Hyper Ledger Fabric v1.0

Samples Setup

**Draft Version v0.1**

**24th July, 2017**

## 

## Balance transfer

A sample Node.js app to demonstrate **fabric-client** & **fabric-ca-client** Node.js SDK APIs

### Prerequisites and setup:

* [Docker](https://www.docker.com/products/overview) - v1.12 or higher
* [Docker Compose](https://docs.docker.com/compose/overview/) - v1.8 or higher
* [Git client](https://git-scm.com/downloads) - needed for clone commands
* **Node.js** v6.9.0 - 6.10.0 ( **Node v7+ is not supported** )
* Download docker images

git clone https://github.com/hyperledger/fabric-samples.git

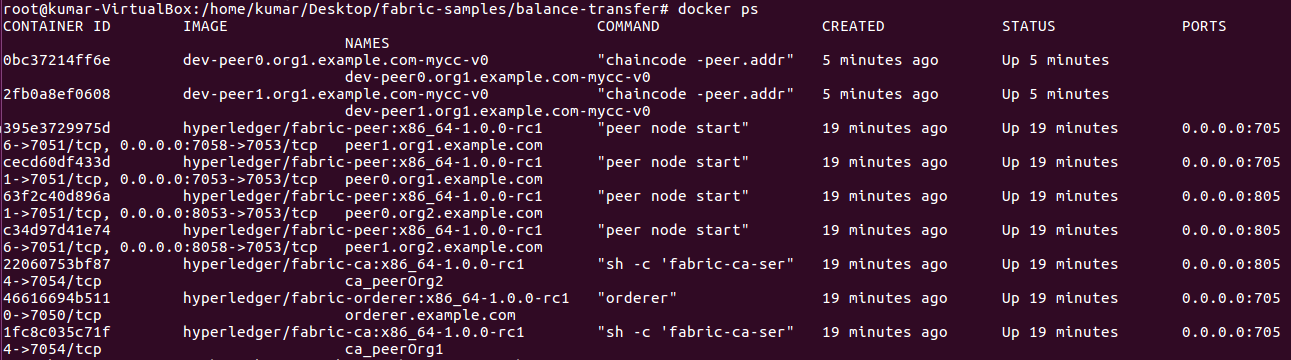
cd fabric-samples/balance-transfer/

docker-compose -f artifacts/docker-compose.yaml up

docker-compose -f artifacts/docker-compose.yaml pull

Once you have completed the above setup, you will have provisioned a local network with the following docker container configuration:

* 2 CAs
* A SOLO orderer
* 4 peers (2 peers per Org)



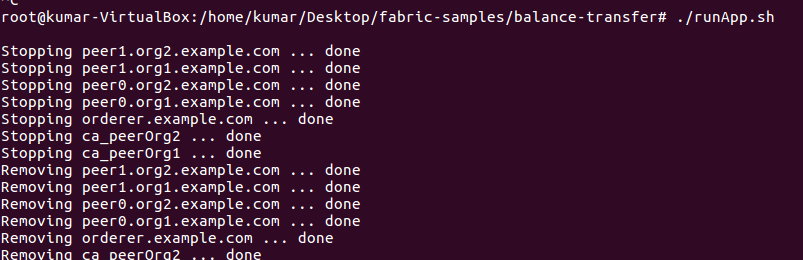
## Running the sample program

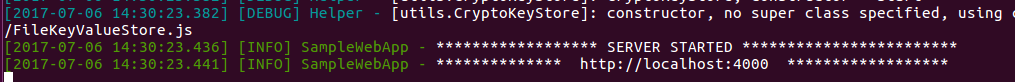
##### **Terminal Window 1**

cd fabric-samples/balance-transfer/

./runApp.sh

* This lauches the required network on your local machine
* Installs the fabric-client and fabric-ca-client node modules
* And, starts the node app on PORT 4000





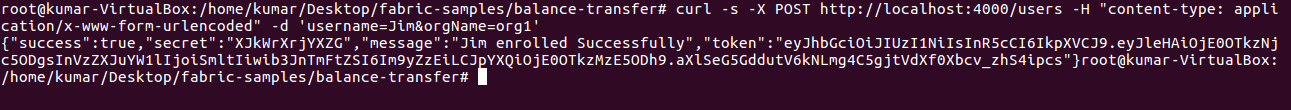
##### **Terminal Window 2**

## Sample REST APIs Requests

### Login Request

* Register and enroll new users in Organization - **Org1**:

curl -s -X POST http://localhost:4000/users -H "content-type: application/x-www-form-urlencoded" -d 'username=Jim&orgName=org1'



