**Hyperledger Fabric A DLT for Business Applications**

Distributed Ledger technologies for business applications In this lecture I ll talk about the characteristics

that are desired in a distributed Ledger technology for creating business applications

At the end of this lecture you will understand at a high level why hyperledger fabric is suitable for

creating distributed ledger technology based business applications

Hyperledger fabric is a distributed Ledger technology for the business

The keyword here is the business that differentiated the hyperledger fabric from other Blockchain technologies

that are geared more toward the public domain Two such dominant public domain distributed technology networks are

the Ethereum and the Bitcoin networks Business application would require the distributed ledger system to

have certain characteristics which are very different from the characteristics desired in a public domain

distributed Ledger technology

There are four characteristics that makes hyper ledger fabric suitable for implementing DLT based business

applications Hyperledger fabric is a permission network It supports confidential transactions and

to participate one does not need cryptocurrency and it is programmable With these characteristics

hyperledger fabric establishes trust transparency and accountability among the participants in the network

Hyperledger fabric allows businesses to create permission network The hyperledger fabric provides ways

by which owners of the network can restrict who can access and do what on the network

It requires participants in the network to be known and to join the network

the participants have to get permissions from some authority

Compare this with the public DLT platform such a ethereum where anyone can participate on the

network as it is a permission less network Hyperledger fabric provide ways by which the rules are assigned

to the participant and actions that can be taken by each of those roles are restricted by way of access

control list Transactions are validated by a known set of validators that the participants trust since

all participants are known in a business setting

It is easy to identify or establish those trust authorities or transaction validators

Compare this with a public network where everyone is anonymous and there is lack of trust among the

participants

That is why in those public networks such as Ethereum resource intensive validation schemes are used

Another characteristics that is desired for business applications is the confidentiality of the transactions

Not all transactions are desired by the business to be visible to all Hyperledger fabric puts participants

in control or the visibility of their transactions

Consider the scenario where A B and C are engaged in carrying out business on a DLT based application

So any transaction initiated by any of these entities will be visible to all of these participants

Now if B and C would like to engage in some kind of business activity that requires them to restrict

the visibility of the transaction to only two of them then they can create a private channel to carry

out such business Any transaction initiated by B or C on this private channel will be visible only to

be B and C Business using hyperledger fabric can create and participate on multiple such private channels

or networks Hyper ledger fabric does not have any concept of crypto currencies

Let me explain to you why it s not needed by comparing it with the public distributed ledger platform such as ethereum

Now ethereum uses cryptocurrency to incentivize the distributed ledger network Transactions on ethereum are

validated by miners who get paid in cryptocurrencies that they can exchange for the fee This kind

of transaction validation scheme is not needed in case of a business DLT application

In other words there is no need to incentivize the network using crypto currencies Hyperledger fabric allows

participants to decide on who the validators would be and what kind of policies will be used for transaction

validation

Typical public domain transaction validation schemes such as proof of work which is very resource intensive

is not applicable and is not needed for distributed ledger business applications

Hyper Ledger fabric is programmable by way of the construct called the chain code Conceptually chain

code is the same as the smart contract

on other distributed ledger technologies Businesses can use the chain code to automate the business processes Chain code

sits next to the ledger and participant of the network can execute the chain code in the context of

a transaction that gets recorded in the ledger Automation of business process by way of chain code

leads to higher efficiency transparency and greater trust among the participants

Let s summarize There are four characteristics that make hyper ledger fabric ideal for building distributed

ledger based business applications Hyper Ledger fabric allows participants to create permission distributed

ledger technology based business applications

Participants may leverage the private channels for managing confidentiality of the transactions

That is transaction visibility is restricted to select parties in the network

Now unlike public networks hyperledger fabric business applications do not require mining and expensive

computations for transaction validation

There is no concept of cryptocurrency in hyperledger fabric The chain code construct available on hyper ledger

fabric can be used for automation of the business processes and for building business logic The

chain code is conceptually the same as the smart contract on other distributed ledger technology platforms

**\*\*\*\*\*\*\*\*\*\*\*\*\*Assets Chaincode Ledger**

Hyperledger concepts In this lecture I will cover assets chaincode and ledger At the end of this lecture

you should be able to explain the relationship between assets chaincode and ledger As discussed in

the lecture on the distributed ledgers assets represents some kind of value that can be exchanged on the

blockchain systems Any object of value in the real world may be represented as an asset on hyperledger fabric

as long as it can be represented digitally On hyperledger fabric the asset representation maybe JSON or in

binary format

For example a simplified representation of the car will have two attributes in the JSON representation

the VIN number field that uniquely identifies the car and the owner of the car

The second field that is the owner can change as a result of a sale of the car

In effect we are saying that the state of the asset may change over time These state changes can take

place on hyperledger fabric only by way of well defined transactions that are coded in chaincode Chaincode

defines the structure of the asset

It also defines the transactions that can be executed against the asset

It has all of the business logic needed for the transaction

In the case of the example of a car there can be a function sell the car defined in the chain code

and call to this function will lead to the transfer of ownership of specific car to the new owner

All transactions are recorded in a ledger Ledger is a data structure that keeps track of all of these

transactions

It also records the state changes taking place in the assets as a result of execution of these transactions

And you already know that the ledger and hyper ledger fabric is distributed

That is

all participants have a replica a copy of the ledger

Let s summarize In this lecture I introduced you to the concept of assets chain code and ledger Assets

represent anything of value in the physical world

Chain code is used for defining the structure of the asset which can be binary or JSON representation

Chain code is also used for coding the transactions that can be executed against the assets When the

chain code is executed

lt leads to the addition of the transaction information in the ledger The ledger

also keeps track of the state of the asset

**\*\*\*\*\*Permissioned Network Members Membership Service Provider**

Hyperledger concepts Fabric is a permission network

What that means is that there is a need to assign identities to the participants in the network

In this lecture at a high level you will learn about how the identities are created and managed on hyperledger

fabric by way of membership service provider component

Businesses deal with known entities

Businesses have B B partners for example suppliers of raw material or purchaser of goods

In some industries by law businesses are supposed to interact with only known entities

For example banking industry Banks must know the identity of every single customer it has

Then there are regulatory agencies that interact with the businesses A distributed ledger technology

based applications for businesses in effect would require support for managing identities on the distributed

ledger network

Now you have already learnt that hyper ledger fabric supports permission blockchain networks

Let s go a little deeper into what it means

It means that unlike public networks such as ethereum anonymous access to blockchain applications

built on hyperledger is not allowed

Business Application defines the roles that are assigned to the participant and access is granted or

restricted by way of these roles An abstract service referred to as the membership service provider

takes care of generating the credentials for the various participants

More on the service in a little while

Let s define the term member In the context of hyperledger member refers to a legally separate or independent

entity

Here A B and C are legally separate and hence defined as the members of the blockchain network

In case of our car example these three entities may be the car manufacturer the car dealer and the

repair shop each of which are legally independent entities

Identity in the hyper ledger network are managed by way of X certificates

When a participant identity is created the certificate is issued to the participant

Anytime a transaction is initiated by the participant certificates private keys used for signing the

transaction and any component in the network can validate be authenticity of the transaction

by using the participant public key Interestingly on hyperledger it is not only the participants that

are issued the certificate

Even the infrastructure components are assigned an identity by way of certificates

This is to prevent a scenario where hackers can add a server to the network for example to disrupt the

network or to make an attempt to manipulate the transactions

In effect every single infrastructure component in the hyperledger network must have a valid certificate

to become part of the network

Members are legally separate entities and so even they are assign an identity by way of a certificate

Certificates follow the typical process of issuance and the revocation by the certification authorities

in the network Members can manage the identities within their organization This aspect will remove the

dependency on a single centralized certification authority

And this is achieved by way of implementing the concept of membership service providers where the member

can use their certificate to create new valid identities that can participate on the network

