HYPERLEDGER FABRIC POC:

* Implementation of fabric network
* Building sample applications on top of this network

Contents

[Introduction 3](#_Toc1942280)

[Conceptual understanding of Hyperledger Fabric and its elements 4](#_Toc1942281)

[Introduction of Hyperledger Fabric specific binaries 6](#_Toc1942282)

[BYFN 🡪 Build your first network 6](#_Toc1942283)

[Insights of Hyperledger Fabric Network Infrastructure 14](#_Toc1942284)

[Insights of Chaincode Lifecycle 15](#_Toc1942285)

[Insights of Fabric CLI 16](#_Toc1942286)

[Next steps 17](#_Toc1942287)

# Introduction

This document is intended for providing:

1. An overall understanding of Hyperledger Fabric framework
2. Fabric Network Implementation POC
3. Building sample applications

For simplicity, this has been further divided into following sub-topics:

* Conceptual understanding of Hyperledger Fabric and its elements. For detailed understanding of Fabric workflow, you can visit

<https://medium.com/@u.avinash070990/hyperledger-fabrics-transaction-endorsement-workflow-e10302b3174>

* Introduction to Hyperledger Fabric specific binaries and insights of setting up Fabric network infrastructure.
* Build Your First Network (byfn) overview and detailed explanation.
* Insights of chaincode lifecycle and steps to install new chaincode.
* Understanding of Fabric CLI

So, let’s get started…!!! ☺

# Conceptual understanding of Hyperledger Fabric and its elements

Hyperledger Fabric is one of the five Hyperledger projects hosted by the Linux Foundation. Hyperledger Fabric is a permissioned blockchain platform aimed at business use. It is contributed by IBM and Digital Asset Holdings and provide plug-n-play modular blockchain consensus.

Hyperledger Fabric is a rapidly-evolving system. Hyperledger Fabric runs Chaincode which is equivalent for Ethereum’s smart contract. The consensus protocol is pluggable.

Hyperledger Fabric Elements –

* Client – Clients are the end-user facing nodes. Hyperledger Fabric architecture provides multiple interfaces to the blockchain. These are presented by the client nodes (i.e. SDK’s and CLI). Users (which may be the application software) send transaction requests to the Hyperledger Fabric network using these interfaces via a client nodes.
* Peer – Peers maintain the state of the ledger. Peers execute the chaincode and participate in consensus formation. Chaincode is installed on peers. They can have several special roles in the network. A peer can be a validating peer, if chaincode is installed on it, or a committing peer. In addition a peer can be an anchor peer, a stable entry point.
* Orderer – Orderer provides Ordering Service (OS). The Ordering Service creates new blocks by ordering the transactions. It provides a shared communication channel to Client and Peers. It has two services i.e. Broadcast (used to inject messages to the system) and Deliver (used for receiving orderer batches). Orderers are responsible for making sure that all the peers in the network have committed a transaction. When a transaction is proposed and committed by a peer, the orderer is informed about the new transaction and it forwards and commits this block to all the adjacent peers.
* Ledger – Ledger includes the history of all transactions. This is used to construct the world state (also referred as state database).
* World State – World State reflects the current data about all the assets in the network. This data is stored in database for efficient access. Current supported databases are Level DB and Couch DB.
* Channels – Channels are data partitioning mechanisms that allow transaction visibility for stakeholders only. Each channel is an independent chain of transaction blocks containing only transactions for that particular channel.
* Chaincode – Chaincode encapsulates both asset definition and business logic (or transactions) for modifying those assets.
* Network – Network is a collection of data processing peers that form a blockchain network. Network is responsible for maintaining a consistently replicated ledger.
* Membership Service Provider (MSP) – MSP manages identity and permissioned access for client and peers. MSP turns verifiable identities into the members of a blockchain network. There are two types of MSP’s. Local MSP and Channel MSP, each peer has its own Local MSP that manages their own users while Channel MSP operates at a higher level involving network administration such as channel configuration and defines participatory rights a channel level.
* Certificate Authority (CA) – Certificate Authority issues certificates to different actors or participants in the network so that we can know who is who in the blockchain network and to ensure security. Fabric comes with cryptographic tools to generate CA for the peers (nodes in the network). One can use different tools like openssl, cryptogen or fabric ca to generate needed certificates and keys. Note that, cryptogen is tool for non-production environment.
* Consensus – Consensus protocols are one of the most important and revolutionary aspect of the blockchain technology. It creates an irrefutable system of agreement between various devices across distributed network. Fabric uses “Kafka” for consensus determining what data can be appended to the ledger.
* Assets – It is the tokenization of physical and other type of assets such as goods, service, currency etc…

# Introduction of Hyperledger Fabric specific binaries

Assuming that all the prerequisites are downloaded and Hyperledger Fabric setup is already done. If not then, you can refer Hyperledger Fabric official documentation.

Upon installation, it will include the following Fabric specific executables:

