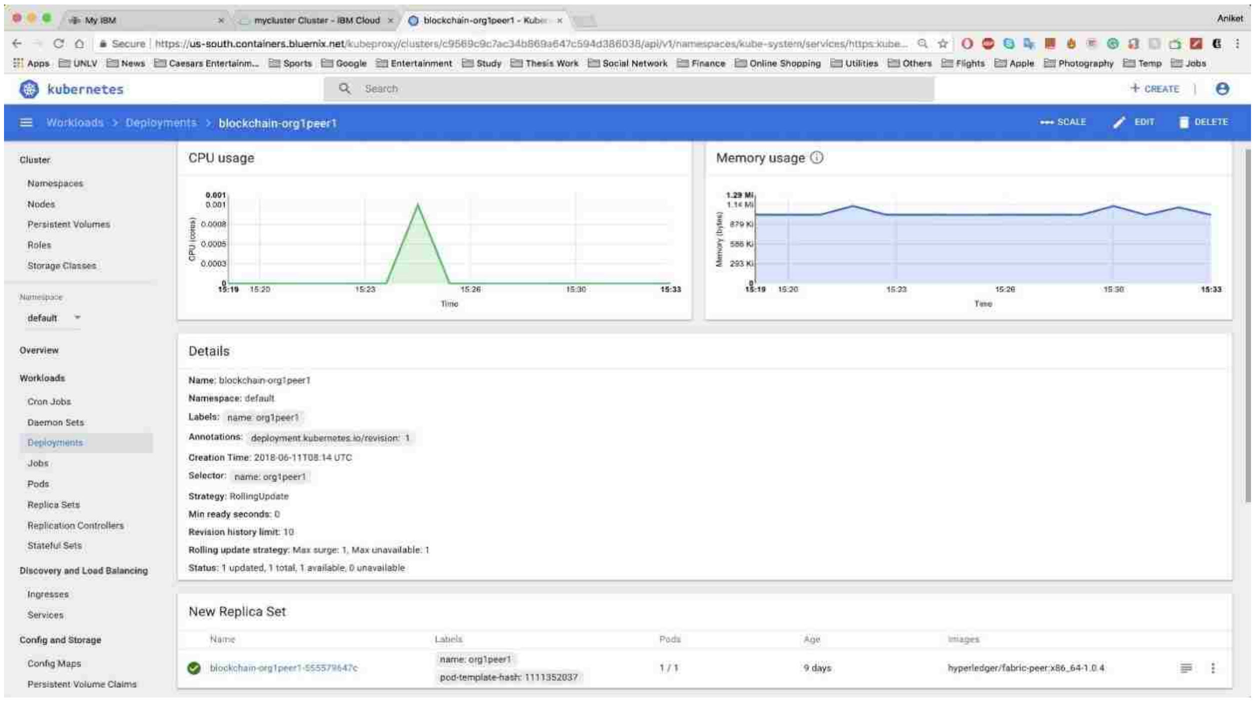
Figure 20 : Deployments on Kubernetes Dashboard



Pods

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
blockchain-org2peer1-df65d6d74-pqljn	10.76.202.212	Running	0	an hour	0	1.043 Mi
blockchain-org3peer1-79f5df6d5b-rr68q	10.76.202.212	Running	0	an hour	0	928 Ki
blockchain-ca-59777df4f7-zt9fm	10.76.202.212	Running	0	an hour	0.001	1.059 Mi
blockchain-orderer-55f7458c48-7jxgt	10.76.202.212	Running	0	an hour	0	1.180 Mi
blockchain-org1peer1-555579647c-7tnm9	10.76.202.212	Running	0	an hour	0	1,024 Ki
blockchain-org4peer1-85f897dd66-p6jjw	10.76.202.212	Running	0	an hour	0	1.020 Mi

