

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Introduction

This module covers the setting up of the Fabric Network for a new problem related to the education sector: the creation and verification of certificates.

The topics covered in this session are as follows:

- Step by step process of setting up an entire Fabric Network
- Look at all configuration files involved in the network setup
- How to customize the configuration files for our use case

After completing this module, you will be able to:

- Model the network based on problem statement
- Build the network in the local computer
- Ready the network for chaincode development

Prerequisites:

- Docker
- Docker compose tool
- NPM

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

The Problem Statement

Consider a scenario of creation and verification of certificates by an education institution. Take the example of IIT as a college which provides education and also issues certificates to students upon completion of the course. These certificates are used by the students to be presented to prospective employers in the industry. These companies will have to spend a lot of time, resources and money to verify the authenticity of these certificates and the grades through third party organizations. This process also lacks the efficiency of digitization which is prevalent across many other sectors. A regulator, for example the Ministry of HRD, is introduced as a solution to this issue. The regulator will ensure that the entire process of exchange of certificates follows certain ethics. The regulator will need visibility of all the certificates being issued and the way it is being used in the entire country. An employer could be any organization like upGrad and the process of validating the certificates will be taken up by the HR of the respective companies during the selection process.

We model this three party certification process using Blockchain to see if this will help in getting certificates from IIT as a college, to be verified by upGrad as a employer and at the same time for MHRD to have complete visibility into the issuance and verification of this entire certification process. Since all the certificates issued will be on the Blockchain as a hash, these certificates can be instantly verified against the original and any manipulations can be immediately identified. For example, when a student presents a certificate to upGrad, it can instantly verify it using the hash to see if this matches and know if the certificate is genuine. As this happens, MHRD can view this entire process in detail and also get to know how the certificates are being used by the students and also the companies in the country.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Following is a model of the user flow that makes the entire process completely auditable and tamper evident.

1. Student sign up process
2. Certificate issuance by the college
The certificate is recorded on the Blockchain. The student details and also the hash of the certificate will be recorded on the ledger
3. Verification process of the certificate once the student applies for a job to the employer
The employer will have some interface to look at the certificate and click to complete the verification instantly. This application will interface with the Blockchain network to fetch and check the original certificate with what is being presented by the student
4. View, audit: MHRD will have the ability to view and audit all the certificates that have been issued and also the verification process
This will also enable analytical data collection on the rate of certification across the country, the usage of these certificates by the employers

Why a Blockchain solution will suit this problem?

There is inherent lack of trust among the stakeholders in the process. The colleges will be competitive and care for the privacy and security of the student data. The colleges would not want all of the student data to be stored in a central database. A central database like the DigiLocker is highly susceptible to data leak and also presents a single point of failure. This will also make a global interface difficult. The employers will also be competitive and be interested to instantly fetch and verify the data that the student shares with them.

As in this Blockchain as a solution, the colleges don't have to put the actual data on the system. Since the Blockchain is tamper-evident, only the hash of the certificates can be put on the chain. This ensures protection of the privacy and security of the original data and identifiable student records. Since this data is spread across several nodes on the network, it avoids the possibility of a single point of failure.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Structure Of The Application

Firstly, the Fabric Network will have the three stakeholders, the colleges represented by IIT, the upGrad which is the employer and the MHRD which will act as the regulator. Each stakeholder will contribute two peers which will be dedicated computing resources to run this network. For example, IIT could deploy two peers one for the students and one for the teachers. upGrad could deploy one for the HR and one for its management. Similarly, MHRD could deploy one for the analytics team and the other for its admin team. There will be a single channel on this network called as the '*certification*' channel. All peers will be a part of this channel and all the data that is shared will be available on this channel. Each organization will have its own MSP. There will be a chaincode or a smart contract. This will define the logic and functions to enable a student join a course, for the colleges to issue a certificate and for the companies to verify the certificates. This contract will be deployed on the channel. All the peers will be committers. The chaincode will interface with the ledgers on the peers. One peer from each organization will act as anchor peers and the peers will also host the chaincode which will become the endorsers on the network.

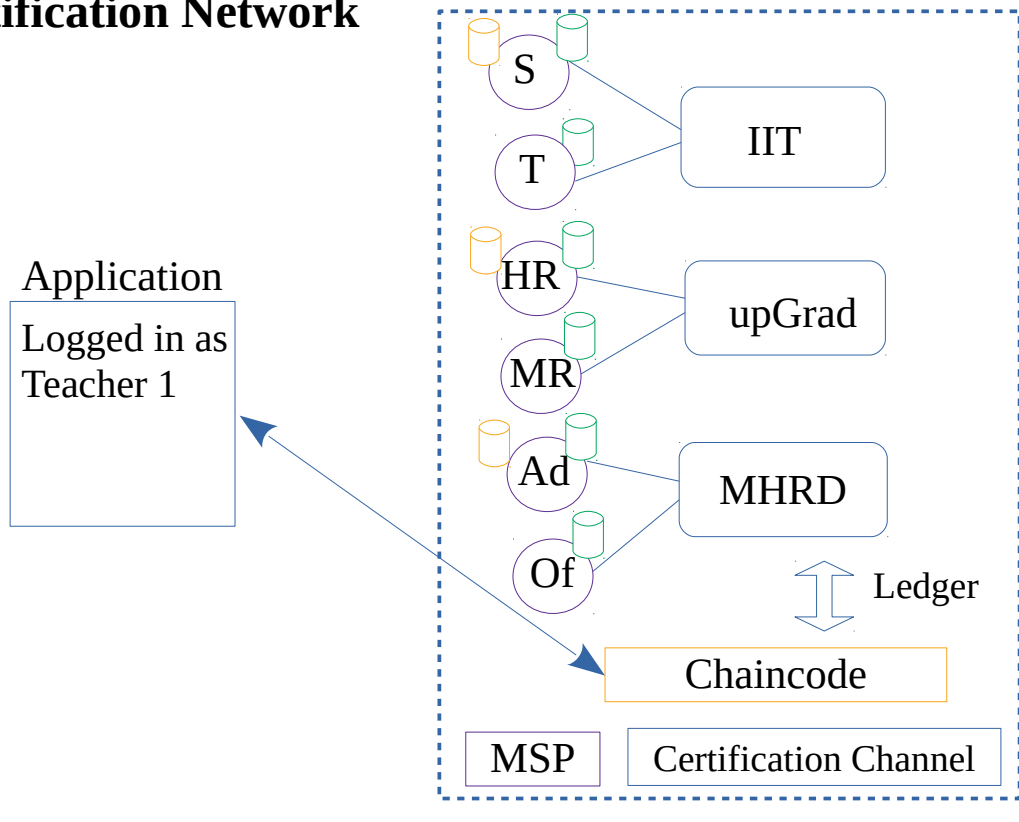
The front end application will interact with the Fabric Network and it could be a web or a mobile interface. It will have a sign-in and the users will be able to access the data on the network based on the roles defined in the MSP.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Certification Network



