

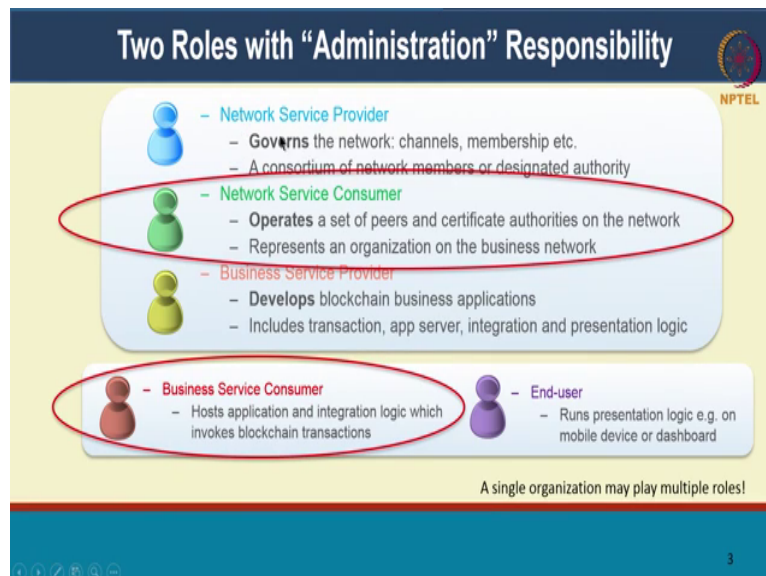
**Blockchains Architecture, Design and Use Cases**  
**Prof. Sandip Chakraborty**  
**Prof. Praveen Jayachandran**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 29**  
**Hyperledger Composer – Network Administration**

Hello everyone, welcome back to the next lecture of our Blockchains course. We will be looking at Hyperledger Composer. We went through some of the concepts and composer how you can easily develop an application on top of composer, and how that connects to a fabric network. So, in this lecture we going to talk about how you would manage your composer network itself, how do you built it up how do you manage it, what are the key concepts and managing a network, and how composer makes an easy for you to manage your network right. Some of the concepts in fabric, in terms of governance of the network in terms of channel administration and all are made easier for you in composer.

So, the main role that is going to be using this network administration as the network service consumer role.

(Refer Slide Time: 00:54)

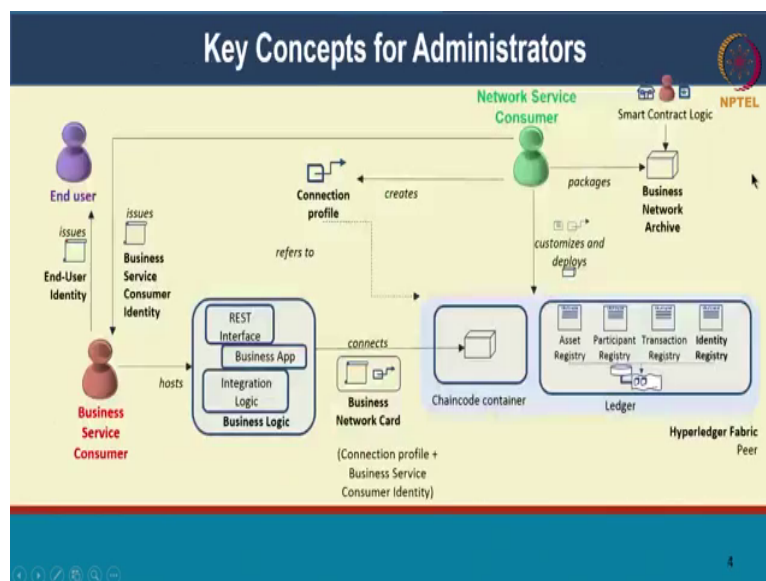


So, as I mentioned this someone who is going to be governing the network that is one role, this governance again I am saying one role, but it could be multiple organizations that all serve this role. They are going to be managing who joins the network who is participating

and so on channels and all that. Network service consumer is going to be operating a set of peers. So, within an organization let us take each organization we have a network service consumer. They going to manage the peers certificates setup the network join the channels and so on.

