**1)Pre-requisites**

1. Fabric-samples
2. Docker

Host1 – AWS – ssh –i ibmblockchain.pem ubuntu@ec2-3-133-85-222.us-east-2.compute.amazonaws.com

Host2 – Azure - ssh [ubuntu@40.114.54.97](mailto:ubuntu@40.114.54.97)

An overlay network is created between the two hosts, through which the communication takes place.

The overlay network is created using docker swarm

The overlay network driver creates a distributed network among multiple Docker daemon hosts. This network sits on top of (overlays) the host-specific networks allows containers connected to it (including swarm service containers) to communicate securely. Docker transparently handles routing of each packet to and from the correct Docker daemon host and the correct destination container.

**2) Steps**

1. Initialize docker swarm mode on HOST1
2. Make all other host network join that swarm as “manager”
3. Create an overlay network so that it could be shared across all the hosts.

**3)Network Topology**

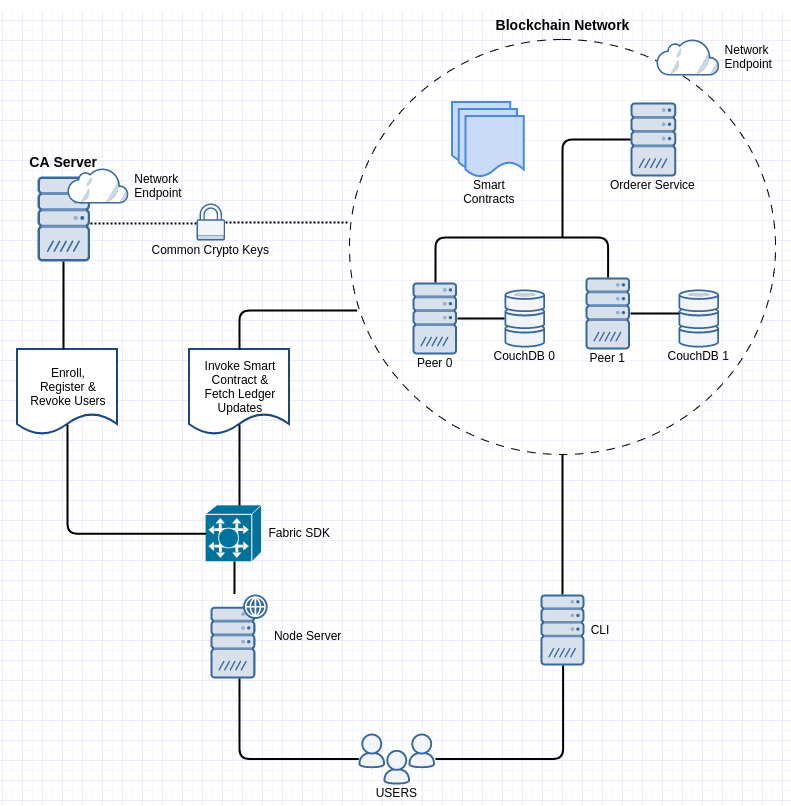
We have 2 hosts HOST1 & HOST2

HOST1 is AWS instance which is manger node

HOST2 is AZURE instance which has to be joined to HOST1 using docker swarm

The container details are as follows:

1. Certificate Authority(CA) – HOST1
2. An Orderer – HOST1
3. 1 peer (peer0) – HOST1
4. 1 peer (peer1) – HOST2
5. CLI – HOST2



**4)STEPS**

1. Login to the HOST1 which is AWS instance

ssh -i ibmblockchain.pem [ubuntu@ec2-3-133-85-222.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-3-133-85-222.us-east-2.compute.amazonaws.com)

1. Initialize docker swarm on HOST1

docker swarm init --advertise-addr 3.133.85.222

1. Run the following command to get the command for adding the other hosts to HOST1

docker swarm join-token manager

The following output will be displayed on HOST1.