In summary, the structure will consist of the following:

- A network involving three organisations, each contributing two peers,
- A single channel,
- One MSP per organisation,
- Chaincode, consisting of all the functions, which will be deployed on the channel, and
- An application that will be responsible for interacting with the Fabric network.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Network Setup

There are three organizations in this network: IIT, upGrad and MHRD. Each of these organizations contribute 2 peers to this network. Each peer would be a computing resource which will form the distributed network. Each peer would be owned and controlled by the respective organizations. Hence this network is distributed and co-owned by all of the organizations on the network. Any combinations of peers is possible even where certain organizations contribute more peers than the others. There are also certificate authorities (CA's) hosted by each organizations to manage the certificates and credentials required for the users, peers and applications for the respective organizations. The organizations will also own and operate separate MSP's and hence this will also be a distributed ownership model.

All the peers will join a single channel named *certification channel* where all the data will be exchanged. The ordering service will be a single orderer using Solo. This single node can be owned by any of the organizations. In a real life production scenario, multiple orderers could be configured with distributed ownership by using the Kafka service.

To build this network on a local network, each of these computing resources must be made to run on a single machine. This is enabled by the containerization provided by Docker. Docker will be used to start the peer instances, the 3 CA's, the orderer and also a CLI container which will be used to interact with all of the other containers. The CLI enables connection to each of the other containers and pass on specific commands to be executed on them.

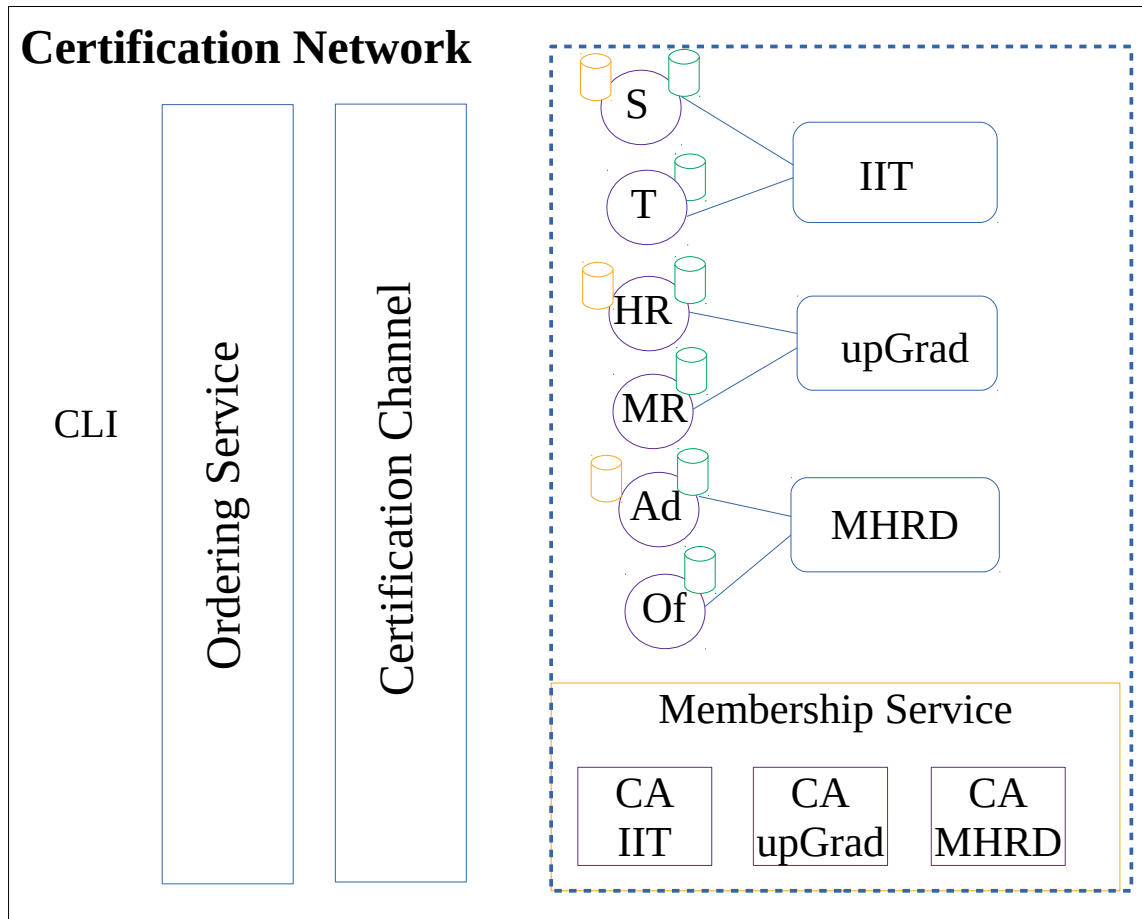
Steps to deploy all the Docker containers on the local computer:

1. There are 11 Docker containers for this network which have specific role and have configurations accordingly. Customized, preset Docker images will be used for the various roles.
2. Crypto materials which are essentially credentials to identify each element in the network.
3. Fabric processes like creating a channel, defining Anchor peers etc. on top of these Docker containers.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP



1. Pre-setup phase
 - Crypto-config generation
 - Channel artifacts creation
2. Network setup phase
 - Docker network creation
 - Channel creation
 - Peer joining
 - Anchor peer updation for each organisation

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Introduction To YAML Language

YAML is an abbreviation, which stands for 'YAML Ain't Markup Language'. It is generally used for configuration files. It is easily human readable and user-friendly. YAML uses the '.yaml' or '.yml' extension for saving YAML files, and it defines data in terms of key-value pairs. YAML is used for all the configuration files using Docker containers to develop the Hyperledger Fabric Network.