* Configtxgen
* Configtxlator
* Cryptogen
* Fabric-ca-client
* Orderer
* Peer

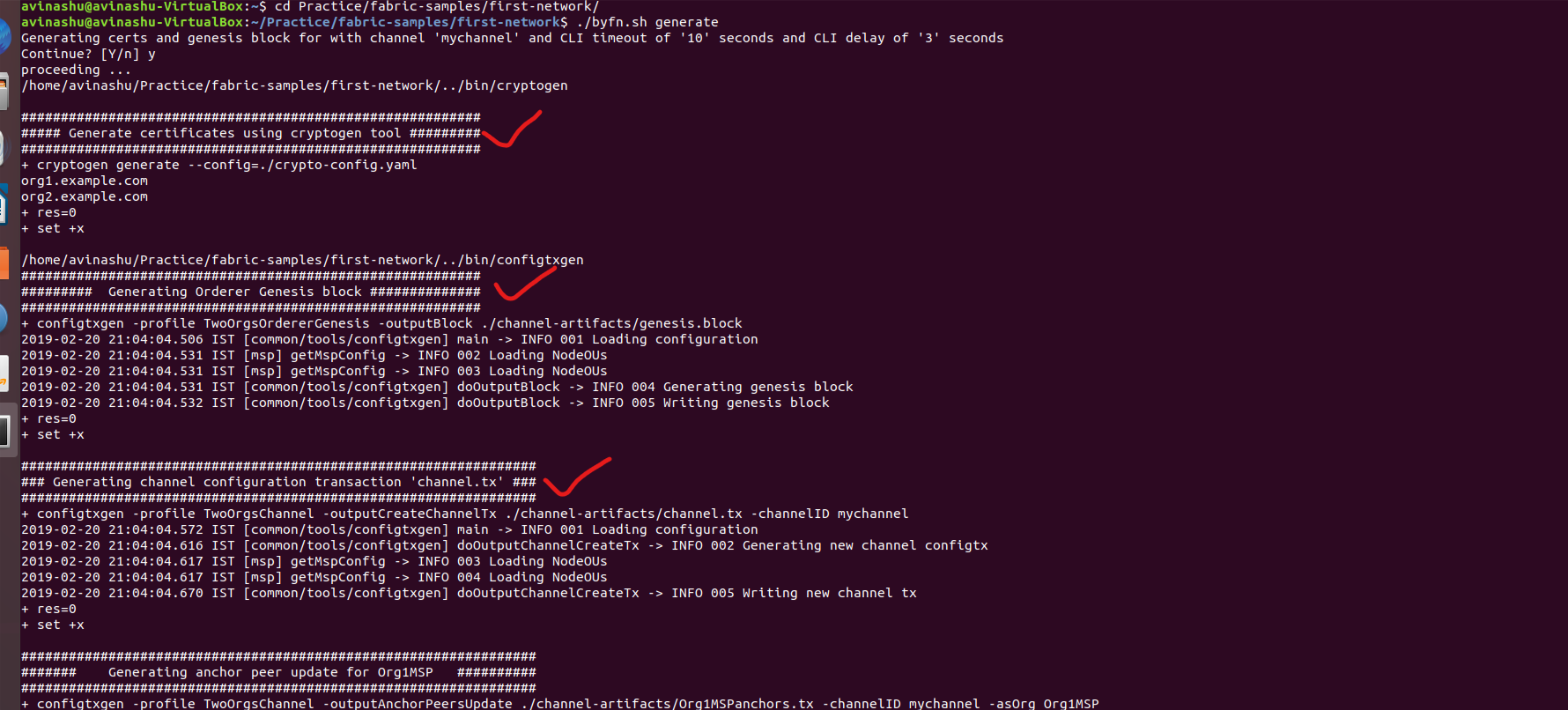
# BYFN 🡪 Build your first network

Now, the first task we will do is to bring up a basic blockchain network with scripts provided by “first-network” under “fabric-samples” repository and it is called as “byfn” (build your first network).

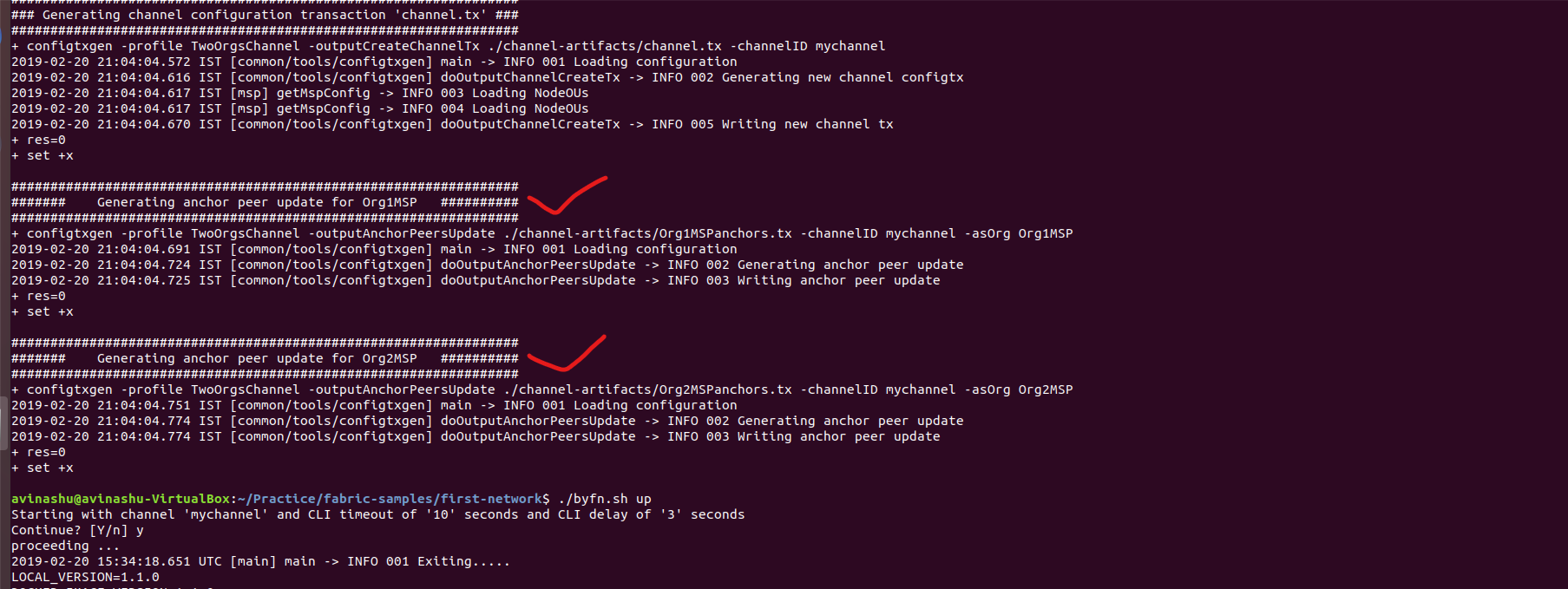
But, before that let me give you an overview of what byfn (build your first network) is all about and its functionality. So, it’s a very simple network which has two organizations i.e. org1 and org2. Each organization has two peers and one of those peers in each organization is called anchor peer. So, just to get started anchor peer is a special kind of peer that allows communication with other organizations, then there is ordering service which is responsible to create consensus in the blockchain and there is a special kind of node, the cli node which allows the script to send commands directly to the network.

