The Chinese University of Hong Kong

Department of Financial Technology

FTEC5520 –Applied Blockchain & Cryptocurrency

Lab2 Guidance – part2

Hyperledger Fabric Setup & Practice on AWS

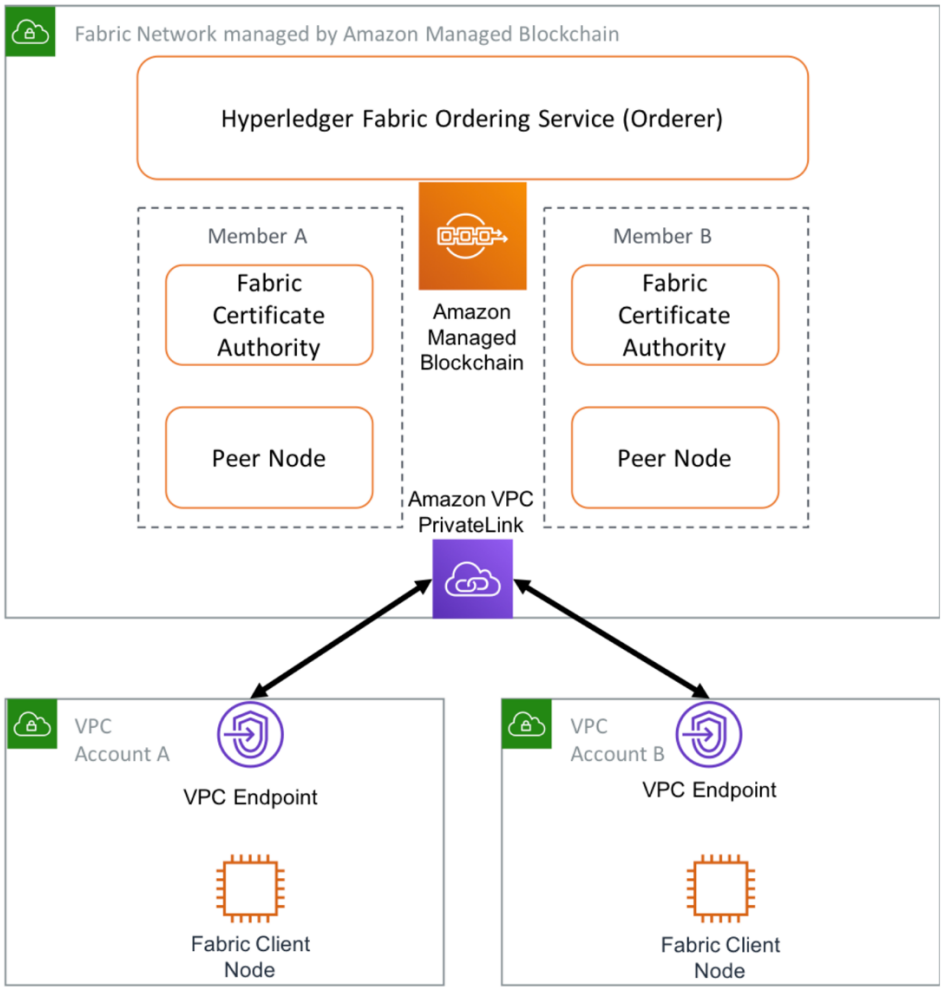
Lab Object

1.1. Building Blockchain With Hyperledger Fabric on AWS

->. A Charity Donation System Application for Non-Profit Organization (NGO)

Main steps:

1. Setup Development Environment, Cloud9, for Building Hyperledger Fabric.
2. Build Hyperledger Fabric Network using AWS Blockchain service
3. Configure Blockchain Network using AWS Blockchain service.
4. Download the NGO repo from GitHub and configure the channel.
5. Run the NGO RESTful API server using Cloud9.
6. Run the NGO Application with User Interface (optional)
7. How to resume and connect back to Cloud9 and fabric client node (optional)

