The Chinese University of Hong Kong

Department of Information Engineering

FTEC5520 –Applied Blockchain & Cryptocurrency

Lab2 Guidance – part1

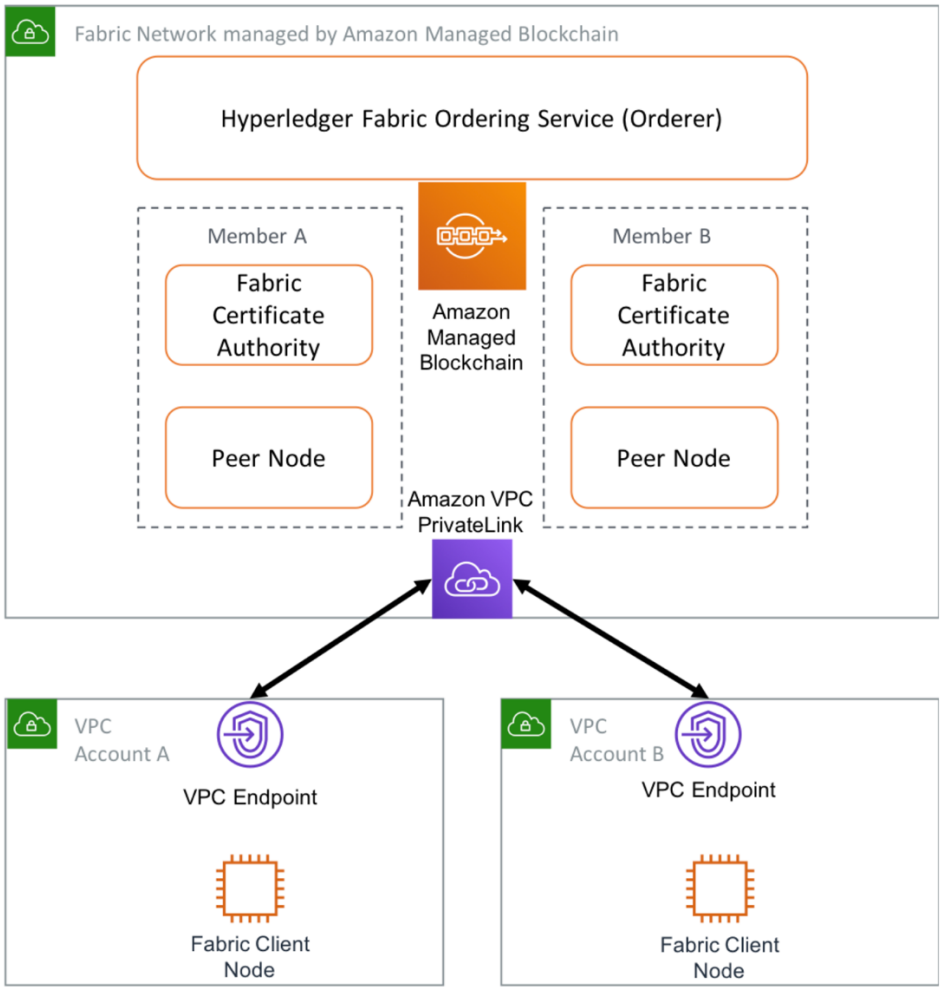
Hyperledger Fabric Setup & Practice on AWS

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 do some basic configurations.
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)



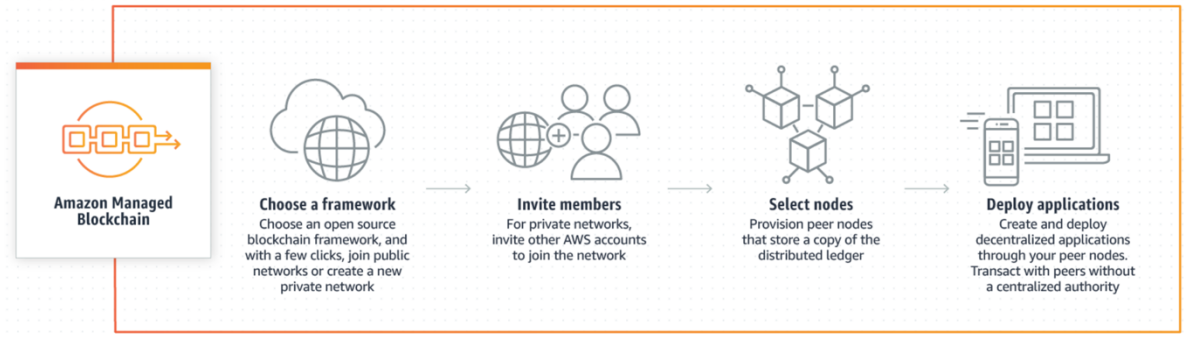
Task0: Preparation Before Starting

1. AWS Educate Account doesn't have the permission to use Amazon Managed Blockchain, so you have to use other account type.
2. Please try register a **personal** free-tier AWS account.

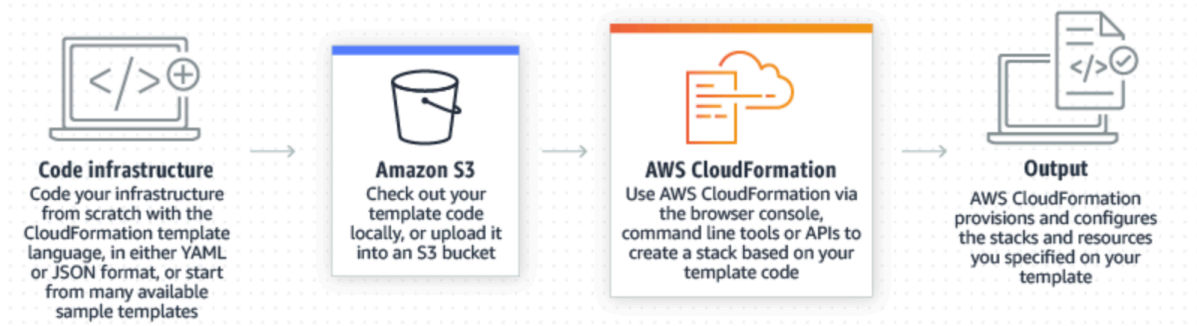
AWS Cloud9: AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

- Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server though SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.
- Using the AWS Cloud9 dashboard, you can create and switch between many different AWS Cloud9 environments, each one containing the custom tools, runtimes, and files for a specific project.

AWS Managed Blockchain: Amazon Managed Blockchain is a fully managed service that makes it easy to join public networks or create and manage scalable private networks using the popular open-source frameworks Hyperledger Fabric and Ethereum.

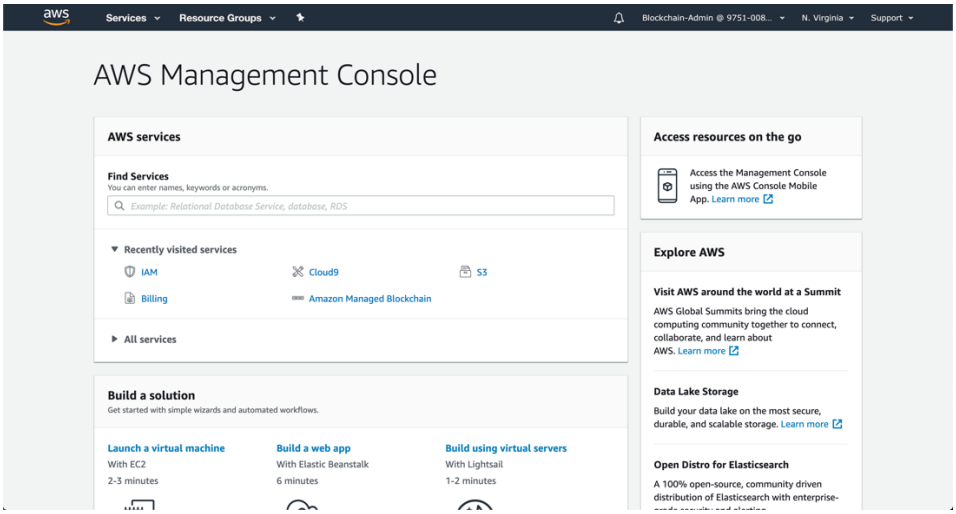


AWS CloudFormation: AWS CloudFormation gives you an easy way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code. A CloudFormation template describes your desired resources and their dependencies so you can launch and configure them together as a stack. You can use a template to create, update, and delete an entire stack as a single unit, as often as you need to, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

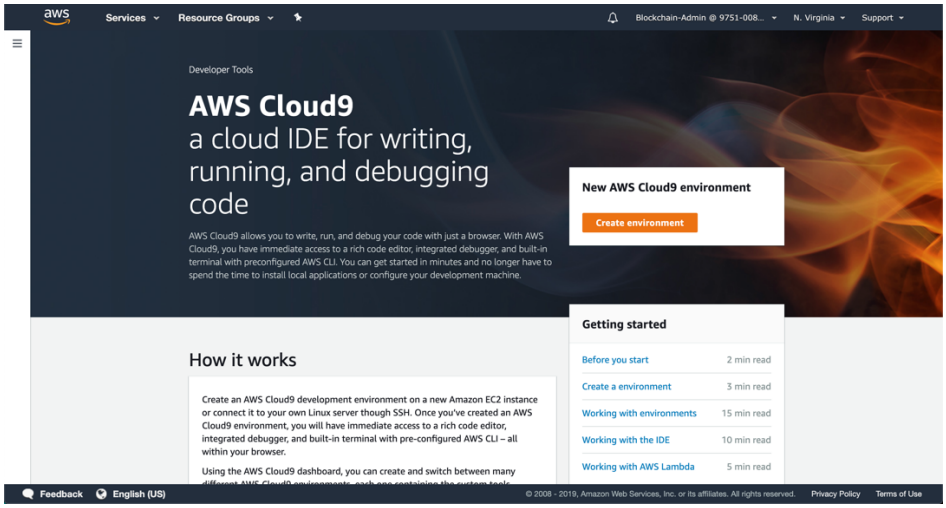


Task1: Setup Environment for Building Hyperledger Fabric on AWS

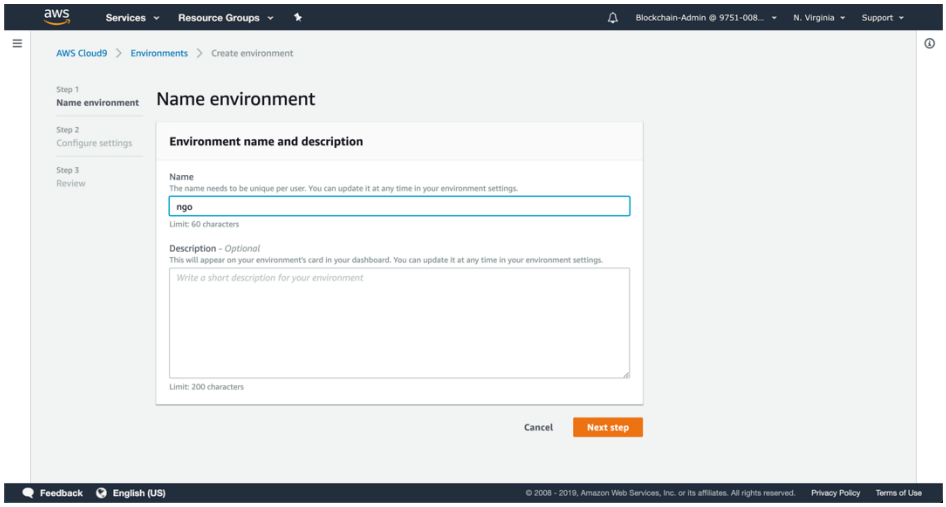
1. Change your region to **N. Virgin** before you go to next step otherwise you may not be able to use Cloud9 correctly.
2. After logging into AWS Management console, type ***Cloud9*** in **Find Services** to start Cloud9 service, the cloud IDE for writing, running and debugging code.



