Lab 1. Creating our first network

In this lab, we will build our first Hyperledger Fabric network.

Install prerequisites.

We have a number of prerequisite software packages that must installed before we can start our hyperledger network.

Specifically we need to install the following:

1. Docker and Docker compose
2. Golang
3. Node.js and NPM

Our hardware platform will be running on Ubuntu Linux Version 16.04.

Step 1.

First we will install docker and docker compose.

1. Add the GPG key for the official docker repository to our system

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
```

2. Add the docker repository to our API sources. (Note, this command is on one line)

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

3 Update the package database

```
$ sudo apt-get update
```

4. Verify that we are going to install docker from the official docker repository rather than the Ubuntu system repo

```
$ apt-cache policy docker-ce
```

.

The above command should produce output like this:

```
      docker-ce:
        Installed: (none)
        Candidate: 17.03.1~ce-0~ubuntu-xenial
        Version table:
           17.03.1~ce-0~ubuntu-xenial 500
              500 https://download.docker.com/linux/ubuntu
      xenial/stable amd64 Packages
           17.03.0~ce-0~ubuntu-xenial 500
         500 https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
```

Note that we haven't yet installed docker or docker compose but the above output indicates that the package will be installed from the correct repository.

5. Now install docker.

```
$ sudo apt-get install -y docker-ce
```

Verify that docker is installed by running the following command:

```
$ sudo systemctl status docker
```

The output of that command should look like this:

```
      ● docker.service - Docker Application Container Engine
         Loaded: loaded (/lib/systemd/system/docker.service;
      enabled; vendor preset: enabled)
         Active: active (running) since Sun 2016-05-01 06:53:52
      CDT; 1 weeks 3 days ago
           Docs: https://docs.docker.com
 Main PID: 749 (docker)
```

Note that the PID will be different for each system.

6. Install docker compose

First we install the Python3 installer.

```
$ sudo apt-get -y install python-pip
```

Then we install docker compose

```
$ pip install docker-compose
```

Step 2. Install golang

Make sure that you are in your home directory. Then download the golang version 1.9 source.

```
$ cd ~
$ curl -O https://storage.googleapis.com/golang/go1.9.linux-
amd64.tar.gz
```

Extract the tarball binary using the tar xvf command, change ownership of the go directory, then move the extracted files to /usr/local/bin

```
$ tar xvf go1.6.linux-amd64.tar.gz
sudo chown -R root:root ./go
sudo mv go /usr/local/bin
```

In your .profile file, set some environment variables. Use whichever editor you prefer.

```
export GOPATH=$HOME/work
export PATH=$PATH:/usr/local/bin/go/bin:$GOPATH/bin
```

Step 3. Install node.js and NPM.

Run the following commands to install node.js

```
$ sudo apt-get update
$ sudo apt-get install nodejs
$ sudo apt-get install npm
```

Step 4. Install Hyperledger Fabric samples and binaries.

Make sure that you are in your home directory. Then use git clone to download and install the hyperledger fabric samples. Change to the fabric-samples directory and download the platform-specific binaries for our fabric network.

```
git clone https://github.com/hyperledger/fabric-samples.git
cd fabric-samples
curl -sSL https://goo.gl/5ftp2f | bash
```

Step 5. Generate our first network

Change into the first-network directory, then run the byfn.sh shell script to generate our network. This network will have four nodes, two organizations and an orderer node. It will launch a docker container that will run an execution script that will join the nodes to a communications channel, deploy and instantiate some chaincode and drive execution of transactions against the deployed chaincode. Note that the default channel name is *mychannel*.

Run the following command. Respond with a 'y' to execute the action

```
$ ./byfn.sh -m generate
```

The output should look like this:

```
Generating certs and genesis block for with channel 'mychannel' and
CLI timeout of '10000'
Continue (y/n)?y
proceeding ...
/Users/xxx/dev/fabric-samples/bin/cryptogen

############################################################
##### Generate certificates using cryptogen tool #########
############################################################
org1.example.com
2017-06-12 21:01:37.334 EDT [bccsp] GetDefault -> WARN 001 Before
using BCCSP, please call InitFactories(). Falling back to bootBCCSP.
...