So in other words a member can create new participants certificate that associate the participants

with the organization or the member by virtue of the certificate chain Also member can create the certificate

for that infrastructure component As a result the hyper ledger fabric network can have one or more memberships

service provider components Hyperledger is a permission network All entities participating on the

network are known and have an identity which is assigned by way of X certificates Certificates are

issued to all participants infrastructure components and members Members are legally separate entities

These are the organizations that have decided to adopt blockchain for process automation

Each of these members are assigned a certificate and depending on their authority they may be able to

use an MSP to create participant and infrastructure component identity within their organization

Blockchain network can have one or more MSPs as shown in this illustration

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Nodes and Channels**

Hyperledger Concepts At the end of this lecture you should be able to explain what is a node and what types

of nodes are there in hyperledger fabric

Also you should be able to explain the concept of channels The concept of Node is common in all blockchain

technologies

Think of node as a communication end point in a blockchain network Nodes connect to other node and that is how

blockchain network is formed

Nodes use some kind of peer to peer protocol for keeping the distributed ledger in sync across the network

In public blockchain networks such as Ethereum and Bitcoins all nodes are equal and the network looks

like this

in case of Ethereum To participate in these public networks one just needs to download the node software

generally called Wallet Create an account and Execute the node

Things at hyperledger are very different

Let s talk about nodes in the context of hyperledger Nodes are the communication entities of the blockchain

Nodes need valid certificates to be able to communicate with the network and the participants

use the apps that connect to the network by way of the nodes Participants

identity is not the same as the nodes identity When the participant execute or invoke a transaction

participants certificate is used for signing that transaction Nodes certificate is used by the network

to check if they should trust the node or not

Let s say for the sake of discussion that this node s certificate is either revoked or has expired

In that case the transaction tho signed by a valid certificate held by the participant is broadcasted

to the network but the transaction will be rejected because the certificate that node is using has expired

or has been revoked In hyperledger fabric

unlike the public domain Blockchain technologies such as Ethereum and Bitcoin all nodes are not equal

There are three distinct type of nodes

First one is the client node

This is the node that applications use for initiating the transactions Peers are the nodes that keep

the ledger in sync across the network Orderers

are the communication backbone for the blockchain network They are responsible for the distribution

of transactions

Members can participate on multiple hyperledger blockchain networks The transaction in each network

is isolated and this is made possible by way of what is referred to as the channel Peers connect with

the channels and they can receive all the transactions that are getting broadcasted on that channel

The channel has its own independent ledger

In other words if there are two channels there are two different ledgers maintained in each of these

channels and there is no visibility for a peer connected to one channel into the ledger of another

channel

Consider this example where there are five members and they have decided to launch a blockchain network

Here as you can see there is a single channel and there is a ledger and the chain code that is available

on that channel to all five members

Now let s say C and E decided to have some kind of deal

where they want their transactions to be private

So what they can do is they can create a private channel

So in this scenario A B and D are connected to the common channel whereas C and E are also part of that

common channel Along with that they have created this private channel The ledger and chain code

for the private channel is independent and isolated from the ledger and chain code for the common

channel

Here s another scenario let s say B C and E have decided to launch a blockchain network and all their

transactions are visible only to B C and E

And then A B C and D also decide to launch a network on which they carry out certain specific kind of transactions

So in this scenario there is no common channel across the members

There are two separate channels used for maybe different type of transactions

And on one channel the visibility of the transactions is restricted to A B C and D whereas on the other channel

the visibility of transactions is restricted to B C and E B and C are the members that are participating

in both channels so they have visibility into both the ledgers in these two channels

Let s summarize In this lecture I talked about the Node and the Channel In the hyperledger fabric all

nodes are not equal

This is different from the public network where all nodes are equal

There are three different type of nodes in hyperledger fabric First one is the client that is used

by the participant to initiate transaction

Second one is the peers that take care of keeping the ledger in sync with the network

And the third one is orderer which provides the communication backbone for the blockchain network Members

can participate on multiple hyperledger blockchain network by way of private channels

Each channel manages its own independent and isolated ledger

**\*\*\*\*\*\*\*\*\*\*\*\*\*Hyperledger Fabric Composer Overview**

In this lecture I ll introduce you to the fabric composer

What is fabric composer Why you should consider using fabric composer

And then I ll give you a high level overview of the various tools and runtime environment that makes

up the fabric composer

I ll also give you a high level overview of the process used for creating business network application

using the composer toolsets

Let s begin with what is composer The composer is a hyperledger open development tool that makes it easy

for teams to create and manage business network applications that are deployed on the hypeledger

technologies As a primary goal and I ll quote the composer dev team

The primary goal is to accelerate the development of blockchain applications on hyperledger

In fact apart from reduce time to market there are other benefits of using composer

It hides the complexity of the underlying infrastructure

Composer also offers business modeling capability by way of modeling language that can easily be used

by non technical team members such as Business Analyst The smart contracts or transactions maybe coded

in javascript which most developers are familiar with

So it becomes easy to write and manage smart contracts The composer development toolset consist of multiple

tools geared towards the developers architects and the operators of the network

These are the folks who need visibility into the network

Then there are tools for the administrators

Those are the folks who would manage the policies on the network or create participant identities on

the network

And then there are tools for the business analyst The developer and the Business Analyst create the

business network applications that get deployed on a composer run time

There are multiple types of composer run times Let me explain to you the development process at

a high level

The domain expert such as the business analyst uses the composer modeling language to create the business network

model The composer modeling language is an object oriented language for defining the domain model for

the business network The developer takes the business network model and codes the transaction specification

in the business network model in javascript to create a final application that consist of the Javascript

based transactions and the business domain model in the composer modeling language The administrator

uses a composer tool to deploy the business network application to the execution runtime The execution

runtime is based on the hyperledger fabric x

And this is the blockchain network on which the application gets deployed

The operator uses the tools to maintain good health of the application on the business network

Now this is just one of the execution runtime

There are two other execution run times which are used by the business analyst and the developers

The second type of runtime environment is referred to as the playground

It is primarily used by the business domain expert and the developers

The playground is available as a web application so there is a web UI that is used by the domain

experts and the developers to create the business network model The business network model created by

way of the playground is stored in the browser local storage

The other purpose for the playground is for simulated testing of the business network application

The third type of execution runtime environment is referred to as the embedded environment

The idea of embedded environment is that the developers can code the business network application and

then deployed in a node based embedded simulator to test out the application

All of the execution is carried out in memory Primarily the embedded environment enables test driven

development and unit testing

Let s summarize In this lecture I introduced the fabric composer

It is an open development tool

The primary goal is to reduce the time for development of the business network application

It offers multiple tools which are geared towards the domain experts developers administrators and

the operators of the network The composer domain modeling language can be used by domain experts to

create the domain model for their applications

Composer toolset also offers three types of execution runtime environments for the business

network applications The hyperledger fabric x infrastructure

think of it as the production or the live network environment Playground which is the web based application

that allows the development of domain models and it also allows simulation based testing of business

network application

The third type is Node based simulator

referred to as the embedded runtime It is primarily used for test driven development and unit

testing

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ledger Implementation**

Ledger implementation Hyperledger fabric is a distributed ledger technology

All peers in the network have a copy replica of the ledger Ledger has two parts transaction log and

state database At the end of this lecture

You should be able to explain what is contained in the transaction log and what is contained in the

state database as well as how these two components are implemented in fabric

In last section I touched on the topic of ledger In fabric

there are two parts to the ledger Transaction log that keeps track of all the transactions invoked against the assets

And then there is the state data The state data is representation of the current state of the asset at any point in time

As in the example of car let s say dealer sells the car to person A transaction is added to the log and state data reflects that A is the owner of the car

Now let s say person A sold the car to B The state data reflects the new owner as B And let s say again B sells to C the state data reflects

C as the owner The transaction log is immutable But the state data is not immutable

This is sometime a source of confusion Now let s discuss what I mean by these statements

