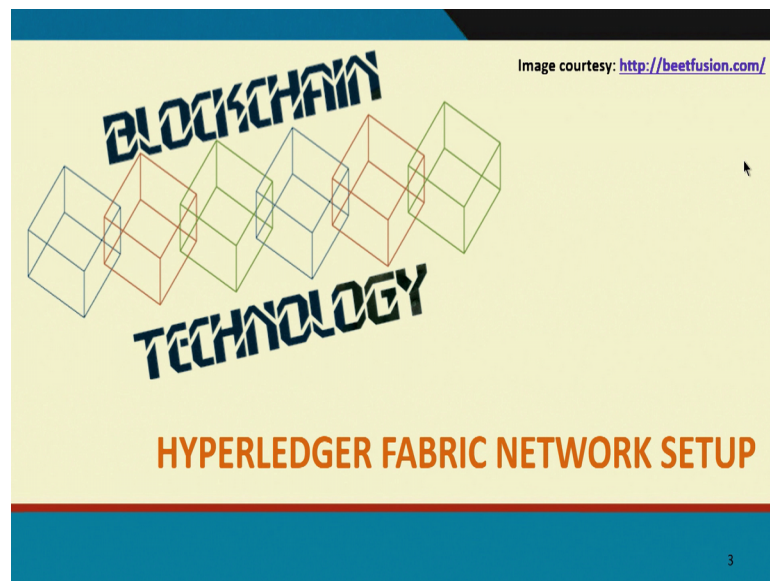


**Blockchains Architecture, Design and Use Cases**  
**Prof. Praveen Jayachandran**  
**Prof. Sandip Chakraborty**  
**Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 24**  
**Hyperledger Fabric Network Setup\_Lec\_06**

Hello, everyone and welcome back to our next lecture in our block chains course Architecture Design and use cases. Before we start I must thank a lot of my colleagues at IBM who have made some of these presentations available over for a period of time. So, I have collected this from many different sources, but a lot of the IBM heirs have helped to prepare some of these lights. So, have many things to say to them. So, we have been looking at some of the details of hyper ledger fabric itself and in this lecture were going to talk about how you can set up your own fabric network.

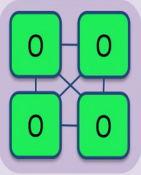
(Refer Slide Time: 00:39)



So, what are some of the things it takes what are the steps it takes to set that up.

(Refer Slide Time: 00:55).

## Step 1/6: Configure & Start Ordering Service



Ordering-Service

Hyperledger Fabric Network

An Ordering Service is configured and started for the network:  
`$ docker-compose [-f orderer.yml] ...`

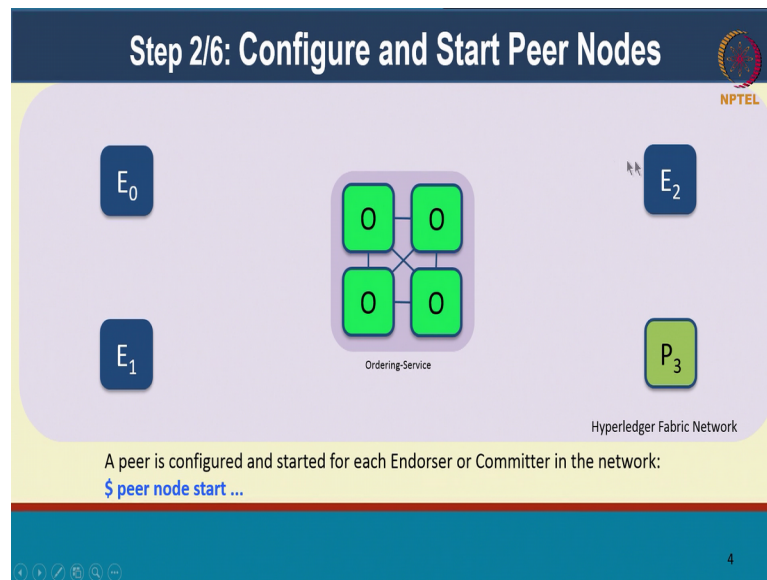
NPTEL

3

So, first step is like I mentioned earlier is the ordering service that is the main component that is going to determine how transactions are ordered in the network and that will be the first thing you set up or you bring up in the network. So, it could be single solo ordering net node or it could be a distributed set of nodes that are running an ordering service let us say a kafka consensus that they execute. So, that is the first ordering service that that is the first component that you bring up.