So, that is the network service consumer. So, that will be the main role who will use these functionalities provided by a hyperledger composer. We looked at the business service provider before, they were the once who are developing the application. And the other consumer is going to be the once who are hosting applications and building applications on top of the network that is provision. So, that is the business service consumer role. So, these are the 2 roles we were going to be using, some of the features that I am going to be talking about in this lecture. And like I mention although I we have abstract these into rules one organization could play multiple of these roles.

(Refer Slide Time: 01:59)



So, what are the some of the key concepts here for your network administrators? Let us start with the network service consumer. So, this is a person who is going to be setting up the set of peers, maybe managing the membership of users and their organization. So, what are the some of the things they going to do. The business service provider has developed plot contract logic. So, they going to be packaging that smart contract logic in some way, and we going to call that the business network archive. So, we will talk about what that is how this whole thing is getting packaged up, so that is the business network archive.

Once that is packaged it is going to get deployed on to the fabric network. So, I mention that composer runs on top of fabric. So, once we have packaged up what your business network specification is; you going to deploy that on to the fabric network. So, each peer will get a copy of that. There is a composer runtime which is a chaincode container. So, it is a the special container that composer implements and this container is going to interpret the business network definition or business network archive all the assets participants transactions that you have to find for your network. So, that is all is deployed on to the fabric network.

Now, that talks about this smart contract logic itself, apart from this the network service consumer is also going to create what is called a connection profile. So, this connection profile is going to talk about what the network is, how should users connect to this network, and it gives you details about who are the peers who are the orderers, what channels it gives you all of that information that is the connection profile. Now, that is the network service consumer you going to define the smart contract logic deployed on the hyperledger fabric peer or peers, and they also create the connection profile. Now end users these are the business service consumer this might be applications, it might be individual end users, who are going to be invoking transactions that are defined in the smart contracts.

They all of course, need some notion of identity right. So, there is a notion of identity so they going to be issued identity. So, the end users have let us say certificates composer gives you easy way for you to create all of that; abstract away the notions of certificate for you do not have to deal like that level you only have to deal with the level of let us say, a member and identifiers for that member. Then the business service consumer is also going to host the application logic. So, this could be a rest interface that users endusers can invoke to say I want to perform this transaction on blockchain. It could be a business application logic that runs there this is not the smart contract logic.

But any application logic that you want to run on top and you could also have the integration logic with other systems that you work with. And this whole application is going to connect with the fabric network and the smart contract that you deploy, how does it connect? There are 2 pieces of information that needs to connect. The first piece of information is the identity of the user performing the transaction. So, that is the consumer identity. So, that needs to be there you also need to be have the connection profile so that is the connection profile I talked

about. So, you need the identity and you need to know which peers to connect to which channel and so on.

So, think of it in if the equivalent in the fabric domain is whatever information is there in your SDK for your hyperledger fabric client, you need to provide information you need to have a wallet that has your private key for signing a transaction that is your consumer identity. You need to know which peers to connect to which orderer, which channel to connect to all that is known in your hyperledger fabric client. So, the same information is going to be packaged up into a connection profile and consumer identity. And together this is going to be called a business network card.

So, this business network card will be available with each end user, there will be using their personal card to connect with the blockchain authenticate with it sign and submit transactions to blockchain, and the blockchain of course, well validate and accept their transactions so that is the overall flow. So, we going to look at some of these concepts in more detail, what is the business network archive, what is the business network card so all of this how is the identity created. So, all of these we going to be talking about in the course of this lecture.

(Refer Slide Time: 06:12)

**Network Service Consumer packages resources in a BNA file**

- Business Network Archive (BNA) is a package of the resources used by Fabric:
  - Model files (.CTO)
  - Transaction processors (.JS)
  - Access Control Lists (.ACL)
  - Static queries (.QRY)
  - Documentation and versioning (.MD)
  - It does *not* contain the client application
- The BNA simplifies deployment of blockchain and promotion between environments
  - c.f. TAR, WAR, EAR, JAR, BAR...
- Create BNA files from Playground or command line
  - Build from filesystem or NPM module

**Business Network Archive**

model.cto   logic.js   permissions.acl

queries.qry   readme.md

```
composer archive create --archiveFile my.bna  
--sourceType module --sourceName myNetwork
```

Now let us talk about the business network archive. So, this is BNA file. So, the BNA file is consists of the following.

