**Section 1: Hyperledger Fabric Refresher**

**Hyperledger project and blockchain for business:**

Hyperledger is not a single project but a collection of projects under the Hyperledger Initiative.

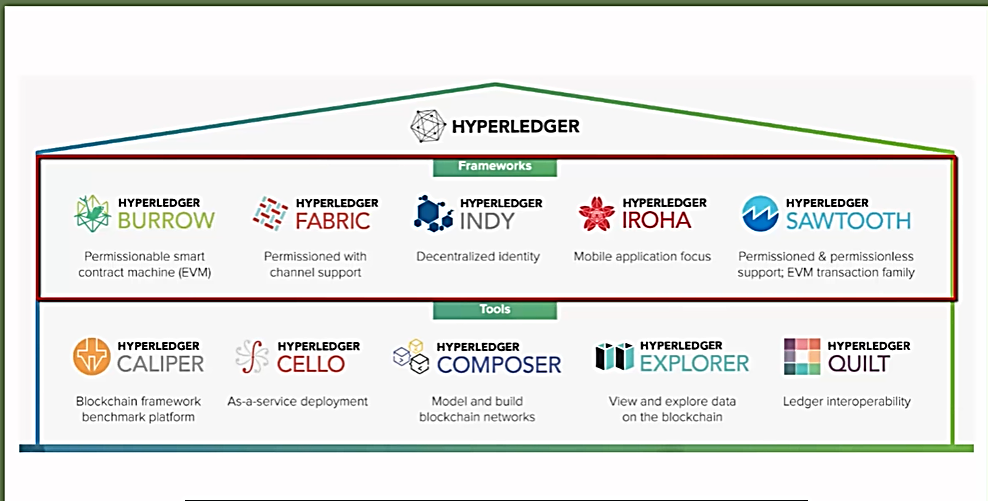
There are two types of projects.

1. **Frameworks:** These are the distributed ledger technology frameworks.

Hyperledger Fabric is a distributed ledger technology framework.

1. **Tools:** Hyperledger Composer is a tool that can accelerate the development of applications on Hyperledger fabric.

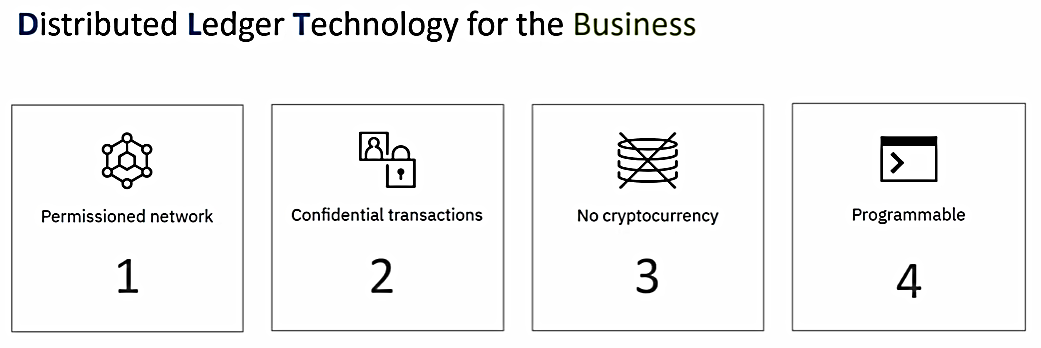
The expectation for the tools is that they will work across the various distributed ledger technology frameworks.



**“Hyperledger Fabric is a distributed ledger technology framework for building business blockchain applications.”**

**Distributed ledger technology for the Business**

There are **four** aspects of Hyperledger fabric distributed ledger technology that makes it suitable for implementing blockchain applications for business, **permissioned network, confidentiality of the transactions**. No need for cryptocurrency, for transactions and programmability.

****

1. **Permissioned blockchain**

 the participants don't have to reveal their identity. They can just download a software and start transacting on the public network.

In the case of a business application, anonymity is not the way things work.

* Member entities are known
* Participants have role based access

Hyperledger Fabric is a permissioned network in which identities are managed and roles are assigned.

1. **confidential transactions**

In the case of a public blockchain network, all transactions are visible to everyone.