You must have heard about the term CRUD create retrieve update and delete

Let s see how CRUD applies to transaction log and state data

You can create transaction records in the transaction log

You can retrieve the existing transaction records from the transaction log but you cannot update an

existing transaction record which is there in the log

And you can not delete any of the transactions which have been added to the log From the state data

perspective

you can carry out any of these operations on the state data for an asset

Now let s talk about how new transactions are created and how the CRUD operations are carried out on

the state data

It all happens by way of the execution of the chain code When the chain code is executed

It leads to the creation of transactions in the transaction log

And at the same time depending on the code in the chain code there may be a change in the state data

So in other words the CRUD operations are implemented in the code for the chain code The transaction

log is implemented using the levelDB LevelDB is a lightweight library for building key value data store

It is used in embedded manner as part of the fabric peer implementation

That means that the levelDB is not launched as a separate process but it is part of the peer process

It is a queryable and highly efficient implementation for insertion of data or creation of data

Peers write the transactions to the level database

Now keep in mind that you cannot replace level database with any other

It is a fixed implementation from the fabric peer implementation perspective

The state data consists of the key value pairs that are version

Let me explain what I mean by that

The asset state is managed in storage variables identified by the keys for example

Key owner

The value is represented by way of arbitrary blobs or binary objects

These blobs maybe in JSON format

Apart from the key value the state data also has a version When the state data is updated the existing

data is not over written

which would cause the old data to be lost

Instead a new version is created for the key value pair and placed in the state database

It is an example let s say the car has two attributes VIN number and the owner information

This is the version one of the cars state data

Now let s say the car is sold by the current owner so the owner data in the state is updated

So at this point the version one of the data is there and it s not over written A version of the state data

is created for the owner and retained in the state database

Same way a version will be created the next time a transaction is executed that updates the owner information

State data can have this representation state name of the key and then the version and value for that

key

The start data by default is managed in LevelDB which is the embedded database

like I explained in the case of transaction log The data will look like this for our car example The

VIN number which never changed

stays at version and then ownership of the car has changed over a period time

So the version was when the car was created

by the manufacturer and then the version is when the ownership of the car was transferred to the dealership

Let s call the dealership Adam dealership and then when the dealership sold the car to John

Then there is a version created in the levelDB

One thing to keep in mind is that chain code owns the data for these keys so the identification of a

specific key is not just the key name but the chain code name and the key name

So in other words the access to the data is restricted to the owner chain code and if there s another

chain code that tries to access it it won t be able to do that

Now this is how it s implemented today

in version

This may change in future

where in the other chain code

maybe allowed to access the chain code data that they don t own

Let s talk about the data held in the transaction log

And the state data Both transaction log and state database by default

use levelDB and levelDB supports such simple queries

Now if these simple queries are going against the transaction log that should be OK because you re not

going to go for complex queries against the transaction logs

The only thing that you have to keep in mind is that when you are querying the transaction log you ll

be querying it against the peer because peer has the levelDB embedded within the peer process

Businesses depend on reports and business intelligence and insights that they gather from the data to

make their decisions With simple queries that are available in levelDB

Businesses will be constrained because they need to write some complex queries to generate these reports

and insights

Unfortunately levelDB will not work in those cases

The good news is that the hyperledger fabric team understands this issue and what they allow you to

do is switch the levelDB with a more mature database with more flexibility around the querying

capabilities

So the state data base is pluggable at the peer level By default it is levelDB which support a simple

query for key value pairs but you can replace it with couch database which is a nosql database that

allows you to execute complex queries and all of this is done by the configuration of the peer

Let s summarize In this lecture

I talked about the ledger Ledger has two parts

The transaction log and the state database The ledger is managed on the peer node The peers receive

the data for the transactions and they update the transaction log and the state database Both transaction log

and the state database are implemented by way of levelDB which is a key value database Transaction

log tracks the transactions in the order they are received Transaction log is immutable

In other words once the transaction is added to the transaction log it can neither be updated nor be

deleted but it can be retrieved Compare it with state data State data can be manipulated by way of the CRUD

operations that can be coded in the chain code State data is versioned

So what that means is any time you update the data or delete the data managed as state data for the asset

a new version is created

Businesses need to create reports and generate insight from the state data and that requires execution

of complex queries

Unfortunately levelDB does not provide the capability for creating and executing complex queries

But the good news is that you can replace the state database which is implemented in levelDB with

couchDB which supports creation of complex queries

This switching of database is carried out by way of configurations on the peer

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Dev Environment Walkthrough Peer CouchDB setup**

Development fabric tools setup In this lecture I ll give you a quick walkthru of the peer and the couchDB

set up for the dev environment setup The peer image already has the peer binary installed When the peer

is started it reads the environment variables for configuration

Here it s configured to use the levelDB for transaction data and the peer is set up to use an instance

of couchDB for state management

The container for Koshti has the binary installed for always DV and it also has the configuration some

of the configuration parameters are read for CouchDB from the environment

Let s take a look at the configuration setup for peer and couchDB All of the container configuration

for the dev topology or the dev fabric tools environment is done in the docker composed yml file under

the composer subfolder or hlfv or hlfv

I have opened the docker compose yml file under the hlfv

Let s look at how the peer is configured The container name is

peer org example com

These are all the environment variables which are made available to the peer binary

These two variables control the logging level for various modules within the peer binary

So you can control the logging level for the chain code by changing this debug from debug to for example

error or info

This variable controls whether the peer would use the levelDB or couchDB for managing the state data

This peer is set up to use the couchDB instance The instance information is made available to the

peer by way of this environment variable When the container is launched

this is the command that gets executed peer node start

And this is where the peer instances are started in the container

These two ports are exposed to the local host or local machine

All of the configuration information is managed under the composer subdirectory of the fabric tools project

That configuration is made available to the components within the container

by way of these volume mappings As discussed earlier the peer binary is already installed in the container

and you will in fact be able to run some of the commands available on this peer binary Just to give you an

idea

These are the commands available on peer binary

For example you can get and set the logging levels by using the peer logging command

You can get the status of the node by peer node status get the version using a peer version command

Let s logon to the container and try out some of these commands

Let s attach a shell to the running container right click attach shell type env to check the environment

This is the environment that we have set up in the docker compose file

You can check the status of the peer by running peer node status and this is the current status

Let s list the channel by using the peer channel list and as you can see this peer has joined the composer

channel Do a ls la here

You will find the composerchannel block that has the channel configuration

this peer is using

Let s take a look at the couchDB now Right click on the couchDB container and show logs and you

will see a message here What this message is saying is that the couchDB instances running in admin

party mode has no security and anyone with a url can connect to it

Let s connect to it

If you are on a Mac or Ubuntu you can simply use local host

But since I am on Windows I ll have to use the IP address of the virtual machine and here couchDB welcome

version

This basically indicates that we are connected to the couchDB instance that is running in the

docker container and the peer instance is connecting to the this couchDB instance for managing the

state data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Peers Nodes Anchors and Endorsers

Peer nodes In the last section I gave you an overview of the three different type of nodes client node

peer node and the orderer node In this lecture the focus is the peer node

In this lecture you ll also learn about two special type of peer nodes the Anchor peers and the Endorser peers

Members in the blockchain network need to set up peers in their infrastructure for participating in the network

Say for example a member A has set up peer nodes

All of these peers need to be configured with the appropriate cryptographic materials such as certificates

and other information Peers in the members organization receive transaction invocation requests from

the clients within the organization As transactions are created in the network and new blocks get generated

These blocks are sent out to the peers by the ordering service and peers receiving these blocks need

to validate and update the ledger managed on the peer node

Inherently this kind of an architectural approach is highly scalable as there is no need for a centralized

effort to scale the network or to scale the infrastructure

Each member organization can look at their needs and set up the infrastructure based on their requirements

So for example the member organization A has three peers whereas B has decided to setup only two

peers because that may be enough for their requirements Member organization can have multiple peers but

not all peers receive the block information from the orderer only the Anchor peer receives the blocks