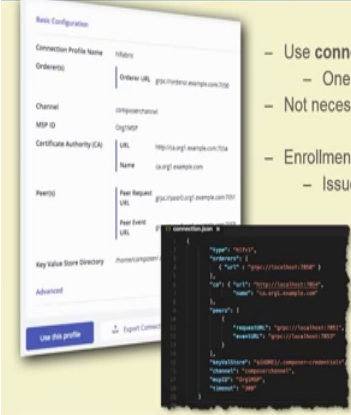
Right, all of these we saw we talked about in the previous lecture. It includes the model files it is called the C T O files. So, this includes your data elements so this is the assets participants all of them are defined in your model files, then there is a transaction processes this is your JavaScript function which captures the business logic to be placed in your smart contracts. So, this is your transaction processes, you can define your access control list and any static queries. So, all of these are captured in different files and in addition to all of these files you can also add any documentation you want. So, let us say someone else queries your contract, business network they can understand what this application is doing.

So, in plain text in actually a dot MD file you can define the documentation for your application. All of these are packaged together into a BNA file, and that is going to be your business network or the application that you have to find on composer. And this significantly simplifies the deployment of blockchain between environments. So, you can package this in tar in a war file ear and so on. So, it really across different environments depending on what you are comfortable with all will be the BNA file and simplify the deployment for you. And it is possible for you to create BNA files from the playground itself. So, while you are playing with it you can create some sample models in the playground.

You can store that as a BNA file load it up later and modify it or you can take it to a different application and use it there. And you can build something in the playground and actually take that to a real network deployment with fabric and use it there right. So, you can build this from the playground build it from file system or through an NPM module so all of that is possible.

(Refer Slide Time: 08:05)

### Connection Profiles to Hyperledger Fabric



The image shows a slide titled "Connection Profiles to Hyperledger Fabric". On the left, there is a screenshot of a web-based configuration interface for a connection profile. The interface has a "Basic Configuration" section with the following fields: "Connection Profile Name" (value: fabric), "Orderers" (value: OrderersURL: grpc://localhost:7051), "Channel" (value: channel), "MSP ID" (value: Org1MSP), "Certificate Authority (CA)" (value: URL: http://localhost:7054, Name: ca.org1.example.com), "Peers" (value: Peer0URL: grpc://localhost:7051, Peer1URL: grpc://localhost:7052), and "Key Value Store Directory" (value: channelcomposer). There are "Use this profile" and "Export Connection Profile" buttons. On the right, there is a list of bullet points: "Use connection profiles to describe Fabric connection parameters" (with sub-points: "One connection profile required per channel", "Not necessary for web-based simulation"), "Enrollment in Hyperledger Fabric network required (see later)" (with sub-point: "Issue Fabric identity from Composer participants"), and "Connection profiles currently used by Composer only" (with sub-point: "Plans to implement common connection profiles that can be used by both Fabric and Composer"). Below the text is a screenshot of a JSON file representing the connection profile configuration.

- Use connection profiles to describe Fabric connection parameters
  - One connection profile required per channel
  - Not necessary for web-based simulation
- Enrollment in Hyperledger Fabric network required (see later)
  - Issue Fabric identity from Composer participants
- Connection profiles currently used by Composer only
  - Plans to implement common connection profiles that can be used by both Fabric and Composer

That was a BNA file now coming to the connection profile right. So, this I mentioned is a capturing the network configuration itself the peers, orderers, channels and so on so how is this done?

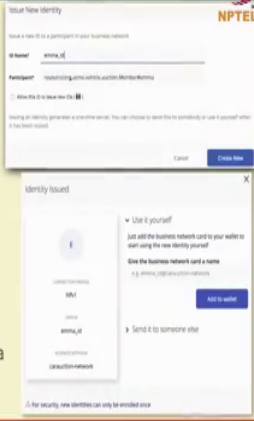
You need one connection profile per channel on your fabric network. So, depending on the number of channels you will be creating as many connection profile. For the web based simulation you cannot have the connection profile all you need is your BNA file. You can just say these are my models. These are my transactions, and access control rules you do not need anything else, because it is not actually connecting with to a actual channel the backend it is only meant as a playground. You are required to enroll users into the fabric network and it has to have a fabric identity, for each of your composer participants. So, we will talk about identity in the next slide, but the connection profile is mainly about the peers and the channels used.