### Create Channel request

curl -s -X POST \

http://localhost:4000/channels \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json" \

-d '{

"channelName":"mychannel",

"channelConfigPath":"../artifacts/channel/mychannel.tx"

}'

### 

### Join Channel request

curl -s -X POST \

http://localhost:4000/channels/mychannel/peers \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json" \

-d '{

"peers": ["localhost:7051","localhost:7056"]

}'

### 

### Install chaincode

curl -s -X POST \

http://localhost:4000/chaincodes \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json" \

-d '{

"peers": ["localhost:7051","localhost:7056"],

"chaincodeName":"mycc",

"chaincodePath":"github.com/example\_cc",

"chaincodeVersion":"v0"

}'

### 

### Instantiate chaincode

curl -s -X POST \

http://localhost:4000/channels/mychannel/chaincodes \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json" \

-d '{

"peers": ["localhost:7051"],

"chaincodeName":"mycc",

"chaincodeVersion":"v0",

"functionName":"init",

"args":["a","100","b","200"]

}'

### 

### Invoke request

curl -s -X POST \

http://localhost:4000/channels/mychannel/chaincodes/mycc \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json" \

-d '{

"peers": ["localhost:7051", "localhost:7056"],

"fcn":"move",

"args":["a","b","10"]

}'

### 

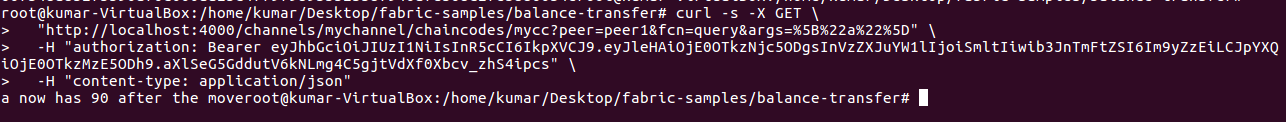
### Chaincode Query

curl -s -X GET \

"http://localhost:4000/channels/mychannel/chaincodes/mycc?peer=peer1&fcn=query&args=%5B%22a%22%5D" \

-H "authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0OTQ4NjU1OTEsInVzZXJuYW1lIjoiSmltIiwib3JnTmFtZSI6Im9yZzEiLCJpYXQiOjE0OTQ4NjE5OTF9.yWaJhFDuTvMQRaZIqg20Is5t-JJ\_1BP58yrNLOKxtNI" \

-H "content-type: application/json"



NOTE:

1. If any of the docker images doesn’t start up, then stop all the images and start using the command “docker rm $(docker ps -a -q)”

# Building First Network

You will have to download and install the hyperledger fabric samples. You will notice that there are a number of samples included in the fabric-samples repository. We will be using the first-network sample. Let’s open that sub-directory now.

**cd first-network**

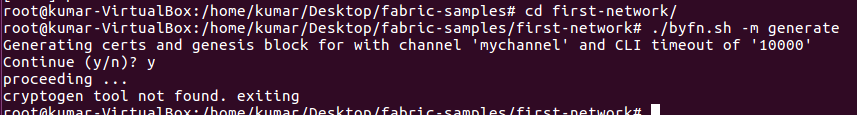
**Lets run it now**

The fully annotated script - byfn.sh - leverages these Docker images to quickly bootstrap a Hyperledger Fabric network comprised of 4 peers representing two different organizations, and an orderer node. It will also launch a container to run a scripted execution that will join peers to a channel, deploy and instantiate chaincode and drive execution of transactions against the deployed chaincode.

### Generate Network Artifacts

**./**byfn**.**sh **-**m generate

This command asks for the acknowledgment to continue give y and enter



If u get the above shown error then you are good to go.