Screen%20Shot%202019-11-01%20at%204.12.00%20PM.png

1. Login to our HOST2 instance which is AZURE instance

ssh ubuntu@40.114.54.97

1. Run the docker swarm join command on HOST2. Add the ip address of the HOST2 along with the docker swarm join command using –advertise-addr as mentioned below

docker swarm join --token SWMTKN-1-3n2asqns5iq8x3d48wzyzxt7s6pqyxjuo3n98d014c99tr61bk-7xvvdruvlz9qfqqtq3okpmzbb 3.133.85.222:2377 --advertise-addr 40.114.54.97

Screen%20Shot%202019-11-01%20at%204.21.24%20PM.png

1. Check from the leader node using the following command if both the networks are ready

docker node ls



1. Create the overlay network from HOST. Here the name of network is “my-net”

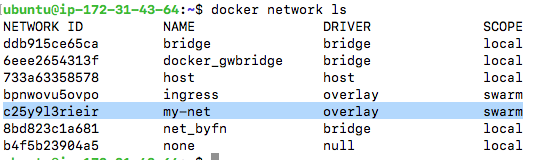
docker network create --attachable --driver overlay my-net

The output of the above command will display the following.

Screen%20Shot%202019-11-01%20at%204.26.08%20PM.png

1. Check using the following command on both the participating networks (HOST1 & HOST2) whether the overlay network is created successfully or not.

docker network ls



1. Clone the following repos on both the hosts

git clone https://code.cognizant.com/IBM-Blockchain-Platform/Multi-Host-Network-Hyperledger.git

1. Generate the network artifacts (crypto Material) – HOST1

cd Multi-Host-Network-Hyperledger/

./bmhn.sh

This will generate network artifacts in” crypto-config” and “channel-artifacts” folder. The network artifatcs should be same on both the hosts. Copy these to the second host. It is important that both the HOSTs must have the same crypto material otherwise the network will not communicate.

**5)Setting Up the Network**

The below scripts have to be run on HOST1. It contains scripts to run CA, orderer couchdb0 and peer0.

All the scripts have to run on separate terminal. The commands should be run inside the “Multi-Host-Network-Hyperledger” folder.

HOST1- AWS

1. CA Sever

In the command below replace **“{put the name of secret key}”** with the name of secret key under **“/crypto-config/peerOrganizations/org1.blockchaingarage.com/ca”**

docker run --rm -it --network="my-net" --name ca. blockchaingarage.com -p 7054:7054 -e FABRIC\_CA\_HOME=/etc/hyperledger/fabric-ca-server -e FABRIC\_CA\_SERVER\_CA\_NAME=ca. blockchaingarage.com -e FABRIC\_CA\_SERVER\_CA\_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1. blockchaingarage.com-cert.pem -e FABRIC\_CA\_SERVER\_CA\_KEYFILE=/etc/hyperledger/fabric-ca-server-config/**{put the name of secret key}** -v $(pwd)/crypto-config/peerOrganizations/org1. blockchaingarage.com/ca/:/etc/hyperledger/fabric-ca-server-config -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=hyp-net hyperledger/fabric-ca sh -c 'fabric-ca-server start -b admin:adminpw -d'

1. Orderer

Execute the following command to run the orderer container

docker run --rm -it --network="my-net" --name orderer.blockchaingarage.com -p 7050:7050 -e ORDERER\_GENERAL\_LOGLEVEL=debug -e ORDERER\_GENERAL\_LISTENADDRESS=0.0.0.0 -e ORDERER\_GENERAL\_LISTENPORT=7050 -e ORDERER\_GENERAL\_GENESISMETHOD=file -e ORDERER\_GENERAL\_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block -e ORDERER\_GENERAL\_LOCALMSPID=OrdererMSP -e ORDERER\_GENERAL\_LOCALMSPDIR=/var/hyperledger/orderer/msp -e ORDERER\_GENERAL\_TLS\_ENABLED=false -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net -v $(pwd)/channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block -v $(pwd)/crypto-config/ordererOrganizations/blockchaingarage.com/orderers/orderer.blockchaingarage.com/msp:/var/hyperledger/orderer/msp -w /opt/gopath/src/github.com/hyperledger/fabric hyperledger/fabric-orderer orderer

1. CouchDB 0 for Peer 0

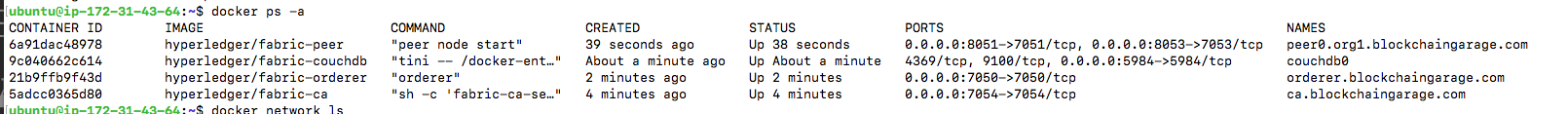
docker run --rm -it --network="my-net" --name couchdb0 -p 5984:5984 -e COUCHDB\_USER= -e COUCHDB\_PASSWORD= -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net hyperledger/fabric-couchdb