So, if you look at this is example here it says these are the orderers, these are the MSP, this is the certificate authority, these are the peers in the in this network, and it gives you the key value store directory. So, all of this is captured in your connection profile and here is an example Json, which gives you all of these the same attributes is captured as a Json. Here it currently is used only by composer, but in future it is also possible for fabric to also consume this, but right now fabric does not consume this asset, but it could be in the future.

(Refer Slide Time: 09:25)

## Participant Identity

- The **Network Service Consumer** issues network participants with an **identity** in order to connect to Hyperledger Fabric
  - Issued as a Hyperledger Fabric userid/secret
  - Automatically swapped for a certificate on first use
  - Packaged in a Business Network Card and supplied when the client application connects
- Composer Participant to Fabric Identity mapping is stored on the blockchain in an **identity registry**
- Usually, only **Business Service Consumers** have a Fabric identity
  - **End-users** log in to the business application using a separately managed identity; blockchain transactions invoked by proxy
- Manage identity from Playground, Javascript, REST or command line
  - For example: Test connection, issue identity, bind an identity to a participant, revoke an identity, list identities



The screenshot shows two overlapping windows from the NPTEL interface. The top window, titled 'Issue New Identity', has a 'Create New' button. The bottom window, titled 'Identity Issued', shows a list of identities with columns for 'Name', 'Secret', and 'Action'. The 'Name' column contains 'anna\_01' and the 'Action' column contains 'Send it to someone else'. There is also a 'Use it yourself' section with an 'Add to wallet' button.

So, the other things you need apart from knowing which peers, orderers to connect to you also have to have an identity on blockchain.

Again composer makes this easier for you. It creates you can create network participants with an identity to connect to hyperledger fabric. So, is issued as a hyperledger fabric user id and secret. And it automatically is wraps it in a certificate whenever you are using it first time right. So, the certificate creation is taken care of for you and this is together with the connection profile it is packaged into a business network card and supplied when the client application is connected. So, like I mentioned every client trying to connect to the blockchain network has to have this business network card that has both this identity as well as the connection profile.

Now, the composer participant to fabric identity mapping so when I said I you are creating a new user in composer be automatically create the id for you in fabric and the certificate is also generated for you. The mapping between the fabric id and the composer participant is also stored as a transaction in the identity registry that the composer on time maintained. So, this is also maintained in blockchain also right, in a descent class fashion. The only the business service consumers have a fabric identity end users for instance can be extracted away by application logic and they can they might just have an application identity.

So, they might not have a identity themselves on blockchain, only the business service consumers would have an identity. So, the end users who do not have an identity will be





multiple of these cards through command line as well as playground. Then you are using these cards to connect to fabric from whether it is playground or command line or from your application, your JavaScript we will be using these cards to connect right.

So, here is a just a small example of how you would connect to hyperledger fabric using a business card.

(Refer Slide Time: 12:54)

The slide is titled "Systems of Record Integration" and features a list of bullet points on the left and a screenshot of an "angular-app" interface on the right. The interface shows a REST API explorer with endpoints for Auctioneer, CloseBidding, Member, Offer, and Vehicle. The Vehicle endpoint is selected, showing a request URL and a JSON response body.

- Domain specific APIs very attractive to mobile and web developers. Resources and operations are business-meaningful
- Composer exploits Loopback framework to create REST APIs: <https://loopback.io/>
- Extensive test facilities for REST methods using loopback
- Secured using JS Passport, giving >400 options for authentication
- Composer provides back-end integration with any loopback compatible product
  - e.g. IBM Integration Bus, API Connect, StrongLoop
  - Outbound and Inbound (where supported by middleware)

The screenshot shows the "angular-app" interface with the following endpoints:

- Auctioneer: A participant named Auctioneer
- CloseBidding: A transaction named CloseBidding
- Member: A participant named Member
- Offer: A transaction named Offer
- Vehicle: An asset named Vehicle

The selected "Vehicle" endpoint shows a request URL of `http://0.0.0.0:3000/api/Vehicle` and a response body of:

```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "id": "1234",
  "owner": "odowd@ibm.com"
}
```

Composer helps you integrate with existing systems of record and it helps you create domain specific APIs. So, this is automatic generation of APIs from the model files that you are created. Based on the assets you created composer automatically can generate APIs for you, and it uses the loopback framework for this. So, any application any other system of record that leverages loopback can be integrated with this. And it also provides you extensive test facilities and with loopback it also creates you creates a swagger API document for you.