In this diagram the anchor peer is this one

So what happens is when the anchor peer receives the block it updates the other peers in the member

organization To avoid single point of failure an organization can create a cluster of anchor peers or

more than one anchor peer Anchor peers are set up and defined as part of the channel configuration and

the anchor peers are by default discoverable

Now what that means is that any peer that is marked as anchor peer is discoverable by the order and other

anchor peers Lets talk about the endorser Peers maybe marked as the endorser or they can take up the role

of endorser in which case they are also known as the endorsing peer A client sends the invocation requests

to the endorsing peer On receiving the request for the invocation the endorsing peer validates the transaction

For example it checks if the end user has used a valid certificate or not If the validation checks out

fine

Then it simulates the chain code

What that means is that the endorsing peer executes the chain code

Now after the execution it does not save the state of the chain code back to the block chain as it is

simply simulating the change

And you will understand much better how this process works as part of the transaction flow

So for the time being just think of simulation as execution but the data that has changed in the chain

code is not saved to the ledger at this point

At the end of the endorsement process the endorser either rejects the transaction and this may happen

due to multiple reasons for example the security aspects

didn t check out or the execution failure or the endorser may respond back with an endorsed transaction

request

More on this later

The primary objective of the endorsing peer or the endorser is to protect the network Now when I say protect

the network it doesn t necessarily means protect from intentional attack on the network but it also

means that it needs to protect the network from a misbehaving or mis configured node on the network

And since every member organization is responsible for configuring their own nodes which is peers there

is the possibility that a mis configured node may be unintentionally added to the network

Another advantage of this mechanism is that only the endorser needs to execute the chain code

And this will improve the overall scalability of the infrastructure

Since there is no need for all nodes to execute the chain code The focus of this lecture was on the peer

node Every member organization need to set up peers in the organization

Peers receive blocks from the network

There is a special kind of peer that is set up to receive the blocks and that peer is known as the anchor

Anchor peer receives the blocks and then provides these blocks to the other peers in the organization

Then there is endorser peer or the endorsing peer The endorsing peer receives the transaction requests

from the client On receiving the transaction requests from the client

It simulates the transaction that is the transaction is executed

The chain code is executed but the state of the chain code is not updated in the ledger

The endorser then rejects or accepts the transaction after it has carried out multiple validation check

The primary objective of this endorsement mechanism is to protect the network from intentional as well

as unintentional attacks

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*035 Clients Node Endorsement Policies-en

Client nodes By now you are already aware of the fact that client nodes are responsible for initiating

transactions

In this lecture my focus is on the endorsement policies Endorsement policies are used by the client

the peer and the orderer to ensure that the transactions are valid before they get added to the ledger

across the network

Client is responsible for initiating the transactions and the client does that by creating the transaction

request and sending it to one or more endorsing peer Clients connect to the endorser and sends the

transaction to be endorsed

Now the question that comes to mind is which peer should be used as an endorser

And the answer is It depends

It depends on the chain code The chain code can associate a policy known as the endorsement policy

and this endorsement policy has two components which peers to use as endorsers

So this is how the client comes to know which endorser it should connect to

for getting the transaction endorsed Another part of the endorsement policy is the criteria for valid

transaction

Let s look at the endorsement policy in a little bit more detail

In this example policies we have set up three peers that have been assigned the role of endorsers Clients

must send the transaction request to all of these endorsing peers and then collect the responses In

the second part of the endorsement policy

there is a criteria for transaction validation

This is used by the peers

I ll explain that aspect in a few minutes

But before that let s go over what is in this criteria In the criteria the chain code developer defines

either the number of endorsements needed for checking the validity of the transaction or same thing can

be specified as a percentage

For example the policy may say that two out of three endorsers must endorse the policy to have the transaction

considered as valid

Or it may simply say percent or above is the criteria used for marking the transaction as valid In

combination you may use logical expression with AND or OR or operators to define expressions that are used

for checking the validity of the transaction

Here is an example

The transaction will be considered valid if Org endorser endorses the transaction and if the Org

member has not endorsed the transaction then both or two and or three endorsers should have endorsed

the transaction request for it to be considered valid

Couple of quick points about the endorsement policy Endorsement policies are optional

In other words the chain code is not required to have an endorsement policy The endorsement policy

is specified at the time of deployment of the chain code and obvious question would be if the policy

is not specified then what is the default policy

The default policy is that any one peer from the client organization can endorse the transaction request

At this point you are already aware that peers are responsible for keeping the ledger in sync with the

network

So peers received the transactions that they need to add to the ledger but before the peer adds the transaction

to the ledger

It uses the endorsement policy to check the validity of that transaction

So there are three high Level things that the peer checks for First is are all the endorsements valid

Second thing is check the criteria

Are there enough endorsement for the transaction to be considered valid

Third thing is are the endorsements coming from the right sources

So in this particular scenario are the endorsements coming from peer peer and peer

If there is a failure in any of these three checks

The transaction is considered invalid

So after this check the transaction is marked as failed if it has failed the validation and added to the transaction

log

If the validation has passed then it is marked as successful transaction and added to the transaction

log

Let s summarize In this lecture I covered one of the most important topics of hyperledger fabric

The endorsement policy The endorsement policies are associated with the chain codes

Clients need to send the transaction request to the endorsing peers and the client gets the list of

endorsing peers from the endorsement policy

Client then sends the request to the endorsing peers which validate the transaction request and simulate

the transaction

Once the transaction is validated the client s sends it out to the orderer which sends it out to the

peers Peer on receiving the transaction does not add it directly to the ledger

What it does is it again checks the transaction against the endorsement policy

If the endorsement policy is not met by the transaction

The transaction is marked as failed

And if the transaction meets the endorsement policy the transaction is marked as successful and then

the peer adds it to the ledger

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Orderer Nodes

Orderer Nodes In this lecture I ll cover the features and functions of the orderer

I ll also talk about how the orderer node is implemented in hyperledger fabric network The orderers provide

the communication backbone for the fabric

It is also referred to as the ordering service and the orderers are responsible for creating a consistent

ledger s state across the network

Consensus is built in fabric using the orderers and the orderers are responsible for maintaining the order

of the transaction The blocks in the fabric network are created by the orders

And once the blocks are created they are delivered to each peer in the network

The orderer guarantees the atomic delivery of each of these blocks

Orderer is implemented on top of our message oriented middleware

There are two options available in version

The first option is to use SOLO

SOLO is a messaging middleware that is set up with a single node only

There is no clustering feature in it

So it s good for development but then it introduces a single point of failure in your network In production

you will not use solo Productionalized version of fabric would use Kafka

Kafka is a messaging software that has high throughput and high scalability features and it is fault tolerant

by way of clustering Both Kafka and SOLO support multiple channels so even if you are testing

with solo you should be able to test out various configurations with multiple channels

So let s take a look at how the orderer fits in the bigger picture The client broadcast the endorsed transaction

to the various peers using the orderer

You have already seen this picture The endorsing peer sends back the endorsement responses to the client Client

is now ready to send it to the various peers in the network and to do that

It invokes the orderers broadcast service

The orderer sends the transaction in a block to be anchor peers in the member organizations

The anchor peers on receiving the block with the transaction then sends it out to the various other

peers within that organization Orderer provides the communication layer for the hyperledger fabric network

It plays an important role in the consensus process and it is responsible for managing the order of

the transactions

Orderer is implemented with messaging system such as SOLO and Kafka

The default implementation available out of the box is solo As the name indicates solo allows you

to create a single instance of the orderer

Obviously having a single instance in a production environment would be unacceptable because of failure

of that single instance of the orderer will bring down the whole network

So in a production environment you would use Kafka

Kafka is a highly scalable and high thruput messaging system

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Membership Service Provider Certification Authority

In this lecture I ll cover the details of membership services and certification authority The membership

services provider is an abstract component of the hyperledger fabric system that provides the credentials

to the clients and peers for them to participate in the hyperledger fabric network

Now the key word here is abstract The idea of abstraction is that alternate implementation of membership

service provider may be plugged in without impacting the core logic or the foundational components

of the hyperledger fabric network

The default MSP implementation is based on the public key infrastructure or a PKI