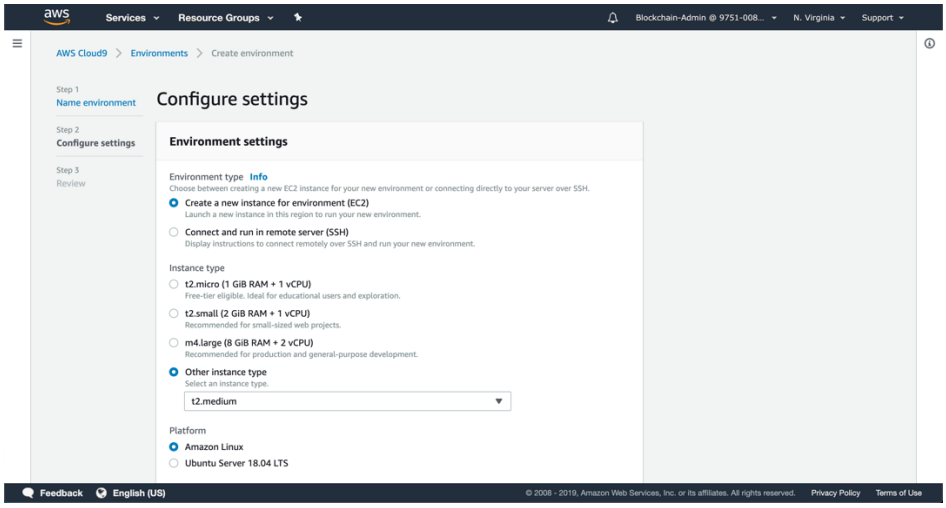
3. Setup AWS Cloud9 environment as the preparation.
   3.1. Click **Create environment** to get started.



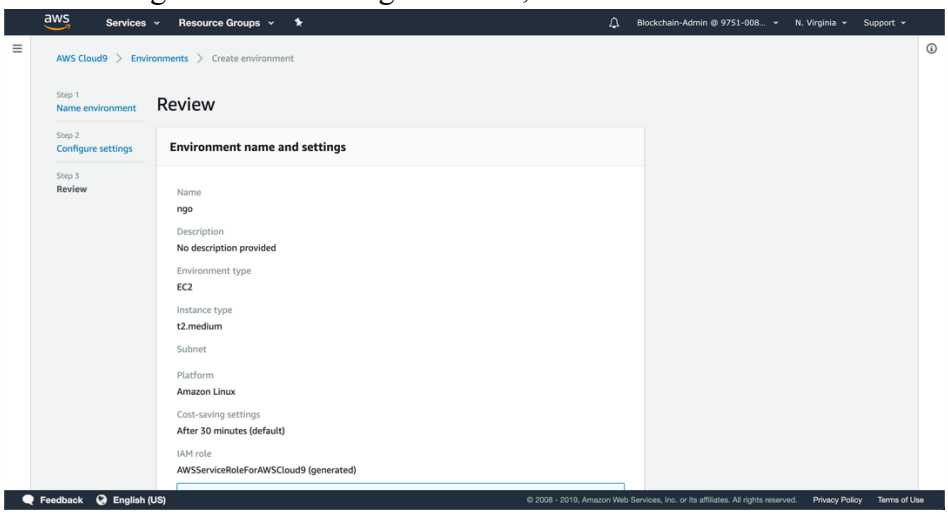
   3.2. Enter *ngo* as the name. Click **Next step**.



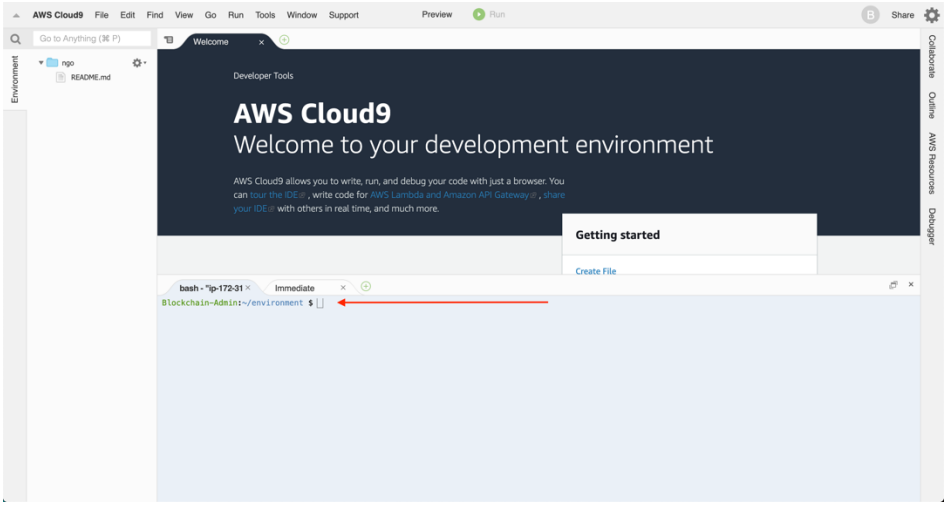
3.3. In the **Configure settings**, **<u>ONLY</u>** change the instance type to **Other instance type** and select **t2.medium**. Click **Next step** then.



3.4. If the setting is same as the figure below, click **Create environment**.



3.5. In the Cloud9 environment, we enter command in the Linux Command Line area.



3.6. Clone non-profit blockchain Git repository and update the AWS CLI to the latest version.

```
cd ~
git clone https://github.com/aws-samples/non-profit-blockchain.git
sudo pip install awscli --upgrade
```

Task 2: Build Hyperledger Fabric Network

There're two ways to launch new Hyperledger Fabric network:

- **CLI:** Cloud9-> **Fabric Client Node**->AWS CLI
- **GUI:** AWS console ->**AWS Manged Blockchain** service portal

2.1 **CLI-based approach**: Cloud9->Fabric Client Node-> AWS CLI

1. Create the Hyperledger Fabric blockchain network

Use Cloud9 to create a Fabric network using the provided CloudFormation template.
In your Cloud9 terminal window:

```
export REGION=us-east-1
export STACKNAME=non-profit-amb
cd ~/non-profit-blockchain/ngo-fabric
./amb.sh
```

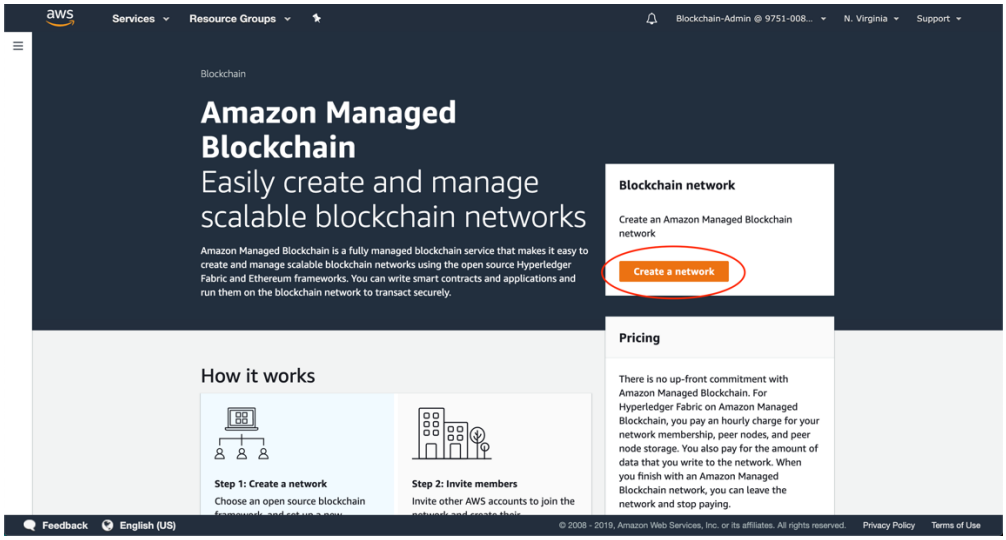
The CloudFormation template, `amb.yaml`, expects a number of parameters. All of these have default values for the purpose of this lab, but can be overridden in the script `amb.sh`. The template defaults to creating a 'Starter' Fabric network with a single small peer node. This would not be suitable for a production network, which would need a '**Standard**' Fabric network with multiple peer nodes spread across AZs.
Check the progress in the AWS CloudFormation console and wait until the stack is **CREATE COMPLETE**. You will find some useful information in the Outputs tab of the CloudFormation stack once the stack is complete. We will use this information in later steps.