Figure 21 : Pods on Kubernetes Dashboard



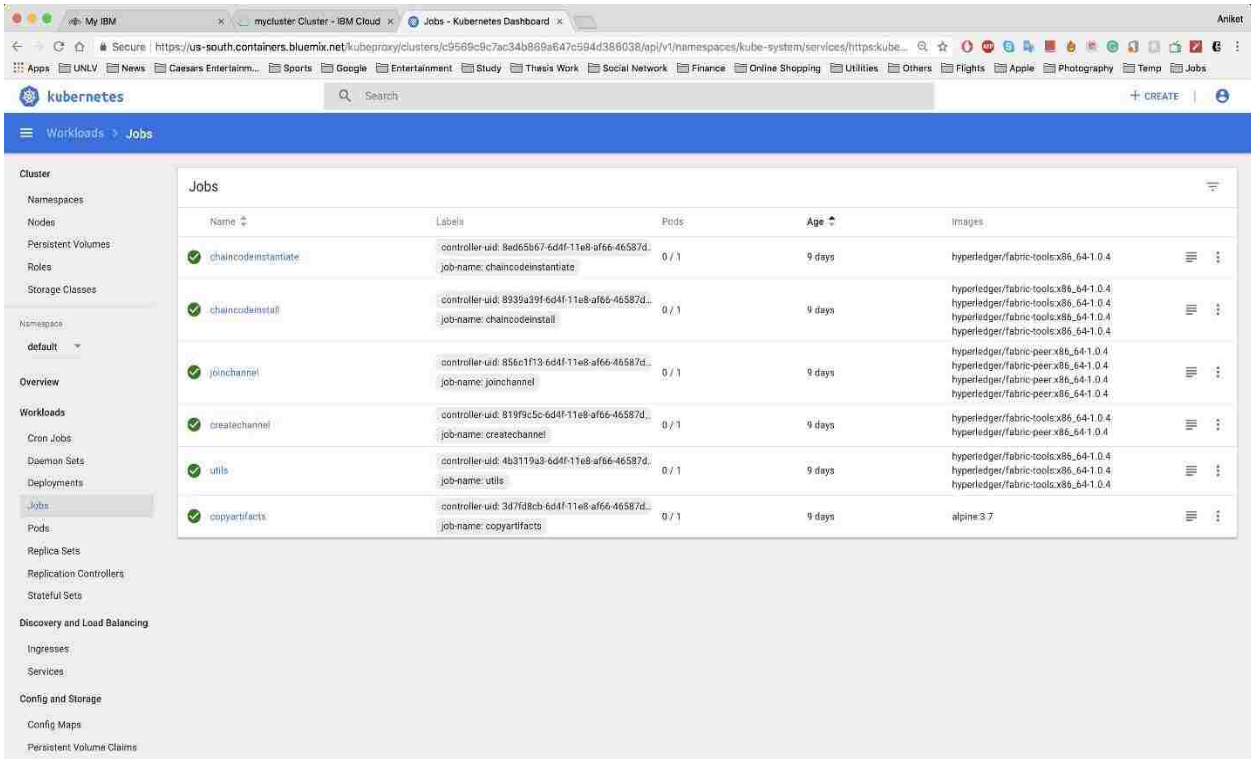
Details

Name: blockchain-org1peer1
Namespace: default
Labels: name: org1peer1
Annotations: deployment.kubernetes.io/revision: 1
Creation Time: 2018-06-11T08:14 UTC
Selector: name: org1peer1
Strategy: RollingUpdate
Min ready seconds: 0
Revision history limit: 10
Rolling update strategy: Max surge: 1, Max unavailable: 1
Status: 1 updated, 1 total, 1 available, 0 unavailable

New Replica Set

Name	Labels	Pods	Age	Images
blockchain-org1peer1-555579647c	name: org1peer1 pod-template-hash: 1111352037	1 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4