So, you can go to swagger and open up your set of APIs and perform your rest queries or rest invocations. And this whole thing is secured using JS Passport, this is a very popular tool and it has over 400 options for authentication. It provides a backend integration with any loop back compatible product. Today we have connected with integration bus and API connect so we will actually node RED so we will talk about that.

(Refer Slide Time: 14:00)

### Exploiting Loopback: Examples

**IBM Integration Bus**

- IIB V10 contains Loopback connector
- Example above takes input from file, SAP or MQ
- Data mapping from CSV, BAPI/IDOC or binary form to JSON model definition

**Node.RED**

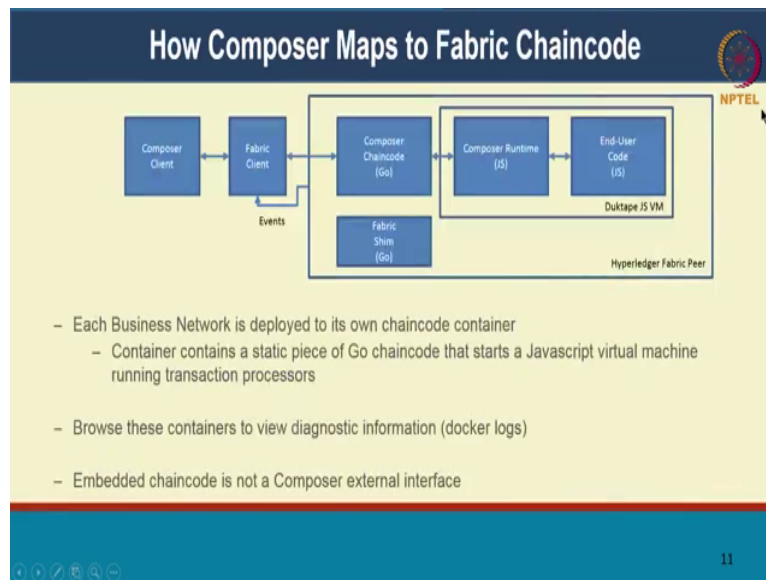
- Pre-built nodes available for Composer
- Connect to hardware devices, APIs and online services
- Install direct from Node.RED UI
  - Manage Palette -> Install -> node-red-contrib-composer

10

So, here is the example of exploiting loopback. So, the integration so if I take IBM integration bus it has a loopback connector. So, we have implemented composer capabilities as a loopback connector.

So, it can take a sample file from sample form SAP your MQ and you can connect that to your composer transactions. So, you can easily map data from let us say CSV file to Json model definition in composer. Likewise, it we also have a loopback connector for node RED. And it gives you pre built node red pre built nodes available for composer, and you can use that to connect to hardware devices may be these are IOT devices that you can connect to a APIs and other services and this is directly available from the node red UI can install it directly from there it is very, very easy to use.

(Refer Slide Time: 14:55)



This is going down a level of detail to see how composer actually maps this into fabric. So, I told you composer runs on top of fabric so how does it do it? So, the composer client itself is actually going to be talking to a fabric client, which is going to be using fabric SDK to connect to a special chaincode that composer executes. So, this chaincode has logic to start up a java virtual machine inside fabric inside the peer. And this java virtual machine is going to be running a JavaScript runtime, and what is runtime is going to do is it is going to interpret any node contract logic you would place there