Step by step Explanation and Execution of byfn.sh:

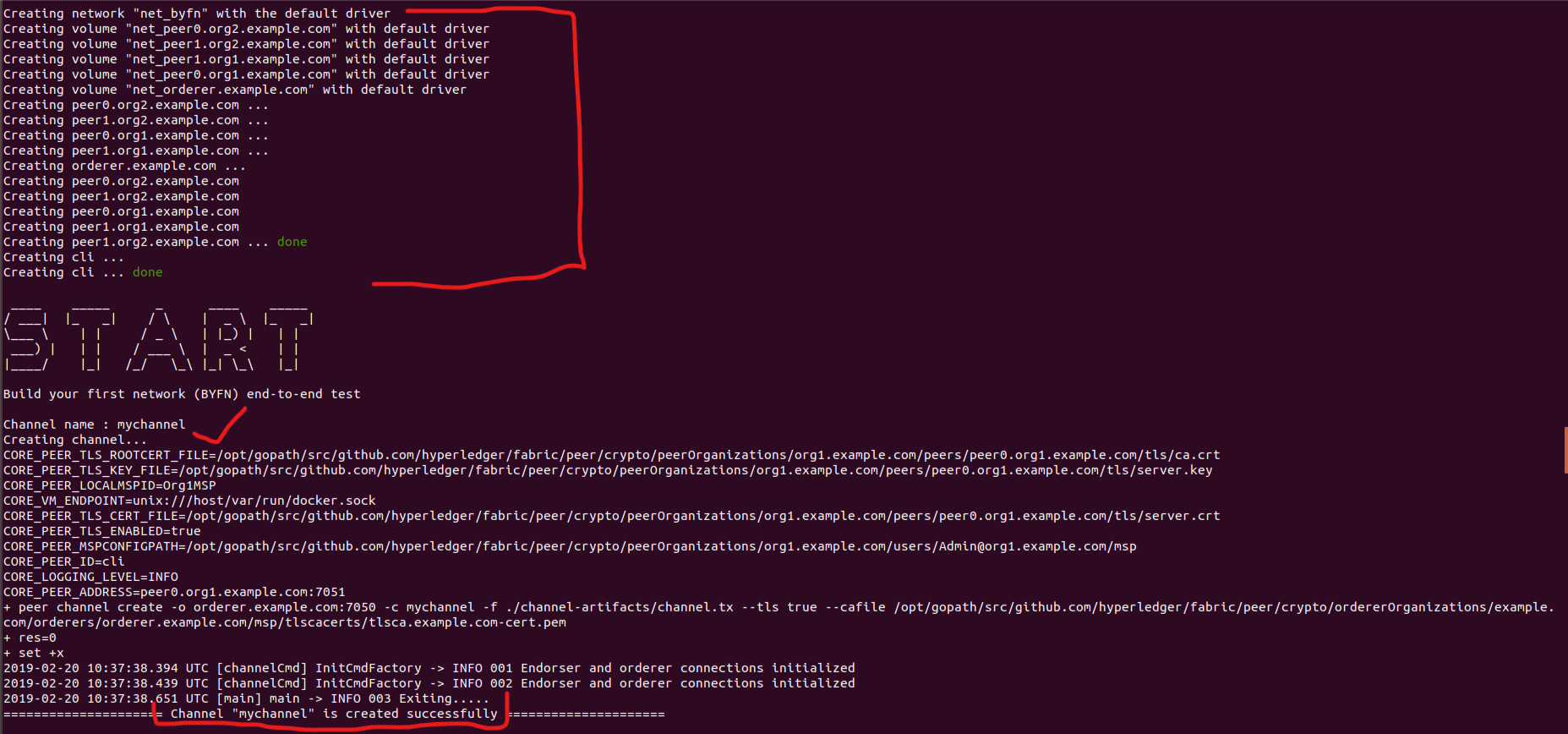
* First bring up the virtual machine and go to the terminal (ctrl + alt + t)
* Now go to the folder containing byfn.sh script (cd fabric-samples/first-network)
* Then perform ./byfn.sh generate. This will generate all the certificates and cryptomaterials.
* So, when we look at the output from the below script we can see that cryptogen generate –config=./crypto-config.yaml command to use. So, we have cryptogen tool which generates the cryptographic artifacts.
* Then in the following steps, we use the configtxgen i.e. configtxgen –profile TwoOrgsOrdererGenesis –outputBlock ./channel-artifacts/genesis.block command to create the so called genesis block which is the first block in the blockchain.
* Then we define a channel, so that, the peers can communicate with each other i.e. configtxgen –profile TwoOrgsChannel –outputCreateChannelTX ./channel-artifacts/channel.tx –channelID mychannel command.