In the case of a business blockchain network, it may not be a desirable thing.

Sometime businesses like to keep their transactions confidential and visible only to the counterparty.

Hyperledger fabric based blockchain network allows the members to transact privately.

That means their transactions are visible only to the counterparties and are not visible to the other members on the network.

* Not all transactions visible to all
* Members can transact privately

1. **No cryptocurrency**

in the case of a public blockchain network, transactions are validated by the members of the network in return for crypto tokens and that is the incentive which keeps the public networks going.

In the case of a business, blockchain network, cryptocurrency or crypto token based incentivization is not needed.

* No incentivization needed.
* No crypto token needed for transactions

1. **programmable**

Also, in the case of a **business blockchain** network, there is no need for crypto tokens to be given bout as a fee for the execution of the transactions.

Hyperledger Fabric does not have the concept of cryptocurrencies.

For that reason, Hyperledger Fabric supports **smart contracts or the chain codes.**

The chain code encapsulates the business logic for the implementation of the business blockchain application.

* Smart contract
* Business logic implementation

**SUMMARY**

To summarize,

* Hyperledger refers to the project under Linux Foundation and
* Hyperledger Fabric is a **distributed ledger technology framework**, which is one of the projects under the Hyperledger Initiative.
* Hyperledger Fabric is built ground up for creating business blockchain applications.
* There are four aspects that make it suitable for business blockchain applications.
* It supports Permissioning, it supports confidential transactions.
* There is no need for cryptocurrency for incentivization as well as for the transaction execution.
* And you can build business logic and chain code

**Hyperledger Fabric Architecture and Components:**

All blockchain technologies have this concept of nodes.

Nodes connect to each other to form the blockchain network.

In public, blockchain networks such as Ethereum and Bitcoin, all nodes are equal,

but in the case **of Hyperledger Fabric**, all nodes may not be equal in the Hyperledger fabric distributed ledger technology or blockchain technology.

There is a concept of members. **Members=single Legal entities**

Members are legally separate entities that join the blockchain network to transact.

These members share one or more distributed ledgers and each of these members host nodes.

These nodes are used for:

* submitting the transactions and
* for managing the state of the ledger within the organization.

Sometime nodes are referred to as the communication entities of the blockchain.

Each node is assigned an identity by way of certificates.

Even the users in the blockchain ecosystem built on Hyperledger fabric are assigned a certificate.

**Ledgers are used for managing the state of assets.**

These assets in the Hyperledger Fabric Network may represent anything in the real world, such as bonds, loans, title to a house, or it can be anything that can be represented digitally.

Nodes initiate the transactions which lead to the execution of the chain code, and the transactions are recorded in the ledger in the order they are received.

These chain code executions also lead to the change of the state of the assets.

There is **one ledger per channel**, so channels are a way by which transaction privacy is achieved in the Hyperledger Fabric Network.

**Assets**

**Assets**

Manage asset state

**Assets**

**Chaincode**



**Node Ledger**

**Ledger**

**Node Ledger**



**Node Ledger**



Manage Transactions

all of the nodes and users in the Hyperledger Fabric Network are assigned an identity, and that is done by way of the membership service Providers and Certification Authority.

There are three types of nodes in the Hyperledger Fabric Network,

1. Order nodes leader peer
2. peer nodes and



1. client nodes Anchor peer

peers may be tagged a role.Peer can be a leader, peer or an anchor peer.

a peer that has not been tagged with the leader or the anchor role may be referred to as the regular peer.

**order nodes**

* Provide the communication channel for the fabric network.
* The order nodes are responsible for creation of the blocks and ordering of the transactions
* They are also known as the ordering service.
* The **primary responsibility** of the order nodes **is to ensure a consistent state of the ledger** across the blockchain network.
* They provide a **consensus mechanism** and
* make sure that the order of the transactions is maintained.
* Order creates the blocks and guarantees atomic delivery of these blocks to the peer nodes in the network.

**Order is implemented with message-oriented middleware.**

Single node=Good for development| dev mode

Single point of failure

Solo, which is a messaging component, is built into the order node.