Follow the below steps to eliminate that

1. Install Go

Run the following commands

**wget** [**https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz**](https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz)

**sudo tar -xvf go1.8.3.linux-amd64.tar.gz**

**sudo mv go /usr/local**

1. Set GOPATH

**export GOROOT=/usr/local/go**

**export GOPATH=/usr/local/go/Project/Go**

**export PATH=$GOPATH/bin:$GOROOT/bin:$PATH**

1. Run the fabric inside GOPATH

Run the following commands:

**cd usr/local/go/**

**mkdir –p $GOPATH/src/github.com/hyperledger**

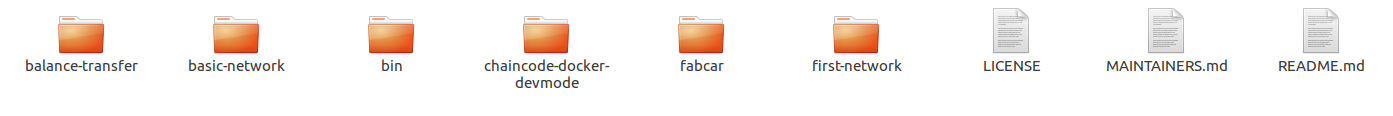
**cd $GOPATH/src/github.com/hyperledger**

**git clone** [**https://github.com/hyperledger/fabric.git**](https://github.com/hyperledger/fabric.git)

**cd fabric**

**make release**

This creates a folder “linux-amd64” file in release folder of fabric. Inside the “linux-amd64” folder there will a bin folder created. Copy that folder and paste inside the fabric samples.



1. Change the releases in the .yaml files to x86\_64-1.0.0 in the below mentioned files as shown in the image for image tag