There are two main services that the MSP provides authentication service and authorization service

Let me give you some examples for these two

Authentication is wherein the user s identity is getting validated

in other word is the user or peers certificate valid

That s for the MSP checks for Is the peer allowed to participate

This is again based on the certificate that the peer presents Authorization referred to various actions

that the user and peers can take

And the MSP again validates the authorization for the peers and users to carry out those actions

For example can this user issue or create new identities can the user deploy a chain code So these are

some examples where MSP plays a critical role in securing the hyperledger fabric network Now in PKI

based implementations there is a need to manage the identity by way of certificates

Certificates are issued validated and revoked

This is where the certification authority fits into the picture Hyperledger fabric development team has done

a great work of putting together a certification authority implementation that is available to you by

default

Now you may already be aware of what certification authorities are but for the benefit of those who are

new to certification authority I ll give you a crash course Certification Authority is a trusted

party that affirms the identity of an entity

By signing the certificate containing the entities public key and the key used for signing the certificate

is the certification authorities own private key

There are two other authorities that are referred to in the context of certification authority the

Registration Authority and the validation authority Lets go through the process of how certificates

are issued by the certification authority

The process starts with the requester raising a request for issuance of identity or signing of the certificate

The requester sends the appropriate documentation to the Registration Authority

Now there s documentation maybe a passport or it may be a HR letter saying that John Doe works for

Acme something that can prove to the registration authority that John is who he is claiming to be

So once the Registration Authority has validated the identity the registration authority informs the

certification authority to issue a certificate The certification authority issues the certificate by

signing the certificate and then sending it back to the requester

And one more thing that the certification authority does is it informs the validation authority about

this new certificate so that anyone can check with the validation authority if the certificate issued

to John Doe is valid or not

Let s talk about certification authorities in the context of hyperledger fabric

Think about a scenario where there was a single certification authority for the whole of the network

In that case the process of issuance of certificates will be very inefficient and may not be very cost

effective

So the way hyperledger fabric works is that a root certificate is issued to

each member in the network and then the root certificate may be authorised to issue new identities so that

the members in the network can manage the identities within their organizations

So in other words there are multiple certification authorities that can be set up in the hyperledger

fabric network As shown here in this illustration

There are three members and each of these members have their own certification authorities that are authorized

to issue certificates within their organization

So in other words the certification authority for member A can not issue a certification which will affiliate

or associate a member of organization A with the organization B or C The fabric CA is implemented in two

parts

The fabric CA server and the fabric CA client The person responsible for managing the certificates use

the fabric CA client to manage the certificates on the fabric CA server Orderers and peers can

validate the certificates using the interfaces exposed by the fabric CA server Apart from the CA client

there is also an SDK available for managing the certificates and there is also a rest API interface

available for the fabric CA server

The fabric CA server manages the identities and certificates in database

By default the CA server uses the SQL light implementation which obviously is not very robust By way of

configuration

you can change the CA server to use mysql or Postgres and it also supports the enterprise LDAP

Let s summarize In this lecture

I talked about the membership services and the certification authorities

The membership services are an abstract component in the hyperledger fabric and they provide the features

and functions for authentication and authorization

The default implementation of member services is based on the public key infrastructure

This default implementation can be switched with some kind of a custom implementation and that s where

the abstraction aspect of the membership services comes into play

The fabric CA implementation has two parts the fabric CA server and the fabric CA client Members are issued

a root certificate that they can use for issuing their own identities within their organizations

So the hyperledger fabric network can have one or more certification authorities to manage the certificates

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Dev Environment Walkthrough Orderer and CA Server

Development environment fabric tools setup In this lecture I will walk you thru the setup of the orderer and

the CA container

The solo based orderer component is already installed in the container for the orderer Configuration information

is provided to the orderer process by way of the environment variables The name of the orderer container is

orderer example com configuration parameters are here and the port on which the order

listens is

This is the command for launching the orderer

Lets attach a shell to the orderer container

Orderer is already installed on the container

We ll run orderer version to check the version

So this is the version of the orderer that is installed in the container for the orderer

You can use the orderer help to check out the various commands available on the orderer

The CA container has the binary for the CA server as well as the CA client The name of the CA container

is ca org example com and the port on which the CA server is listening is

When the container is launched the fabric CA server is started with the crypto material which is

on the local drive and mapped to the volume Lets attach a shell to the CA container

fabric ca server help

There are two commands available for the fabric CA server init and start The container calls the start command

with the options

There s a long list of options

You can go through these options on your own

Let s look at the fabric CA client and same minus minus help you can get all the commands available

for the fabric CA client

You see these commands available enroll getcacert reenroll register revoke

Some of these commands we will use from the composer command line interface

in later lectures When you are developing applications using the composer framework you don t have to

use the fabric CA client you can use the composer toolset to carry out the same activities

\*\*\*\*\*\*\*\*\*\*\* Chaincode Development

Chaincode In this lecture I ll cover at a very high level the structure of the chain code how

the chain code is developed and the execution runtime for the chain code Chain code on hyperledger fabric

maybe written in Go lang Java and node JS There are two parts to the chain code The first part is

the asset definition which is the digital representation of the asset

Think of it as a structure or a class definition that you would create In the case of nodeJS the structure

or the class for the asset is defined as part of Business Network modeling

So here s an example of the sample asset which has two attributes asset ID and value

Once you have created the asset definition you have to put together the transactions which will create

the asset and manage the state of the assets To do that

You would write the code for managing the state of the asset and that code can be written in javascript

in the case of node runtime The code in the transactions implement the business logic

This code carries out the typical operations against the various assets defined in the business network

application

The create retrieve update and delete operations

Let s talk about the development workflow for chain code

The developer writes the chain code in Java or go Lang or nodeJS and then uses the common software practices

to iteratively compile and test the code

till he or she is satisfied that the code is working As the next step the developer deploys the chain

code to the peer using a deployed transaction As part of the deployment

the developer can also put together the endorsement policy for the chain code The deployment transaction

is propagated to the network and once the deployment transaction is successful the transaction log and

the state data gets updated

Participants in the network can use applications to invoke the chain code and all such invocations are

recorded in the transaction log

All the state changes are recorded in the state database The deployment transaction deploys the chain

code instance in its own container and the execution or invocation of the chain code also happens within

the independent containers for each of these chain code instances

So here the example shows that chain code A and chain code B have their own execution runtime environments

Let me show you how that works

As you can see here I have my development environment up

You can see these four containers Now I am going to deploy a business network application and that would launch

an additional container here So the deploy command has created an additional container dev peer

and the container has the runtime for the application that has been deployed

This course covers the business network application development using composer

You will be learning about all the composer tools You will be using the nodeJS environment for creating

your business network application and all the chain code that you will develop in this course will be

written in javascript

We will dig deeper into the composer tool

starting next section and the section after that you will start coding the chain code in Javascript

Let s summarize In this lecture I talked about chain code Chain code on fabric can be developed

using go lang Java or nodeJS Chain code is developed as part of the business network application

There are two parts in the chain code

The first part is the asset definition which is the digital representation of the asset that you would

manage on the blockchain The second part is the transaction definition and implementation wherein you

would code the transactions for creation of the assets as well as management of the asset

Typically you would code the CRUD operations or create retrieve update and delete operations on

the various assets that you would define for your business network application

When the business network application is deployed the chain code gets deployed and this chain code gets

deployed in its own docker container So at the runtime the chain code execution occurs in the isolated

docker container environment that is dedicated to the business network application

Starting from next section

Our focus will be on developing business network application as well as chain code using the Node runtime

and all the chain code that you will develop will be in

Javascript

Ledger implementation Hyperledger fabric is a distributed ledger technology

All peers in the network have a copy replica of the ledger Ledger has two parts transaction log and

state database At the end of this lecture

You should be able to explain what is contained in the transaction log and what is contained in the

state database as well as how these two components are implemented in fabric

In last section I touched on the topic of ledger In fabric

there are two parts to the ledger Transaction log that keeps track of all the transactions invoked against

the assets

And then there is the state data The state data is representation of the current state of the asset

at any point in time

As in the example of car let s say dealer sells the car to person A transaction is added to the log and

state data reflects that A is the owner of the car

Now let s say person A sold the car to B The state data reflects the new owner as B