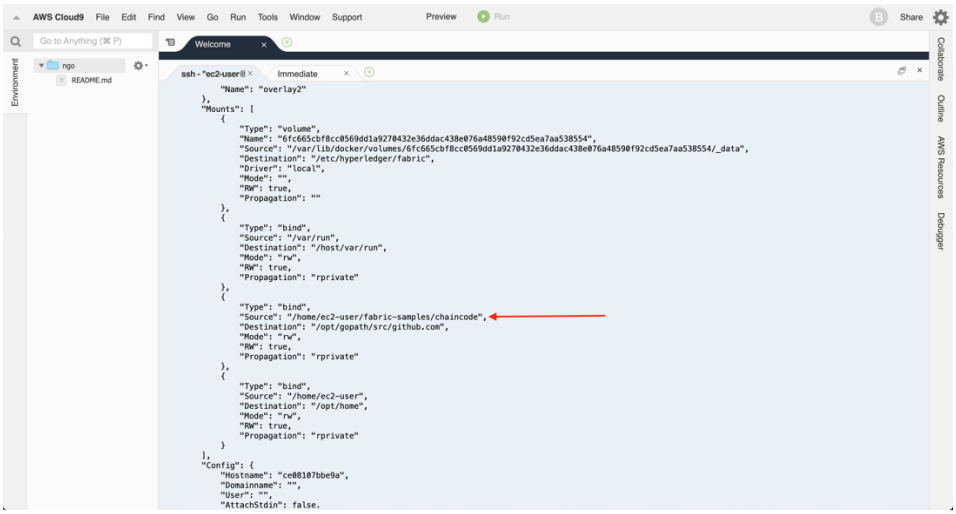


4 Non-profit (NGO) Chaincode Deployment

1. On the Fabric client node. Inspect the docker config.

```
docker inspect cli
```

In the **Mounts** section, the Fabric CLI container mounts a folder from the Fabric client node EC2 instance: /home/ec2-user/fabric-samples/chaincode.



2. Copy the **chaincode** repo from the repository to the fabric-samples directory in order to make it available to CLI.

```
cd ~
```

```
mkdir -p ./fabric-samples/chaincode/ngo
```

```
cp ./non-profit-blockchain/ngo-chaincode/src/* ./fabric-samples/chaincode/ngo
```

3. Install the chaincode on your peer.

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER" cli peer chaincode install -n ngo -l node -v v0 -p /opt/gopath/src/github.com/ngo
```

(Chaincode name: ngo, in language: node.js, with path: /opt/gopath/src/github.com/ngo)

You will see the command with result:

```
[ec2-user@ip-10-0-104-173 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER"  \
>     cli peer chaincode install -n ngo -l node -v v0 -p /opt/gopath/src/github.com/ngo
2019-06-20 09:56:27.016 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-06-20 09:56:27.016 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-06-20 09:56:27.016 UTC [container] WriteFolderToTarPackage -> INFO 003 rootDirectory = /opt/gopath/src/github.com/ngo
2019-06-20 09:56:27.025 UTC [chaincodeCmd] install -> INFO 004 Installed remotely response:<status:200 payload:"OK" >
```

```
[ec2-user@ip-10-0-175-133 ~]$ ls
admin-msp               fabric-samples              managedblockchain-tls-chain.pem  peer-exports.sh
configtx.yaml           go                          mychannel.pb
docker-compose-cli.yaml go1.10.3.linux-amd64.tar.gz non-profit-blockchain
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER"  \
>     cli peer chaincode install -n ngo -l node -v v0 -p /opt/gopath/src/github.com/ngo
2019-10-25 12:13:06.014 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-10-25 12:13:06.014 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-10-25 12:13:06.015 UTC [container] WriteFolderToTarPackage -> INFO 003 rootDirectory = /opt/gopath/src/g
ithub.com/ngo
2019-10-25 12:13:06.024 UTC [chaincodeCmd] install -> INFO 004 Installed remotely response:<status:200 payloa
d:"OK" >
[ec2-user@ip-10-0-175-133 ~]$
```

4. Instantiate the chaincode in order to bind ngo chaincode to mychannel.

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER" cli peer chaincode instantiate -o $ORDERER -C mychannel -n ngo -v v0 -c '{"Args":["init"]}' --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER"  \
>     cli peer chaincode instantiate -o $ORDERER -C mychannel -n ngo -v v0 -c '{"Args":["init"]}' --cafile /o
pt/home/managedblockchain-tls-chain.pem --tls
2019-10-25 12:13:43.277 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-10-25 12:13:43.277 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
[ec2-user@ip-10-0-175-133 ~]$
```

5. Query the chaincode to check whether it is installed.

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDonors"]}'
```

You will get the following result with no donor:

```
2019-06-20 10:00:13.381 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
[ec2-user@ip-10-0-104-173 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDonors"]}'
[]
```

```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDonors"]}'
[]
[ec2-user@ip-10-0-175-133 ~]$
```

6. Invoke a transaction to add 2 donors.

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode invoke -C mychannel -n ngo -c  '{"Args":["createDonor","{\"donorUserName\": \"edge\", \"email\": \"edge@def.com\", \"registeredDate\": \"2018-10-22T11:52:20.182Z\"}"]}' -o  $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode invoke -C mychannel -n ngo \
>     -c  '{"Args":["createDonor","{\"donorUserName\": \"edge\", \"email\": \"edge@def.com\", \"registeredDat
e\": \"2018-10-22T11:52:20.182Z\"}"]}' -o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
2019-10-25 12:14:53.787 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. re
sult: status:200
[ec2-user@ip-10-0-175-133 ~]$
```