Indentation is very important in YAML as in the example below:

employee:

name: "Herald"

empolyee_number: 1004

designation: "Blockchain Developer"

tax_paid: true

prog_languages: ["Node.js", "JavaScript", "Solidity"]

Crypto-Materials

Pre-setup phase: crypto material generation

A tool named *cryptogen* is used to create crypto materials. This tool will have been installed along with other binaries of Hyperledger Fabric. Crypto materials are certificates or public-private key pairs to identify the various organizations or other entities in the network. It is mandatory for Fabric to identify all the Docker containers which form a part of the network. The cryptogen tool takes as argument a configuration file written in YAML to generate the certificates required for the various components of the network.

These crypto materials will be used in the Fabric network configuration files and also in applications which interface with this network.

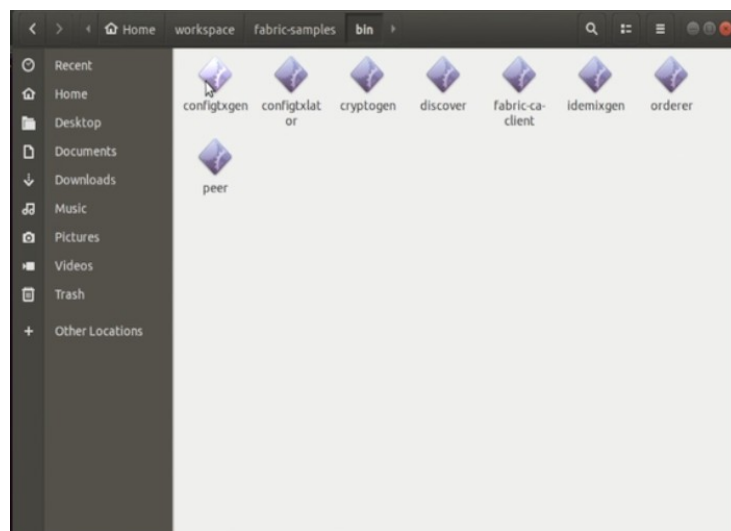
HYPERLEDGER FABRIC & COMPOSER

MODULE 2

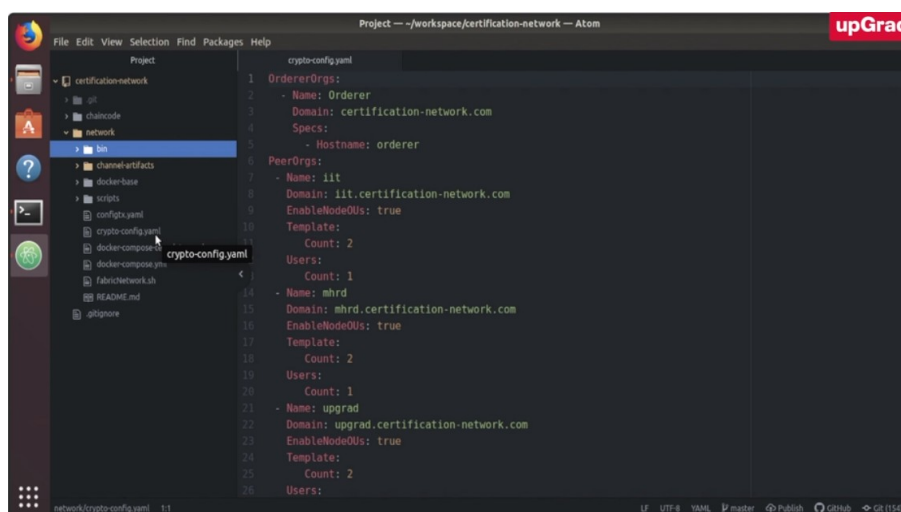
FABRIC NETWORK SETUP

Generating Crypto-Materials

When Fabric was installed or if the setup that comes along with the Virtual Box is being used, a Fabric-Samples directory would become available which will contain the binaries that will be needed to run the various commands.



The folder structure of the 'certification-network' project consists of the 'chaincode' folder which contains the smart contract and the 'network' folder which has all the configuration files for the Fabric network. The network folder has the 'crypto-config.yaml' file which is the YAML configuration file to generate the crypto-materials.



HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Following is the structure of the crypto-config.yaml file to generate the crypto material for the network

a. The orderer organization, which has the following components:

- Name of the orderer
- Domain name of the network for the orderer
certification-network.com will be the domain name we will use
- Hostname of the orderer

Note: A domain name is a group of hostnames. For example, google.com is a domain name. So addresses like gmail.google.com, maps.google.com, etc., are hostnames of the google.com domain.

b. The peer organizations, which has the following components:

- Name of the peer
- A domain name for all the peers
standard way of naming peer is:
organization<dot>domain of ordering service<dot>com
example: iit.certification-network.com

- Template

This is the number of peers inside an organization with the naming convention *peer0<dot>org<dot>domain<dot>com*

- Users

Number of users for every host created as in the template. By default an Admin user is created for every peer. This is the number of additional users apart from the Admin

- EnableNodeOUs: OU stands for organizational units. The certificates that will be generated using cryptogen tool have a property called organizational units. Suppose there is an organization with an admin department and several different units under that department like HR, Tech, Finance, etc. All the different units under the admin department would be identified using organizational units of their certificates.