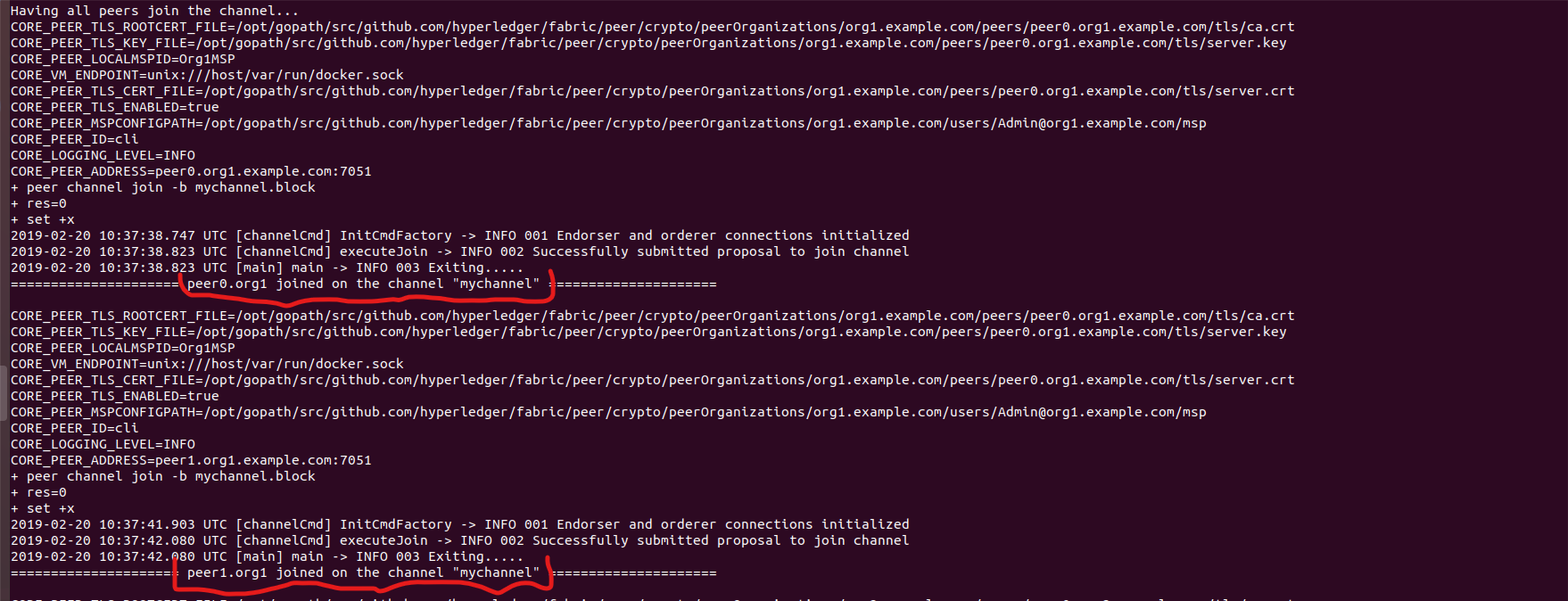
* After that we define or specify two anchor peers, one for each organization i.e. configtxgen –profile TwoOrgsChannel –outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx –channelID mychannel –asOrg Org1MSP command for Organization1 and configtxgen –profile TwoOrgsChannel –outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx –channelID mychannel –asOrg Org2MSP command for Organization2.