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCE
RT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_ADD
RESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPA
TH=$MSP_PATH" cli peer chaincode invoke -C mychannel -n ngo -c  '{"A
rgs":["createDonor","{\"donorUserName\": \"braendle\", \"email\": \"
braendle@def.com\", \"registeredDate\": \"2018-11-05T14:31:20.182Z\
"}"]}' -o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pe
m --tls
```

```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode invoke -C mychannel -n ngo \
>     -c  '{"Args":["createDonor","{\"donorUserName\": \"braendle\", \"email\": \"braendle@def.com\", \"regis
teredDate\": \"2018-11-05T14:31:20.182Z\"}"]}' -o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem
 --tls2019-10-25 12:15:32.502 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successf
ul. result: status:200
[ec2-user@ip-10-0-175-133 ~]$
```

You may meet this error, try to check the command you input just now:

7. Query the chaincode to view the results.

   Query All Users:

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCE
RT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_ADD
RESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPA
TH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Arg
s":["queryAllDonors"]}'
```

Query with result:


```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/op
t/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDonors"]}'[{"Key":"donorbraendle","R
ecord":{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","do
cType":"donor"}},{"Key":"donoredge","Record":{"donorUserName":"edge","email":"edge@def.com","registeredDate":
"2018-10-22T11:52:20.182Z","docType":"donor"}}]
[ec2-user@ip-10-0-175-133 ~]$
```

   Query Specific User:

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCE
RT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PEER_ADD
RESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPA
```

```
TH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Arg
s":["queryDonor","{\"donorUserName\": \"edge\"}"]}'
```

Query with result:

```
[ec2-user@ip-10-0-104-173 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryDonor","{\"donorUserName\": \"edge\"}"]}'
{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}
```

```
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem"     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH"     cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryDonor","{\"donorUserName\": \"edge\"}"]}'{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}
[ec2-user@ip-10-0-175-133 ~]$
```

5 Run the RESTful API server.

1. (On the Fabric client node) Install the Nodejs.

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

```
[ec2-user@ip-10-0-175-133 ~]$ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh |
bash  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11329  100 11329    0     0  73795      0 --:--:-- --:--:-- --:--:-- 73564
=> Downloading nvm from git to '/home/ec2-user/.nvm'
=> Cloning into '/home/ec2-user/.nvm'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 7537 (delta 4), reused 12 (delta 4), pack-reused 7520
Receiving objects: 100% (7537/7537), 2.47 MiB | 23.90 MiB/s, done.
Resolving deltas: 100% (4759/4759), done.
* (HEAD detached at v0.33.0)
  master
=> Compressing and cleaning up git repository
Counting objects: 7537, done.
Compressing objects: 100% (7481/7481), done.
Writing objects: 100% (7537/7537), done.
Total 7537 (delta 5041), reused 2272 (delta 0)

=> Appending nvm source string to /home/ec2-user/.bashrc
=> bash_completion source string already in /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:


export NVM_DIR="/home/ec2-user/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ec2-user@ip-10-0-175-133 ~]$
```

2. Install load version manager.

```
. ~/.nvm/nvm.sh
```

```
nvm install lts/carbon
```

```
[ec2-user@ip-10-0-175-133 ~]$ nvm install lts/carbon
Downloading and installing node v8.16.2...
Downloading https://nodejs.org/dist/v8.16.2/node-v8.16.2-linux-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v8.16.2 (npm v6.4.1)
nvm_ensure_default_set: a version is required
[ec2-user@ip-10-0-175-133 ~]$
```

```
nvm use lts/carbon
```

```
[ec2-user@ip-10-0-175-133 ~]$ nvm use lts/carbon
Now using node v8.16.2 (npm v6.4.1)
[ec2-user@ip-10-0-175-133 ~]$
```

You should see:

```
[ec2-user@ip-10-0-104-173 ~]$ . ~/.nvm/nvm.sh
[ec2-user@ip-10-0-104-173 ~]$ nvm install lts/carbon
Downloading and installing node v8.16.0...
Downloading https://nodejs.org/dist/v8.16.0/node-v8.16.0-linux-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v8.16.0 (npm v6.4.1)
nvm_ensure_default_set: a version is required
[ec2-user@ip-10-0-104-173 ~]$ nvm use lts/carbon
Now using node v8.16.0 (npm v6.4.1)
```

3. Install g++.