And the simple CLI command for you to bring that up or you can also use the SDK that is available in multiple languages in java node Js and python.

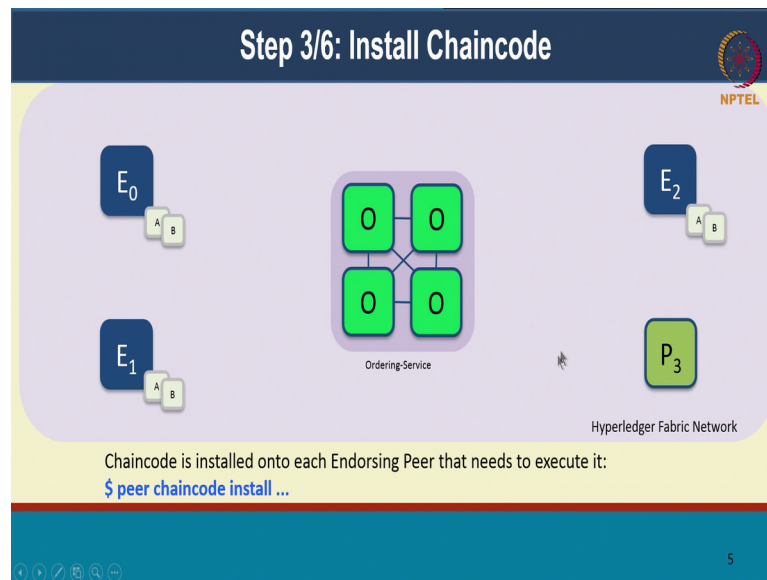
(Refer Slide Time: 01:31).



So, once the ordering service is up, then the next step is to configure and start peer nodes. So, you could have multiple organizations that are part of this block chain network each of those organizations could run one or more peers. So, you will have of course, all the identities have to be set up when I say, you are bringing up each of these entities I am assuming that we or when you brought up the ordering service the identity for the ordering service is set up and likewise for the peers each of these peers in their respective organizations have been have their identities right with an MSP and provided by your certificate authority.

So, each of these peers have to be brought up. So, again there are simple CLI commands and SDK commands that you can use to bring these services up. So, each of these peers like I said could belong to different organizations. So, that now the peers are up, but they are still not can to each other in anyway.

(Refer Slide Time: 02:23).



Once you bring up these pure processes, the next step may be to install one or more chain codes on these peers again when I say install this is just to bring up the process that will run the business logic again this is not connected in any way to the rest of the system. So, this pure will then have a container that runs the chain code A again another container to run chain code B likewise for all the other peers.

Again a very simple command both from CLI and from SDK, will enable you to set up to install these chain codes on the respective peers. And also note the fact that the chain codes can be installed on only a subset of the peers in the network it does not have to be installed on all the nodes. So, you can decide who are the endorsing peers for each chain code and you can install the chain code only on the endorsing peers. So, this just this does give you a notion of privacy in the sense that the code itself the business logic that you are executing in a decentralized fashion, can be executed amongst a subset of the organizations you have in the network, it does not have to be all the organizations or all the peers ok.

So, now you have the ordering service, you have a bunch of peers, you have chain codes installed in them now is when you start constructing the network itself and the communication between them.