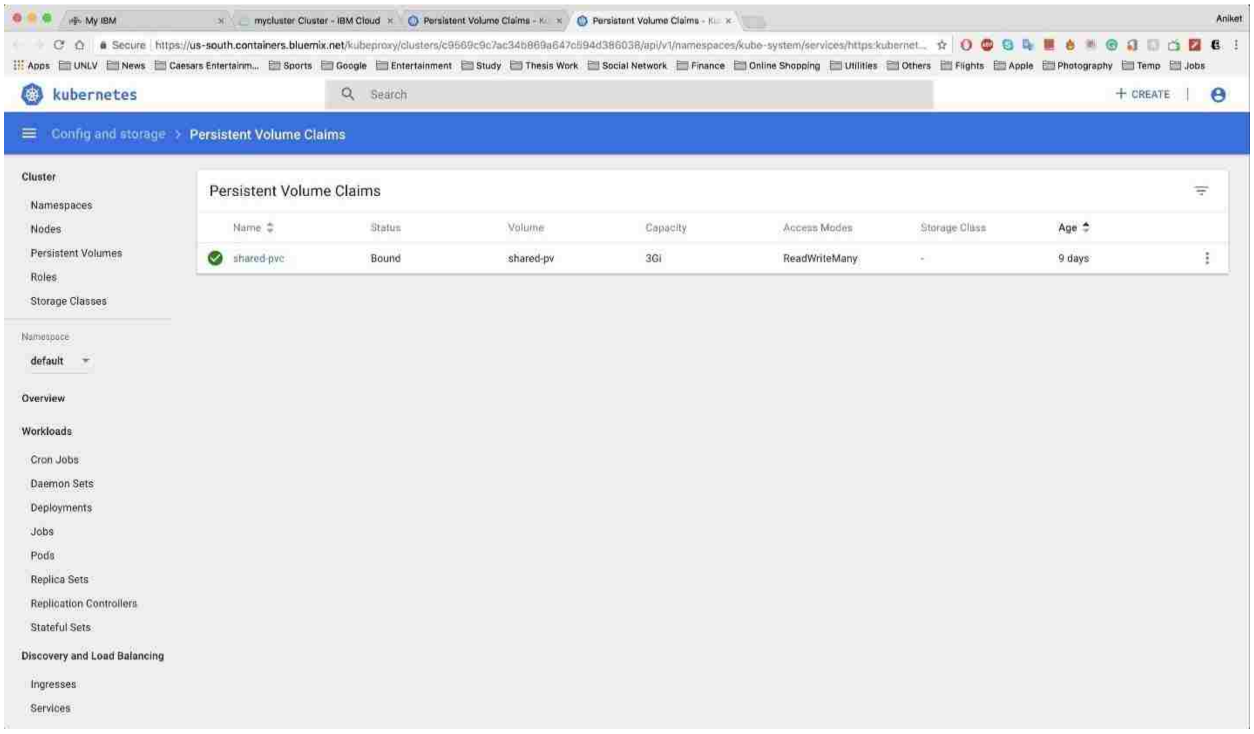
Figure 22 : Peer ‘blockchain-org1peer1-555579647c-7tnm9’ on Kubernetes Dashboard



Jobs

Name	Labels	Pods	Age	Images
✓ chaincodeinstantiate	controller-uid: 8ed65b67-6d4f-11e8-af66-46587d... job-name: chaincodeinstantiate	0 / 1	9 days	hyperledger/fabric-tools:x86_64-1.0.4
✓ chaincodeinstall	controller-uid: 8939a39f-6d4f-11e8-af66-46587d... job-name: chaincodeinstall	0 / 1	9 days	hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-tools:x86_64-1.0.4
✓ joinchannel	controller-uid: 856c1f13-6d4f-11e8-af66-46587d... job-name: joinchannel	0 / 1	9 days	hyperledger/fabric-peer:x86_64-1.0.4 hyperledger/fabric-peer:x86_64-1.0.4 hyperledger/fabric-peer:x86_64-1.0.4 hyperledger/fabric-peer:x86_64-1.0.4
✓ createchannel	controller-uid: 819f9c5c-6d4f-11e8-af66-46587d... job-name: createchannel	0 / 1	9 days	hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-peer:x86_64-1.0.4
✓ utils	controller-uid: 4b3119a3-6d4f-11e8-af66-46587d... job-name: utils	0 / 1	9 days	hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-tools:x86_64-1.0.4 hyperledger/fabric-tools:x86_64-1.0.4
✓ copyartifacts	controller-uid: 3d7fd8cb-6d4f-11e8-af66-46587d... job-name: copyartifacts	0 / 1	9 days	alpine:3.7