```
sudo yum install gcc-c++ -y
```

```
[ec2-user@ip-10-0-175-133 ~]$ sudo yum install gcc-c++ -y
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                                                 | 2.1 kB  00:00:00
amzn-updates                                                              | 2.5 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package gcc-c++.noarch 0:4.8.5-1.22.amzn1 will be installed
--> Processing Dependency: gcc = 4.8.5-1.22.amzn1 for package: gcc-c++-4.8.5-1.22.amzn1.noarch
--> Processing Dependency: gcc48-c++ >= 4.8.5 for package: gcc-c++-4.8.5-1.22.amzn1.noarch
--> Processing Dependency: libstdc++48 >= 4.8.5 for package: gcc-c++-4.8.5-1.22.amzn1.noarch
--> Running transaction check
---> Package gcc.noarch 0:4.8.5-1.22.amzn1 will be installed
---> Package gcc48-c++.x86_64 0:4.8.5-28.142.amzn1 will be installed
---> Package libstdc++48.x86_64 0:4.8.5-28.142.amzn1 will be installed
--> Finished Dependency Resolution
```

……

```
Installed:
  gcc-c++.noarch 0:4.8.5-1.22.amzn1

Dependency Installed:
  gcc.noarch 0:4.8.5-1.22.amzn1                    gcc48-c++.x86_64 0:4.8.5-28.142.amzn1
  libstdc++48.x86_64 0:4.8.5-28.142.amzn1

Complete!
[ec2-user@ip-10-0-175-133 ~]$
```

4. Install the dependencies with *npm install* to download packages with Node Package Manager.

```
cd ~/non-profit-blockchain/ngo-rest-api
```

```
npm install
```

```
[ec2-user@ip-10-0-175-133 ngo-rest-api]$ npm install
[                ..........] | diffTrees: sill install generateActionsToTake
```

…

```
node-pre-gyp WARN Using request for node-pre-gyp https download
[grpc] Success: "/home/ec2-user/non-profit-blockchain/ngo-rest-api/node_modules/grpc
/src/node/extension_binary/node-v57-linux-x64-glibc/grpc_node.node" is installed via
 remote
[                .....] - postinstall:ipaddr.js: info lifecycle ipaddr.js@1.8.0~postins
[                .....] | postinstall:ansi-regex: info lifecycle ansi-regex@3.0.0~posti
[                .....] | postinstall:os-tmpdir: info lifecycle os-tmpdir@1.0.2~postins
[                .....] | install:ngo-rest-api: sill linkStuff ngo-rest-api@1.0.0 has /
[                .....] | prepare:ngo-rest-api: info lifecycle ngo-rest-api@1.0.0~prepa
[                .....] | prepare:ngo-rest-api: info lifecycle ngo-rest-api@1.0.0~prepa
npm WARN ngo-rest-api@1.0.0 No repository field.

added 877 packages from 1330 contributors and audited 3666 packages in 1571.178s
found 70 vulnerabilities (1 low, 5 moderate, 63 high, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
[ec2-user@ip-10-0-175-133 ngo-rest-api]$
```

5. Generate the connection profile used to connect to Fabric network.

```
cd ~/non-profit-blockchain/ngo-rest-api/connection-profile
```

```
./gen-connection-profile.sh
```

```
cat ~/non-profit-blockchain/tmp/connection-profile/ngo-connection-profile.yaml
```

You can see the content of the *ngo-connection-profile.yaml* with the endpoint of the order, peer and CA. The REST API uses the Fabric SDK to connect to order, peer and CA with these endpoints.

6. Check the config file used by app.js.

```
cd ~/non-profit-blockchain/ngo-rest-api
```

```
vim config.json
```

```
{
    "host":"localhost",
    "port":"3000",
    "channelName":"mychannel",
    "chaincodeName":"ngo",
    "eventWaitTime":"30000",
    "peers":[
        "peer1"
    ],
    "admins":[
       {
          "username":"admin",
          "secret":"Adminpwd1!"
       }
    ]
 }
```

Make sure the peer name in config.json (under 'peers:') is the same as the peer name in the connection profile.

//Where can I get the peer name from the connection profile? -by Ricci

** If you use different username and password for this Fabric Network account, change the **username** and **secret** under **admins** to ***your own set of username and password*** in this config.json file.

7. Make sure everything is set correctly. Then run the application.