1. Peer 0

docker run --rm -it --link orderer.blockchaingarage.com:orderer.blockchaingarage.com --network="my-net" --name peer0.org1.blockchaingarage.com -p 8051:7051 -p 8053:7053 -e CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb0:5984 -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME= -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD= -e CORE\_PEER\_ADDRESSAUTODETECT=true -e CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock -e CORE\_LOGGING\_LEVEL=DEBUG -e CORE\_PEER\_NETWORKID=peer0.org1.blockchaingarage.com -e CORE\_NEXT=true -e CORE\_PEER\_ENDORSER\_ENABLED=true -e CORE\_PEER\_ID=peer0.org1.blockchaingarage.com -e CORE\_PEER\_PROFILE\_ENABLED=true -e CORE\_PEER\_COMMITTER\_LEDGER\_ORDERER=orderer.blockchaingarage.com:7050 -e CORE\_PEER\_GOSSIP\_IGNORESECURITY=true -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net -e CORE\_PEER\_GOSSIP\_EXTERNALENDPOINT=peer0.org1.blockchaingarage.com:7051 -e CORE\_PEER\_TLS\_ENABLED=false -e CORE\_PEER\_GOSSIP\_USELEADERELECTION=false -e CORE\_PEER\_GOSSIP\_ORGLEADER=true -e CORE\_PEER\_LOCALMSPID=Org1MSP -v /var/run/:/host/var/run/ -v $(pwd)/crypto-config/peerOrganizations/org1.blockchaingarage.com/peers/peer0.org1.blockchaingarage.com/msp:/etc/hyperledger/fabric/msp -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-peer peer node start

Run the following command to check if the containers are up and running on HOST1.

docker ps –a



HOST2- AZURE

1. CouchDB 1 for peer1

docker run --rm -it --network="my-net" --name couchdb1 -p 6984:5984 -e COUCHDB\_USER= -e COUCHDB\_PASSWORD= -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net hyperledger/fabric-couchdb

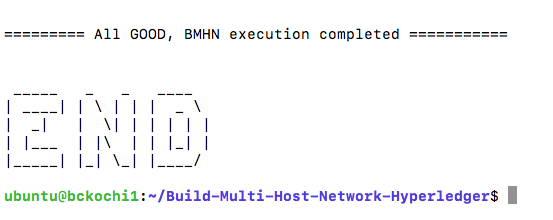
1. Peer 1

docker run --rm -it --network="my-net" --link orderer.blockchaingarage.com:orderer.blockchaingarage.com --link peer0.org1.blockchaingarage.com:peer0.org1.blockchaingarage.com --name peer1.org1.blockchaingarage.com -p 9051:7051 -p 9053:7053 -e CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb1:5984 -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME= -e CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD= -e CORE\_PEER\_ADDRESSAUTODETECT=true -e CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock -e CORE\_LOGGING\_LEVEL=DEBUG -e CORE\_PEER\_NETWORKID=peer1.org1.blockchaingarage.com -e CORE\_NEXT=true -e CORE\_PEER\_ENDORSER\_ENABLED=true -e CORE\_PEER\_ID=peer1.org1.blockchaingarage.com -e CORE\_PEER\_PROFILE\_ENABLED=true -e CORE\_PEER\_COMMITTER\_LEDGER\_ORDERER=orderer.blockchaingarage.com:7050 -e CORE\_PEER\_GOSSIP\_ORGLEADER=true -e CORE\_PEER\_GOSSIP\_EXTERNALENDPOINT=peer1.org1.blockchaingarage.com:7051 -e CORE\_PEER\_GOSSIP\_IGNORESECURITY=true -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net -e CORE\_PEER\_GOSSIP\_BOOTSTRAP=peer0.org1.blockchaingarage.com:7051 -e CORE\_PEER\_GOSSIP\_USELEADERELECTION=false -e CORE\_PEER\_TLS\_ENABLED=false -v /var/run/:/host/var/run/ -v $(pwd)/crypto-config/peerOrganizations/org1.blockchaingarage.com/peers/peer1.org1.blockchaingarage.com/msp:/etc/hyperledger/fabric/msp -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-peer peer node start