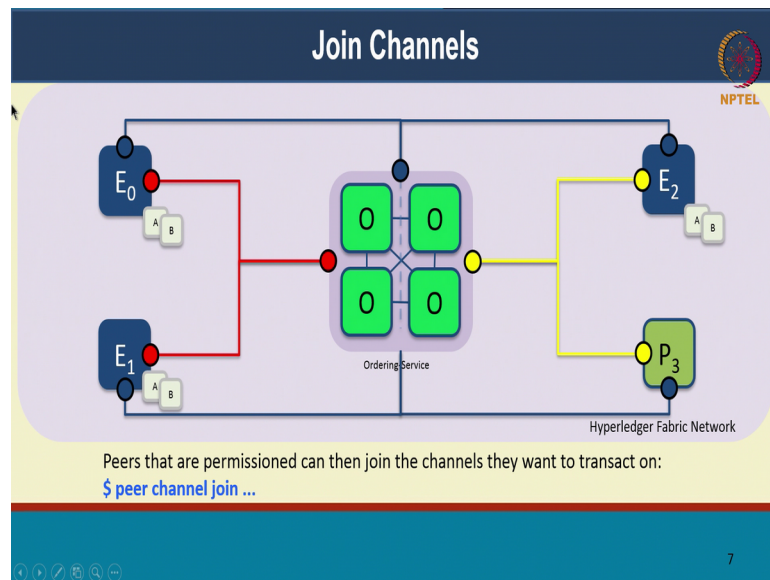
(Refer Slide Time: 03:45).

The slide, titled "Step 4/6: Create Channels", features the NPTEL logo in the top right corner. The main diagram shows a central "Ordering-Service" consisting of four green square nodes arranged in a 2x2 grid, interconnected by lines. Three colored dots (red, blue, and yellow) are positioned around the nodes, representing channels. To the left, three blue boxes labeled E<sub>0</sub>, E<sub>1</sub>, and E<sub>2</sub> are shown, each with two smaller boxes labeled A and B below it. To the right, a blue box labeled E<sub>3</sub> is shown with two smaller boxes labeled A and B below it. Below the diagram, the text reads "Channels are created on the ordering service:" followed by the terminal command `$ peer channel create -o [orderer] ...`. A presenter's head and shoulders are visible in the bottom right corner of the slide frame.

So, you are going to be creating some channels the ordering service is where these channels are configured. So, of course, you could also have multiple ordering services for these different channels, but in this example let us assume that there is just one ordering service in again that ordering service has multiple nodes. So, it is weird ordering service and it has multiple channels. So, in this case there is a red a channel a blue channel and a yellow channel.

So, all these channels are configured with the ordering service and when you are configuring the channels with the ordering service you are also going to be joining peers in that channel as the next step.

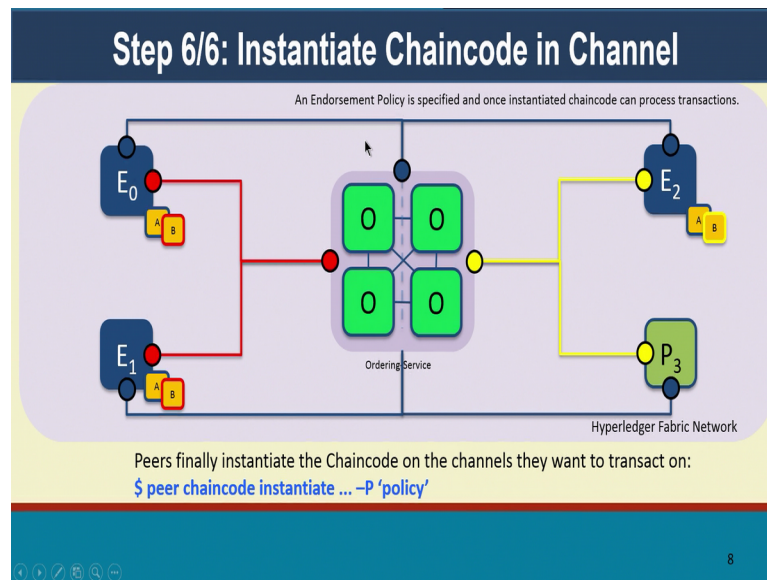
(Refer Slide Time: 04:18).