```
cd ~/non-profit-blockchain/ngo-rest-api
```

```
nvm use lts/carbon
```

```
node app.js &
```

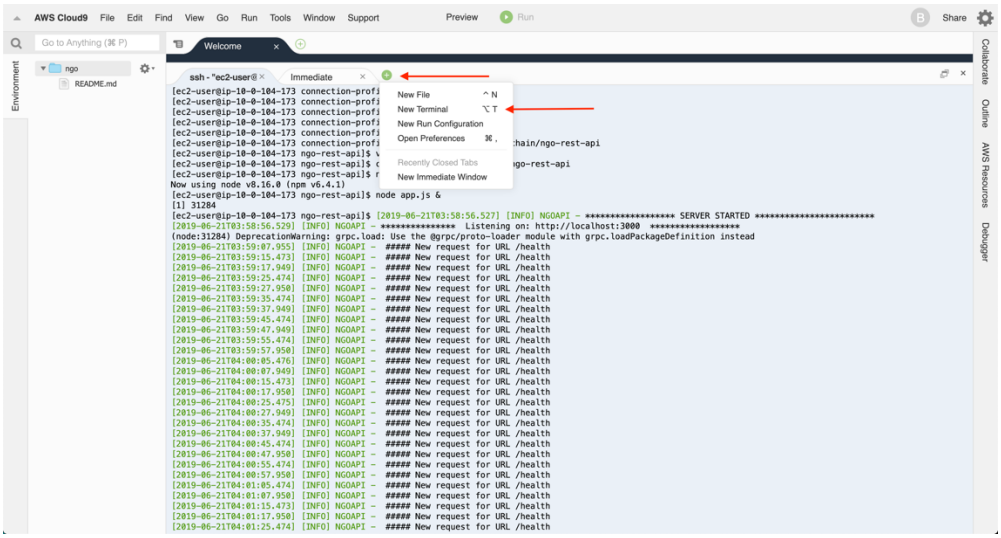
The server will run like this:

```
[ec2-user@ip-10-0-104-173 ngo-rest-api]$ cd ~/non-profit-blockchain/ngo-rest-api
[ec2-user@ip-10-0-104-173 ngo-rest-api]$ nvm use lts/carbon
Now using node v8.16.0 (npm v6.4.1)
[ec2-user@ip-10-0-104-173 ngo-rest-api]$ node app.js &
[1] 31284
[ec2-user@ip-10-0-104-173 ngo-rest-api]$ [2019-06-21T03:58:56.527] [INFO] NGOAPI - ********** SERVER STARTED **********
[2019-06-21T03:58:56.529] [INFO] NGOAPI - **********  Listening on: http://localhost:3000  **********
(node:31284) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
```

```
[ec2-user@ip-10-0-175-133 ngo-rest-api]$ node app.js
[2019-10-25T13:21:55.072] [INFO] NGOAPI - ********** SERVER STARTED ********
**********
[2019-10-25T13:21:55.074] [INFO] NGOAPI - **********  Listening on: http://loca
lhost:3000  **********
(node:5734) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with gr
pc.loadPackageDefinition instead
[2019-10-25T13:22:14.679] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:14.685] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:24.675] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:24.684] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:34.677] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:34.683] [INFO] NGOAPI -  ##### New request for URL /health
[2019-10-25T13:22:39.280] [INFO] NGOAPI -  ##### New request for URL /users
[2019-10-25T13:22:39.280] [INFO] NGOAPI - ========== POST on Users
[2019-10-25T13:22:39.280] [INFO] NGOAPI - ##### End point : /users
[2019-10-25T13:22:39.280] [INFO] NGOAPI - ##### POST on Users- username : michael
[2019-10-25T13:22:39.280] [INFO] NGOAPI - ##### POST on Users - userorg  : Org1
[2019-10-25T13:22:39.281] [INFO] Connection - ========== START getRegisteredUser -
 for org Org1 and user michael
[2019-10-25T13:22:39.281] [INFO] Connection - ========== START getClientForOrg for
 org Org1 and user undefined
[2019-10-25T13:22:39.281] [INFO] Connection - ##### getClient - Loading connection p
rofiles from file: ../tmp/connection-profile/ngo-connection-profile.yaml and ../tmp/
connection-profile/org1/client-org1.yaml
[2019-10-25T13:22:39.602] [INFO] Connection - ========== END getClientForOrg for o
rg Org1 and user undefined


[2019-10-25T13:22:39.603] [INFO] Connection - ##### getRegisteredUser - User michael
 was not enrolled, so we will need an admin user object to register
[2019-10-25T13:22:39.604] [INFO] Connection - ##### getRegisteredUser - Got hfc { [F
unction: Client]
  Peer: [Function: Peer],
  ChannelEventHub: [Function: ChannelEventHub],
  Orderer: [Function: Orderer],
  Channel: [Function: Channel],
  User: [Function: User] }
[2019-10-25T13:22:39.604] [INFO] Connection - ##### getRegisteredUser - Got admin pr
operty [ { username: 'admin', secret: 'Adminpwd1!' } ]
[2019-10-25T13:22:39.844] [ERROR] Connection - ##### getRegisteredUser - Failed to g
et registered user: michael with error: Error: Enrollment failed with errors [[{"cod
```

8. On the top of the command line, click the '+' button and choose **New Terminal** to open a new terminal window.



9. SSH to the client node in the new terminal window with another session. The EC2URL can be found in the CloudFoundation. [ngo-fabric-client-node (in **Stacks**) > Outputs]