2. Check the network is AVAILABLE

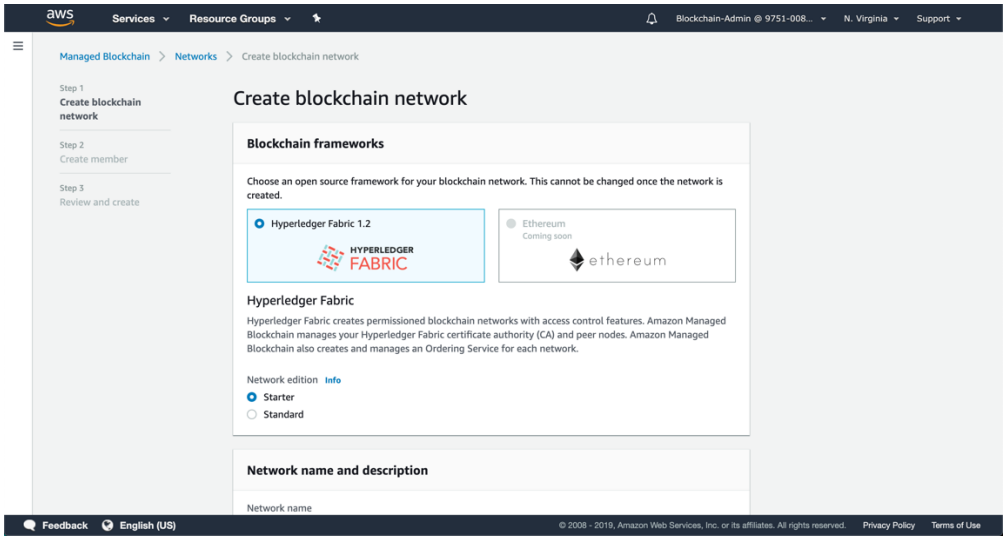
Before continuing, check to see that your Fabric network has been created and is Available. It does take quite a while to create the network, so grab a coffee in the meantime. You will need an Available network before continuining with the next steps.
You can check the status of your network in two places:
In the AWS CloudFormation Console: https://console.aws.amazon.com/cloudformation. Check the resources for your stack. You are waiting for the member and the peer node to be CREATE_COMPLETE.
In the Amazon Managed Blockchain Console: https://console.aws.amazon.com/managedblockchain. For you network, you are waiting for the network, the member and the member's peer node to be Available.
Once your Managed Blockchain network is available, move on to the next step.

2.2 **AWS Managed Blockchain service portal** (optional)

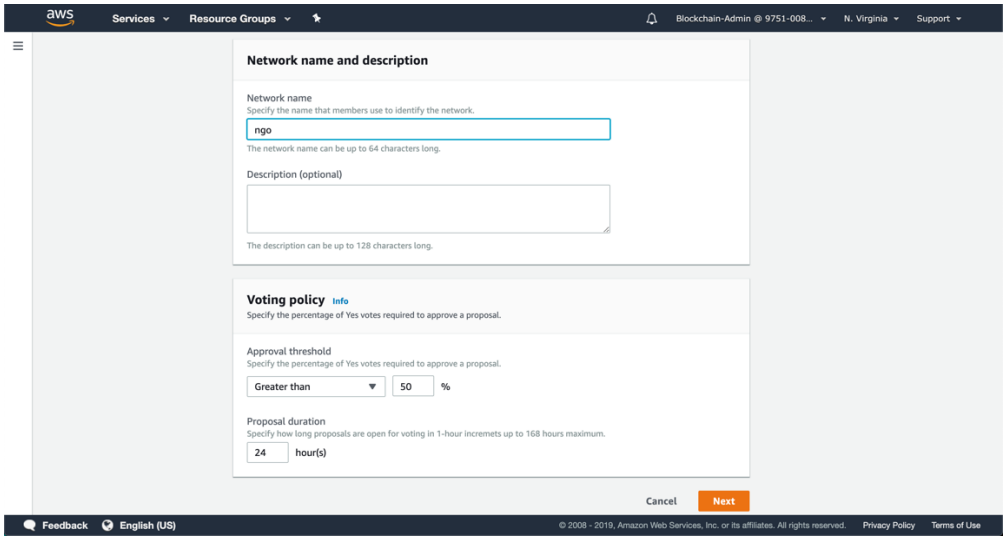
1. Make sure you are in the correct AWS region (**us-east-1**, also known as N. Virginia) before building anything else.
2. Visit https://console.aws.amazon.com/managedblockchain to start next step in Amazon Managed Blockchain Console.
3. Select **Create a network** to start creating a blockchain network.



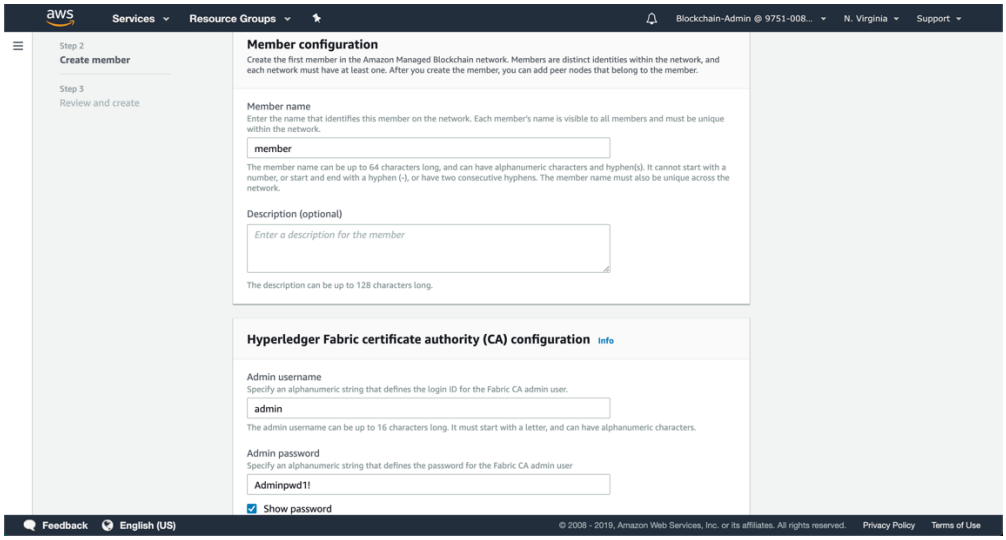
4. In the **Create blockchain network** panel, select **Hyperledger Fabric 1.2** as the blockchain framework and **Starter** as the network edition.



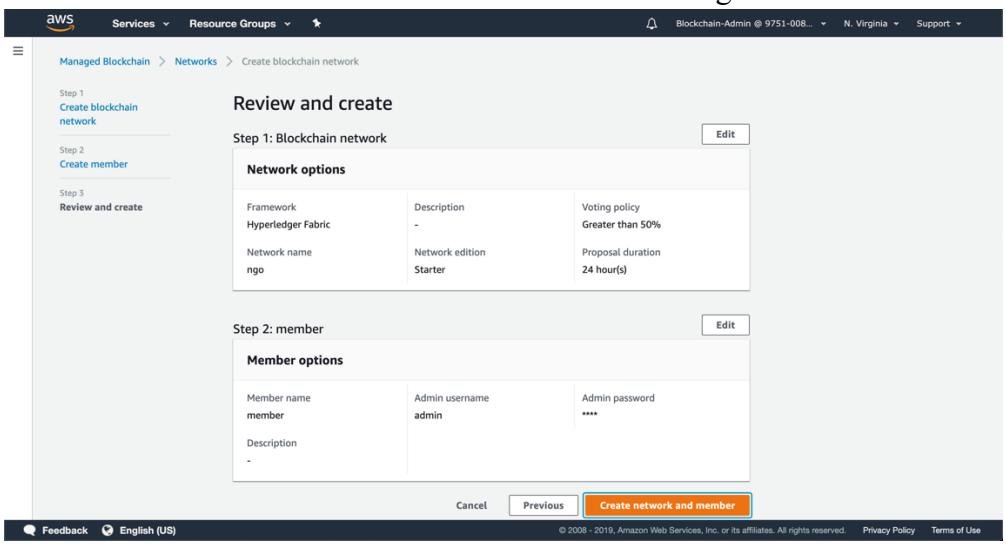
5. Enter *ngo* as the network name and leave the other setting as default. Click **Next** then.



6. In the **Create member** panel, follows the settings. The admin account is crucial in later development. The credentials are supposed to be easily remembered.
   Member configuration:
   Member name: ***member***
   Certificate authority (CA) configuration:
   Admin username: ***admin***
   Admin password: ***Adminpwd1!***



7. Make sure the settings are the same as follows before next step. If the settings are correct, click **Create network and member** to continue creating the blockchain network.



Note: The AWS Amazon Managed Blockchain service charges by the number and used time of nodes, to save money, please only create one member here. The creation of the nodes may take around 15 minutes.

3 Configure Blockchain Network

** Make sure the Blockchain Network just created is having status of Available before continuing to configure the blockchain network. Otherwise, the following steps will fail.
Similar to the creation of AWS Managed Blockchain network, there're two ways to configure the network we created before.

- **CLI:** Cloud9-> **Fabric Client Node**->AWS CLI
- **GUI:** AWS console ->**AWS Manged Blockchain** service portal

3.1 CLI-based operations: Cloud9->Fabric Client Node->AWS CLI

1. Create Fabric Client node

In your Cloud9 terminal window, to create the Fabric client node, which will host the Fabric CLI. You will use the CLI to admnister the Fabric network. The Fabric client node will be created in its own VPC in your AWS account, with VPC endpoints pointing to the Fabric network you created in Step 1 above. AWS CloudFormation will be used to create the Fabric client node, the VPC and the VPC endpoints. Note that the CloudFormation template includes an AMI that is available in us-east-1 only. If you want to run this workshop in a different AWS region, you will need to copy the AMI to your region and replace the AMI ID in the CloudFormation template.
The AWS CloudFormation template requires a number of parameter values. The script you run below will make sure these are available as export variables before calling CloudFormation.
If you see the following error when running the script below: `An error occurred (InvalidKeyPair.NotFound)`, ignore it. This is caused by the script creating a keypair, and ensuring it does not overwrite it if it does exist.
In Cloud9:

```
export REGION=us-east-1
```

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
./vpc-client-node.sh
```

Check the progress in the AWS CloudFormation console and wait until the stack is **CREATE COMPLETE**. You will find some useful information in the Outputs tab of the CloudFormation stack once the stack is complete. We will use this information in later steps.

2. Prepare the Fabric client node and enroll an identity

On the Fabric client node.
Prior to executing any commands on the Fabric client node, you will need to export ENV variables that provide a context to Hyperledger Fabric. These variables will tell the client node which Fabric network to use, which peer node to interact with, which TLS certs to use, etc.
From Cloud9, SSH into the Fabric client node. The key (i.e. the .PEM file) should be in your home directory. The DNS of the Fabric client node EC2 instance can be found in the output of the CloudFormation stack you created in Step 3 above.
Answer 'yes' if prompted: Are you sure you want to continue connecting (yes/no)

```
cd ~
```

```
ssh ec2-user@<dns of EC2 instance> -i ~/<Fabric network name>-keypair.pem
```

Clone the repo:

```
cd ~
```

```
git clone https://github.com/aws-samples/non-profit-blockchain.git
```

In future steps you will need to refer to different configuration values in your Fabric network. In this step we export these values so you don't need to copy them from the console, or look them up elsewhere. Source the file that includes the ENV export values that define your Fabric network configuration so that the exports are applied to your current session. If you exit the SSH session and re-connect, you'll need to source the file again.

```
export REGION=us-east-1
```

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
cp templates/exports-template.sh fabric-exports.sh
```

```
source fabric-exports.sh
```

```
source ~/peer-exports.sh
```

Sourcing the file will do two things:

- export the necessary ENV variables
- create another file which contains the export values you need to use when working with a Fabric peer node. This can be found in the file: `~/peer-exports.sh`, which we also source here.