So, the channel configuration will include the set of organizations and set of peers there are part of that channel. So, in this case E<sub>0</sub> and E<sub>2</sub> have joined the blue channel E<sub>0</sub> and E<sub>1</sub> have joined the red channel E<sub>2</sub> and this P<sub>3</sub> which is not an endorsing node have joined the yellow channel and actually they yah the blue channel also has E<sub>1</sub> and P<sub>3</sub>. So, the blue channel has E<sub>0</sub>, E<sub>1</sub>, E<sub>2</sub> and P<sub>3</sub>. So, all the nodes are part of the blue channel.

So, this way it is possible to have different subsets of nodes join different channels and channels like I mentioned are a notion of provide a notion of privacy across the different transactions and the different state you want to maintain in the network. So, the same peer can be part of multiple channels. So, again there is a simple CLI command and again you can do the same thing same thing using SDK. So, it could be a client application as part of setting up the network could be executing all of these steps or it could be different nodes. So, basically these nodes are part of different organizations their respective client applications would have to join their peers to the channel.

(Refer Slide Time: 05:33).



The next step is to instantiate the chain codes on channel. So, you have to say you have the chain codes running on the peers, but you have to say which chain code belongs to which channel. So, again this is highlighted; if you see as the border color here in each of these boxes if you take the channel red then chain code B is part of the channel red. So, the chain code B is instantiated on that channel. So, it will have it is ledger and it can add transactions to the chain to the block chain in the red channel.

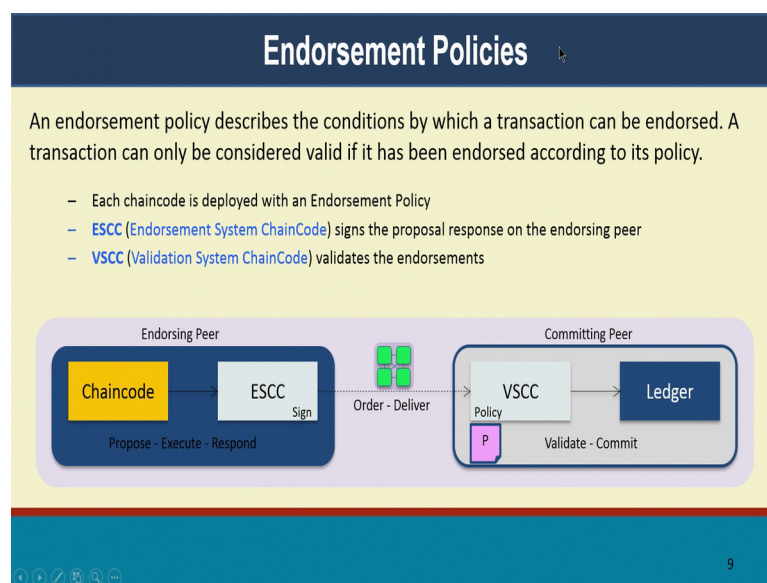
Likewise the yellow channel again has chain code B alone. So, the yellow channel has only one endorsing P 1 which is E 2 it alone runs this chain code B and it is connected to the yellow channel. So, it will have it is lecture and a block of transactions. The blue channel has all the peers and chain code A is part of the blue channel. So, you can see the chain code A has this blue border. So, the chain code will be part of the blue channel and all transactions on the blue channel will be functions that are invoked on chain code A.

So, this is just an example network to show the flexibility of the system. So, you can define who which peers you can define the channels first, you can define which peers which organizations are part of which channels you can also define which chain codes are part of which channel and all of these are segregated separately. So, each chain code is a chain code state is separate from another in each channel right. So, that is how it is maintained and all the chain code state across the peers in a channel are all consistent.

So, all the state will be maintained consistently using consensus again a very simple CLI command is available for you to execute and setup set this up.