```
ssh -i ~/ngo-keypair.pem <the value of EC2URL>
```

```
Blockchain-Admin:~ $ ssh -i ~/ngo-keypair.pem ec2-54-167-192-212.compute-1.amazonaws.com
```

10. Test the REST API.

10.1. Register/enroll a user.

- Call the REST resource users and create a user called michael.

```
curl -s -X POST http://localhost:3000/users -H "content-type: applic
ation/x-www-form-urlencoded" -d 'username=michael&orgName=Org1'
```

```
ec2-user:~/environment $ cd
ec2-user:~ $ ssh -i *.pem ec2-54-86-76-144.compute-1.amazonaws.com
Last login: Fri Oct 25 12:33:05 2019 from fw-9807.ie.cuhk.edu.hk

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
20 package(s) needed for security, out of 34 available
Run "sudo yum update" to apply all updates.
cli is up-to-date
[ec2-user@ip-10-0-175-133 ~]$ curl -s -X POST http://localhost:3000/users -H "conten
t-type: application/x-www-form-urlencoded" -d 'username=michael&orgName=Org1'
{"success":false,"message":"failed Error: Enrollment failed with errors [[{\"code\":
20,\"message\":\"Authorization failure\"}]]"}[ec2-user@ip-10-0-175-133 ~]$

[ec2-user@ip-10-0-175-133 ~]$ curl -s -X POST http://localhost:3000/users -H "conten
t-type: application/x-www-form-urlencoded" -d 'username=michael&orgName=Org1'{"succe
ss":true,"secret":"","message":"michael enrolled Successfully"}[ec2-user@ip-10-0-175
-133 ~]$
```

Result of this REST API call:

```
{"success":true,"secret":"","message":"michael enrolled Successfully"}
```

10.2. Post a Donor through REST API instead of in Fabric layer.

- Do the same as create a donor in chaincode.

```
curl -s -X POST "http://localhost:3000/donors" -H "content-type: application/json" -d '{"donorUserName": "edge2", "email": "edge2@def.com", "registeredDate": "2018-10-22T11:52:20.182Z"}'
```

You will get back a transaction ID upon successful calling:

```
[ec2-user@ip-10-0-104-173 ~]$ curl -s -X POST http://localhost:3000/users -H "content-type: application/x-www-form-urlencoded" -d 'username=michael&orgName=Org1'
{"success":true,"secret":"","message":"michael enrolled Successfully"}[ec2-user@ip-10-0-104-173 ~]$ curl -s -X POST "http://localhost:3000/donors" -H "content-type: application/json" -d '{
>    "donorUserName": "edge2",
>    "email": "edge2@def.com",
>    "registeredDate": "2018-10-22T11:52:20.182Z"
> }'
{"transactionId":"506cdbe60e2b91d6fc15f45f4d950de63ac422a8c44429d045aef6f06c2f83d9"}[ec2-user@ip-10-0-104-173 ~]$
```

```
[ec2-user@ip-10-0-175-133 ~]$ curl -s -X POST "http://localhost:3000/donors" -H "content-type: application/json" -d '{
>    "donorUserName": "edge2",
>    "email": "edge2@def.com",
>    "registeredDate": "2018-10-22T11:52:20.182Z"
> }'{"transactionId":"af4e3ff639d12de61f8d291f6b6a5c33399bad6f1d61c879414f8b5a3728465b"}
[ec2-user@ip-10-0-175-133 ~]$
```

10.3. Query all the donors with REST API.