Check the `source` worked. You should see values for both of the ENV variables below:

```
$ echo $PEERSERVICEENDPOINT
```

```
nd-4MHB4EKFCRF7VBHXZE2ZU4F6GY.m-B7YYBFY4GREBZLPCO2SUS4GP3I.n-WDG36TT
UD5HEJORZUPF4REKMBI.managedblockchain.us-east-1.amazonaws.com:30003
```

```
$ echo $MSP
```

```
m-MKPVAFNPQ5BV7ADVZ4MP2QWA3M
```

For interest, you can check that the peer export file exists and that it contains a number of export keys with values:

```
cat ~/peer-exports.sh
```

Get the latest version of the Managed Blockchain PEM file. This will overwrite the existing file in the home directory with the latest version of this file:

```
aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchain-tls
-chain.pem  /home/ec2-user/managedblockchain-tls-chain.pem
```

Enroll an admin identity with the Fabric CA (certificate authority). We will use this identity to administer the Fabric network and perform tasks such as creating channels and instantiating chaincode.

```
export PATH=$PATH:/home/ec2-user/go/src/github.com/hyperledger/fabri
c-ca/bin
```

```
cd ~
```

```
fabric-ca-client enroll -u https://$ADMINUSER:$ADMINPWD@$CASERVICEEN
DPOINT --tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pe
m -M /home/ec2-user/admin-msp
```

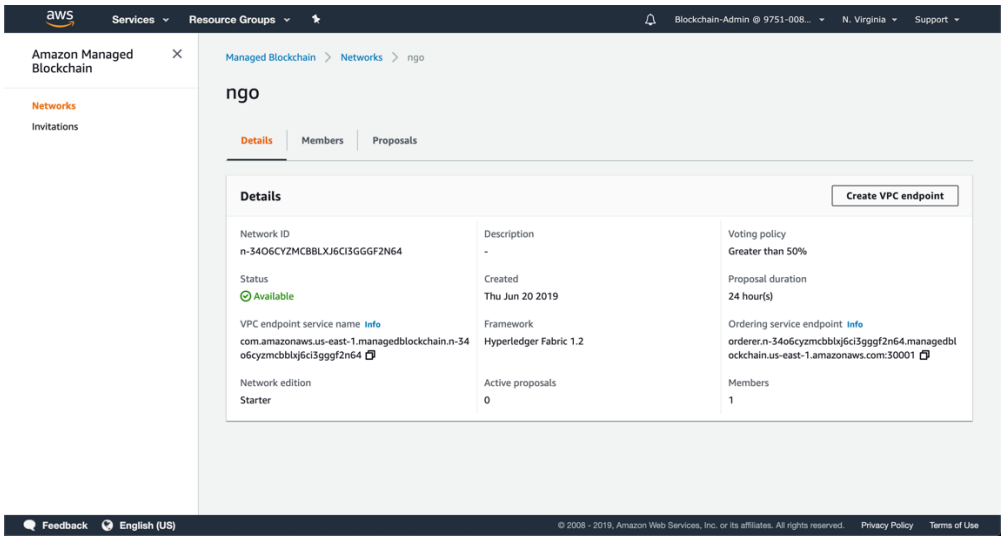
Some final copying of the certificates is necessary:

```
mkdir -p /home/ec2-user/admin-msp/admincerts
```

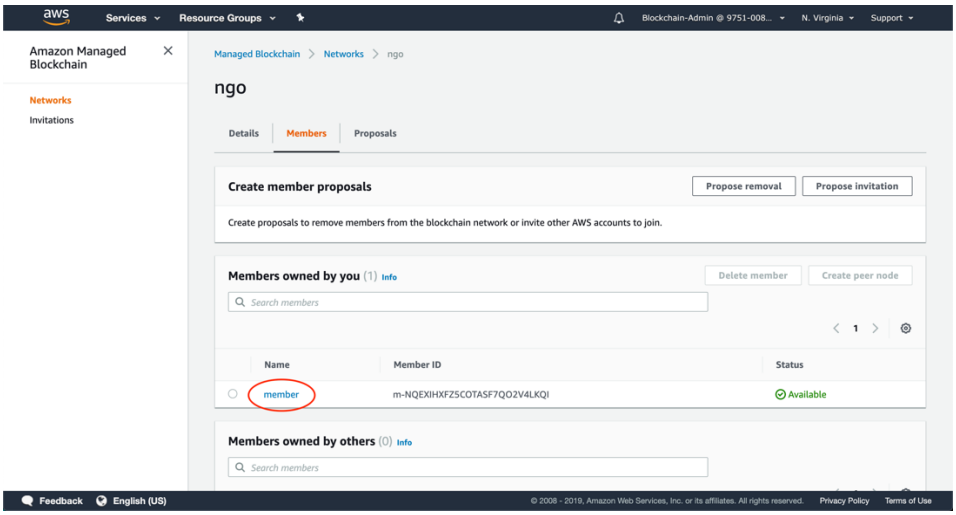
```
cp ~/admin-msp/signcerts/* ~/admin-msp/admincerts/
```

```
cd ~/non-profit-blockchain/ngo-fabric
```

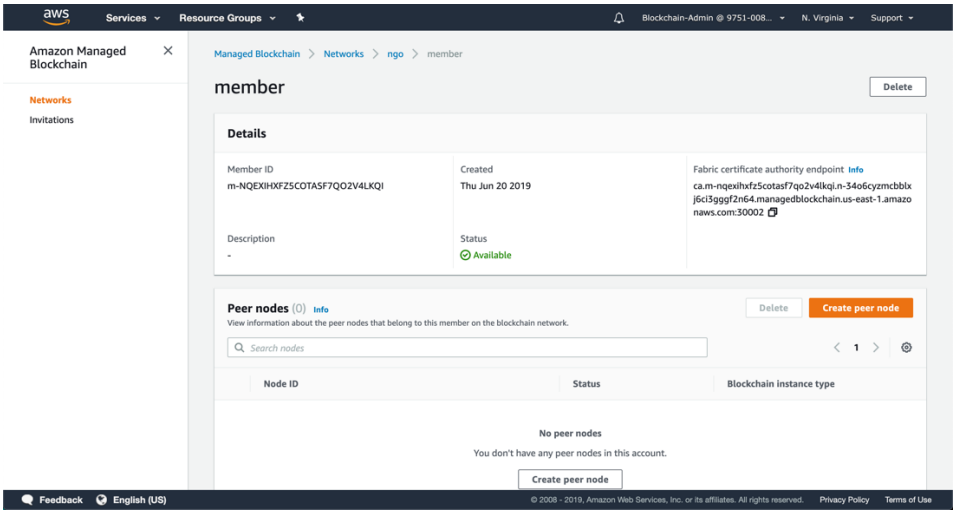
3.2 GUI-based operations: AWS Console-> AWS Managed Blockchain |CloudFormation (optional)



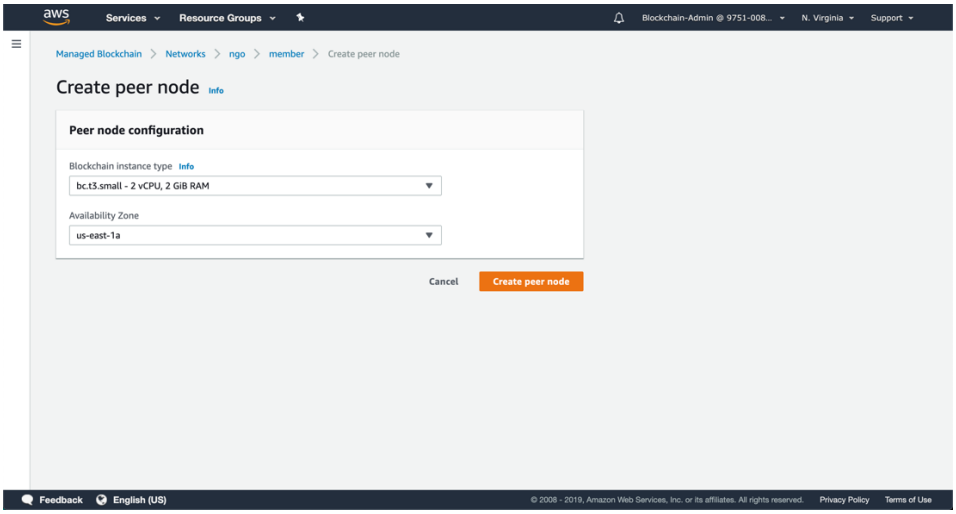
1. Create Fabric peer node.
   1.1. In the new network *ngo* just created, click **Members** to switch to the Members section.
   1.2. In Members section, click **member** to open the details of this newly created member.



   1.3. In the details of this member, click **Create peer node** on the right of the **Peer nodes** panel.



1.4. In **Create Peer Node** panel, select **bc.t3.small** as the Blockchain instance type and **us-east-1a** as the Availability Zone. Select **Create peer node** to create the peer node then.



2. (* No need to wait for the creation of peer node to be completed before proceeding.) Create Fabric client node.

2.1. Go back to Cloud9 IDE.

Type the following commands in the Linux command line.