And let s say again B sells to C the state data reflects

C as the owner The transaction log is immutable

But the state data is not immutable

This is sometime a source of confusion

Now let s discuss what I mean by these statements

You must have heard about the term CRUD create retrieve update and delete

Let s see how CRUD applies to transaction log and state data

You can create transaction records in the transaction log

You can retrieve the existing transaction records from the transaction log but you cannot update an

existing transaction record which is there in the log

And you can not delete any of the transactions which have been added to the log From the state data

perspective

you can carry out any of these operations on the state data for an asset

Now let s talk about how new transactions are created and how the CRUD operations are carried out on

the state data

It all happens by way of the execution of the chain code When the chain code is executed

It leads to the creation of transactions in the transaction log

And at the same time depending on the code in the chain code there may be a change in the state data

So in other words the CRUD operations are implemented in the code for the chain code The transaction

log is implemented using the levelDB LevelDB is a lightweight library for building key value data store

It is used in embedded manner as part of the fabric peer implementation

That means that the levelDB is not launched as a separate process but it is part of the peer process

It is a queryable and highly efficient implementation for insertion of data or creation of data

Peers write the transactions to the level database

Now keep in mind that you cannot replace level database with any other

It is a fixed implementation from the fabric peer implementation perspective

The state data consists of the key value pairs that are version

Let me explain what I mean by that

The asset state is managed in storage variables identified by the keys for example

Key owner

The value is represented by way of arbitrary blobs or binary objects

These blobs maybe in JSON format

Apart from the key value the state data also has a version When the state data is updated the existing

data is not over written

which would cause the old data to be lost

Instead a new version is created for the key value pair and placed in the state database

It is an example let s say the car has two attributes VIN number and the owner information

This is the version one of the cars state data

Now let s say the car is sold by the current owner so the owner data in the state is updated

So at this point the version one of the data is there and it s not over written A version of the state data

is created for the owner and retained in the state database

Same way a version will be created the next time a transaction is executed that updates the owner information

State data can have this representation state name of the key and then the version and value for that

key

The start data by default is managed in LevelDB which is the embedded database

like I explained in the case of transaction log The data will look like this for our car example The

VIN number which never changed

stays at version and then ownership of the car has changed over a period time

So the version was when the car was created

by the manufacturer and then the version is when the ownership of the car was transferred to the dealership

Let s call the dealership Adam dealership and then when the dealership sold the car to John

Then there is a version created in the levelDB

One thing to keep in mind is that chain code owns the data for these keys so the identification of a

specific key is not just the key name but the chain code name and the key name

So in other words the access to the data is restricted to the owner chain code and if there s another

chain code that tries to access it it won t be able to do that

Now this is how it s implemented today

in version

This may change in future

where in the other chain code

maybe allowed to access the chain code data that they don t own

Let s talk about the data held in the transaction log

And the state data Both transaction log and state database by default

use levelDB and levelDB supports such simple queries

Now if these simple queries are going against the transaction log that should be OK because you re not

going to go for complex queries against the transaction logs

The only thing that you have to keep in mind is that when you are querying the transaction log you ll

be querying it against the peer because peer has the levelDB embedded within the peer process

Businesses depend on reports and business intelligence and insights that they gather from the data to

make their decisions With simple queries that are available in levelDB

Businesses will be constrained because they need to write some complex queries to generate these reports

and insights

Unfortunately levelDB will not work in those cases

The good news is that the hyperledger fabric team understands this issue and what they allow you to

do is switch the levelDB with a more mature database with more flexibility around the querying

capabilities

So the state data base is pluggable at the peer level By default it is levelDB which support a simple

query for key value pairs but you can replace it with couch database which is a nosql database that

allows you to execute complex queries and all of this is done by the configuration of the peer

Let s summarize In this lecture

I talked about the ledger Ledger has two parts

The transaction log and the state database The ledger is managed on the peer node The peers receive

the data for the transactions and they update the transaction log and the state database Both transaction log

and the state database are implemented by way of levelDB which is a key value database Transaction

log tracks the transactions in the order they are received Transaction log is immutable

In other words once the transaction is added to the transaction log it can neither be updated nor be

deleted but it can be retrieved Compare it with state data State data can be manipulated by way of the CRUD

operations that can be coded in the chain code State data is versioned

So what that means is any time you update the data or delete the data managed as state data for the asset

a new version is created

Businesses need to create reports and generate insight from the state data and that requires execution

of complex queries

Unfortunately levelDB does not provide the capability for creating and executing complex queries

But the good news is that you can replace the state database which is implemented in levelDB with

couchDB which supports creation of complex queries

This switching of database is carried out by way of configurations on the peer

Development fabric tools setup In this lecture I ll give you a quick walkthru of the peer and the couchDB

set up for the dev environment setup The peer image already has the peer binary installed When the peer

is started it reads the environment variables for configuration

Here it s configured to use the levelDB for transaction data and the peer is set up to use an instance

of couchDB for state management

The container for Koshti has the binary installed for always DV and it also has the configuration some

of the configuration parameters are read for CouchDB from the environment

Let s take a look at the configuration setup for peer and couchDB All of the container configuration

for the dev topology or the dev fabric tools environment is done in the docker composed yml file under

the composer subfolder or hlfv or hlfv

I have opened the docker compose yml file under the hlfv

Let s look at how the peer is configured The container name is

peer org example com

These are all the environment variables which are made available to the peer binary

These two variables control the logging level for various modules within the peer binary

So you can control the logging level for the chain code by changing this debug from debug to for example

error or info

This variable controls whether the peer would use the levelDB or couchDB for managing the state data

This peer is set up to use the couchDB instance The instance information is made available to the

peer by way of this environment variable When the container is launched

this is the command that gets executed peer node start

And this is where the peer instances are started in the container

These two ports are exposed to the local host or local machine

All of the configuration information is managed under the composer subdirectory of the fabric tools project

That configuration is made available to the components within the container

by way of these volume mappings As discussed earlier the peer binary is already installed in the container

and you will in fact be able to run some of the commands available on this peer binary Just to give you an

idea

These are the commands available on peer binary

For example you can get and set the logging levels by using the peer logging command

You can get the status of the node by peer node status get the version using a peer version command

Let s logon to the container and try out some of these commands

Let s attach a shell to the running container right click attach shell type env to check the environment

This is the environment that we have set up in the docker compose file

You can check the status of the peer by running peer node status and this is the current status

Let s list the channel by using the peer channel list and as you can see this peer has joined the composer

channel Do a ls la here

You will find the composerchannel block that has the channel configuration

this peer is using

Let s take a look at the couchDB now Right click on the couchDB container and show logs and you

will see a message here What this message is saying is that the couchDB instances running in admin

party mode has no security and anyone with a url can connect to it

Let s connect to it

If you are on a Mac or Ubuntu you can simply use local host

But since I am on Windows I ll have to use the IP address of the virtual machine and here couchDB welcome

version

This basically indicates that we are connected to the couchDB instance that is running in the

docker container and the peer instance is connecting to the this couchDB instance for managing the

state data

Peer nodes In the last section I gave you an overview of the three different type of nodes client node

peer node and the orderer node In this lecture the focus is the peer node

In this lecture you ll also learn about two special type of peer nodes the Anchor peers and the Endorser peers

Members in the blockchain network need to set up peers in their infrastructure for participating in the network

Say for example a member A has set up peer nodes

All of these peers need to be configured with the appropriate cryptographic materials such as certificates

and other information Peers in the members organization receive transaction invocation requests from

the clients within the organization As transactions are created in the network and new blocks get generated