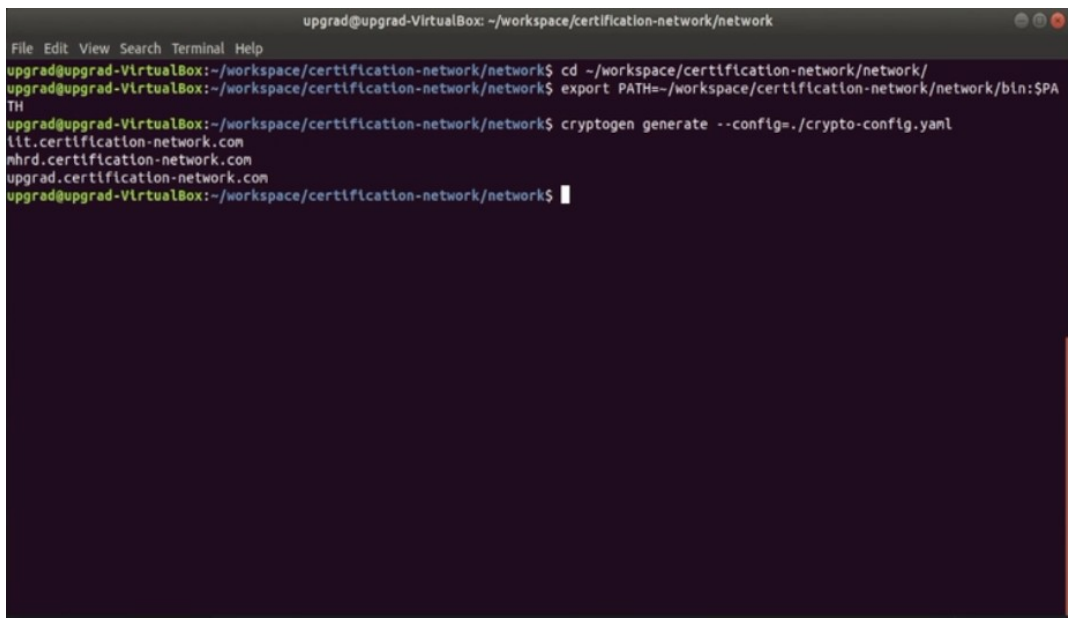
HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Following are the steps to run the cryptogen command:

- In the terminal window, navigate to the network directory
- use *export* command to define the PATH variable to the bin folder
- run the *cryptogen* command by passing the YAML configuration file

A terminal window screenshot showing the execution of the `cryptogen generate` command. The terminal title is `upgrad@upgrad-VirtualBox: ~/workspace/certification-network/network`. The command prompt shows the user navigating to the network directory and setting the PATH variable to include the bin directory. The `cryptogen generate --config=./crypto-config.yaml` command is executed, resulting in the creation of a `crypto-config` directory containing subdirectories for `ca`, `msp`, and `peer`. The terminal output lists the generated certificates for `l1t.certification-network.com`, `mhrd.certification-network.com`, and `upgrad.certification-network.com`.

```
upgrad@upgrad-VirtualBox: ~/workspace/certification-network/network
File Edit View Search Terminal Help
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ cd ~/workspace/certification-network/network/
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ export PATH=~/workspace/certification-network/network/bin:$PA
TH
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ cryptogen generate --config=./crypto-config.yaml
l1t.certification-network.com
mhrd.certification-network.com
upgrad.certification-network.com
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$
```

A new folder named *crypto-config* will have been created. This will have the certificates for the following:

- Orderer organizations such as CA, MSP (Admin, CA, TLS, Users), Orderer, TLSCA and Users
- Peer organizations such as IIT, MHRD, upGrad as per our YAML configuration file

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Structure of a Peer Organization with an example:

Let's take an example to understand the structure of peer organisations further. Suppose you work in a company called solutionFactory that builds software solutions. There are three teams in your company: the development team, the testing team and the deployment team. Now, you wish to create a Hyperledger Fabric network for your company. Suppose all these teams act as peer organisations in your network. The development team has two roles: tech lead and software developer. Now, there are two tech leads and two software developers in the organisation. So, how do you define the development team as a peer organisation in your crypto-config.yaml file?

Suppose you name the development team TechTeam and choose the domain name "techteam.solutionFactory.com" for this team. Let's now define the peer organisation.

- Name: TechTeam

Domain: techteam.solutionFactory.com

In this domain name, 'tech team' refers to the development team and is appended to the name of the parent company, i.e., solutionFactory.com

To define the remaining features, such as the roles in the team and the number of people, you will have to enable a flag called EnableNodeOUs. The value of the flag needs to be true for defining the roles inside the peer organisation. Otherwise, the network will take the default structure. So, this is how our structure will look:

- Name: TechTeam

Domain: techteam.solutionFactory.com

EnableNodeOUs: true

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Now we need to define the roles in the tech team. We will use a list item called template for this purpose. The template defines the number of different peer nodes inside one peer organisation. In our example, there will be two peers in the development team: one, the software developer and another, the tech lead. Therefore, we will set the count of templates to 2 as follows:

- Name: TechTeam
Domain: techteam.solutionFactory.com
EnableNodeOUs: true
Template:
 Count: 2

Finally, we need to set the number of users in each peer. Here, a tech lead is acting as a peer, and the number of people employed as a tech lead is the number of users. We have two tech leads and two software developers, which means that we have two users per peer. So, the count of users shall be set to 2. This is how our final structure for the development team will look:

- Name: TechTeam
Domain: techteam.solutionFactory.com
EnableNodeOUs: true
Template:
 Count: 2
Users:
 Count: 2

Note: The Count under Template defines the number of nodes in that organisation and the Count under Users defines the number of non-admin users.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Channel Artifacts

Channel Artifacts are created using a tool called *configtxgen*. This tool is also available along with other binaries during Fabric installation. Channel Artifacts are configuration files used at different phases during network setup.

Genesis:

 this is used to set up the genesis block of the entire blockchain.

Channel:

 configuration file to create the channel on the network.