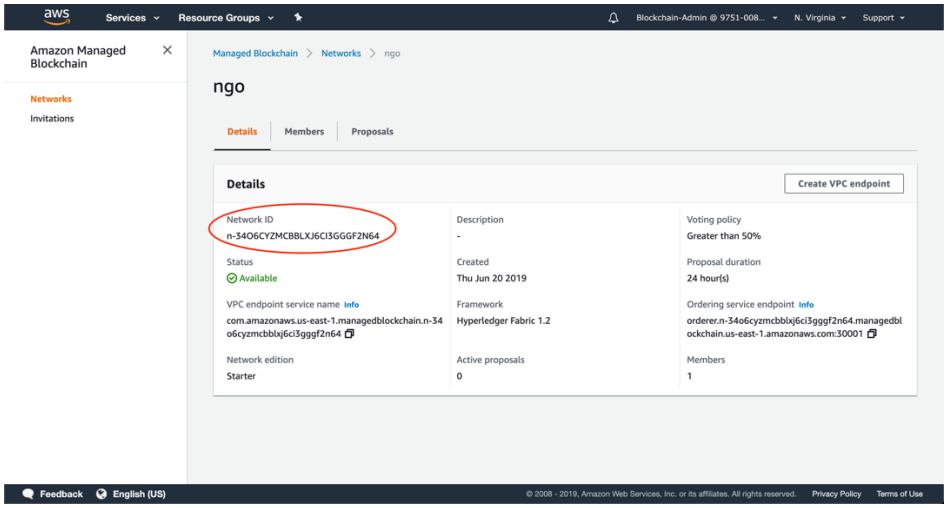
```
export REGION=us-east-1
```

```
export NETWORKID=<the network ID you created in Step1, from the Amaz
on Managed Blockchain Console>
```

```
export NETWORKNAME=ngo
```

```
ec2-user:~ $ export REGION=us-east-1
ec2-user:~ $ export NETWORKID=n-NJP3Q3KYCNGXTLSTY5OJCEGPUI
ec2-user:~ $ export NETWORKNAME=ngo-blockchain
```

** The network ID can be found in the Amazon Managed Blockchain Console, network ***ngo*** Details tab as shown below.



2.2. Set VPC endpoint.

```
export VPCENDPOINTSERVICENAME=$(aws managedblockchain get-network --region $REGION --network-id $NETWORKID --query 'Network.VpcEndpointServiceName' --output text)
```

```
echo $VPCENDPOINTSERVICENAME
```

```
Blockchain-Admin:~/environment $ export VPCENDPOINTSERVICENAME=$(aws managedblockchain get-network --region $REGION --network-id $NETWORKID --query 'Network.VpcEndpointServiceName' --output text)
Blockchain-Admin:~/environment $ echo $VPCENDPOINTSERVICENAME
com.amazonaws.us-east-1.managedblockchain.n-34o6cyzmcbblxj6ci3gggf2n64
Blockchain-Admin:~/environment $
```

2.3. If VPC endpoint is populated with a value, create CloudFoundation stack.

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
ec2-user:~ $ export VPCENDPOINTSERVICENAME=$(aws managedblockchain get-network --region $REGION --network-id $NETWORKID --query 'Network.VpcEndpointServiceName' --output text)
ec2-user:~ $ cd ~/non-profit-blockchain/
ec2-user:~/non-profit-blockchain (master) $ ls
CODE_OF_CONDUCT.md  CONTRIBUTING.md  images  LICENSE  new-member  ngo-chaincode  ngo-fabric  ngo-lambda  ngo-rest-api  ngo-ui  NOTICE  README.md
ec2-user:~/non-profit-blockchain (master) $ cd ngo-fabric/
```

```
./3-vpc-client-node.sh
```

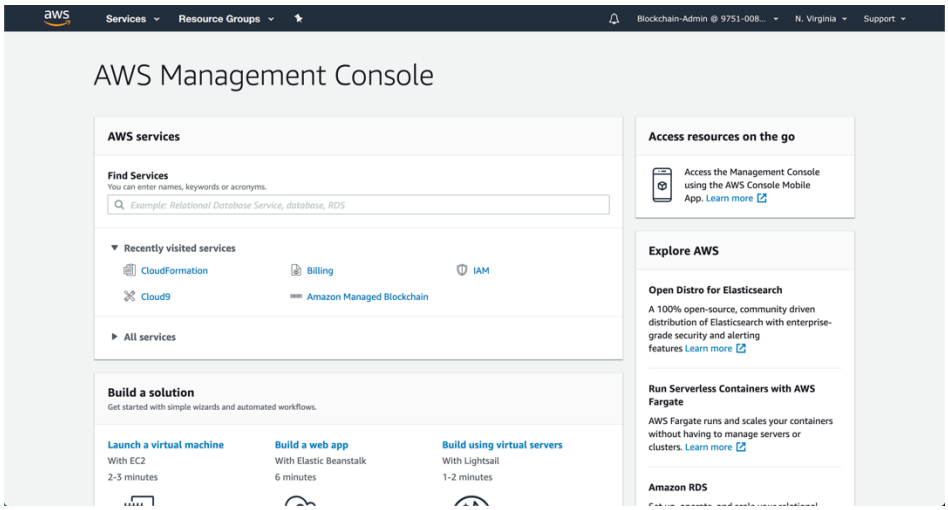
```
ec2-user:~/non-profit-blockchain/ngo-fabric (master) $ ./3-vpc-client-node.sh
Creating VPC - TODO. Create the VPC, subnets, security group, EC2 client node, VPC endpoint
Create a keypair
Searching for existing keypair named ngo-blockchain-keypair

An error occurred (InvalidKeyPair.NotFound) when calling the DescribeKeyPairs operation: The key pair 'ngo-blockchain-keypair' does not exist
Creating a keypair named ngo-blockchain-keypair. The .pem file will be in your /home/ec2-user directory
Create the VPC, the Fabric client node and the VPC endpoints

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - ngo-blockchain-fabric-client-node
ec2-user:~/non-profit-blockchain/ngo-fabric (master) $
```

* If there is an error saying key pair does not exist, this error is expected and can be ignored.

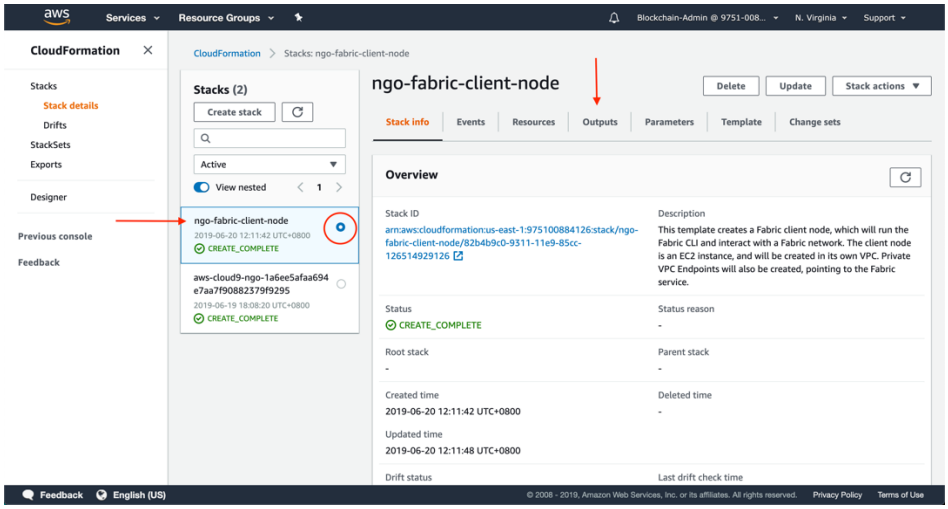
Note: This step will take around 5 minutes to complete.

3. Prepare a fabric client node and enroll an identity.

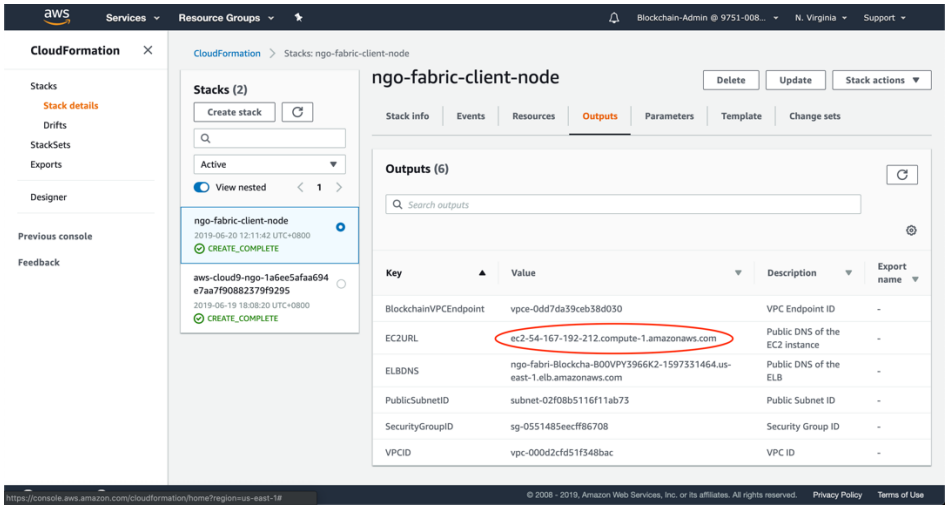
3.1. Switch the browser to AWS Management console. In **the Find Services** dialog, enter ***CloudFormation*** to open CloudFormation service.



3.2. In CloudFormation Stacks, make sure **ngo-fabric-client-node** is selected. On the right showing the details of ngo-fabric-client-node, select **Outputs**.



3.3. Copy the DNS of ngo-fabric-client-node by locating key-value pair with key **EC2URL**. (Everyone should have different DNS)



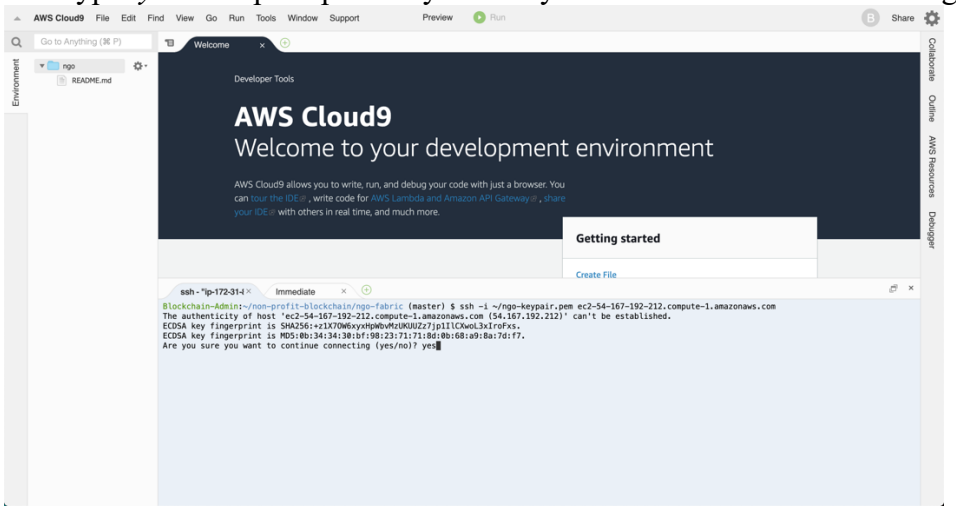
3.4. In the Cloud9 command line, ssh into the client node.