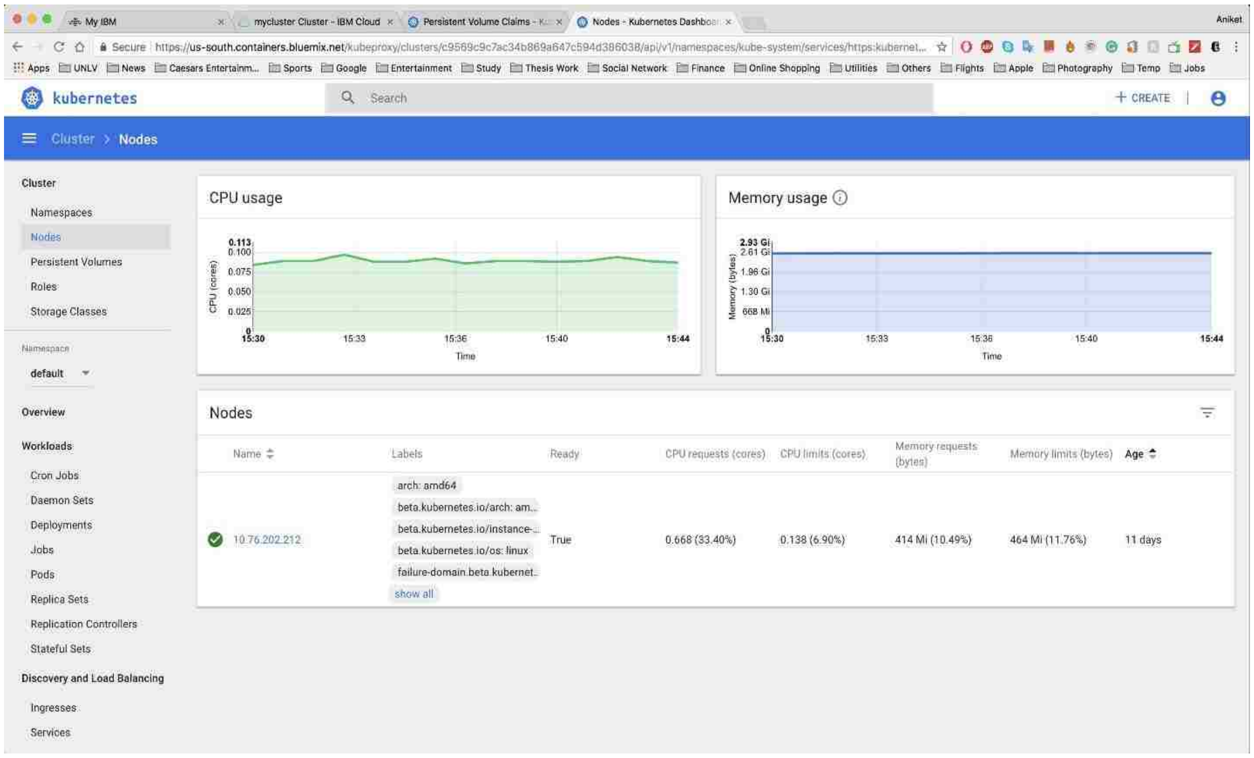
Figure 23 : Jobs on Kubernetes Dashboard



Persistent Volume Claims

Name	Status	Volume	Capacity	Access Modes	Storage Class	Age
✓ shared-pvc	Bound	shared-pv	3Gi	ReadWriteMany	-	9 days