Anchor:

 Each organization must have at least one anchor peer. Once all the peers start to run, this configuration will be used to define the Anchor peers out of them.

Certificate Authority (CA) And Membership Service Provider (MSP):

Certificate Authority is an entity within the MSP of each organization which is authorised to issue certificates for different entities such as Admin, Client, peer, and Orderer in the organization. The entities of the organization need to show these certificates to carry out any operation(such as querying the ledger or invoking the transaction) on the network.

For example, if the client application needs to invoke any transaction then it must have the certificate issued by the CA of that organization. An important thing to note is that CA is a part of the MSP which is responsible for issuing certificates to different entities of the organization.

Membership Service Provider is a pluggable service running within each organization that is primarily responsible to validate the certificates of the entities trying to carry out any operation on the network. Imagine an entity is trying to read the ledger of a particular peer belonging to a particular organization. Now, it is the responsibility of the MSP of that organization to check if that entity has the reading permission or not.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

MSP Roles / Principals:

There are four roles allowed by the Fabric network and these were already split when the certificates were created by the cryptogen tool.

- Client: An application that will be accessing or transacting on the network
- Peer: one of the nodes or computing resource that will be contributed by the organization on the network
- Admin: Entity that will be allowed to make changes to the configuration or to add more channel, peers to the network
- Orderer: peer who's job is to act as the ordering service

All of the above roles are called as Members roles. Rules can be defined specific to any of the available roles. Member role is used to provide a generalized access to any role.

configtx.yaml:

This is the file which contains the configurations required to create the channel artifacts. These configurations are required for three important tasks:

create the genesis block: allows the orderer to start the ordering process.

define channel: to enable other peers join this channel.

define anchor: define anchor peers for cross-communication between peers.

The *configtx.yaml* file defines the various capabilities that will be allowed for the organization, the channel, the orderer and the applications. This will also link the MSP's to each organization in the network. This also defines the rules or policies that will be used to govern the organizations.

Organizations:

This defines the MSP and the related policies for each organization. This is not a definition of the organization itself, instead it is only the policies which the organization will be governed by. This needs to be defined for the orderer besides the three organizations (IIT, upGrad, MHRD) in this network.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Configurations used for organizations:

Name: name of the MSP

ID: unique ID of the MSP's

Directory: path to the configuration for the MSP's. These already have been created by the cryptogen tool

Policies:

defines readers, writers and writers of the organization. These policies are written in the form of a conditional statement, whether all or some of the entities have to provide signatures for this policy to be true.

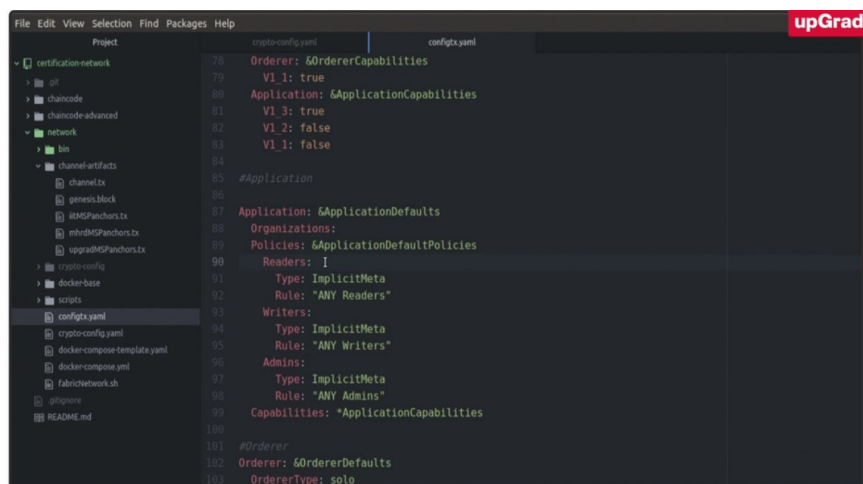
Anchor Peers: for the organizations other than the orderer, Anchor peer's are also defined. These parameters will have to be exactly the same ones used in the docker containers used to start the network.

Configurations used for capabilities:

This allows to differentiate between the levels of binaries are allowed to be used in different parts of the network. In some cases, there may be some recent versions of Fabric binaries which may have some features that are incompatible with prior versions. To avoid this scenario, a particular version of Fabric binary is defined here.

Configurations or policies for the applications:

Here, the implicit way of defining policies is used where as long as the organization identifies a particular entity as a reader, writer or admin, the application will accept that as it is.



```
78 Orderer: &OrdererCapabilities
79   V1_1: true
80 Application: &ApplicationCapabilities
81   V1_3: true
82   V1_2: false
83   V1_1: false
84
85 #Application
86
87 Application: &ApplicationDefaults
88 Organizations:
89   Policies: &ApplicationDefaultPolicies
90   Readers: I
91     Type: ImplicitMeta
92     Rule: "ANY Readers"
93   Writers:
94     Type: ImplicitMeta
95     Rule: "ANY Writers"
96   Admins:
97     Type: ImplicitMeta
98     Rule: "ANY Admins"
99 Capabilities: "ApplicationCapabilities"
100
101 #Orderer
102 Orderer: &OrdererDefaults
103 OrdererType: solo
```


HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Configurations for the orderer:

This config file is used to create the channel artifact which gives the genesis block to be used by the ordering service to start the blockchain. These parameters define the generation of the genesis block and the following generation of blocks.

Type: it can be Solo / Khafka / Raft

Address: this specifies the address of the orderer. In our case, there is only one
Time, Size, Capabilities and Policies

Block validation: This defines which entities are allowed to sign a transaction

Configurations for the channel:

Here also, the implicit way of defining the policies is used.

Profiles:

These profiles will be used when running the channel artifact creation command. Two profiles used are as below. The names are arbitrary, however, the same have to be used when running the CLI commands:

OrdererGenesis: to create the genesis block

CertificationChannel: configurations to create the channel on the network

The various parameters defined before which were placeholders, will be imported into these profiles.