```
ssh -i ~/ngo-keypair.pem <dns of EC2 instance>
```

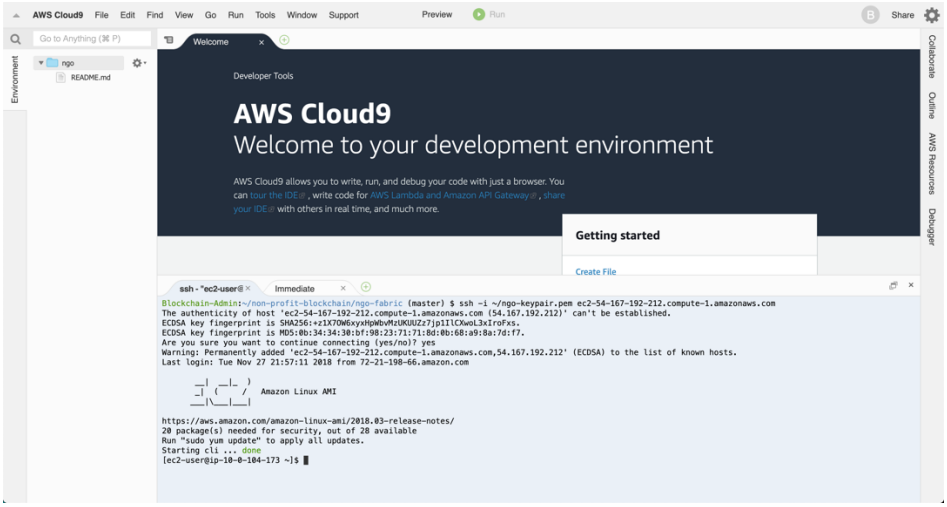
* This dns of EC2 instance is the DNS copied in 3.3.

```
ec2-user:~/non-profit-blockchain/ngo-fabric (master) $ cd
ec2-user:~ $ ls
environment                 node_modules           package-lock.json
ngo-blockchain-keypair.pem  non-profit-blockchain
ec2-user:~ $ ssh -i ~/ngo-blockchain-keypair.pem ec2-3-86-250-38.compute-1.amazonaws.com
The authenticity of host 'ec2-3-86-250-38.compute-1.amazonaws.com (3.86.250.38)' can't be est
ablished.
ECDSA key fingerprint is SHA256:6sKb9K2rwhCJGkxyo0EScOgbEibraOffX1n+ui2Mnrk.
ECDSA key fingerprint is MD5:f1:2a:e5:4b:01:81:8b:32:f1:81:cf:99:9a:96:ac:e7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-86-250-38.compute-1.amazonaws.com,3.86.250.38' (ECDSA) to t
he list of known hosts.
Last login: Tue Nov 27 21:57:11 2018 from 72-21-198-66.amazon.com

       __|  __|_  )
       _|  (     /   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.
Starting cli ... done
[ec2-user@ip-10-0-175-133 ~]$
```

And type '*yes*' if it prompts Are you sure you want to continue connecting (yes/no).



3.5. Then you SSH to the Fabric client node.



3.6. Clone the repository of non-profit blockchain as before.

```
cd ~
```

```
An error occurred (InvalidKeyPair.NotFound) when calling the DescribeKeyPairs operation: The key pair 'ngo-keypair' does not exist
Creating a keypair named ngo-keypair. The .pem file will be in your /home/ec2-user directory
Create the VPC, the Fabric client node and the VPC endpoints

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - ngo-fabric-client-node
ec2-user:~/non-profit-blockchain/ngo-fabric (master) $ ssh -i ~/n
ngo-keypair.pem        node_modules/          non-profit-blockchain/
ec2-user:~/non-profit-blockchain/ngo-fabric (master) $ ssh -i ~/ngo-keypair.pem ec2-54-209-181-61.compute-1.amazonaws.com
The authenticity of host 'ec2-54-209-181-61.compute-1.amazonaws.com (54.209.181.61)' can't be established.
ECDSA key fingerprint is SHA256:j8TLnlGLd08QIPD93pYY1c6mKBfrrhoiLuRCa/Ir34w.
ECDSA key fingerprint is MD5:af:65:c5:08:b6:68:88:79:74:0e:74:a4:23:2e:30:96.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-209-181-61.compute-1.amazonaws.com,54.209.181.61' (ECDSA) to the list of known hosts.
Last login: Tue Nov 27 21:57:11 2018 from 72-21-198-66.amazon.com

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

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
23 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.
Starting cli ... done
[ec2-user@ip-10-0-95-111 ~]$ pwd
/home/ec2-user
[ec2-user@ip-10-0-95-111 ~]$ ls
docker-compose-cli.yaml  fabric-samples  go  go1.10.3.linux-amd64.tar.gz  managedblockchain-tls-chain.pem
ay a menu r@ip-10-0-95-111 ~]$
```

```
git clone https://github.com/aws-samples/non-profit-blockchain.git
```

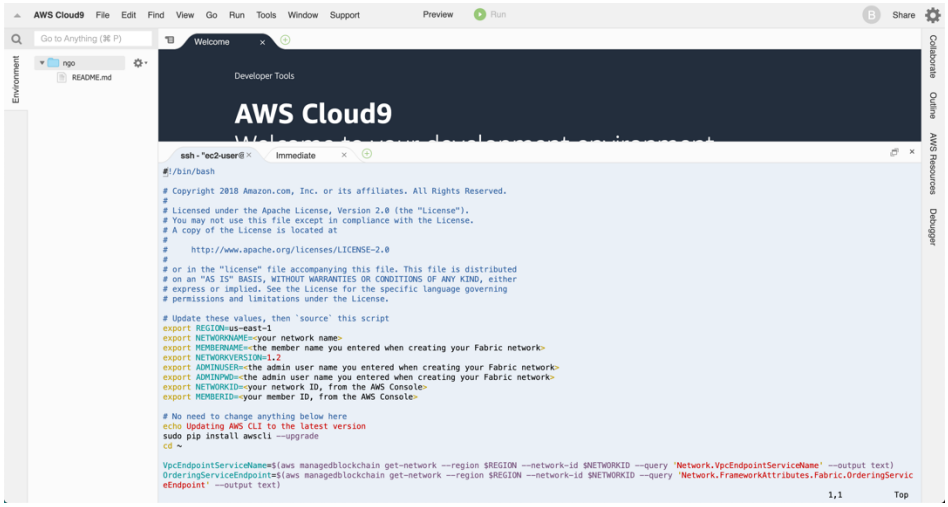
3.7. Create a file to include ENV export values that define your Fabric network configuration.

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
cp templates/exports-template.sh fabric-exports.sh
```

```
vi fabric-exports.sh
```

3.8. At the very beginning of the *fabric-exports.sh*. Before anything change, the code looks like this.



To enter edit mode, type ***i***.

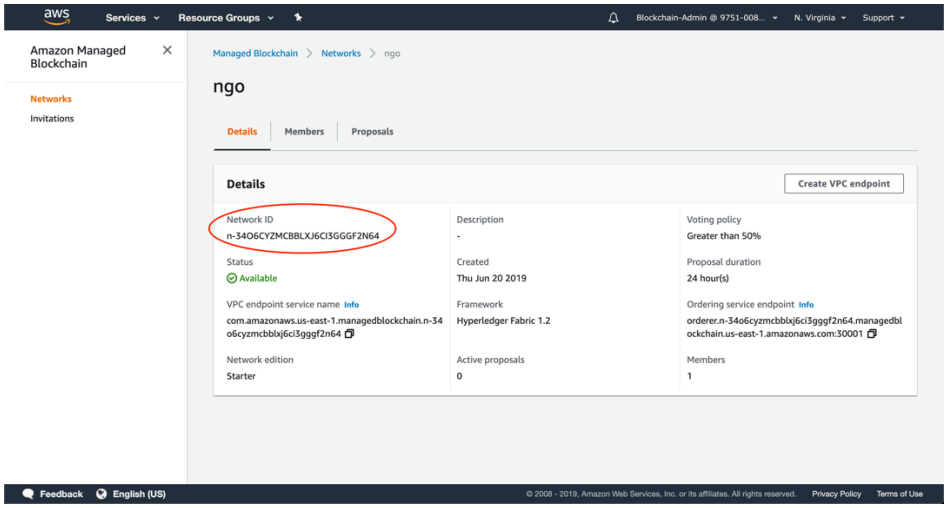
After editing the file, type ***Esc***, then ***:wq*** to save the changes or ***:q!*** to leave without saving.

3.8.1. Change the NETWORKID.

```
export NETWORKID=<your network ID, from the AWS Console>
```

➔ The network ID is same as the one in 2.1. (starting with n-…)

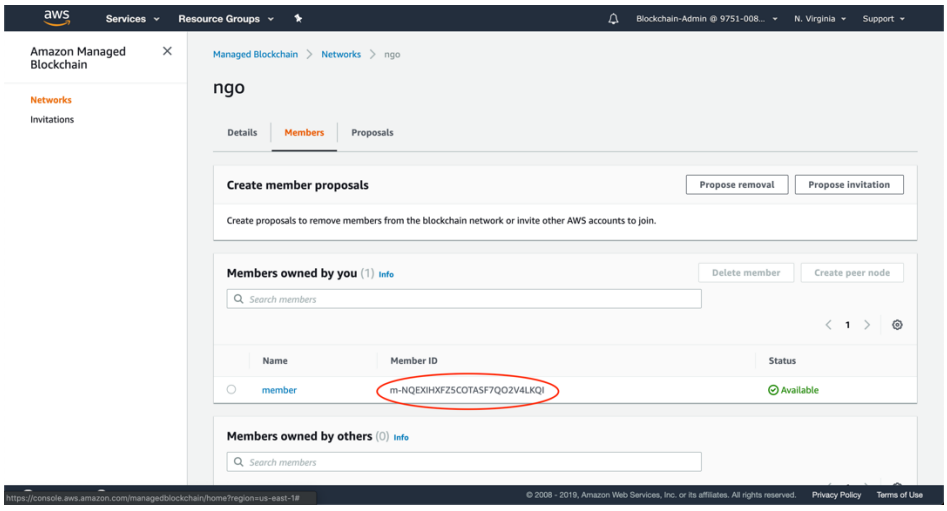
➔ Managed Blockchain Console > ngo > Details



3.8.2. Change the MEMBERID.

```
export MEMBERID =<your member ID, from the AWS Console>
```

➔ Managed Blockchain Console > ngo > Members > member



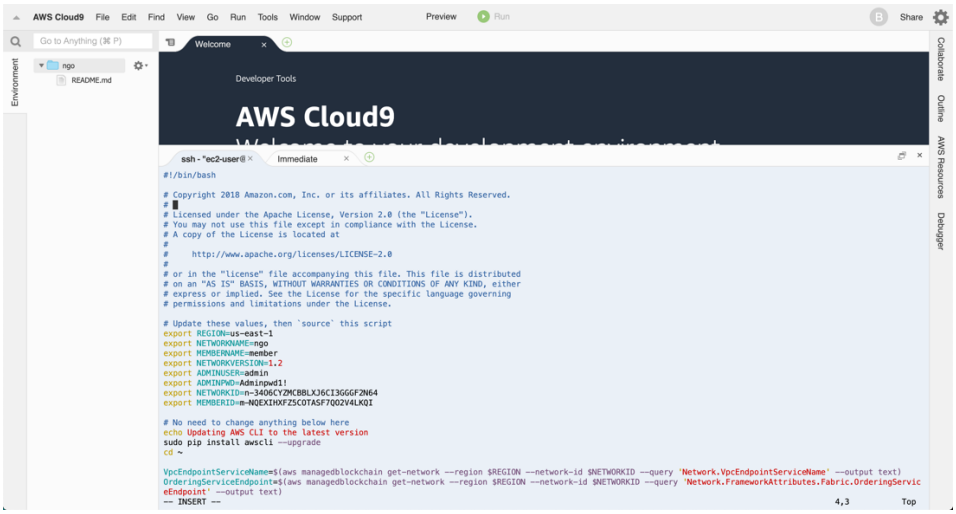
3.8.3. Change the ADMINUSER and ADMINPWD.

```
export ADMINUSER=admin
export ADMINPWD=Adminpwd1!
```

3.8.4. Change the NETWORKNAME and MEMBERNAME.

```
export NETWORKNAME=ngo
export MEMBERNAME=member
```

Final code should look the same as below.