So, with this if you see the whole network is set up and now users can perform transactions on these on this network. So, on the red channel users can invoke functions of chain code B again same thing for the yellow channel users can invoke functions of chain code B on the blue channel they can invoke functions of chain code A. So, that is how it is separated out.

(Refer Slide Time: 07:50)



I mentioned that it is possible to have just a subset of the peers executing chain codes on each channel and that is governed by the endorsement policy right. So, the endorsement policy states which peers have to execute and endorse a transaction before it will be deemed as valid to be added to the block chain.

So, each chain code defines an endorsement policy and this can be different for each channel that the chain code belongs to now there are two important system chain codes. So, these are the system chain codes are chain codes that are implemented internal to fabric itself that perform certain important functions we will talk about two of these system chain codes, there are I think three or four more of these system chain codes that have been added again to these system chain codes also provide the notion of plug ability. So, although I call I mentioned these two system chain codes, you can always



think about other implementations with other properties that these two system chain codes provide.

So, the first is in chain code that is of importance is the endorsement system chain code what it does is. So, this system chain code is going to be running within every endorsing peer. So, when a client submits a transaction to appear. So, this is an endorsing peer here when a client submits a transaction to the endorsing peer. It is a transaction that is invoking a particular function that is in the chain code and it first comes to the ESCC.

So, the ESCC the endorsement system chain code is the one that will invoke the chain code will execute the transaction, it will compute the read write set of the transaction. So, that what is the state information that was read in the chain code in that function? What is the state information that was written? All of that is captured by the ESCC and it will then sign the proposal response.

So, it will then sign this and say ok. So, when I run this transaction, this is the read set, this was the write set and here is my signature saying that I was the one who run this and I endorse this. So, this is done by every endorsing peer and this system chain code does that signing part of it. Now client collects all of these signatures from multi layer all the endorsements it will then submit the transaction to the order. The order will include it in the block, deliver it to the peer and now all committing peers are now going to do certain functions right.

So, that is where the validation system chain code comes in which is going to validate these endorsements. So, what does the validation system chain code do it will look up the policy for this particular chain code. So, this policy might say organization one two and three in the network have to sign this transaction.

So, it will then check whether those three signatures are available the other policy we look at some of these policies that how the syntax works for some of these policies, but the validation system chain code does the job of going through the endorsements that of that are available in the transaction that have been submitted with the transaction and seeing if that set of endorsements satisfying the policy that was specified for the chain code in that channel.

So, that is the validation system chain code. Now it is always possible that the endorsement, you want to use some other logic for the endorsement or for the validation you want to use some other logic for the validation or even you might want to implement that in a different language rather than go lang. So, all those possibilities exist and this whole module is pluggable and you can bring in your own implementation for these things based on your application needs.

But for most applications we deem that these existing implementations of endogenous system chain code and validation system chain code will be sufficient for most in most applications that we see in the enterprise world to give you a comparison with bit coin. The validation that bit coin does in some sense is the validation whether the transaction is a duplicate or not. So, this is the UTXO model whether the transaction is unspent amongst all the previous transactions that this network has seen right. So, that is the UTXO model all it does is verify whether this transaction is unspent or not.

So, here we are not following the UTXO model. We are following what is popularly called as the state based model. So, the validation is based on state information that we are storing in as part of the ledger and the validation here specifically one part of the validation is the validation of the endorsement policy. So, this is different from what bit coin has and this is very specific to hyper ledger fabric and it is also one of the differentiators of hyper ledger fabric.

(Refer Slide Time: 12:44).

**Endorsement Policy Syntax**

```
$ peer chaincode instantiate
-C mychannel
-n mycc
-v 1.0
-p chaincode_example02
-c '{"Args":["init","a", "100", "b", "200"]}'
-P "AND('Org1MSP.member')"
```

Instantiate the chaincode `mycc` on channel `mychannel` with the policy `AND('Org1MSP.member')`

Policy Syntax: `EXPR[E, E...]`  
Where `EXPR` is either AND or OR and `E` is either a principal or nested EXPR

Principal Syntax: `MSP.ROLE`  
Supported roles are: member and admin  
Where `MSP` is the MSP ID, and `ROLE` is either "member" or "admin"

N-out-of-K policy specification also possible (e.g., 3 out of 5 peers in the channel must endorse)

NPTEL

10

So, looking at a little more detail about the endorsement policy itself, what how do you define it how do you specify it? When you are instantiating a chain code this is just an example of a chain code instantiation, you are providing first the channel on which you are instantiating this chain code mycc is the chain code name you have a version for the chain code.