These blocks are sent out to the peers by the ordering service and peers receiving these blocks need

to validate and update the ledger managed on the peer node

Inherently this kind of an architectural approach is highly scalable as there is no need for a centralized

effort to scale the network or to scale the infrastructure

Each member organization can look at their needs and set up the infrastructure based on their requirements

So for example the member organization A has three peers whereas B has decided to setup only two

peers because that may be enough for their requirements Member organization can have multiple peers but

not all peers receive the block information from the orderer only the Anchor peer receives the blocks

In this diagram the anchor peer is this one

So what happens is when the anchor peer receives the block it updates the other peers in the member

organization To avoid single point of failure an organization can create a cluster of anchor peers or

more than one anchor peer Anchor peers are set up and defined as part of the channel configuration and

the anchor peers are by default discoverable

Now what that means is that any peer that is marked as anchor peer is discoverable by the order and other

anchor peers Lets talk about the endorser Peers maybe marked as the endorser or they can take up the role

of endorser in which case they are also known as the endorsing peer A client sends the invocation requests

to the endorsing peer On receiving the request for the invocation the endorsing peer validates the transaction

For example it checks if the end user has used a valid certificate or not If the validation checks out

fine

Then it simulates the chain code

What that means is that the endorsing peer executes the chain code

Now after the execution it does not save the state of the chain code back to the block chain as it is

simply simulating the change

And you will understand much better how this process works as part of the transaction flow

So for the time being just think of simulation as execution but the data that has changed in the chain

code is not saved to the ledger at this point

At the end of the endorsement process the endorser either rejects the transaction and this may happen

due to multiple reasons for example the security aspects

didn t check out or the execution failure or the endorser may respond back with an endorsed transaction

request

More on this later

The primary objective of the endorsing peer or the endorser is to protect the network Now when I say protect

the network it doesn t necessarily means protect from intentional attack on the network but it also

means that it needs to protect the network from a misbehaving or mis configured node on the network

And since every member organization is responsible for configuring their own nodes which is peers there

is the possibility that a mis configured node may be unintentionally added to the network

Another advantage of this mechanism is that only the endorser needs to execute the chain code

And this will improve the overall scalability of the infrastructure

Since there is no need for all nodes to execute the chain code The focus of this lecture was on the peer

node Every member organization need to set up peers in the organization

Peers receive blocks from the network

There is a special kind of peer that is set up to receive the blocks and that peer is known as the anchor

Anchor peer receives the blocks and then provides these blocks to the other peers in the organization

Then there is endorser peer or the endorsing peer The endorsing peer receives the transaction requests

from the client On receiving the transaction requests from the client

It simulates the transaction that is the transaction is executed

The chain code is executed but the state of the chain code is not updated in the ledger

The endorser then rejects or accepts the transaction after it has carried out multiple validation check

The primary objective of this endorsement mechanism is to protect the network from intentional as well

as unintentional attacks

Client nodes By now you are already aware of the fact that client nodes are responsible for initiating

transactions

In this lecture my focus is on the endorsement policies Endorsement policies are used by the client

the peer and the orderer to ensure that the transactions are valid before they get added to the ledger

across the network

Client is responsible for initiating the transactions and the client does that by creating the transaction

request and sending it to one or more endorsing peer Clients connect to the endorser and sends the

transaction to be endorsed

Now the question that comes to mind is which peer should be used as an endorser

And the answer is It depends

It depends on the chain code The chain code can associate a policy known as the endorsement policy

and this endorsement policy has two components which peers to use as endorsers

So this is how the client comes to know which endorser it should connect to

for getting the transaction endorsed Another part of the endorsement policy is the criteria for valid

transaction

Let s look at the endorsement policy in a little bit more detail

In this example policies we have set up three peers that have been assigned the role of endorsers Clients

must send the transaction request to all of these endorsing peers and then collect the responses In

the second part of the endorsement policy

there is a criteria for transaction validation

This is used by the peers

I ll explain that aspect in a few minutes

But before that let s go over what is in this criteria In the criteria the chain code developer defines

either the number of endorsements needed for checking the validity of the transaction or same thing can

be specified as a percentage

For example the policy may say that two out of three endorsers must endorse the policy to have the transaction

considered as valid

Or it may simply say percent or above is the criteria used for marking the transaction as valid In

combination you may use logical expression with AND or OR or operators to define expressions that are used

for checking the validity of the transaction

Here is an example

The transaction will be considered valid if Org endorser endorses the transaction and if the Org

member has not endorsed the transaction then both or two and or three endorsers should have endorsed

the transaction request for it to be considered valid

Couple of quick points about the endorsement policy Endorsement policies are optional

In other words the chain code is not required to have an endorsement policy The endorsement policy

is specified at the time of deployment of the chain code and obvious question would be if the policy

is not specified then what is the default policy

The default policy is that any one peer from the client organization can endorse the transaction request

At this point you are already aware that peers are responsible for keeping the ledger in sync with the

network

So peers received the transactions that they need to add to the ledger but before the peer adds the transaction

to the ledger

It uses the endorsement policy to check the validity of that transaction

So there are three high Level things that the peer checks for First is are all the endorsements valid

Second thing is check the criteria

Are there enough endorsement for the transaction to be considered valid

Third thing is are the endorsements coming from the right sources

So in this particular scenario are the endorsements coming from peer peer and peer

If there is a failure in any of these three checks

The transaction is considered invalid

So after this check the transaction is marked as failed if it has failed the validation and added to the transaction

log

If the validation has passed then it is marked as successful transaction and added to the transaction

log

Let s summarize In this lecture I covered one of the most important topics of hyperledger fabric

The endorsement policy The endorsement policies are associated with the chain codes

Clients need to send the transaction request to the endorsing peers and the client gets the list of

endorsing peers from the endorsement policy

Client then sends the request to the endorsing peers which validate the transaction request and simulate

the transaction

Once the transaction is validated the client s sends it out to the orderer which sends it out to the

peers Peer on receiving the transaction does not add it directly to the ledger

What it does is it again checks the transaction against the endorsement policy

If the endorsement policy is not met by the transaction

The transaction is marked as failed

And if the transaction meets the endorsement policy the transaction is marked as successful and then

the peer adds it to the ledger

Orderer Nodes In this lecture I ll cover the features and functions of the orderer

I ll also talk about how the orderer node is implemented in hyperledger fabric network The orderers provide

the communication backbone for the fabric

It is also referred to as the ordering service and the orderers are responsible for creating a consistent

ledger s state across the network

Consensus is built in fabric using the orderers and the orderers are responsible for maintaining the order

of the transaction The blocks in the fabric network are created by the orders

And once the blocks are created they are delivered to each peer in the network

The orderer guarantees the atomic delivery of each of these blocks

Orderer is implemented on top of our message oriented middleware

There are two options available in version

The first option is to use SOLO

SOLO is a messaging middleware that is set up with a single node only

There is no clustering feature in it

So it s good for development but then it introduces a single point of failure in your network In production

you will not use solo Productionalized version of fabric would use Kafka

Kafka is a messaging software that has high throughput and high scalability features and it is fault tolerant

by way of clustering Both Kafka and SOLO support multiple channels so even if you are testing

with solo you should be able to test out various configurations with multiple channels

So let s take a look at how the orderer fits in the bigger picture The client broadcast the endorsed transaction

to the various peers using the orderer

You have already seen this picture The endorsing peer sends back the endorsement responses to the client Client

is now ready to send it to the various peers in the network and to do that

It invokes the orderers broadcast service

The orderer sends the transaction in a block to be anchor peers in the member organizations

The anchor peers on receiving the block with the transaction then sends it out to the various other

peers within that organization Orderer provides the communication layer for the hyperledger fabric network

It plays an important role in the consensus process and it is responsible for managing the order of

the transactions

Orderer is implemented with messaging system such as SOLO and Kafka

The default implementation available out of the box is solo As the name indicates solo allows you

to create a single instance of the orderer

Obviously having a single instance in a production environment would be unacceptable because of failure

of that single instance of the orderer will bring down the whole network

So in a production environment you would use Kafka

Kafka is a highly scalable and high thruput messaging system

In this lecture I ll cover the details of membership services and certification authority The membership

services provider is an abstract component of the hyperledger fabric system that provides the credentials

to the clients and peers for them to participate in the hyperledger fabric network