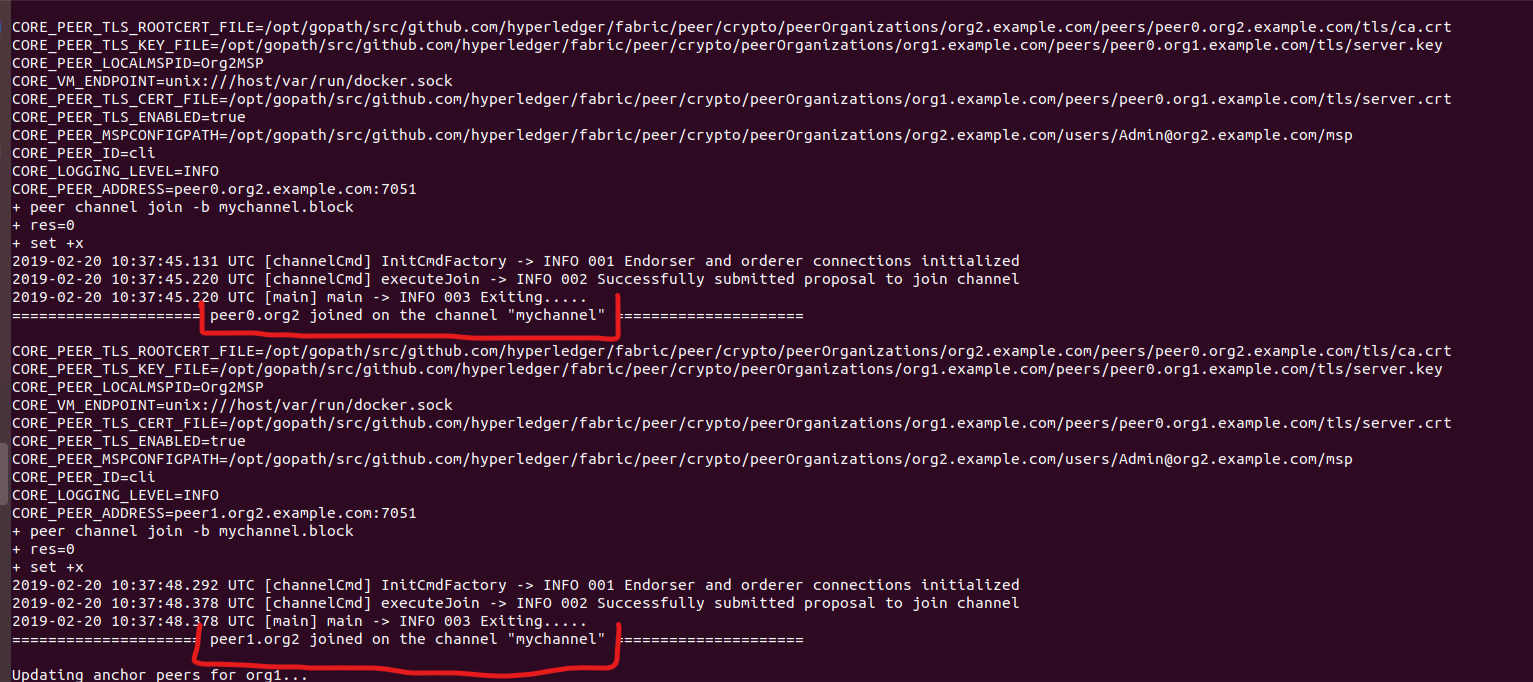


* So, to summarize cryptogen creates a crypto-config folder inside first-network folder and the configtxgen creates channel-artifacts folder and here in this channel-artifacts folder you can see some of the binary files for the genesis block, for the channel, and for the anchors. Entering the crypto-config folder, we have the certificates and private key etc… etc… all the materials we need to communicate in the hyperledger fabric blockchain.
* Now, we can run the build your first network by ./byfn.sh up
* Here the network is already setup with one channel named mychannel with parameters including keys, MSP, peer address etc… etc… and with that a lot of things will start to happen.
* The below screenshot shows that a blockchain network named “net\_byfn” has been created and it is launching the following nodes/peers and services:
  + Organization1 (Org1) has two peers (nodes)
    - peer0.org1.example.com
    - peer1.org1.example.com
  + Organization2 (Org2) also has two peers (nodes)
    - peer0.org2.example.com
    - peer1.org2.example.com
  + Then we have Orderer named orderer.example.com
  + And a CLI (Command Line Interface for Fabric)
* The first step is to start docker with docker services, the docker-compose command creates a lot of things i.e. creating peer0.org1.example.com, creating peer1.org1.example.com, creating peer0.org2.example.com, peer1.org2.example.com, creating cli and orderer.example.com.
* So, all togeather we have six of them and you can do a dockerps to verify that they are up and running.
* Now we start with the actual scenario. So, the first thing we need to create is the channel and in the output you can see: peer channel create –o orderer.example.com:7050 –c mychannel –f ./channel-artifacts/channel.tx --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
* Here, its peer tool with option channel create and then a bunch of parameters also note that often parameters are passed in using system environment variables. Also you can see here the artifacts we have created (channel.tx), they are passed into the tool to access these parameters. We can see that docker creates volumes.

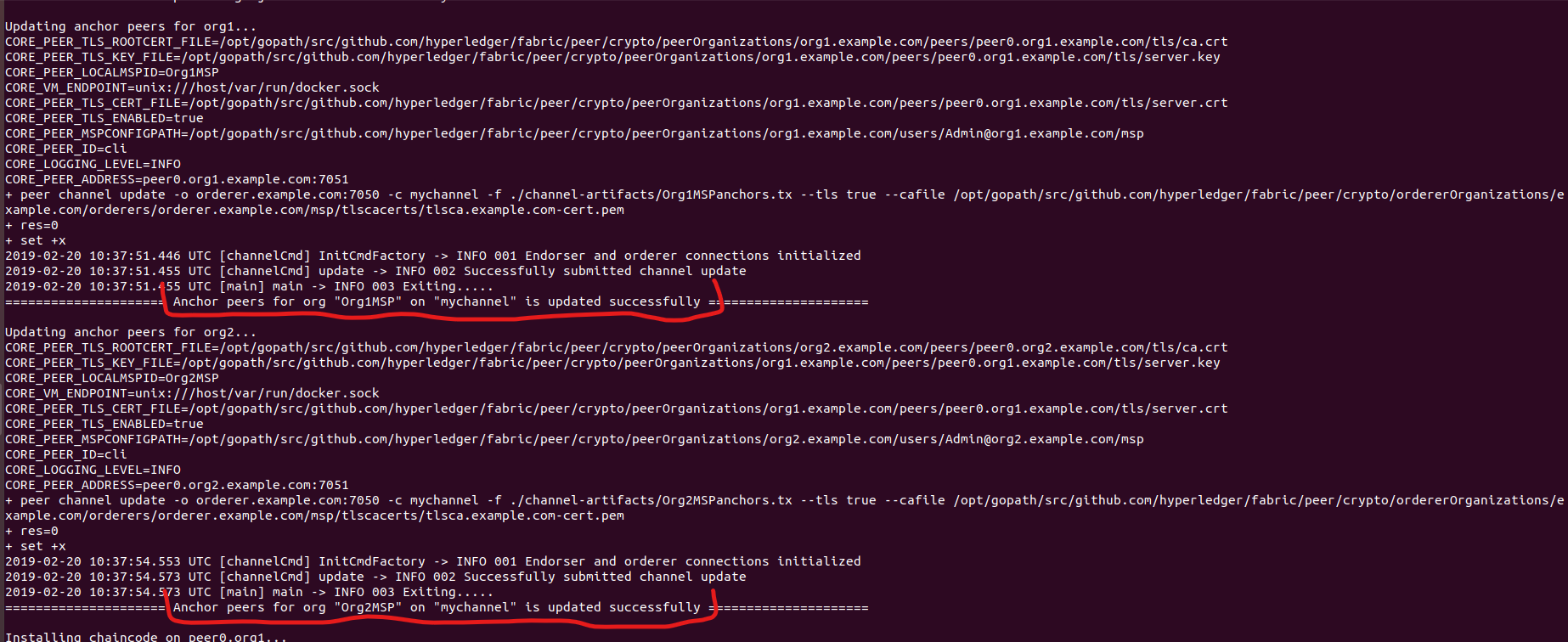


* After we have created the channel, we let all the peers join the channel via peer channel join –b mychannel.block command and the address is passed in this environment variable CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051
* And let’s do this for all the peers i.e. peer0, peer1 for both organization1 and organization2.