And if you use solo, you can have only a single order instance, which is okay for development, but obviously will lead to a single point of failure in production, which is not the way to go.

In a live network, you would use the **raft consensus.**

The raft consensus mechanism is built into the order of binary like solo.

Another option for setting up the ordering service in production is the **Kafka** setup.

But the challenge with Kafka is that you need to set up the Kafka brokers across the multiple member organizations, and this adds complexity in the network and a number of additional moving parts.

So at this point, the recommendation is to go with Raft in a live network.

Each of the pier instance in a network maintains its own copy of the ledger.

There are two parts in this ledger.

1. transaction log
2. the state database.

Both of these are implemented by default using Leveldb, which is an in-process database.

**Transaction log is immutable:** What that means is that once a transaction is added to the transaction log, it can neither be deletednor it can be changed.

Peer may be configured to use Couchdb for state database.That way you can run complex queries against the state data maintained for the chain code.

**ledger**

State Database

Transaction log

Trn log

log is immutable levelDB->CouchDB

This is a logical representation of a peer.

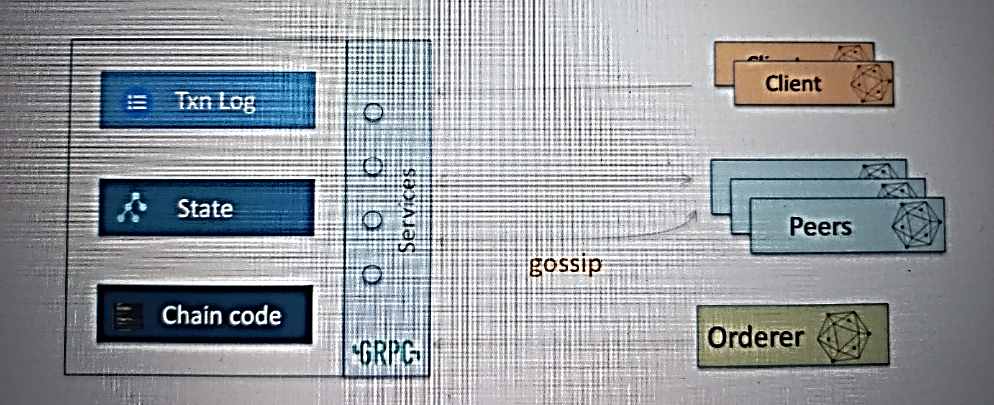
**A peer node maintains:**

* the transaction log,
* the state database, and
* it also manages the chain code deployed to it.

The peer node exposes services and these services are built on gRPC. These services are invoked by the clients, by the other peers and by the orderer for sending blocks to the peer.

peers, exchange block data by way of gossip, data dissemination protocol.