3.9. Source the shell file.

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
source fabric-exports.sh
```

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$ source fabric-exports.sh
Updating AWS CLI to the latest version
Collecting awscli
  Downloading https://files.pythonhosted.org/packages/76/f4/63853217c2065cffaf84e7ca76d861509
ee59c058959473f6b562cdda32f/awscli-1.16.265-py2.py3-none-any.whl (2.4MB)
    100% |████████████████████████████████| 2.4MB 528kB/s
Collecting docutils<0.16,>=0.10 (from awscli)
  Downloading https://files.pythonhosted.org/packages/3a/dc/bf2b15d1fa15a6f7a9e77a61b74ecbbae
7258558fcda8ffc9a6638a6b327/docutils-0.15.2-py2-none-any.whl (548kB)
    100% |████████████████████████████████| 552kB 2.2MB/s
Collecting rsa<=3.5.0,>=3.1.2 (from awscli)
  Downloading https://files.pythonhosted.org/packages/e1/ae/baedc9cb175552e95f3395c43055a6a5e
125ae4d48a1d7a924baca83e92e/rsa-3.4.2-py2.py3-none-any.whl (46kB)
    100% |████████████████████████████████| 51kB 9.4MB/s
Collecting s3transfer<0.3.0,>=0.2.0 (from awscli)
  Downloading https://files.pythonhosted.org/packages/16/8a/1fc3dba0c4923c2a76e1ff0d52b305c44
606da63f718d14d3231e21c51b0/s3transfer-0.2.1-py2.py3-none-any.whl (70kB)
    100% |████████████████████████████████| 71kB 9.4MB/s
```

After running the command, the useful information used in Cloud9 will be printed.
It does two things:

- export the necessary ENV variables
- create another file which contains the export values you need to use when working with a Fabric peer node. This can be found in the file: ~/peer-exports.sh

```
Useful information used in Cloud9
REGION: us-east-1
NETWORKNAME: ngo-blockchain
NETWORKVERSION: 1.2
ADMINUSER: ftec5520
ADMINPWD: FTEC5520password
MEMBERNAME: member-1
NETWORKID: n-NJP3Q3KYCNGXTLSTY5OJCEGPUI
MEMBERID: m-7WWGANWTUFHFTKP5N2NLEMFZ4I
ORDERINGSERVICEENDPOINT: orderer.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30001
ORDERINGSERVICEENDPOINTNOPORT: orderer.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com
VPCENDPOINTSERVICENAME: com.amazonaws.us-east-1.managedblockchain.n-njp3q3kycngxtlsty5ojcegpui
CASERVICEENDPOINT: ca.m-7wwganwtufhftkp5n2nlemfz4i.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30002
PEERNODEID: nd-2TN3QV2M4VH2PCSBN5FSKXFRMI
PEERSERVICEENDPOINT: nd-2tn3qv2m4vh2pcsbn5fskxfrmi.m-7wwganwtufhftkp5n2nlemfz4i.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30003
PEERSERVICEENDPOINTNOPORT: nd-2tn3qv2m4vh2pcsbn5fskxfrmi.m-7wwganwtufhftkp5n2nlemfz4i.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com
PEEREVENTENDPOINT: nd-2tn3qv2m4vh2pcsbn5fskxfrmi.m-7wwganwtufhftkp5n2nlemfz4i.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30004
[ec2-user@ip-10-0-175-133 ~]$
```

** If you exit the SSH session and re-connect back to this client node, you need to repeat this step to source the file again.

3.10. Check if the peer export file exists and all the export keys are with values.

```
cat ~/peer-exports.sh
```

```
[ec2-user@ip-10-0-175-133 ~]$ cat ~/peer-exports.sh
export MSP_PATH=/opt/home/admin-msp
export MSP=m-7WWGANWTUFHFTKP5N2NLEMFZ4I
export ORDERER=orderer.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30001
export PEER=nd-2tn3qv2m4vh2pcsbn5fskxfrmi.m-7wwganwtufhftkp5n2nlemfz4i.n-njp3q3kycngxtlsty5ojcegpui.managedblockchain.us-east-1.amazonaws.com:30003
export CHANNEL=mychannel
export CAFILE=/opt/home/managedblockchain-tls-chain.pem
export CHAINCODENAME=mycc
export CHAINCODEVERSION=v0
export CHAINCODEDIR=github.com/chaincode_example02/go
[ec2-user@ip-10-0-175-133 ~]$
```

If the file has values for all keys, source it:

```
source ~/peer-exports.sh
```

If not, review if there is any typo in doing 3.8.

```
[ec2-user@ip-10-0-175-133 ~]$ source peer-exports.sh
[ec2-user@ip-10-0-175-133 ~]$
```

** If you exit the SSH session and re-connect back to this client node, you need to repeat sourcing peer-exports.sh file again as well.

3.11. Get the latest version of the Managed Blockchain PEM file.

```
aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchain-tls-chain.pem /home/ec2-user/managedblockchain-tls-chain.pem
```

```
[ec2-user@ip-10-0-175-133 ~]$ aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchai
ntls-chain.pem /home/ec2-user/managedblockchain-tls-chain.pem
fatal error: An error occurred (404) when calling the HeadObject operation: Key "etc/managedb
lockchaintls-chain.pem" does not exist
[ec2-user@ip-10-0-175-133 ~]$
[ec2-user@ip-10-0-175-133 ~]$ aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchai
n-tls-chain.pem /home/ec2-user/managedblockchain-tls-chain.pem
download: s3://us-east-1.managedblockchain/etc/managedblockchain-tls-chain.pem to ./managedbl
ockchain-tls-chain.pem
[ec2-user@ip-10-0-175-133 ~]$
```

3.12. Enroll an admin identity with the Fabric CA.

We will use this identity to administer the Fabric network and perform tasks such as creating channels and instantiating chaincode.

```
export PATH=$PATH:/home/ec2-user/go/src/github.com/hyperledger/fabri
c-ca/bin
```

```
[ec2-user@ip-10-0-175-133 bin]$ export PATH=$PATH:/home/ec2-user/go/src/github.com/hyperledge
r/fabric-ca/bin/
[ec2-user@ip-10-0-175-133 bin]$
```

```
cd ~
```

```
fabric-ca-client enroll -u https://$ADMINUSER:$ADMINPWD@$CASERVICEEN
DPOINT --tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pe
m -M /home/ec2-user/admin-msp
```

```
[ec2-user@ip-10-0-175-133 ~]$ fabric-ca-client enroll -u https://$ADMINUSER:$ADMINPWD@$CASERV
ICEENDPOINT --tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem -M /home/ec2-user/
admin-msp
2019/10/24 15:06:12 [INFO] Created a default configuration file at /home/ec2-user/.fabric-ca-
client/fabric-ca-client-config.yaml
2019/10/24 15:06:12 [INFO] TLS Enabled
2019/10/24 15:06:12 [INFO] generating key: &{A:ecdsa S:256}
2019/10/24 15:06:12 [INFO] encoded CSR
2019/10/24 15:06:13 [INFO] Stored client certificate at /home/ec2-user/admin-msp/signcerts/ce
rt.pem
2019/10/24 15:06:13 [INFO] Stored root CA certificate at /home/ec2-user/admin-msp/cacerts/ca-
m-7wwganwtufhftkp5n2nlemfz4i-n-njp3q3kycngxtlsty5ojcegpui-managedblockchain-us-east-1-amazona
ws-com-30002.pem
[ec2-user@ip-10-0-175-133 ~]$
```

3.13. Do final copying of the certificates.

```
mkdir -p /home/ec2-user/admin-msp/admincerts
```

```
cp ~/admin-msp/signcerts/* ~/admin-msp/admincerts/
```

```
cd ~/non-profit-blockchain/ngo-fabric
```

```
[ec2-user@ip-10-0-175-133 ~]$ mkdir -p admin-msp/admincerts
[ec2-user@ip-10-0-175-133 ~]$ cd admin-msp/
[ec2-user@ip-10-0-175-133 admin-msp]$ cd
[ec2-user@ip-10-0-175-133 ~]$ cp ~/admin-msp/signcerts/* ~/admin-msp/admincerts/
[ec2-user@ip-10-0-175-133 ~]$ cd ~/non-profit-blockchain/ngo-
-bash: cd: /home/ec2-user/non-profit-blockchain/ngo-: No such file or directory
[ec2-user@ip-10-0-175-133 ~]$ cd ~/non-profit-blockchain/ngo-fabric/
[ec2-user@ip-10-0-175-133 ngo-fabric]$ ls
3-vpc-client-node.sh  configtx.yaml           fabric-client-node.yaml  README.md
cli                   docker-compose-cli.yaml fabric-exports.sh        templates
[ec2-user@ip-10-0-175-133 ngo-fabric]$
```

4. Update the configtx channel configuration. (On the Fabric client node)

4.1. Check if member ID is in ENV variable.

```
echo $MEMBERID
```

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$ echo $MEMBERID
m-7WWGANWTUFHFTKP5N2NLEMFZ4I
[ec2-user@ip-10-0-175-133 ngo-fabric]$
```

If the member ID is empty, refer to 3.8.2 for retrieving the ID.

4.2. Update the configtx.yaml file. *Make sure you edit the configtx.yaml file you copy to your home directory below, NOT the one in the repo:

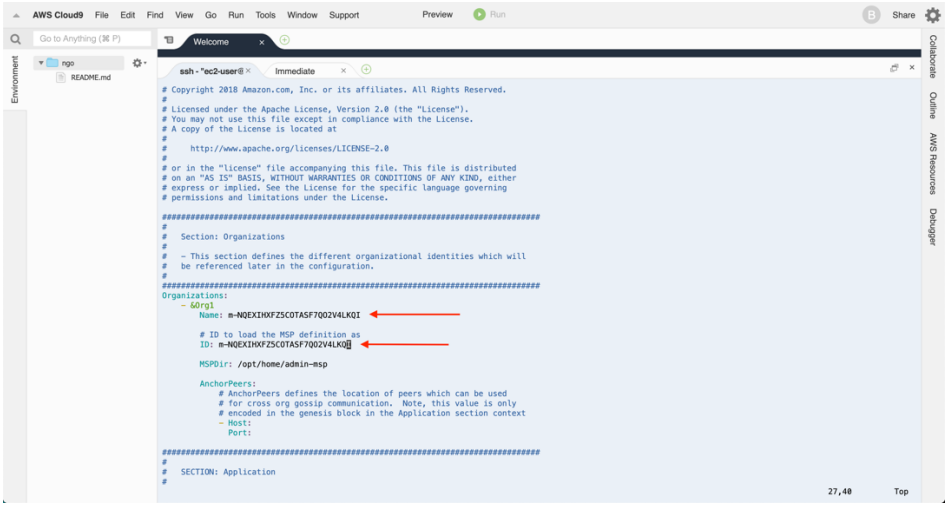
```
cp ~/non-profit-blockchain/ngo-fabric/configtx.yaml  ~
```

(copy the member ID before executing next command)