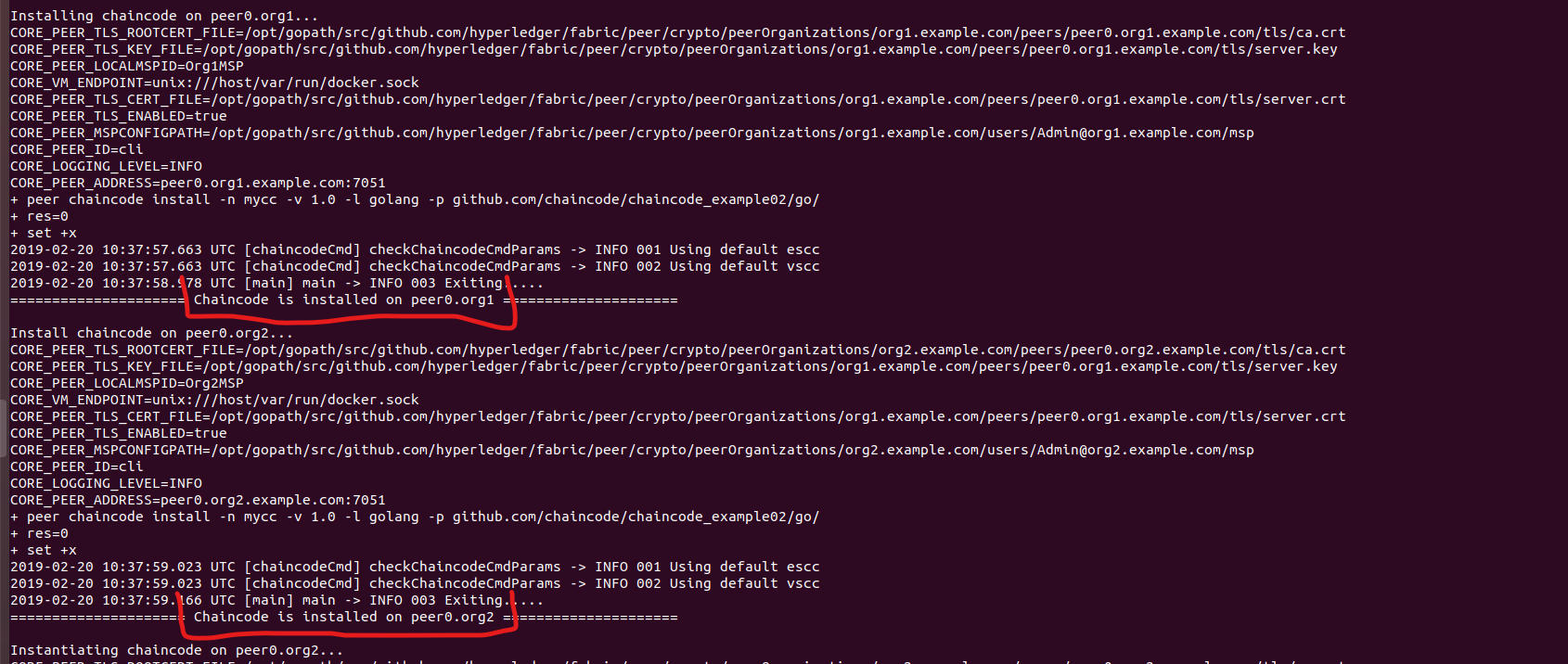




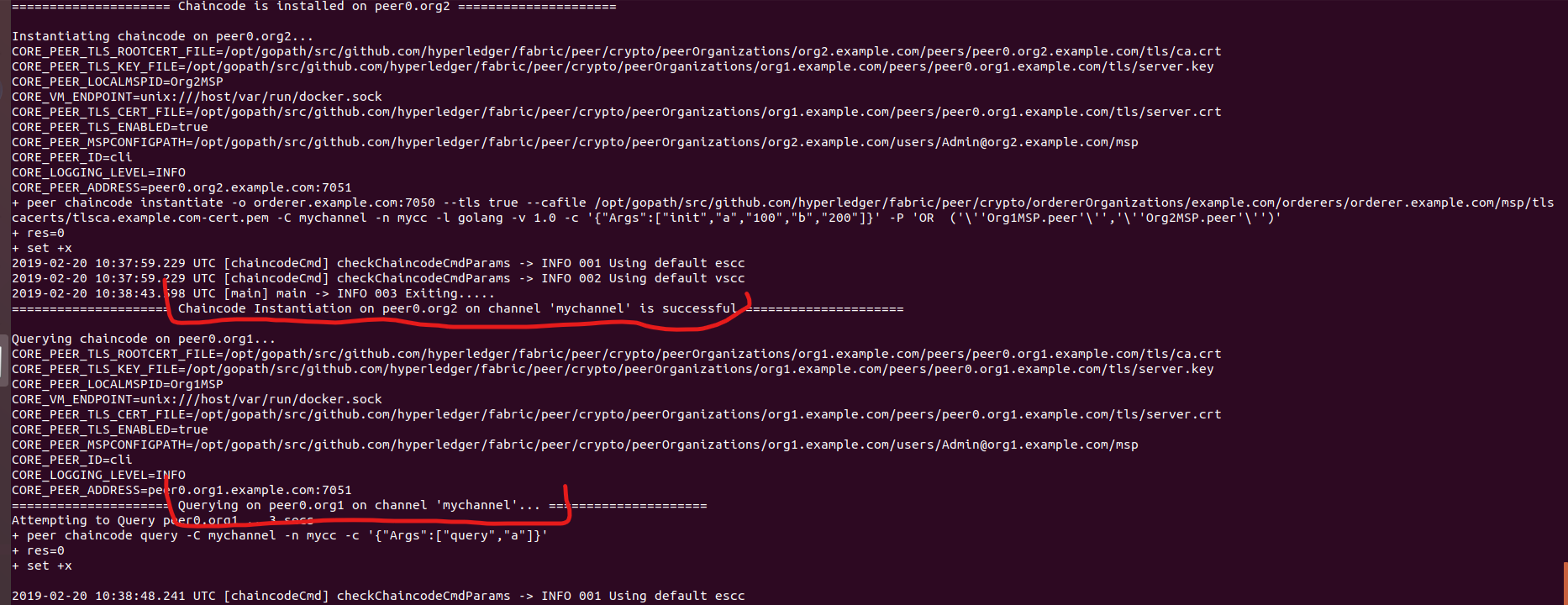
* Now we need to update the anchor peers and we can do this with peer channel update –f ./channel-artifacts/Org1MSPanchors.tx –tls true – cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/msp/tlscacerts/tlsca.example.com-cert.pem
* And we have to do this for both the anchor peers i.e. anchor peer for Org1 and for Org2.