So, it is possible for you to deploy multiple versions of the chain code that may be improvements of the chain code over time maybe changes in the business logic can be captured. So, the chain code itself can follow full lifecycle and there is a separate system chain code for that a life strike a life cycle system chain cod, that manages the lifes life cycle or the different versions of a chain code so, that is a separate, so, a version of the chain code

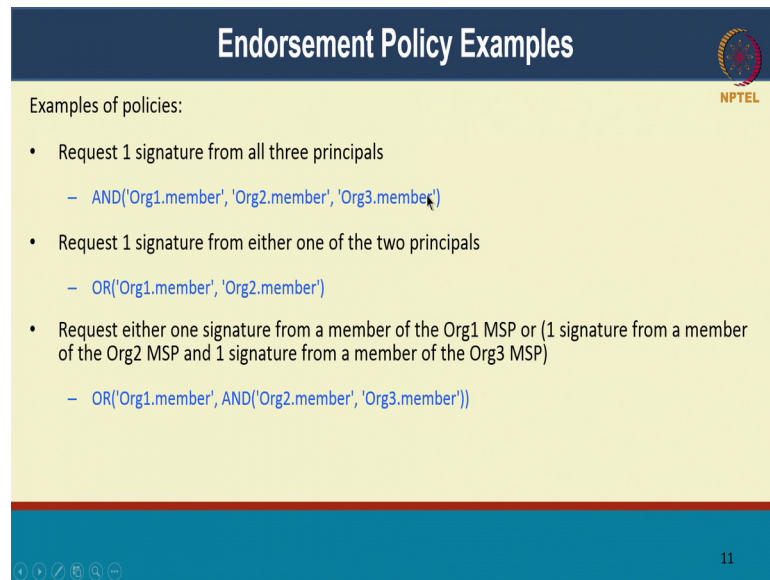
And then you have a particular code that it refers to. So, that is the particular spot contract that implementation and then the input arguments for that for that chain code right. So, these are the initialization arguments. So, it is called the you calling the init function with a certain set of parameters. So, this could be any parameters that you choose to pass at the time of initializing the chain. So, this chain code takes for initialization arguments. So, I think this is this chain code example 0 2 if you are looking up the fabric code and the examples there, it actually sets the account balance of a to be 100 and an account balance of b to be 200. So, that is the initialization that were doing and the – P option here gives you the endorsement policy that is the that is what we are looking at here.

So, what is this endorsement policy says the endorsement policy says that Org1MSP any entity of Org1 can sign this since I can sign this transaction. So, this is the and policy. So, it is possible the syntax for the internal policy is any expression are using AND and OR gates and using any principles or a nested expression. So, you can; I can say Org1MSP.member and Org 2 MSP members.

So, any so, both of these entities must sign and the principles that I am called out here is really a particular role it could be member it could be admin these are the supported roles today and the MSP is a particular MSP ID. So, there is an MSP called Org1MSP and there is a role called member within the Org1MSP and anyone who has the member roles in Org1MSP can sign this can endorse this transaction.

Now it is also possible to have a N out of K policy. So, that is possible to say 3 out of the 5 organizations you know you specify the list of five organization and say any 3 out of 5 if they are signing this transaction, then I will consider that as well. So, the N out K signing is also possible that is also part of the policy syntax.