So here I have introduced two terms **gossip and gRPC.**





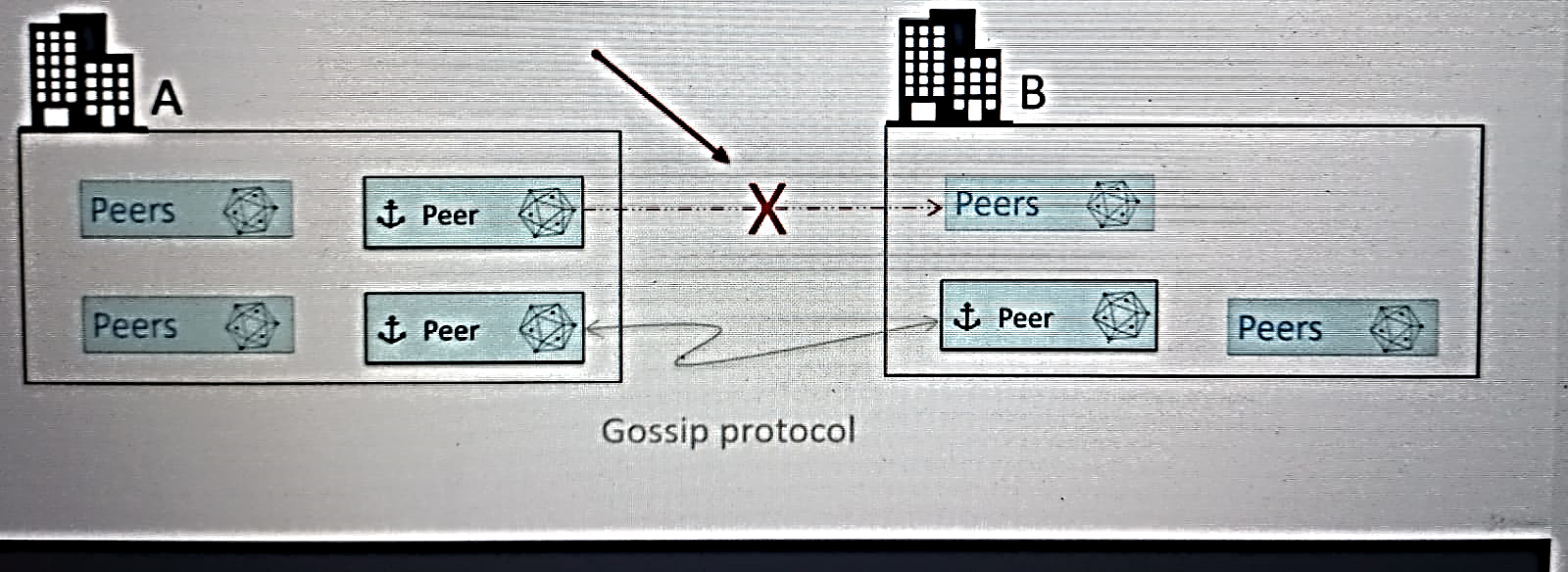
**Anchor peers are known outside organization:**

An organization may set up multiple peers for scalability, isolation and performance reasons, but only the anchor peers are known outside the organization.

In other words, they are the only peer instances within an organization that are **discoverable**.

**Each organization must have at least one anchor peer.**

Here is an example



* two organization Organization A that has two anchor peers and organization B, which has one anchor peer.
* The anchor peer is an organization A and B are discoverable.
* As a result, the peers may engage in gossip, protocol based data dissemination protocol.
* But since the regular peers in Organization B are not discoverable by the anchor peer in organization, a such connection between the two peers is not possible.

**Leader peer receive Blocks from orderers**

* Not all peers in the network connect directly to the order for receiving the blocks.
* Only the leader peers in the organization receive blocks from the orders.

Regular peer is tagged as a leader Either statically or the administrator of the organization can set up the nodes in the organization to dynamically elect a leader. Leadership assignment is at a channel level.

The idea is that a peer node can join multiple channels, so this assignment is done at the channel level.

Here is how the infrastructure for an organization may look like.

There will be an anchor peer, a leader peer, which may be static or dynamically assigned, and the there will be a set of regular peers.

Leader Peer will receive the new blocks from the order and then by way of the gossip data dissemination protocol, send it to the other peers within the organization.

Client Node acts on behalf of the end user.

They are created by way of SQS, such as the Golang SDK for Hyperledger Fabric or the Node SDK.

The client node is also known as the submitting client.

**Client node are the one that submit the transaction requests to the network.**

Channels are a way by which transaction privacy is achieved in the Hyperledger Fabric based blockchain network.

All transactions in a channel are isolated. In other words, transactions created on that channel are available only to the members of that channel.

**Chaincode is deployed to a channel, not to the network.**

There is a special system channel known as the Ordering System Channel. It is also referred to as the bootstrap channel. This channel gets automatically created at the time of the network initialization peers and hence the organizations may join multiple channels.

* Ordering system channel= Bootstrap Channel
* Organizations| peers may join multiple channels

Each of these channels has its own distributed ledger and state database.

Here's an example setup.

Let's say there are five members in a blockchain network and they are part of a common channel. This common channel will have its own instance of the ledger and the chain code instances.

Let's say Organization C(seller) and E would like to transact privately.

In other words, they don't want their transactions to be visible to the rest of the members in the network.

In that case, they can create a separate channel for which there will be a isolated, dedicated ledgerand chain code instances not available to the members A, B and D.

Here's another example where the organization's B, C, and E are part of a channel.And then there is another channel in which A, B, C, D are members.