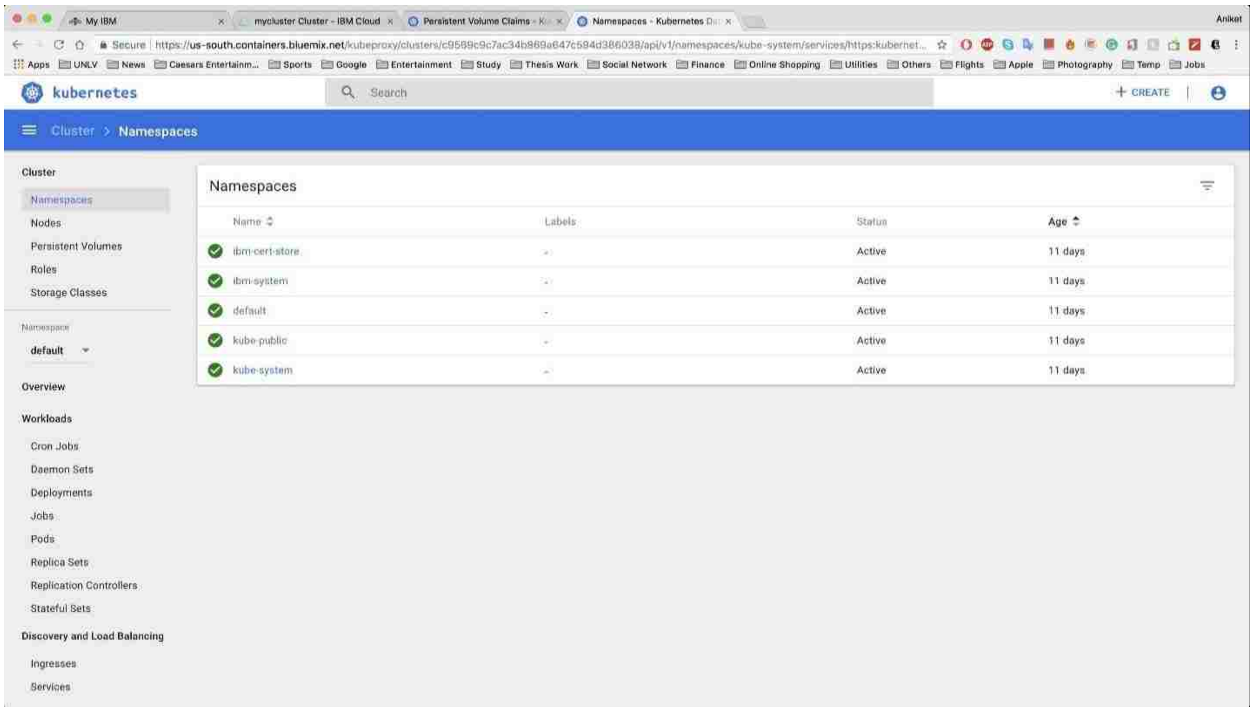
Figure 24 : Persistent Volume Claims on Kubernetes Dashboard



Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
10.76.202.212	arch: amd64 beta.kubernetes.io/arch: am... beta.kubernetes.io/instance-... beta.kubernetes.io/os: linux failure-domain.beta.kubernet... show all	True	0.668 (33.40%)	0.138 (6.90%)	414 Mi (10.49%)	464 Mi (11.76%)	11 days

Figure 25 : Nodes on Kubernetes Dashboard



Namespaces

Name	Labels	Status	Age
ibm-cert-store	-	Active	11 days
ibm-system	-	Active	11 days
default	-	Active	11 days
kube-public	-	Active	11 days
kube-system	-	Active	11 days

Figure 26 : Namespaces on Kubernetes Dashboard

CHAPTER 8

CONCLUSION

We studied several platforms offering Blockchain-as-a-Service (BaaS) and container orchestration tools. In our case scenario, we found that IBM's Hyperledger BaaS system and Kubernetes are good as both goes hand in hand. Both, Hyperledger Fabric and Kubernetes when used together provides us a powerful, reliable and a secure platform for processing blockchain transactions.

We were successfully able to deploy the Hyperledger Fabric network on Kubernetes using IBM Cloud Service. We can develop and build our own blockchain applications, smart contracts and chaincodes on this deployed network.