(Refer Slide Time: 15:49)



The slide is titled "Endorsement Policy Examples" and features the NPTEL logo in the top right corner. It lists three examples of policies:

- Request 1 signature from all three principals
  - `AND('Org1.member', 'Org2.member', 'Org3.member')`
- Request 1 signature from either one of the two principals
  - `OR('Org1.member', 'Org2.member')`
- Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP)
  - `OR('Org1.member', AND('Org2.member', 'Org3.member'))`

At the bottom of the slide, there are navigation icons and the number 11.

Here is just some examples of policies. So, one example could be AND of Org1 member Org 2 member and Org3 member. So, in this case Org1 Org2 Org3 are really MSP identifiers for their respective organizations member is the role within those organizations. So, a member of Org1 has to sign a member of Org2 has to sign a member of Org3 has to sign all of these together now will satisfy the policy.

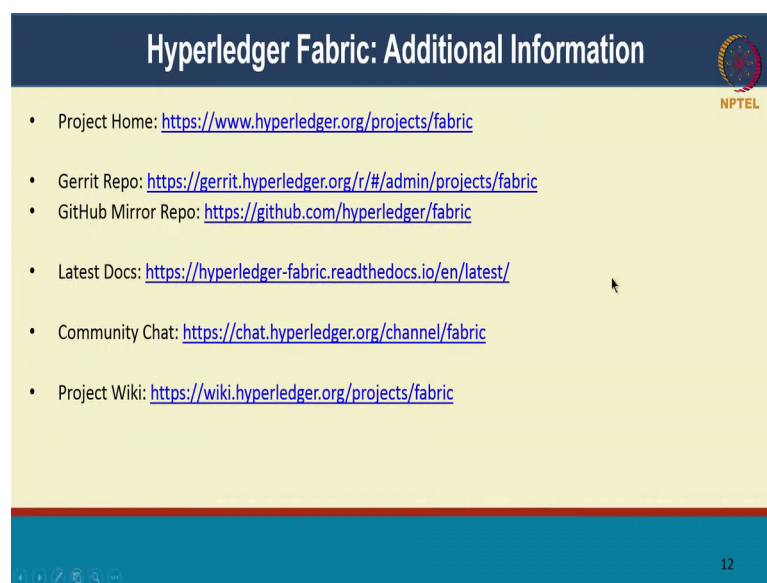
The other policy you can do is or I can say Org1 member or Org2 member must sign this transaction that will be the endorsement and it is also possible for you to include arbitrary combinations of AND and OR like in this example what this example. Here this example says is first let us look at the inside thing Org2 and Org3 must sign it Org1 must sign it. So, that is the condition. So, it is either Org1 or a combination of Org2 and Org3 must endorse this transaction or it to be valid.

So, you can have any arbitrary combinations of this of this endorsement. So, based on for each transaction who has signed it the specific members who have signed it then we will compare as part of the validation system chain code we will compare it with this, this

logic this policy that you have specified and validate whether the policy is satisfied or not. So, this is just a satisfying satisfaction Boolean satisfaction.

And if it is satisfied then we will admit that into well admit that as a valid transaction. So, each at the time of the validation stage each peer is going to execute this and validate whether it is satisfied. So, that again likewise there can be out of K policy also I have not given the example here, but you can look up the documentation for that. So, it is also possible to define say N out of K members in the network have to sign.

(Refer Slide Time: 17:46)



The slide is titled "Hyperledger Fabric: Additional Information" and features an NPTEL logo in the top right corner. It contains a list of six links:

- Project Home: <https://www.hyperledger.org/projects/fabric>
- Gerrit Repo: <https://gerrit.hyperledger.org/r/#/admin/projects/fabric>
- GitHub Mirror Repo: <https://github.com/hyperledger/fabric>
- Latest Docs: <https://hyperledger-fabric.readthedocs.io/en/latest/>
- Community Chat: <https://chat.hyperledger.org/channel/fabric>
- Project Wiki: <https://wiki.hyperledger.org/projects/fabric>

The slide number "12" is visible in the bottom right corner.