Notice that B and C are part of both the channels.

As a result, they will have visibility to the transactions on both channels.

Fabric uses **PKI(public key infrastructure)** for identity management.

It follows the typical process used for identity management by way of certification authority. Hyperledger fabric has an implementation of the certification authority CA server.

The server exposes services for

* identity registration,
* identity enrollment and
* certificate management.

**SUMMARY**

* I covered four topics peers, clients, channels, and fabric.
* The Hyperledger infrastructure in an organization has multiple type of peers.

**The regular peers, The leader Peers and the anchor peers.**

* Anchor peers are discoverable outside of the organization.
* As a result, gossip protocol based data dissemination can take place between the anchor peers belonging to different organizations.
* Leaders are the only peers that receive new blocks from the orderer.
* Once the leader receives the block, it send those blocks to the rest of the peers in the organization.
* By way of the gossip, clients submit the transactions to the network.
* **Channels** are a way by which transaction privacy is achieved in the Hyperledger Fabric Network.
* Peers in the organization may join multiple channels.
* Fabric provides a implementation known as the server Services exposed by the server are used for identity management.

**Hyperledger Fabric Software(Binaries and Configuration)**

This will cover two things the

1. **that Hyperledger Fabric software has on external components and software.**
2. **Overview of the binaries that are part of the Hyperledger Fabric Architecture.**

HF is open source project written in Go language

Dependencies-> docker

Docker is used by Hyperledger fabric in three ways.

1. **Launched in isolated containers:** at runtime, the deployed chain code on the Hyperledger Fabric Network is launched in isolated Docker containers.
2. **Components available as containers:** architectural components of Hyperledger fabric, such as peer and orderer, are made available as Docker images.
3. **Container Orchestration:**The third one is that Docker Compose may be used for container container orchestration for setting up Hyperledger fabric infrastructures.

Some of the binaries in Hyperledger Fabric already have a **built in database** which is based on the leveldb. Since it's a in-process database, there is no separate server or software required for leveldb.

But if you are switching the state database to Couchdb, in that case you would need to install Couchdband make it available to the Hyperledger fabric binary.(optional)

Hyperledger fabric software components may be categorized under three headings.

1. core components:

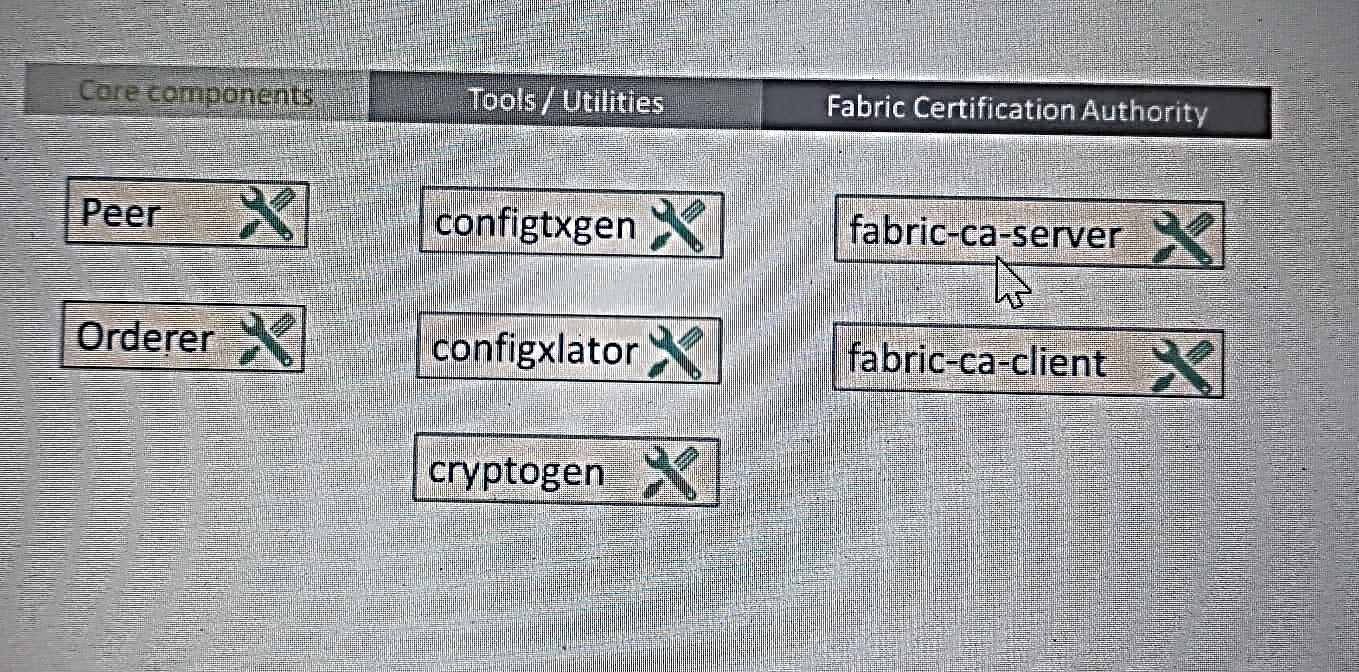
* peer
* order.