Consortiums:

These are a group of organizations to come together to create a channel. Before organizations can become a part of a channel, they first must be defined as a consortium. In this network, we name it as CertificationConsortium where all the three organizations IIT, upGrad and MHRD are defined as a part of it.

CertificationChannel:

This is the profile for creating the channel. This defines the consortium, the capabilities of the applications and also the organizations which are allowed as clients to interact with this channel.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Steps to create the channel artifacts:

Please use the separate file given on the portal which gives all the step-by-step commands to create the channel artifacts.

```
upgrad@upgrad-VirtualBox: ~/workspace/certification-network/network
File Edit View Search Terminal Help
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ configtxgen -profile CertificationChannel -outputAnchorPeersU
pdate ./channel-artifacts/lttMSPanchors.tx -channelID certificationnetwork -asOrg lttMSP
2019-09-16 22:18:04.340 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2019-09-16 22:18:04.365 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/upgrad/workspace/
certification-network/network/configtx.yaml
2019-09-16 22:18:04.413 IST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 orderer type: solo
2019-09-16 22:18:04.413 IST [common.tools.configtxgen.localconfig] LoadTopLevel -> INFO 004 Loaded configuration: /home/upgrad/wo
rkspace/certification-network/network/configtx.yaml
2019-09-16 22:18:04.413 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 005 Generating anchor peer update
2019-09-16 22:18:04.414 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 006 Writing anchor peer update
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ configtxgen -profile CertificationChannel -outputAnchorPeersU
pdate ./channel-artifacts/mhrdMSPanchors.tx -channelID certificationnetwork -asOrg mhrdMSP
2019-09-16 22:18:34.385 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2019-09-16 22:18:34.428 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/upgrad/workspace/
certification-network/network/configtx.yaml
2019-09-16 22:18:34.459 IST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 orderer type: solo
2019-09-16 22:18:34.460 IST [common.tools.configtxgen.localconfig] LoadTopLevel -> INFO 004 Loaded configuration: /home/upgrad/wo
rkspace/certification-network/network/configtx.yaml
2019-09-16 22:18:34.460 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 005 Generating anchor peer update
2019-09-16 22:18:34.460 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 006 Writing anchor peer update
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ configtxgen -profile CertificationChannel -outputAnchorPeersU
pdate ./channel-artifacts/upgradMSPanchors.tx -channelID certificationnetwork -asOrg upgradMSP
2019-09-16 22:18:51.189 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2019-09-16 22:18:51.134 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/upgrad/workspace/
certification-network/network/configtx.yaml
2019-09-16 22:18:51.173 IST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 orderer type: solo
2019-09-16 22:18:51.173 IST [common.tools.configtxgen.localconfig] LoadTopLevel -> INFO 004 Loaded configuration: /home/upgrad/wo
rkspace/certification-network/network/configtx.yaml
2019-09-16 22:18:51.174 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 005 Generating anchor peer update
2019-09-16 22:18:51.174 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 006 Writing anchor peer update
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$
```

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Docker and Fabric Network Setup

This session covers:

Docker containers and how to start them on the local computer.

Steps to start the network:

- Creating the channel
- Making all the peers to join the channel
- Updating all anchor peers for each organization

Docker will be used to set up the entire Certification network which consists of a total of 11 nodes which are 6 peers (which belong to 3 organizations), 3 CA's, Orderer and the CLI terminal. Docker allows each of these independent nodes to run on a single machine. One Docker container will be used for each of these nodes. These containers will work together to form the Fabric network. *Docker Compose* is the tool that is used to create all the containers. This tool allows the nodes to be created as services. The Docker Compose YAML configuration file will list down the various parameters for these services.

- An image which is provided by Fabric will be used for each of the services.

The environment variables in these images will be configured to suit this network.

- Volume mapping will enable the network data to be available for these services.

- Run commands to start the specific nodes.

docker-compose.yaml:

This is a YAML configuration file which will be used to define all the parameters related to each of the services as a part of the network. Independent docker containers will need to be started for each of the nodes. Docker provides CLI tool commands which will be used to start or stop all of these services or nodes in a single command on the CLI terminal.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

version:

This is the version of the YAML configuration file. This just an arbitrary number to track the version.

volumes:

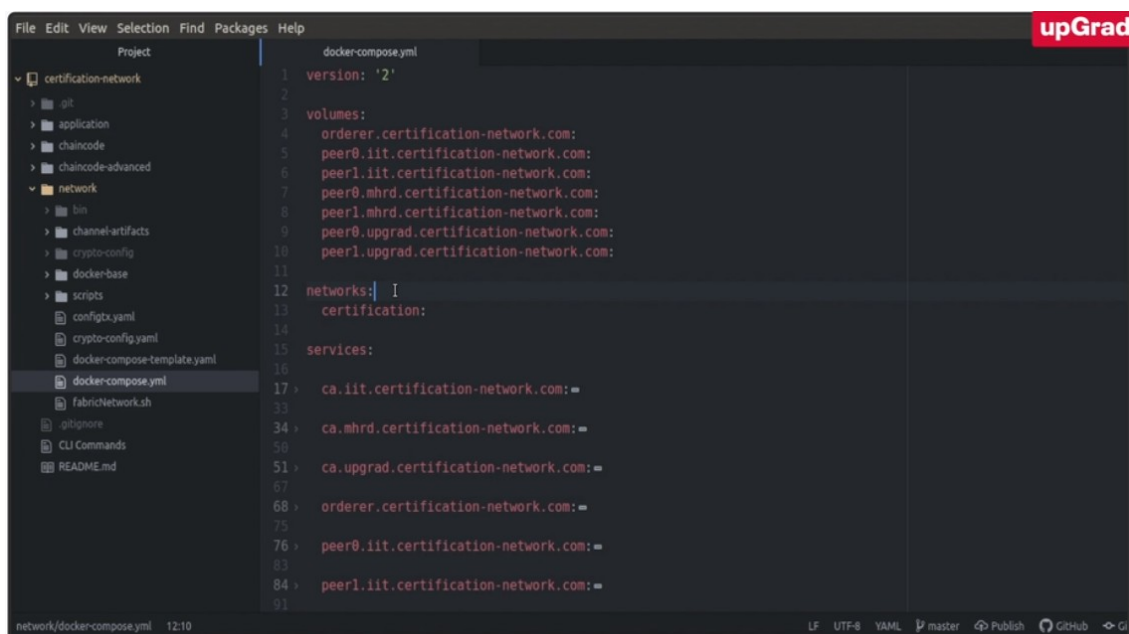
These are the disk spaces which will be connected to the services. For each of the volumes defined here, there will be an independent disk mapping which will be attached to the services. Here, only the names are listed. The other parameters will be defined along with the parameters of the individual nodes.