So, this actually concludes all of the material for understanding what really fabric is how it is designed how the network setup is made. So, there is a lot of oh that is available for you to quickly understand get to the next level of detail right. So, the homepage for fabric itself is under the hyper ledger project it is available there. You can look up the code and there is some interesting examples that are that are shared along with the code. So, there is a gerrit repo as well as a github mirror of that gerrit repo.

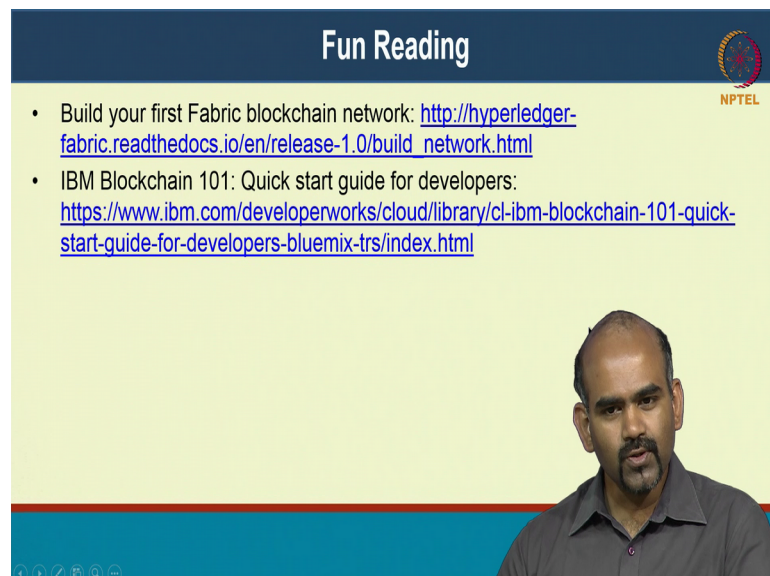
Some of the latest docs this keeps updating based on new features new features are getting added almost on a weekly basis and there are multiple features that are being proposed actively and they are also being developed actively. So, we are right now in version one point one of the fabric version one point two is being planned and there will be a later release later this year there is a community chat.

If you are a developer you want to you have some issues with some understanding or some part of your code is not working you have a defect that you want resolved you can always go to chat and there is a very active community out there that takes an interest in answering in others questions.

So, very active community that that helps all developers mutually so, and there is also a wiki, where you can also find more information over the next few lectures what we will do is we will go through some demos, I will try and show you fabric in action. So, there are many ways in which you can set up fabric you can set it up on the cloud very easily you can set up on your own laptop or on your favorite development environment set of VMS Docker containers wherever you want to set up there are instructions for you to get started very easily. So, you can set it set up your network in a matter of maybe tens of minutes maximum and on the cloud it is just a couple of clicks you can have your you can get going.

So, I will show you some of those and I will also show you how a network is set up maybe show you some examples of how a smart contract is written. So, well go through some of those demos over subsequent lectures.

(Refer Slide Time: 19:56)



The slide is titled "Fun Reading" and features the NPTEL logo in the top right corner. It contains two bullet points with hyperlinks:

- Build your first Fabric blockchain network: [http://hyperledger-fabric.readthedocs.io/en/release-1.0/build\\_network.html](http://hyperledger-fabric.readthedocs.io/en/release-1.0/build_network.html)
- IBM Blockchain 101: Quick start guide for developers: <https://www.ibm.com/developerworks/cloud/library/cl-ibm-blockchain-101-quick-start-guide-for-developers-bluemix-trs/index.html>

In the bottom right corner, there is a video inset showing a man with a beard and a dark shirt speaking.

So, to conclude some of the again the fun reading section some of the additional information you can get. There is very easy instructions to if you are a do it yourself person kind of person very easy instructions for you to set up your own network in any

kind of development environment, any kind of operating system that should be. There is also as a quick start guide for developers on how to you; how you can get started having very easy instructions for you to get started. So, with that thank you that concludes our lectures on hyper ledger fabric, see you in the next set of lectures for seeing some demos of fabric in action.

Thank you.