1. tools and utilities:

* config, gen config, translator and crypto gen tool.

1. Fabric Certification Authority, which is composed of the

* fabric core serverbinary
* fabric CA client binary.



Almost all of these binaries require configuration information to be provided in Yaml format.This is the standard for describing the configuration that the Hyperledger fabric developers have decided to adopt.

there are additional tools and utilities available for Hyperledger fabric, but these components are the absolutely required set of Hyperledger fabric binaries for setting up Hyperledger fabric infrastructure for experimentation as well as production.

**overview of all the binaries:**

**Core Components:**

1. **PEER Binary:**

The peer binary is used in two ways.

1. when it is launched as a process, it becomes a node in the network and it may be tagged as an anchor peer or as the leader.

Launches as process=node in network

May be tagged as Anchor|Leader

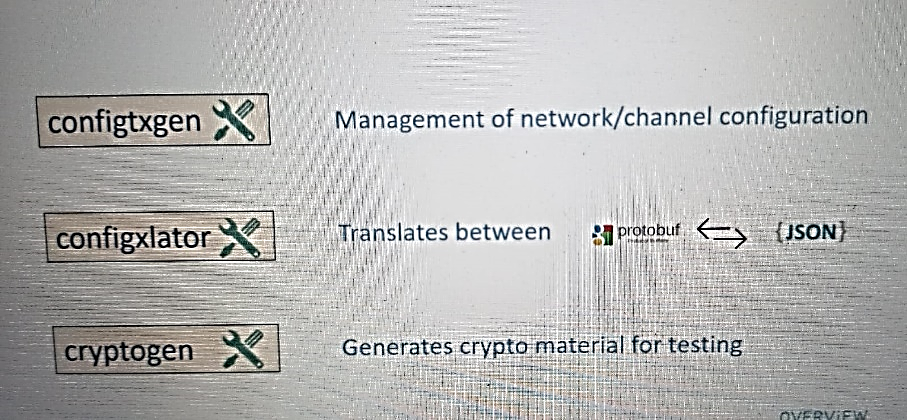
1. it is used as a tool to manage network and channel configurations.
2. **Orderer Binary:**

Orderer Binary can be set up to use **solo** for messaging.

Solo is built into the order and then order Binary can also be configured to use the Kafka cluster.

As mentioned earlier, both peer and the order binaries expect configuration files in Yaml format.

**Tool/Utilities:**



1. The **config gen** tool is used for managing the network and channel configuration.

This tool also requires the configuration information to be available in Yaml format.

1. **Config translator** tool is used for translation between the protocol buffer format and Json format.

Now protocol buffer format is a binary format that is used by gRPC.

1. Third tool is the **crypto gen** tool.It generates the crypto material for testing very useful tool that we will be using for creating test infrastructures.

**Fabric Certification authority:**

1. **Fabric-ca-server binary:**

This binary is launched as a server as a standalone process that exposes services for managing identities and certificates.

1. **Fabric-ca-server binary:**

The fabric client binary is a command line interface tool that is used for executing services on the fabric server.