1. CLI

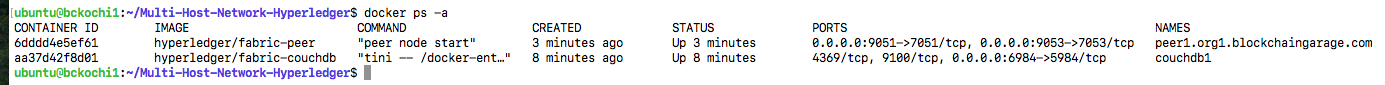
docker run --rm -it --network="my-net" --name cli --link orderer.blockchaingarage.com:orderer.blockchaingarage.com --link peer0.org1.blockchaingarage.com:peer0.org1.blockchaingarage.com --link peer1.org1.blockchaingarage.com:peer1.org1.blockchaingarage.com -p 12051:7051 -p 12053:7053 -e GOPATH=/opt/gopath -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_TLS\_ENABLED=false -e CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock -e CORE\_LOGGING\_LEVEL=DEBUG -e CORE\_PEER\_ID=cli -e CORE\_PEER\_ADDRESS=peer0.org1.blockchaingarage.com:7051 -e CORE\_PEER\_NETWORKID=cli -e CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/users/Admin@org1.blockchaingarage.com/msp -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net -v /var/run/:/host/var/run/ -v $(pwd)/chaincode/:/opt/gopath/src/github.com/hyperledger/fabric/examples/chaincode/go -v $(pwd)/crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ -v $(pwd)/scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/ -v $(pwd)/channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-tools /bin/bash -c './scripts/script.sh'

The CLI script utilises the script.sh file inside the scripts directory and it install the chaincode. If the following is displayed as the output while running the command for CLI, then the execution is completed.



Run the following command to check if the containers are up and running on HOST2

docker ps –a



The script will execute the following:

* Create channel: **mychannel**
* Make peer0 and peer1 join the channel
* After joining successfully, anchor peer is updated (peer0 in this network)
* Install chaincode on both the peers.

**6)Testing the Network**

1. We will again hit the CLI container from HOST2, this time we will exec into it so that we can run the chaincode.

docker run --rm -it --network="my-net" --name cli --link orderer.blockchaingarage.com:orderer.blockchaingarage.com --link peer0.org1.blockchaingarage.com:peer0.org1.blockchaingarage.com --link peer1.org1.blockchaingarage.com:peer1.org1.blockchaingarage.com -p 12051:7051 -p 12053:7053 -e GOPATH=/opt/gopath -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_TLS\_ENABLED=false -e CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock -e CORE\_LOGGING\_LEVEL=DEBUG -e CORE\_PEER\_ID=cli -e CORE\_PEER\_ADDRESS=peer0.org1.blockchaingarage.com:7051 -e CORE\_PEER\_NETWORKID=cli -e CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/users/Admin@org1.blockchaingarage.com/msp -e CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=my-net -v /var/run/:/host/var/run/ -v $(pwd)/chaincode/:/opt/gopath/src/github.com/hyperledger/fabric/examples/chaincode/go -v $(pwd)/crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ -v $(pwd)/scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/ -v $(pwd)/channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-tools /bin/bash

The following will be displayed as output.

Screen%20Shot%202019-11-01%20at%205.10.20%20PM.png

Now we have entered the CLI container, we will execute the commands to instantiate, invoke and query the chaincode.

1. Instantiate chaincode on Peer0

We have to set the environmental variables for peer0. For that run the following command.

# Environment variables for PEER0  
  
CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/users/Admin@org1.blockchaingarage.com/msp