network:

this is the name of the internal network. All of the nodes will connect to and use this network to communicate with each other.

services:

These are the different Docker containers that will be started to run this network. For this network, we have individual services for 3 CA's, the orderer, the 6 different peers (2 peers for each organization), the CLI and the chaincode. The CLI is an external node to the network and helps to setup and manage the network. The CLI can connect to any one of the nodes and can run specific commands on them. The chaincode container will help make the chaincode development easier and quicker.



```
File Edit View Selection Find Packages Help
Project
certification-network
  .git
  application
  chaincode
  chaincode-advanced
  network
    bin
    channel-artifacts
    crypto-config
    docker-base
    scripts
    configtx.yaml
    crypto-config.yaml
    docker-compose-template.yaml
    docker-compose.yml
    fabricNetwork.sh
  .gitignore
  CLI Commands
  README.md
network/docker-compose.yml
12:10
LF UTF-8 YAML master Publish GitHub G

docker-compose.yml
1 version: '2'
2
3 volumes:
4   orderer.certification-network.com:
5   peer0.iit.certification-network.com:
6   peer1.iit.certification-network.com:
7   peer0.mhrd.certification-network.com:
8   peer1.mhrd.certification-network.com:
9   peer0.upgrad.certification-network.com:
10  peer1.upgrad.certification-network.com:
11
12 networks:
13   certification:
14
15 services:
16
17   ca.iit.certification-network.com:
18
19   ca.mhrd.certification-network.com:
20
21   ca.upgrad.certification-network.com:
22
23   orderer.certification-network.com:
24
25   peer0.iit.certification-network.com:
26
27   peer1.iit.certification-network.com:
28
```

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

There will be different containers performing similar tasks on the network. For example, there are 3 CA's and 6 peers but belong to different organizations. Some of the codes and parameters for these services will be common. To avoid the repetition of this common code, inheritance will be used to define them separately and import them to other parent configuration files.

Following are the parameters for the CA (Certificate Authority):

- image:

The container services will be based on the image that will be pulled from the Docker cloud. These are the base operating systems with base parameters by default. The base images for various roles in the network are already available in Fabric through the Docker hub. This command will download the specific image version which can later be customized separately. This configuration is done through *environment variables*.

- environment:

these are the environment variables which need to be overridden. Here the home path, server name and the TLS variables are set. We will keep the TLS disabled for this network. TLS needs separate certificates and keys to function.

- ports:

this is the mapping between the port number on the local computer and the port number of the Docker container.

- command:

this will define the commands that will be run after the service starts. This will run the binaries which are already downloaded as the image for this container.

- volumes:

this is the definition to map the paths of the disk spaces on the local computer to file paths inside the container. This allows to access the code files from the container.

- container_name:

this is the name of the container to identify the container.

- networks:

this is the network that the container will connect to.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Following are the parameters for the ordering service and the peers:

- extends:

Here, we will be importing the definitions from a parent file which is inside the *dockerbase* folder. The services defined in the base compose file will be imported.

- container_name:

This is the name of this container.

- networks:

This is the name of the network that this container will connect to.

Following are the parameters for the CLI container:

- container_name:

Some arbitrary name of the container

- image:

This is the image that will be used for the CLI container. This image will be downloaded from the Docker hub.

- tty, stdin_open:

These will be set to *true* to enable the CLI open a shell and be able to run commands.

- environment:

These are the configuration parameters that need to be overridden. Here, the `CORE_PEER_ADDRESS`, `CORE_PEER_LOCALMSPID` and other file paths for the certificates are set.

- working dir:

this will set the current working directory for this container.

- command:

this will be the command that will be run as soon as the container starts

- volumes:

these are the volume mappings between the local computer and the file paths inside the container. Here we will need to map the file parts where the chaincodes and channel artifacts are stored.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

- `depends_on`:

this parameter makes sure that this particular container will only start after the defined containers here are started.

- `networks`:

this is the network that this container will connect to.

Following are the parameters for the chaincode container:

- `container_name`:

custom name for the container

- `image`:

this is the image to be used for this container

- `tty`:

enables container to open a shell to run commands on it

- `environment`:

parameters to be customized

- `working_dir`:

this sets the working directory

- `command`:

this sets the command which need to be run at the start

- `volumes`:

these are the volumes of the paths that need to be mapped

- `networks`:

this is the network that this container will connect to

docker-compose-base.yaml and docker-compose-peer:

This file will have the base services for the orderer and each of the peers. The `docker-compose.yaml` file inherits properties from the `docker-compose-base.yaml` file, which, in turn, inherits some properties from the `docker-compose-peer.yaml` file. This splitting of configuration files is not mandatory but this is only used to be efficient with code re-usability.

HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

This file also defines the same set of parameters such as container_name, image, environment variables, working_dir, command, volumes and port. The environment variables will be unique to each peer or the specific service.

To aid the quick set up of the network through these configuration files, the examples provided in the Fabric Samples can be used. Changes can be made to these parameters to suit the network being newly developed. This way, we don't need to start from scratch. You can use the online Hyperledger Fabric documentation to look at the default configuration of the environment variables in the Docker images.