```
curl -s -X GET   "http://localhost:3000/donors" -H "content-type: application/json"
```

You will see the command with the results:

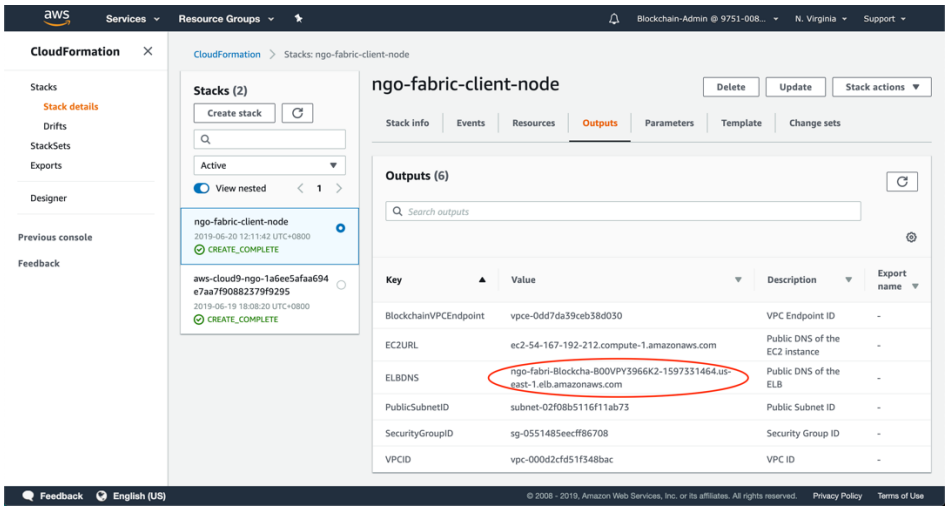
```
[ec2-user@ip-10-0-104-173 ~]$ curl -s -X GET   "http://localhost:3000/donors" -H "content-type: application/json"
[{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"},{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"},{"donorUserName":"edge2","email":"edge2@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}][ec2-user@ip-10-0-104-173 ~]$
```

```
[ec2-user@ip-10-0-175-133 ~]$ curl -s -X GET   "http://localhost:3000/donors" -H "content-type: application/json"[{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"},{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"},{"donorUserName":"edge2","email":"edge2@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}][ec2-user@ip-10-0-175-133 ~]$
```

10.4. Load the workshop test data.

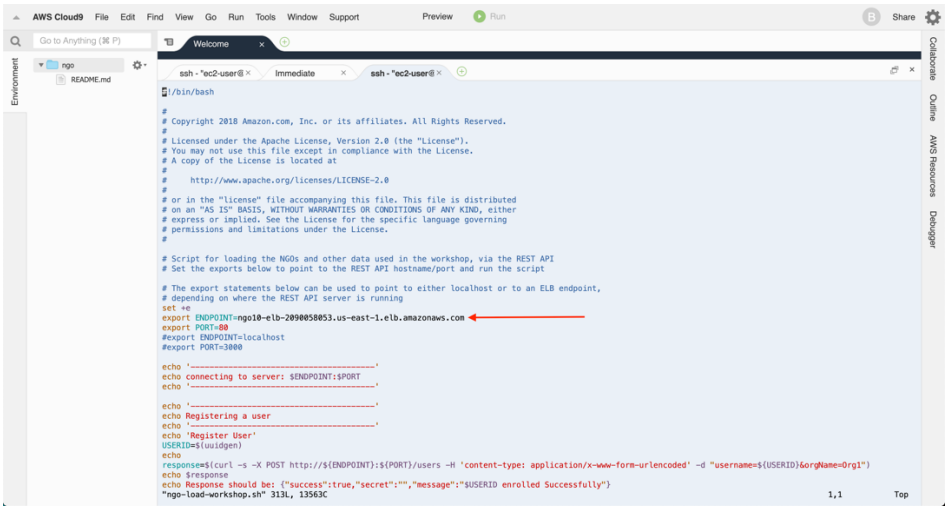
```
cd ~/non-profit-blockchain/ngo-rest-api
vi ngo-load-workshop.sh
```

On the other side, go to the CloudFormation with **ngo-fabric-client-node** and Outputs (where you get the EC2URL). Copy the value of key **ELBDNS** as shown below.



Then paste it to the first export (at the very beginning) in the ngo-load-workshop.sh file (as shown below) and save the file.

* It could point to localhost if you run this on the Fabric client node. If you use localhost you also need to change the port to 3000.



10.5. Run the script to run the workshop test data.

```
cd ~/non-profit-blockchain/ngo-rest-api
./ngo-load-workshop.sh
```

- It creates some donors and some donations.

Last part of the result would be like this:

```
Transaction ID is {"transactionId":"09b160895707032dab2ce431033742beed14e15c81d7ec85765a1abc1256ec80"}

Transaction ID is {"transactionId":"9af32cacf3a6005cf7f5e84a90b543a4cab2518f23655b4195e871cb9b0b9a0a"}

Transaction ID is {"transactionId":"22845a130e318569980d06fdecfed66faab12b0adfd9903f1c69d1124d8c0b75"}

Transaction ID is {"transactionId":"4048783b9ac21f3697fa23a09840490da6853560bb6faeac569caddf5c955bef"}

Transaction ID is {"transactionId":"ab0fbe9001eedff33544785224f3f1cead73710b3097f2124dff09381ad0e885"}