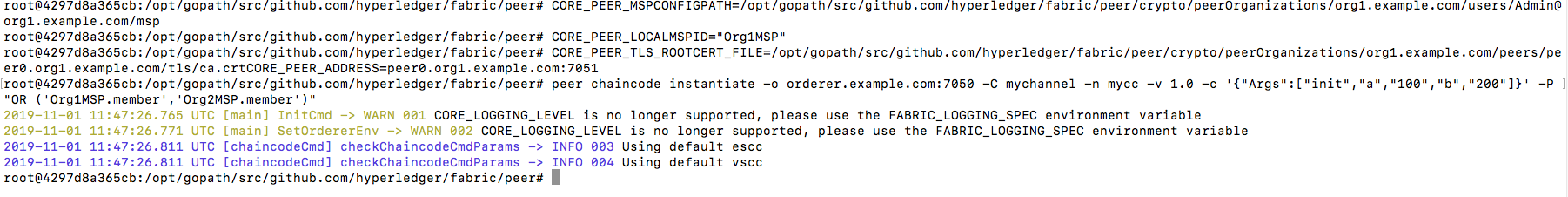
CORE\_PEER\_LOCALMSPID="Org1MSP"

CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/peers/peer0.org1.blockchaingarage.com/tls/ca.crt

CORE\_PEER\_ADDRESS=peer0.org1.blockchaingarage.com:7051

After that execute the below command to instantiate the chaincode.

peer chaincode instantiate -o orderer.blockchaingarage.com:7050 -C mychannel -n mycc -v 1.0 -c '{"Args":["init","a","100","b","200"]}' -P "OR ('Org1MSP.member','Org2MSP.member')"



This will instantiate the chaincode and populate it with a=100 and b=200.

1. Query chaincode on peer1

Environment variables have to be changed for peer1

# Environment variables for Peer1

CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/users/Admin@org1.blockchaingarage.com/msp

CORE\_PEER\_LOCALMSPID="Org1MSP"

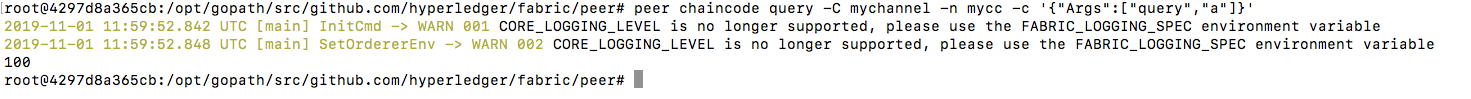
CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/peers/peer0.org1.blockchaingarage.com/tls/ca.crt

CORE\_PEER\_ADDRESS=peer1.org1.blockchaingarage.com:7051

Run the following command to query the chaincode

peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'

It will display 100 as result



Invoke the chaincode on PEER0

Change the environment variables.

# Environment variables for PEER0

CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/users/Admin@org1.blockchaingarage.com/msp

CORE\_PEER\_LOCALMSPID="Org1MSP"

CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchaingarage.com/peers/peer0.org1.blockchaingarage.com/tls/ca.crt

CORE\_PEER\_ADDRESS=peer0.org1.blockchaingarage.com:7051

Run the following command to invoke the chaincode. The value **10** will be moved form **a** to **b.**

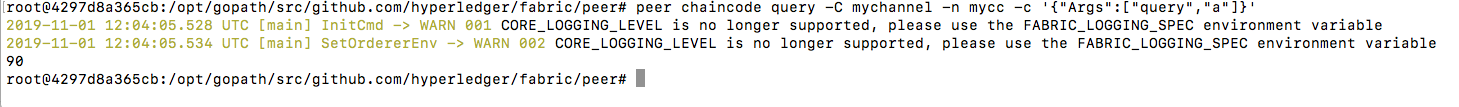
peer chaincode invoke -o orderer.blockchaingarage.com:7050 -C mychannel -n mycc -c '{"Args":["invoke","a","b","10"]}'

Query the chaincode

Let’s confirm that our previous invocation executed properly. We initialized the key a with a value of 100 and just removed 10 with our previous invocation. Therefore, a query against a should reveal 90. The syntax for query is as follows. (we are querying on peer0 so no need to change the environment variables).

# be sure to set the -C and -n flags appropriately

peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'



At this point, as your ledger is populated, you can view the transactions at **(open it on browser on PC1)**

**Peer 0 (PC 1):** <http://ec2-3-133-85-222.us-east-2.compute.amazonaws.com:5984/_utils/#/database/mychannel_/_all_docs>

**Peer1 (PC 2):** <http://40.114.54.97:6984/_utils/#/database/mychannel/_all_docs>

Reference Link

<https://medium.com/@wahabjawed/hyperledger-fabric-on-multiple-hosts-a33b08ef24f>

<https://medium.com/@malliksarvepalli/hyperledger-fabric-1-4-on-multiple-hosts-using-docker-swarm-and-compose-ec668db0bad5>