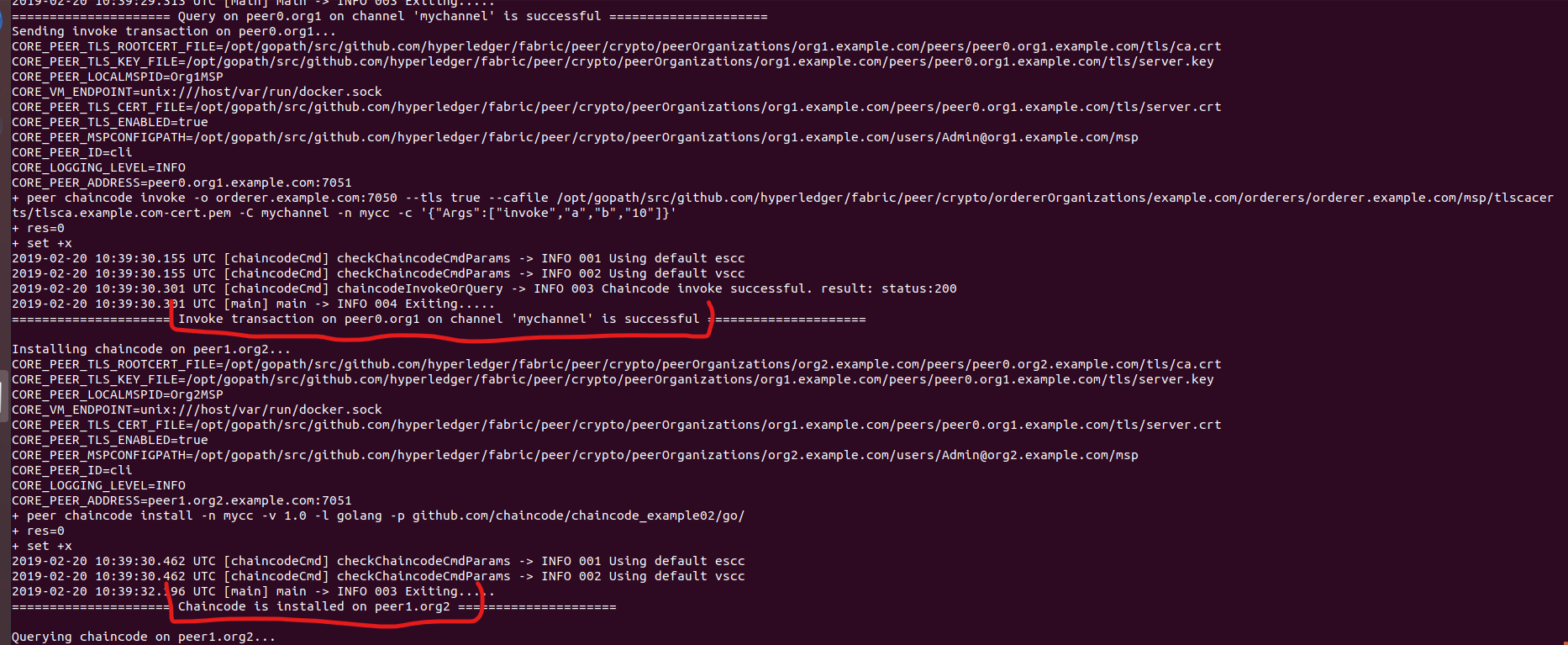
* Now we are actually ready to run our scenario, but before that we need to install chaincode on the peers and we can have a look at the chaincode used in this network, it can be found at fabric-samples folder🡪chaincode folder 🡪chaincode\_example02🡪chaincode\_example02.go (it is written in golang).
* So, here we need to install chaincode on peers and we run the command: peer chaincode install –n mycc –v 1.0 –l golang –p github.com/chaincode/chaincode\_example02/go/ to this address CORE\_PEER\_ADDRESS= peer0.org1.example.com:7051 and provide some other parameters and one of them is the actual chaincode (source code) github.com/chaincode/chaincode\_example02/go/
* We execute the above command on both peers i.e. peer0 of org1 and peer0 of org2 (peer0.org1 and peer0.org2)



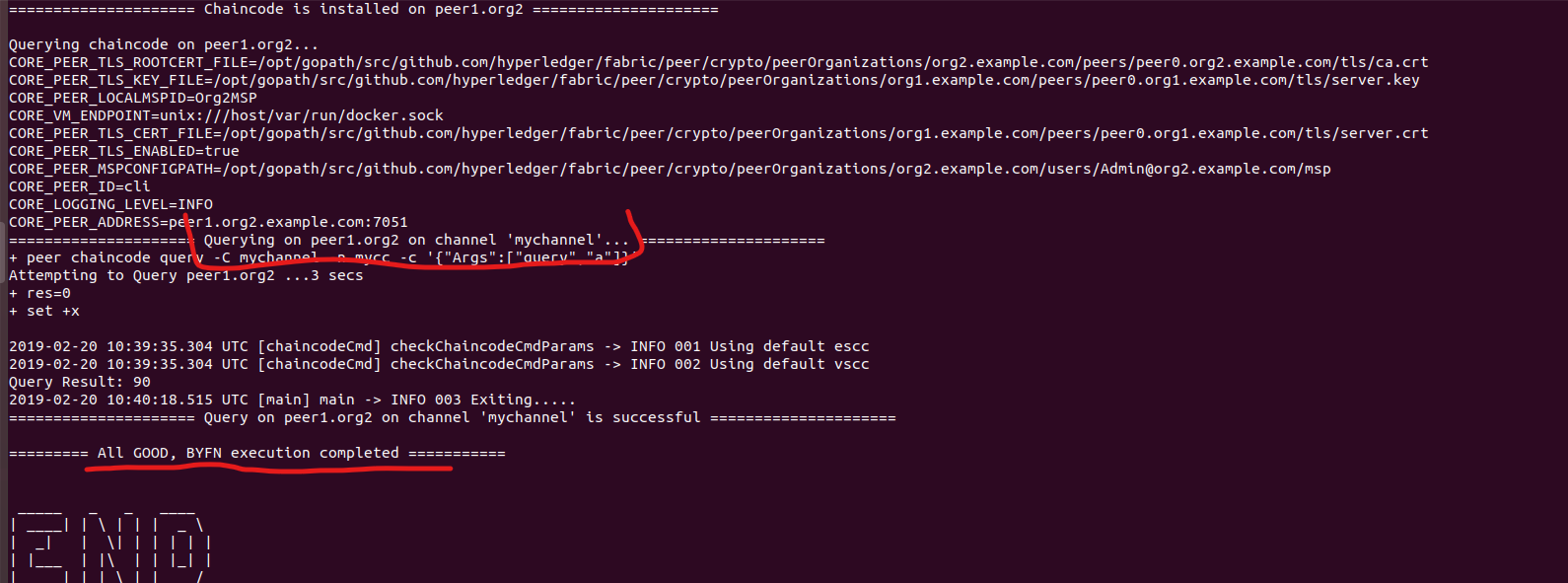
* Now, we have installed the chaincode and now we need to instantiate the chaincode atleast on one peer.
* Instantiating chaincode on peer0.org2 peer chaincode instantiate –o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem –C mychannel –n mycc –l golang –v 1.0 –c ‘{“Args”: [“init”,“a”,“100”,“b”,“200”]}’ –P ‘OR (‘\’ ‘Org1MSP.peer’\‘ ’, ‘\’ ‘Org2MSP.peer’\‘ ’)’
* And the important part is ‘{“Args”: [“init”,“a”,“100”,“b”,“200”]}’ where we send the arguments to this method. So, first is the “init” (the name of the method) and the next is the name of two entities a, b and their values.
* Then we query the chaincode and please note that query is done on peer0.org1 i.e. CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051
* Here we are querying on peer0 from org1 but we in earlier step we have instantiated the chaincode on peer0 from org2 via command peer chaincode query –C mychannel –n mycc –c ‘{“Args”:[“query”, “a”]}’
* Query Result: 100 and this is what query returns.