Transaction ID is {"transactionId":"88be22f15204c5c41540a941c5ae9f2e39fe3c24a173147dc82592341e0408c3"}

Transaction ID is {"transactionId":"2304c9e50a229e190c93eb4f29e701118564846dca8d619a118a355eb15348f9"}

Transaction ID is {"transactionId":"dfeedc6cadacd3262feba4444a14fffe178e3f6c3d32adf5b252808a3efc4956"}

Transaction ID is {"transactionId":"dfbffe037b44f09e608e326845816f8683ef61f7f456decb7495d47586e81df3"}

Transaction ID is {"transactionId":"708559e6d53c94e65a8855c2b118ed84e707d4c42fafb958818ccabbc75005bd"}
Query all Donations

[{"donationId":"318dd663-2e5c-4b20-8ef4-7df78ebaa24a","donationAmount":520,"donationDate":"2018-09-20T12:41:59.582Z","donorUserName":"edge","ngoRegistrationNum
ber":"1101","docType":"donation"},{"donationId":"420775f8-69f6-49fe-98d5-2b1178efb4e7","donationAmount":200,"donationDate":"2018-09-18T07:41:59.582Z","donorUse
rName":"edge","ngoRegistrationNumber":"1103","docType":"donation"},{"donationId":"5513ce6b-1809-4ded-929c-bc1055683011","donationAmount":430,"donationDate":"20
18-08-09T09:32:59.582Z","donorUserName":"braendle","ngoRegistrationNumber":"1103","docType":"donation"},{"donationId":"99bb028b-4a88-4e83-b9cb-b9ad704d776c","d
onationAmount":760,"donationDate":"2018-09-18T07:41:59.582Z","donorUserName":"jane","ngoRegistrationNumber":"1105","docType":"donation"},{"donationId":"b61bc58
4-6dcb-4851-ae9c-3a3fcc37603f","donationAmount":900,"donationDate":"2018-09-09T06:32:59.582Z","donorUserName":"louisa","ngoRegistrationNumber":"1103","docType"
:"donation"},{"donationId":"b8cd5ec2-e974-44e8-9e93-04495ee70e99","donationAmount":25,"donationDate":"2018-09-09T06:32:59.582Z","donorUserName":"louisa","ngoRe
gistrationNumber":"1101","docType":"donation"},{"donationId":"d138bc17-f34e-4c89-b6f8-5f03b3a68117","donationAmount":255,"donationDate":"2018-09-18T07:41:59.58
2Z","donorUserName":"jane","ngoRegistrationNumber":"1105","docType":"donation"},{"donationId":"d8873b2e-79c5-40f8-b86c-297e9c559431","donationAmount":100,"donа
tionDate":"2018-09-20T12:41:59.582Z","donorUserName":"edge","ngoRegistrationNumber":"1102","docType":"donation"},{"donationId":"f41a9dc7-0971-42a3-b9f1-4cdbbf0
a73a9","donationAmount":120,"donationDate":"2018-09-18T07:41:59.582Z","donorUserName":"edge","ngoRegistrationNumber":"1104","docType":"donation"},{"donationId"
:"fffdc361-b55a-415b-869f-c4dd96e514d0","donationAmount":44,"donationDate":"2018-08-09T09:32:59.582Z","donorUserName":"braendle","ngoRegistrationNumber":"1103"
,"docType":"donation"}]----------------------------------------
Spend
----------------------------------------
Create Spend

Transaction ID is {"transactionId":"ee5eedba7f5e7ddd906b8324158040dc22165f41e254a96ecc7c2fea6deae767"}

Transaction ID is {"transactionId":"8502e3c7cc4ec64e1a9443d93621beb0492d72564671dc69dbb5f3c3a701a7f1"}
```

```
[ec2-user@ip-10-0-175-133 ngo-rest-api]$ ./ngo-load-workshop.sh
----------------------------------------
connecting to server: ngo-block-Blockcha-1CIAWIJTJO7F6-1589927185.us-east-1.elb.amaz
onaws.com:3000
----------------------------------------
----------------------------------------
Registering a user
----------------------------------------
Register User


Response should be: success:true secret: message:f09f9510-a77f-4c78-a3ff-38ecc5648df
c enrolled Successfully
----------------------------------------
NGOs
----------------------------------------
Creating NGO - 1101

Transaction ID is
Creating NGO - 1102

Transaction ID is
Creating NGO - 1103

Transaction ID is
Creating NGO - 1104

Transaction ID is
Creating NGO - 1105

Transaction ID is
Checking that the data has been loaded
Query all NGOs


Query specific NGOs - looking for NGO 1103

----------------------------------------
Rating
----------------------------------------
Create Rating

Transaction ID is
```