Now the key word here is abstract The idea of abstraction is that alternate implementation of membership

service provider may be plugged in without impacting the core logic or the foundational components

of the hyperledger fabric network

The default MSP implementation is based on the public key infrastructure or a PKI

There are two main services that the MSP provides authentication service and authorization service

Let me give you some examples for these two

Authentication is wherein the user s identity is getting validated

in other word is the user or peers certificate valid

That s for the MSP checks for Is the peer allowed to participate

This is again based on the certificate that the peer presents Authorization referred to various actions

that the user and peers can take

And the MSP again validates the authorization for the peers and users to carry out those actions

For example can this user issue or create new identities can the user deploy a chain code So these are

some examples where MSP plays a critical role in securing the hyperledger fabric network Now in PKI

based implementations there is a need to manage the identity by way of certificates

Certificates are issued validated and revoked

This is where the certification authority fits into the picture Hyperledger fabric development team has done

a great work of putting together a certification authority implementation that is available to you by

default

Now you may already be aware of what certification authorities are but for the benefit of those who are

new to certification authority I ll give you a crash course Certification Authority is a trusted

party that affirms the identity of an entity

By signing the certificate containing the entities public key and the key used for signing the certificate

is the certification authorities own private key

There are two other authorities that are referred to in the context of certification authority the

Registration Authority and the validation authority Lets go through the process of how certificates

are issued by the certification authority

The process starts with the requester raising a request for issuance of identity or signing of the certificate

The requester sends the appropriate documentation to the Registration Authority

Now there s documentation maybe a passport or it may be a HR letter saying that John Doe works for

Acme something that can prove to the registration authority that John is who he is claiming to be

So once the Registration Authority has validated the identity the registration authority informs the

certification authority to issue a certificate The certification authority issues the certificate by

signing the certificate and then sending it back to the requester

And one more thing that the certification authority does is it informs the validation authority about

this new certificate so that anyone can check with the validation authority if the certificate issued

to John Doe is valid or not

Let s talk about certification authorities in the context of hyperledger fabric

Think about a scenario where there was a single certification authority for the whole of the network

In that case the process of issuance of certificates will be very inefficient and may not be very cost

effective

So the way hyperledger fabric works is that a root certificate is issued to

each member in the network and then the root certificate may be authorised to issue new identities so that

the members in the network can manage the identities within their organizations

So in other words there are multiple certification authorities that can be set up in the hyperledger

fabric network As shown here in this illustration

There are three members and each of these members have their own certification authorities that are authorized

to issue certificates within their organization

So in other words the certification authority for member A can not issue a certification which will affiliate

or associate a member of organization A with the organization B or C The fabric CA is implemented in two

parts

The fabric CA server and the fabric CA client The person responsible for managing the certificates use

the fabric CA client to manage the certificates on the fabric CA server Orderers and peers can

validate the certificates using the interfaces exposed by the fabric CA server Apart from the CA client

there is also an SDK available for managing the certificates and there is also a rest API interface

available for the fabric CA server

The fabric CA server manages the identities and certificates in database

By default the CA server uses the SQL light implementation which obviously is not very robust By way of

configuration

you can change the CA server to use mysql or Postgres and it also supports the enterprise LDAP

Let s summarize In this lecture

I talked about the membership services and the certification authorities

The membership services are an abstract component in the hyperledger fabric and they provide the features

and functions for authentication and authorization

The default implementation of member services is based on the public key infrastructure

This default implementation can be switched with some kind of a custom implementation and that s where

the abstraction aspect of the membership services comes into play

The fabric CA implementation has two parts the fabric CA server and the fabric CA client Members are issued

a root certificate that they can use for issuing their own identities within their organizations

So the hyperledger fabric network can have one or more certification authorities to manage the certificates

Development environment fabric tools setup In this lecture I will walk you thru the setup of the orderer and

the CA container

The solo based orderer component is already installed in the container for the orderer Configuration information

is provided to the orderer process by way of the environment variables The name of the orderer container is

orderer example com configuration parameters are here and the port on which the order

listens is

This is the command for launching the orderer

Lets attach a shell to the orderer container

Orderer is already installed on the container

We ll run orderer version to check the version

So this is the version of the orderer that is installed in the container for the orderer

You can use the orderer help to check out the various commands available on the orderer

The CA container has the binary for the CA server as well as the CA client The name of the CA container

is ca org example com and the port on which the CA server is listening is

When the container is launched the fabric CA server is started with the crypto material which is

on the local drive and mapped to the volume Lets attach a shell to the CA container

fabric ca server help

There are two commands available for the fabric CA server init and start The container calls the start command

with the options

There s a long list of options

You can go through these options on your own

Let s look at the fabric CA client and same minus minus help you can get all the commands available

for the fabric CA client

You see these commands available enroll getcacert reenroll register revoke

Some of these commands we will use from the composer command line interface

in later lectures When you are developing applications using the composer framework you don t have to

use the fabric CA client you can use the composer toolset to carry out the same activities

Chaincode In this lecture I ll cover at a very high level the structure of the chain code how

the chain code is developed and the execution runtime for the chain code Chain code on hyperledger fabric

maybe written in Go lang Java and node JS There are two parts to the chain code The first part is

the asset definition which is the digital representation of the asset

Think of it as a structure or a class definition that you would create In the case of nodeJS the structure

or the class for the asset is defined as part of Business Network modeling

So here s an example of the sample asset which has two attributes asset ID and value

Once you have created the asset definition you have to put together the transactions which will create

the asset and manage the state of the assets To do that

You would write the code for managing the state of the asset and that code can be written in javascript

in the case of node runtime The code in the transactions implement the business logic

This code carries out the typical operations against the various assets defined in the business network

application

The create retrieve update and delete operations

Let s talk about the development workflow for chain code

The developer writes the chain code in Java or go Lang or nodeJS and then uses the common software practices

to iteratively compile and test the code

till he or she is satisfied that the code is working As the next step the developer deploys the chain

code to the peer using a deployed transaction As part of the deployment

the developer can also put together the endorsement policy for the chain code The deployment transaction

is propagated to the network and once the deployment transaction is successful the transaction log and

the state data gets updated

Participants in the network can use applications to invoke the chain code and all such invocations are

recorded in the transaction log

All the state changes are recorded in the state database The deployment transaction deploys the chain

code instance in its own container and the execution or invocation of the chain code also happens within

the independent containers for each of these chain code instances

So here the example shows that chain code A and chain code B have their own execution runtime environments

Let me show you how that works

As you can see here I have my development environment up

You can see these four containers Now I am going to deploy a business network application and that would launch

an additional container here So the deploy command has created an additional container dev peer

and the container has the runtime for the application that has been deployed

This course covers the business network application development using composer

You will be learning about all the composer tools You will be using the nodeJS environment for creating

your business network application and all the chain code that you will develop in this course will be

written in javascript

We will dig deeper into the composer tool

starting next section and the section after that you will start coding the chain code in Javascript

Let s summarize In this lecture I talked about chain code Chain code on fabric can be developed

using go lang Java or nodeJS Chain code is developed as part of the business network application

There are two parts in the chain code

The first part is the asset definition which is the digital representation of the asset that you would

manage on the blockchain The second part is the transaction definition and implementation wherein you

would code the transactions for creation of the assets as well as management of the asset

Typically you would code the CRUD operations or create retrieve update and delete operations on

the various assets that you would define for your business network application

When the business network application is deployed the chain code gets deployed and this chain code gets

deployed in its own docker container So at the runtime the chain code execution occurs in the isolated

docker container environment that is dedicated to the business network application

Starting from next section

Our focus will be on developing business network application as well as chain code using the Node runtime

and all the chain code that you will develop will be in

Javascript