* Now, we need to do an invoke i.e. peer chaincode invoke –o orderer.example.com:7050 –tls true –cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem –C mychannel –n mycc –c ‘{“Args”: [“invoke”, “a”, “b”, “10”]}’
* Sending invoke transaction on peer0.org1 with following arguments ‘{“Args”: [“invoke”, “a”, “b”, “10”]}’ which means send 10 of currency from a to b.
* Now we install the chaincode on another peer i.e. peer1.org2 via command: peer chaincode install –n mycc –v 1.0 –l golang –p github.com/chaincode/chaincode\_example02/go/ and CORE\_PEER\_ADDRESS=peer1.org2.example.com:7051



* And at last query that chaincode on the above peer i.e. peer1.org2 via command: peer chaincode query –C mychannel –n mycc –c ‘{“Args”: [“query”, “a”]}’ and the query result will be 90.



# Insights of Hyperledger Fabric Network Infrastructure

Inside Hyperledger Fabric Network Infrastructure we will be focusing on

* How to create our own fabric network?
* How we can configure the peers?
* How we can configure the orderer?
* How we can create a channel?
* How peer can join one or multiple channel?
* How to install and instantiate the chaincode?
* How we can execute operations over this chaincode?

Before proceeding, we need to identify who is who in our network and from that configuration we can use a tools provided by hyperledger to our own customed network.

# Insights of Chaincode Lifecycle

In management wise, Chaincode has two main categories. One is called System Chaincode and the other one is called User Chaincode.

System Chaincode are binaries which interacts with a given blockchain. User Chaincode is what we developers/programmers create.

Several System Chaincodes are integrated in Hyperledger Fabric:

* VSCC – Validation System Chaincode implements the transaction validation policy.
* LSCC – Lifecycle System Chaincode for deploying and managing chaincode.
* ESCC – Endorsement System Chaincode signs transaction proposal response. It generates transactions and sends them to the ordering service via an orderer.
* CSCC – Configuration System Chaincode is used to configure the channel.
* QSCC – Query System Chaincode handles queries on ledger.

User Chaincode has four lifecycle of “Install”, “Instantiate”, “Invoke”, and “Query”.

* “Install” essentially maps the chaincodes location/path. Thus, when need arises it can be found and utilized.
* “Instantiate” would create a container image to support all future invocations and query needs of the particular chaincode.
* “Invoke” write data or put data onto the blockchain.
* “Query” get data or retrieve data from the blockchain.

# Insights of Fabric CLI

In continuation to above successful execution of byfn (build your first network), now let’s open another terminal window (ctrl + alt + t) and run the command: docker exec –it cli bash. This command is a way to go to the Fabric CLI prompt. Here in this command exec is for execute, -it option for interactive with current terminal and, cli bash for asking bash shell to help to bring up the cli prompt.

So, now we are in the Fabric CLI prompt and let’s check a few things out, querying and invoking. First let’s find out the current balance for both “a” and “b” with the following commands:

Peer chaincode –query –C mychannel –n mycc –c ‘{“Args”: [“query”, “a”]}’ and

Peer chaincode –query –C mychannel –n mycc –c ‘{“Args”: [“query”, “b”]}’

And thereafter, we get the output as “a” has a balance of 90 and “b” has a balance of 210.

Now let’s suppose, all of a sudden “b” becomes generous and decides to send a balance of 50 to “a”. So for that to be executed the command will be:

peer chaincode invoke –o orderer.example.com:7050 –C mychannel –n mycc –c ‘{“Args”: [“invoke”, “b”, “a”, 50]}’

and post execution of above command you can query the “a” and “b” balances via Peer chaincode –query –C mychannel –n mycc –c ‘{“Args”: [“query”, “a”]}’ and Peer chaincode –query –C mychannel –n mycc –c ‘{“Args”: [“query”, “b”]}’ and the result of the query will be a = 140 and b = 160.

Note: Fabric comes with CLI (Command Line Interface), which is fit for basic development, testing and debugging purpose. It can be very handy if something goes wrong unexpectedly.

peer invoke and query commands only work for peers that have chaincode installed and instantiated. For instance, for byfn network, it would work for peer0.org1, peer0.org2 and peer1.org2 CLI but if you go to peer1.org1 prompt by the command: docker exec –it –e CORE\_PEER\_ADDRESS=peer1.org1.example.com:7051 cli bash and after execution of this command, if you try to use query and invoke commands as

done above they would fail and the reason would be that the chaincode in not installed and instantiated in peer1.org1.

# Next steps

Currently I am working on the setting up POCs for the below topics:

* Building blockchain network with multiple servers.
* Creating web UI interacting with blockchain.
* AWS 🡪 Blockchain as a service (BaaS)

References PDF