/Users/xxx/dev/fabric-samples/bin/configtxgen
############################################################
#########  Generating Orderer Genesis block ##############
############################################################
2017-06-12 21:01:37.558 EDT [common/configtx/tool] main -> INFO 001
Loading configuration
2017-06-12 21:01:37.562 EDT [msp] getMspConfig -> INFO 002
intermediate certs folder not found at [/Users/xxx/dev/byfn/crypto-
config/ordererOrganizations/example.com/msp/intermediatecerts].
Skipping.: [stat /Users/xxx/dev/byfn/crypto-
config/ordererOrganizations/example.com/msp/intermediatecerts: no such
file or directory]
...
```

```
2017-06-12 21:01:37.588 EDT [common/configtx/tool] doOutputBlock ->
INFO 00b Generating genesis block
2017-06-12 21:01:37.590 EDT [common/configtx/tool] doOutputBlock ->
INFO 00c Writing genesis block

#################################################################
### Generating channel configuration transaction 'channel.tx' ###
#################################################################
2017-06-12 21:01:37.634 EDT [common/configtx/tool] main -> INFO 001
Loading configuration
2017-06-12 21:01:37.644 EDT [common/configtx/tool]
doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2017-06-12 21:01:37.645 EDT [common/configtx/tool]
doOutputChannelCreateTx -> INFO 003 Writing new channel tx

#################################################################
#######    Generating anchor peer update for Org1MSP   ##########
#################################################################
2017-06-12 21:01:37.674 EDT [common/configtx/tool] main -> INFO 001
Loading configuration
2017-06-12 21:01:37.678 EDT [common/configtx/tool]
doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2017-06-12 21:01:37.679 EDT [common/configtx/tool]
doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update

#################################################################
#######    Generating anchor peer update for Org2MSP   ##########
#################################################################
2017-06-12 21:01:37.700 EDT [common/configtx/tool] main -> INFO 001
Loading configuration
2017-06-12 21:01:37.704 EDT [common/configtx/tool]
doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2017-06-12 21:01:37.704 EDT [common/configtx/tool]
doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
```

This command generates cryptography certificates and keys for all of our nodes as well as create our *genesis block* and a collection of configuration transactions needed to configure a channel.

Step 6. Bring up the network.

Run the following command:

```
$ ./byfn.sh -m up
```

Again, respond with a 'y' when asked if you wish to continue.

You should get output that looks like this:

```
Starting with channel 'mychannel' and CLI timeout of '10000'
Continue (y/n)?y
proceeding ...
Creating network "net_byfn" with the default driver
Creating peer0.org1.example.com
Creating peer1.org1.example.com
Creating peer0.org2.example.com
Creating orderer.example.com
Creating peer1.org2.example.com
Creating cli



 ____    _____      _      ____    _____
/ ___|  |_   _|    / \    |  _ \  |_   _|
\___ \    | |     / _ \   | |_) |   | |
 ___) |   | |    / ___ \  |  _ <    | |
|____/    |_|   /_/   \_\ |_| \_\   |_|

Channel name : mychannel
Creating channel...
.
.   <More log data here>
.
.

2017-05-16 17:08:01.366 UTC [msp] GetLocalMSP -> DEBU 004 Returning
existing local MSP
2017-05-16 17:08:01.366 UTC [msp] GetDefaultSigningIdentity -> DEBU
005 Obtaining default signing identity
2017-05-16 17:08:01.366 UTC [msp/identity] Sign -> DEBU 006 Sign:
plaintext:
0AB1070A6708031A0C08F1E3ECC80510...6D7963631A0A0A0571756572790A0161
2017-05-16 17:08:01.367 UTC [msp/identity] Sign -> DEBU 007 Sign:
digest:
E61DB37F4E8B0D32C9FE10E3936BA9B8CD278FAA1F3320B08712164248285C54
Query Result: 90
2017-05-16 17:08:15.158 UTC [main] main -> INFO 008 Exiting.....
===================== Query on PEER3 on channel 'mychannel' is
successful =====================

===================== All GOOD, BYFN execution completed
=====================



 _____   _   _   ____
| ____| | \ | | |  _ \
|  _|   |  \| | | | | |
| |___  | |\  | | |_| |
|_____| |_| \_| |____/
```

Step 7. Bring down the network.

To bring down the network, simply type the following command:

```
./byfn.sh -m down
```

As before, type 'y' to continue when prompted.