So this is the tool that we will use for managing identities and certificates.

**SUMMARY**

* discussed the dependencies that Hyperledger Fabric software has on other components or external components and software.
* provided a quick overview of these binaries which are absolutely required for setting up Hyperledger fabric infrastructure for experimentation as well as for production.
* Most of these binaries require configuration to be provided in Yaml format.

**Hyperledger Fabric Distribution and Infrastructure options:**

First, I'll discuss **the two infrastructure setup options for Hyperledger Fabric**,

There are two options for setting up the Hyperledger Fabric infrastructure.

1. **Container based setup**: The first option is based on Docker containers.
2. **Native setup:** The second option is to **go native.** That is, install all the binaries in the VMs to set up the infrastructure.
3. **Container based setup**:

In the container based setup, you will use **Docker images** available for Hyperledger Fabric.

check out the public Docker image repository at hub.docker.com to browse through the various Hyperledger fabric images.

The container instances may be hosted in shared or dedicated virtual machines.

Recall that the Build Your first Network tutorial uses a single virtual machine to host all of the components, but for obvious reasons, you would not like to do that in case of a production environment There you will have to break apart the various container instances into separate virtual machines.

Typically you would use the Docker compose file for the orchestration of the containers.

So the way the infrastructure will look like is you would have the Docker container instances for the order.

Then you will have one or more containers for peers.

And then you would also need to have fabric server container instance for managing the identities.

One common question that comes up at this point is can I use Kubernetes?

Yes, you can use Kubernetes to achieve **high availability** in a production environment built on Docker containers for Hyperledger fabric.

1. **Native binary install**

* the binaries for Hyperledger fabric are installed directly on the virtual machine.
* You may use orchestration, tools and scripts.

The infrastructure setup would require you to procure multiple virtual machines.

You will install the order.

Then you would have peers installed on one or more virtual machines, and then you would have the fabric server instances on their own virtual machines.

**Order-> peer-> fabric ca server**

Here is a side by side comparison of the two options( container based setup docker and native binary install)

|  |  |
| --- | --- |
| **container based setup** | **native binary install** |
| is carried out by way of Docker compose.  If you are isolating the various components or installing on multiple virtual machines, you will be creating multiple Docker compose files | You would have to use the scripts and DevOps tooling for the installation of binaries on the virtual machine. |
| faster to set up Docker based environment | the installation of binaries on virtual machine is a time consuming process. If you're not using scripting and DevOps tooling, |
| hides a lot of details and is a little bit complex for people who don't have a good understanding of Docker, Docker Compose and other Docker related tools. | Direct binary install on VM is simple to understand irrespective of which way you go, you still have to understand how the Hyperledger fabric binaries are configured and how do they work. |

You’ll learn:

1. how the binaries fit in the overall Hyperledger fabric architecture, how they are implemented, what are the dependencies that these binaries have?
2. And then you will learn how to configure these binaries for your specific use.
3. trying out multiple infrastructure configurations using the native

binary install.

* We will be configuring these infrastructure on guest VMs on your development machine.
* Then you will also configure the native infrastructure on a cloud platform such as AWS

1. with the knowledge of the binaries as well as the knowledge of Docker toolset, you will be in a position to configure and set up Docker container based Hyperledger infrastructure.

Overall you’ll be learning:

1. all the Hyperledger fabric binaries that you need for setting up Hyperledger fabric
2. infrastructure.
3. how to set up the network configurations for different components using the Yaml files.
4. You will try out a number of these network configurations by building small infrastructure footprints on your machine and on the cloud.

Blockchain networks are typically owned by consortiums, a group of organizations that work together to create and maintain the network.

At any point when there is a need to make changes to the network, members of the blockchain network collaborate to decide if the changes should go ahead or not.

This configure operate process involves a very complex workflow and it requires you to understand all the tools that are needed for making these configuration changes.

You will see that I am using **recipes and exercises**.

Recipes will show you how to carry out common infrastructure level tasks on Hyperledger fabric and the exercises are for you to carry out.

Now let's talk about what is not covered in this course.

This course is not on Hyperledger fabric development, so you will not learn how to write chain code.