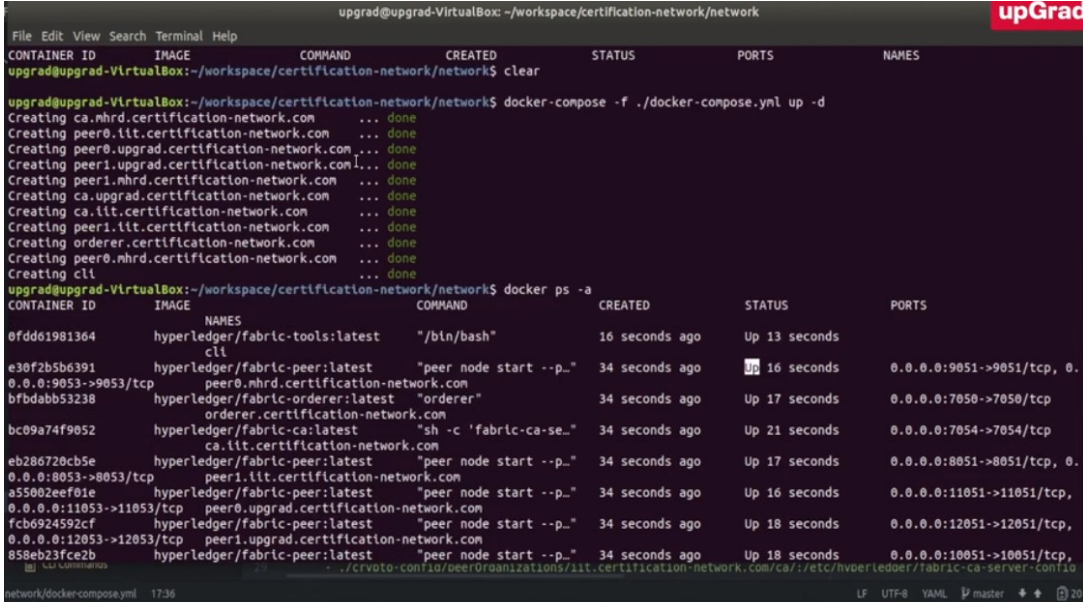
Starting the Docker Containers:

docker-compose is a command tool provided by Docker to start or stop the services defined in the configuration file which will be an input to this command. Commands to use are below.

After running this command, various containers will start getting created and a done statement will appear in front of the containers that are successfully created. Once the entire docker network is up and running you need to make sure that all the status of all the containers is UP.

```
# docker ps -a
```

```
# docker-compose f ./dockercompose.yml up -d
```



```
upgrad@upgrad-VirtualBox: ~/workspace/certification-network/network
File Edit View Search Terminal Help
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ clear

upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ docker-compose -f ./docker-compose.yml up -d
Creating ca.mhrc.certification-network.com ... done
Creating peer0.lit.certification-network.com ... done
Creating peer0.upgrad.certification-network.com ... done
Creating peer1.upgrad.certification-network.com ... done
Creating peer1.mhrc.certification-network.com ... done
Creating ca.upgrad.certification-network.com ... done
Creating ca.lit.certification-network.com ... done
Creating peer1.lit.certification-network.com ... done
Creating orderer.certification-network.com ... done
Creating peer0.mhrc.certification-network.com ... done
Creating cli ... done
upgrad@upgrad-VirtualBox:~/workspace/certification-network/network$ docker ps -a
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
0fdd61981364    hyperledger/fabric-tools:latest    "/bin/bash"    16 seconds ago    Up 13 seconds
e30f2b5b6391    hyperledger/fabric-peer:latest    "peer node start --p..."    34 seconds ago    Up 16 seconds    0.0.0.0:9051->9051/tcp, 0.0.0.0:9053->9053/tcp
bfbdbb53238    hyperledger/fabric-orderer:latest    "orderer"    34 seconds ago    Up 17 seconds    0.0.0.0:7050->7050/tcp
bc09a74f9052    hyperledger/fabric-ca:latest    "sh -c 'fabric-ca-se..."    34 seconds ago    Up 21 seconds    0.0.0.0:7054->7054/tcp
eb28672bcb5e    hyperledger/fabric-peer:latest    "peer node start --p..."    34 seconds ago    Up 17 seconds    0.0.0.0:8051->8051/tcp, 0.0.0.0:8053->8053/tcp
a55002eef01e    hyperledger/fabric-peer:latest    "peer node start --p..."    34 seconds ago    Up 16 seconds    0.0.0.0:11051->11051/tcp, 0.0.0.0:11053->11053/tcp
fcb6924592cf    hyperledger/fabric-peer:latest    "peer node start --p..."    34 seconds ago    Up 18 seconds    0.0.0.0:12051->12051/tcp, 0.0.0.0:12053->12053/tcp
858eb23fce2b    hyperledger/fabric-peer:latest    "peer node start --p..."    34 seconds ago    Up 18 seconds    0.0.0.0:10051->10051/tcp, 0.0.0.0:10053->10053/tcp
network/docker-compose.yml    1736    LF UTF-8 YAML master 20
```


HYPERLEDGER FABRIC & COMPOSER

MODULE 2

FABRIC NETWORK SETUP

Starting the Network:

After having SSHed into the CLI container using the *docker exec* command, run the various commands needed to:

- Create the channel,
- Make the peers join the channel, and
- Update the anchor peers for the organizations

Please use the documentation on the portal which lists down the step-by-step process as above.

To check logs for a particular peer you can use the command with the following syntax.

```
# docker logs -f <address of peer>
```

example:

```
# docker logs f peer0.iit.certificationnetwork.com
```

Summary:

- | | |
|-----------------------|-----------------------|
| 1. Pre Setup | |
| Crypto materials | - cryptogen |
| Channel artifacts | - configtxgen |
| 2. Docker compose | - docker-compose up |
| 3. SSH into CLI | - docker exec |
| 4. Create channel | - peer channel create |
| 5. Join peers | - peer channel join |
| 6. Anchor peer update | - peer channel update |

Commands for steps 1, 2 and 3 need to be run on the local computer while the next three commands 4, 5 and 6 needs to be run on the CLI container.