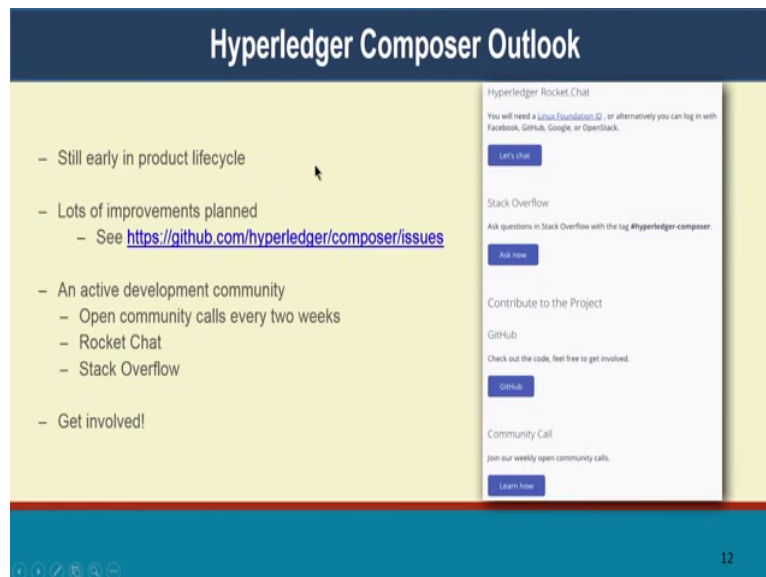
So, maybe explaining from the other way around all of these components are pre deployed when you are deploying composer. When you are deploying a smart contract all your adding is this end user JavaScript code, these are your transaction files your module files all of that are getting deployed as a smart contract on to hyperledger fabric. A composer runtime is going to interpret that JavaScript logic. And all of this is contained within a chaincode that is connected to this JavaScript peer.

And there is of course, it is connect to the peer, the chaincode connects to the fabric peer and connects to the ledger from there. So, this whole runtime is how your code that is written in a higher level business abstraction is getting interpreted as part of chaincode chaincode execution and fabric and is decentralized in it is execution across all the nodes in the fabric network. And each business network that you are defining will be deployed in it is own chaincode container. So, it is isolated in that way and it has a static piece of go and chaincode

that is going to be executing this. And it is possible for you to look through docker logs and browse these containers to view any diagnostic information.

So, if something fails or something you can quickly go lookup docker logs to see why it failed. And you can debug from there and the embedded chaincode this chaincode itself is not exposed as an interface to the outside world. So, this is actually integrated into fabric as and it is deployed into fabric, and that does not have a immediate interface to the outside world. So, the way you connect to it will be through the fabric client and the SDK to connect to this chaincode.

(Refer Slide Time: 17:10)



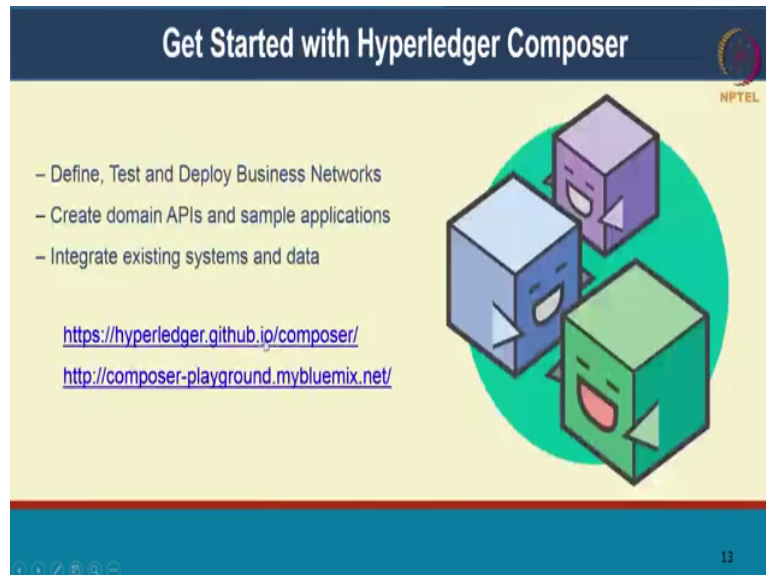
The slide is titled "Hyperledger Composer Outlook" and is divided into two main sections. The left section, on a yellow background, lists several points: "Still early in product lifecycle", "Lots of improvements planned" (with a link to <https://github.com/hyperledger/composer/issues>), "An active development community" (including "Open community calls every two weeks", "Rocket Chat", and "Stack Overflow"), and "Get involved!". The right section, on a white background, features four promotional cards: "Hyperledger Rocket Chat" (with a "Let's chat" button), "Stack Overflow" (with an "Ask now" button), "Contribute to the Project" (with a "GitHub" button), and "Community Call" (with a "Learn how" button). A blue footer bar at the bottom contains navigation icons and the number "12".

So, composer is inactive development just 2 or 3 months back end of 2017, we actually released the first version the V 1 version fraction version of hyperledger composer and there a lot of improvements that are still being planned.