With the implementation of Hyperledger Fabric business blockchain network on IBM Cloud Kubernetes service we can have multiple users work on the same setup. We can use this environment repeatedly for several blockchain applications and run several isolated Fabric instances on our Kubernetes platform. This makes easier for us to develop and test the blockchain applications.

REFERENCES

[1] Alexandru Stanciu, “Blockchain based distributed control system for Edge Computing”, IEEE 2017 - 21st International Conference on Control Systems and Computer Science (CSCS)

[2] A Bessani, J Sousa, M Vukolić, “A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform”, Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, ISBN: 978-1-4503-5173-7

[3] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong, “Performance Analysis of Private Blockchain Platforms in Varying Workloads”, July 2017 IEEE

[4] Oscar Novo , “Blockchain Meets IoT: an Architecture for Scalable Access Management in IoT”, IEEE Internet of Things Journal (Volume: 5, Issue: 2, April 2018)

[5] Hyperledger Architecture - Volume 2 - Smart Contracts, Hyperledger Whitepapers

[6] Casimer DeCusatis, Marcus Zimmermann, Anthony Sager, “Identity-based Network Security for Commercial Blockchain Services”, 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)

[7] “Hyperledger Fabric Docs Master Documentation release-1.1 - 2018”, http://hyperledger-fabric.readthedocs.io/en/release-1.1/index.html

[8] Deborah Dobson, “The 4 Types of Blockchain Networks Explained”, International Legal Technology Association

[9] “Blockchains & Distributed Ledger Technologies”, https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/

[10] Harish Sukhwani, Jose M. Martinez, Xiaolin Chang, Kishor S. Trivedi, and Andy Rindos, “Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)”, 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)

[11] “Difference between various blockchain protocols”, https://hype.codes/difference-between-various-blockchain-protocols

[12] Ittay Eyal, “Blockchain Technology Transforming Libertarian Cryptocurrency Dreams to Finance and Banking Realities”, Computer (Volume: 50, Issue: 9, 2017)

[13] Leemon Baird, Mance Harmon, Tom Trowbridge, Jordan Fried, Natalie Furman, “Hedera Hashgraph Whitepaper”, Hashgraph Consortium, LLC, 2018

[14] Ajitesh Kumar, “List of Hyperledger Tools & Frameworks for Blockchain Apps”, https://vitalflux.com/list-hyperledger-tools-frameworks-blockchain-apps/

[15] Hyperledger Architecture, Volume 1 – Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus, Hyperledger Whitepapers

[16] C. Cachin, "Architecture of the Hyperledger blockchain fabric," in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[17] Shashank, “Hyperledger Fabric – A Platform for Business Solutions”, https://www.edureka.co/blog/hyperledger-fabric/

[18] Nitin Gaur, Luc Desrosiers, Venkatraman Ramakrishna, Petr Novotny, Dr. Salman A. Baset Anthony O'Dowd, “Hands-On Blockchain with Hyperledger”, June 2018

[19] Varun Raj, “Understanding Hyperledger Fabric Architecture”, https://www.skcript.com/svr/understanding-hyperledger-fabric-s-architecture/

[20] Karthik K, “Why choose Hyperledger Fabric for your Enterprise Blockchain”, https://www.skcript.com/svr/5-advantages-of-using-hyperledger-fabric-for-your-enterprise-blockchain/

[21] Henry Zhang, Luke Chen,” How to Deploy Hyperledger Fabric on Kubernetes”, Hackernoon, 2017

CURRICULUM VITAE

GRADUATE COLLEGE
UNIVERSITY OF NEVADA, LAS VEGAS

ANIKET YEWALE
aniketyewale29@gmail.com

Degrees:

- Bachelor of Engineering in Computer Engineering, 2015

  Savitribai Phule Pune University

- Master of Science in Computer Science, 2018

  University of Nevada, Las Vegas

Thesis Title: **Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes**

Thesis Examination Committee:

- Chair Person, Dr. Yoohwan Kim, Ph.D
- Committee Member, Dr. Laxmi Gewali, Ph.D
- Committee Member, Dr. Wolfgang Bein, Ph.D
- Graduate College Representative, Dr. Sean Mulvenon, Ph.D