docker-compose-cli.yaml

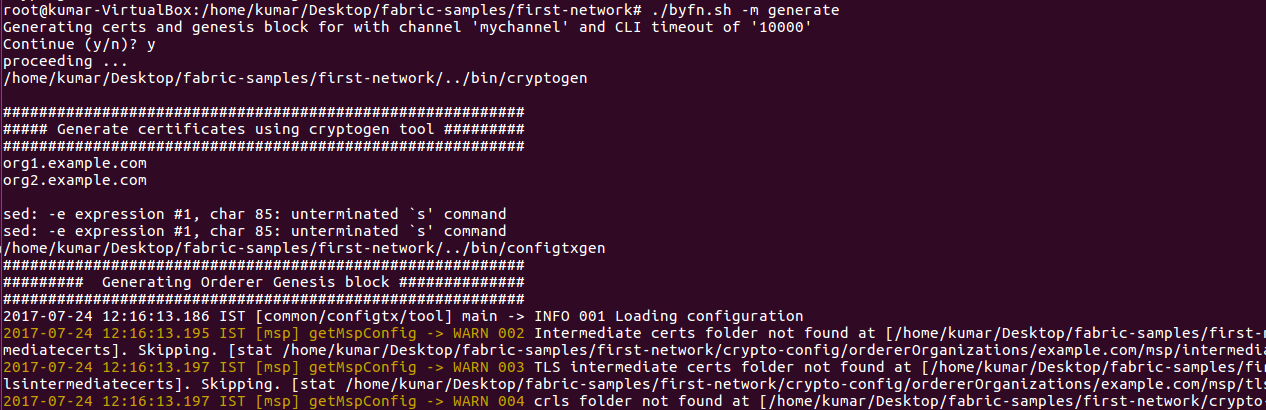
base/docker-compose-base.yaml

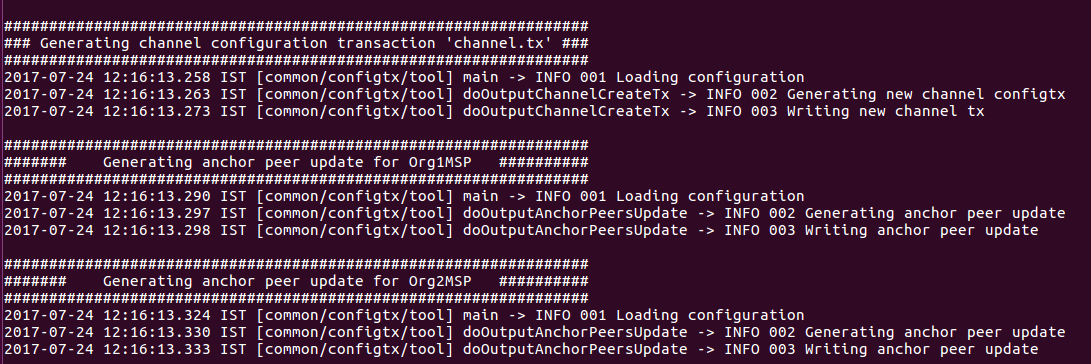
base/peer-base.yaml



Now do the same step as above It will start running as shown in the image below

**./byfn.sh -m generate**

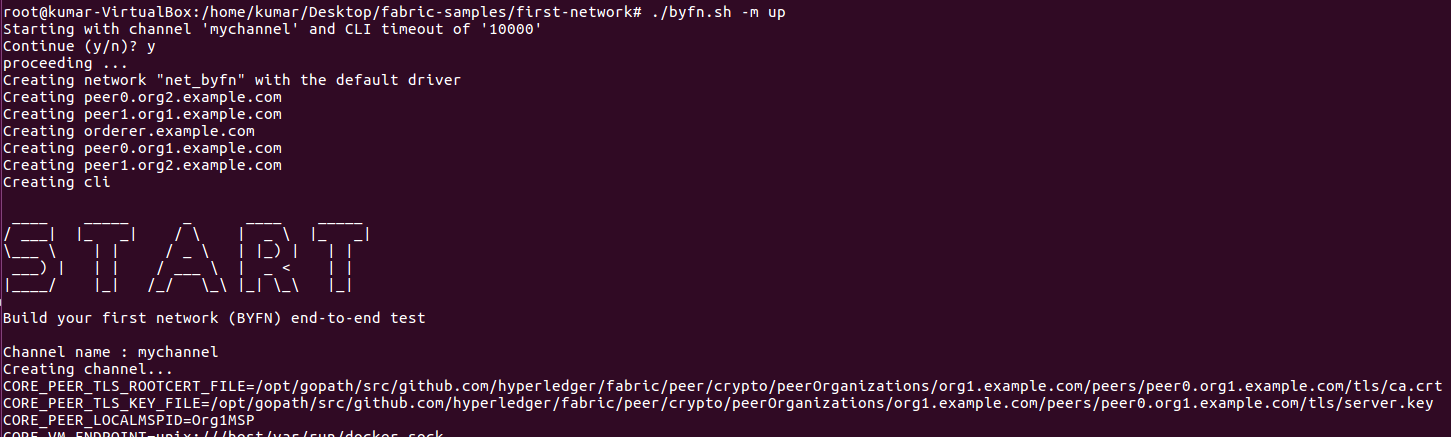




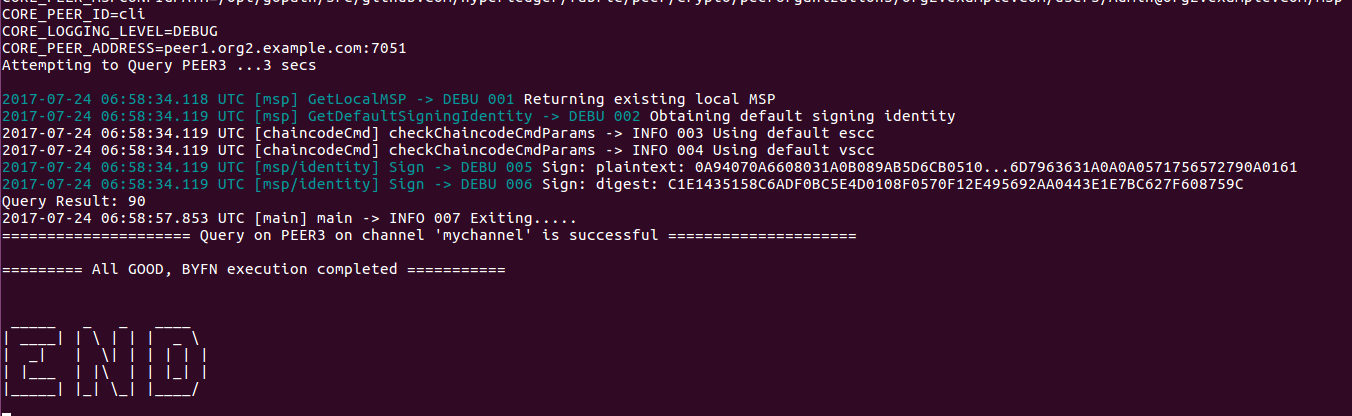
After it stops at this point run the command

**./**byfn**.**sh **-**m up

Acknowledge by typing y and click enter you will see the logs as shown in the image below



The logs will continue from there. This will launch all of the containers, and then drive a complete end-to-end application scenario. Upon successful completion, it should report the following in your terminal window:



**Troubleshooting**

In case of any issues running it next time u have to stop and delete all the images that are created and bring down the network using the command.

**./byfn.sh -m down**

Once this is done below commands will start the network

**./byfn.sh -m generate**

**./byfn.sh -m up**