And they are active (Refer Time: 17:27) these are active discussion in the community about prioritization of these improvements. And there is active development going on to improve some of these features and hyperledger composer. And it is a open community they run a call every 2 weeks. So, anyone is was is invited to participate and contribute to the community, and just like hyperledger fabric they will very active rocket chat. So, it is a online chat forum where you can go ask doubts you can help others you can answer questions for others, there is also stack overflow where people ask questions and people answer there as well.

And I would encourage all of you to get involved right. So, it is a great way for you to start building your face blockchain applications.

(Refer Slide Time: 18:08)



**Get Started with Hyperledger Composer**

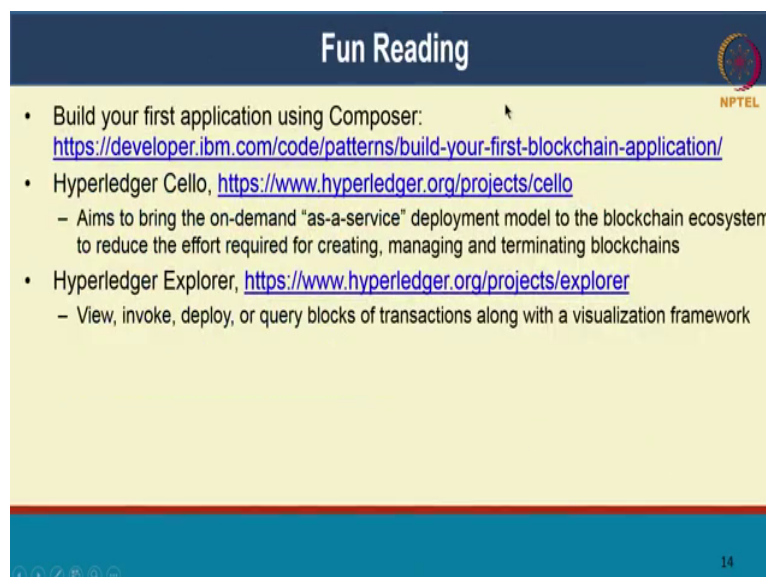
- Define, Test and Deploy Business Networks
- Create domain APIs and sample applications
- Integrate existing systems and data

<https://hyperledger.github.io/composer/>  
<http://composer-playground.mybluemix.net/>

13

Here are some quick links can go lookup composer this is a top level page for the GitHub and composer and you can go through the code. You can go to the playground where you can quickly try out their creating and application yourself.

(Refer Slide Time: 18:21)



**Fun Reading**

- Build your first application using Composer: <https://developer.ibm.com/code/patterns/build-your-first-blockchain-application/>
- Hyperledger Cello, <https://www.hyperledger.org/projects/cello>
  - Aims to bring the on-demand "as-a-service" deployment model to the blockchain ecosystem to reduce the effort required for creating, managing and terminating blockchains
- Hyperledger Explorer, <https://www.hyperledger.org/projects/explorer>
  - View, invoke, deploy, or query blocks of transactions along with a visualization framework

14

So, for the fun readings section of this course we for this lecture; how do you build a first application using composer this is a very simple tutorial that you can follow it is an developer

works in IBM so you can check that out. And apart from that I think there are I would like to highlight 2 other hyperledger projects that could also be very useful for you in your development journey right as a developer. One is hyperledger cello right, cello is the hyperledger project that aims to bring as a service deployment model for blockchain ecosystem.

So, currently I think they are supporting only fabric if I am not mistaken and it helps you reduce the effort required to create manage and terminate blockchain networks. So, it makes very easy for you to join a network create a channel and so on for fabric. Hyperledger explorer is another very interesting project. It gives you a good visualization tool for you to look at you are blockchain itself. So, you can query the blockchain you can find out what transactions are there, you can invoke deploy, look at blocks of your blocks of transactions and it has a neat visualization.

So, you can follow transactions that around from one transaction to another in a nice pluggable visualization framework. So, I would encourage you to look at both of these projects and for any application you might develop you could use both cello and explorer. Cello will help you set up the network and manage it, explorer will help you visualize what is happening in your network and monitor what transactions are going on. I would explore I would encourage you to explore at least these 2 projects there are, at least half a dozen other hyperledger projects.

And we will also be talking about some of the other blockchain platforms later in the course with that.

Thank you thanks for your time.