```
vi ~/configtx.yaml
```

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$  cp ~/non-profit-blockchain/ngo-fabric/configtx.yaml
~
[ec2-user@ip-10-0-175-133 ngo-fabric]$ vim configtx.yaml
```

4.3. In the file, scroll to the Section: Organization. Paste the member ID to name and ID of &Org1. Type *:wq* to save the changes to the file.




```
Organizations:
    - &Org1
        Name: m-7WWGANWTUFHFTKP5N2NLEMFZ4I

        # ID to load the MSP definition as
        ID: m-7WWGANWTUFHFTKP5N2NLEMFZ4I

        MSPDir: /opt/home/admin-msp
```

3.3 Create a Fabric Channel and Deploy the chaincode

Note that before the further steps, you should have finished 3.1 or 3.2.

Generate the **configtx channel** configuration by executing the following script.

```
docker exec cli configtxgen -outputCreateChannelTx /opt/home/$CHANNEL.pb -profile OneOrgChannel -channelID $CHANNEL --configPath /opt/home/
```

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$ docker exec cli configtxgen -outputCreateChannelTx /op
t/home/ $CHANNEL.pb -profile OneOrgChannel -channelID $CHANNEL -configPath /opt/home/
2019-10-24 15:11:36.475 UTC [common/tools/configtxgen] main -> WARN 001 Omitting the channel
ID for configtxgen is deprecated.  Explicitly passing the channel ID will be required in the
future, defaulting to 'testchainid'.
2019-10-24 15:11:36.475 UTC [common/tools/configtxgen] main -> INFO 002 Loading configuration
2019-10-24 15:11:36.832 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 003 Ge
nerating new channel configtx
2019-10-24 15:11:36.833 UTC [common/tools/configtxgen] main -> CRIT 004 Error on outputChanne
lCreateTx: config update generation failure: cannot define a new channel with no Application
section
[ec2-user@ip-10-0-175-133 ngo-fabric]$
```

When the channel is created, this channel configuration will become the genesis block (i.e. **block 0**) on the channel.

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$ docker exec cli configtxgen -outputCreateChannelTx /op
t/home/$CHANNEL.pb -profile OneOrgChannel -channelID $CHANNEL --configPath /opt/home/
2019-10-24 15:16:35.991 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-10-24 15:16:35.993 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Ge
nerating new channel configtx
2019-10-24 15:16:35.993 UTC [common/tools/configtxgen/encoder] NewApplicationGroup -> WARN 00
3 Default policy emission is deprecated, please include policy specificiations for the applic
ation group in configtx.yaml
2019-10-24 15:16:35.994 UTC [common/tools/configtxgen/encoder] NewApplicationOrgGroup -> WARN
 004 Default policy emission is deprecated, please include policy specificiations for the app
lication org group <REPLACE WITH MEMBER_ID> in configtx.yaml
2019-10-24 15:16:35.995 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Wr
iting new channel tx
[ec2-user@ip-10-0-175-133 ngo-fabric]$
```

You should see:

```
2019-06-20 07:57:31.219 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-06-20 07:57:31.225 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2019-06-20 07:57:31.225 UTC [common/tools/configtxgen/encoder] NewApplicationGroup -> WARN 003 Default policy emission is deprecated, please include policy spe
cificiations for the application group in configtx.yaml
2019-06-20 07:57:31.226 UTC [common/tools/configtxgen/encoder] NewApplicationOrgGroup -> WARN 004 Default policy emission is deprecated, please include policy
specificiations for the application org group m-NQEXIHXFZ5COTASF7QO2V4LKQI in configtx.yaml
2019-06-20 07:57:31.227 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Writing new channel tx
```

3.3.Check if the channel configuration has been generated.

```
ls -lt ~/$CHANNEL.pb
```

```
[ec2-user@ip-10-0-104-173 ngo-fabric]$ ls -lt ~/$CHANNEL.pb
-rw-r--r-- 1 root root 327 Jun 20 07:57 /home/ec2-user/mychannel.pb
```

```
[ec2-user@ip-10-0-175-133 ngo-fabric]$ ls -lt ~/$CHANNEL.pb
-rw-r--r-- 1 root root 319 Oct 24 15:16 /home/ec2-user/mychannel.pb
[ec2-user@ip-10-0-175-133 ngo-fabric]$
```

4. Create Fabric channel. (On the Fabric client node)

4.3.Execute the script to create Fabric channel.

```
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 channel create -c $CHANNEL -f /opt/home/$CHAN
NEL.pb -o $ORDERER --cafile $CAFILE --tls --timeout 900s
```

You will see the output is like this.

```
2019-06-20 08:09:02.639 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-06-20 08:09:02.726 UTC [cli/common] readBlock -> INFO 002 Got status: &{NOT_FOUND}
2019-06-20 08:09:02.737 UTC [channelCmd] InitCmdFactory -> INFO 003 Endorser and orderer connections initialized
2019-06-20 08:09:02.939 UTC [cli/common] readBlock -> INFO 004 Got status: &{NOT_FOUND}
2019-06-20 08:09:02.953 UTC [channelCmd] InitCmdFactory -> INFO 005 Endorser and orderer connections initialized
2019-06-20 08:09:03.157 UTC [cli/common] readBlock -> INFO 006 Got status: &{NOT_FOUND}
2019-06-20 08:09:03.166 UTC [channelCmd] InitCmdFactory -> INFO 007 Endorser and orderer connections initialized
2019-06-20 08:09:03.368 UTC [cli/common] readBlock -> INFO 008 Got status: &{NOT_FOUND}
2019-06-20 08:09:03.381 UTC [channelCmd] InitCmdFactory -> INFO 009 Endorser and orderer connections initialized
2019-06-20 08:09:03.583 UTC [cli/common] readBlock -> INFO 00a Got status: &{NOT_FOUND}
2019-06-20 08:09:03.595 UTC [channelCmd] InitCmdFactory -> INFO 00b Endorser and orderer connections initialized
2019-06-20 08:09:03.798 UTC [cli/common] readBlock -> INFO 00c Got status: &{NOT_FOUND}
2019-06-20 08:09:03.808 UTC [channelCmd] InitCmdFactory -> INFO 00d Endorser and orderer connections initialized
2019-06-20 08:09:04.010 UTC [cli/common] readBlock -> INFO 00e Got status: &{NOT_FOUND}
2019-06-20 08:09:04.021 UTC [channelCmd] InitCmdFactory -> INFO 00f Endorser and orderer connections initialized
2019-06-20 08:09:04.223 UTC [cli/common] readBlock -> INFO 010 Got status: &{NOT_FOUND}
2019-06-20 08:09:04.233 UTC [channelCmd] InitCmdFactory -> INFO 011 Endorser and orderer connections initialized
2019-06-20 08:09:04.436 UTC [cli/common] readBlock -> INFO 012 Got status: &{NOT_FOUND}
2019-06-20 08:09:04.451 UTC [channelCmd] InitCmdFactory -> INFO 013 Endorser and orderer connections initialized
2019-06-20 08:09:04.654 UTC [cli/common] readBlock -> INFO 014 Got status: &{NOT_FOUND}
2019-06-20 08:09:04.666 UTC [channelCmd] InitCmdFactory -> INFO 015 Endorser and orderer connections initialized
2019-06-20 08:09:04.870 UTC [cli/common] readBlock -> INFO 016 Got status: &{NOT_FOUND}
2019-06-20 08:09:04.882 UTC [channelCmd] InitCmdFactory -> INFO 017 Endorser and orderer connections initialized
2019-06-20 08:09:05.087 UTC [cli/common] readBlock -> INFO 018 Received block: 0
2019-10-25 11:58:43.627 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Writing new channel tx
[ec2-user@ip-10-0-175-133 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/manag
edblockchain-tls-chain.pem" \
>     -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
>     cli peer channel create -c $CHANNEL -f /opt/home/$CHANNEL.pb -o $ORDERER --cafile $CAFILE --tls --timeout 900s
2019-10-25 11:58:52.584 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-10-25 11:58:52.664 UTC [cli/common] readBlock -> INFO 002 Got status: &{NOT_FOUND}
2019-10-25 11:58:52.674 UTC [channelCmd] InitCmdFactory -> INFO 003 Endorser and orderer connections initialized
2019-10-25 11:58:52.876 UTC [cli/common] readBlock -> INFO 004 Got status: &{NOT_FOUND}
2019-10-25 11:58:52.887 UTC [channelCmd] InitCmdFactory -> INFO 005 Endorser and orderer connections initialized
2019-10-25 11:58:53.093 UTC [cli/common] readBlock -> INFO 006 Got status: &{NOT_FOUND}
2019-10-25 11:58:53.104 UTC [channelCmd] InitCmdFactory -> INFO 007 Endorser and orderer connections initialized
2019-10-25 11:58:53.306 UTC [cli/common] readBlock -> INFO 008 Got status: &{NOT_FOUND}
2019-10-25 11:58:53.317 UTC [channelCmd] InitCmdFactory -> INFO 009 Endorser and orderer connections initialized
2019-10-25 11:58:53.519 UTC [cli/common] readBlock -> INFO 00a Got status: &{NOT_FOUND}
2019-10-25 11:58:53.530 UTC [channelCmd] InitCmdFactory -> INFO 00b Endorser and orderer connections initialized
2019-10-25 11:58:53.733 UTC [cli/common] readBlock -> INFO 00c Got status: &{NOT_FOUND}
2019-10-25 11:58:53.743 UTC [channelCmd] InitCmdFactory -> INFO 00d Endorser and orderer connections initialized
2019-10-25 11:58:53.946 UTC [cli/common] readBlock -> INFO 00e Got status: &{NOT_FOUND}
2019-10-25 11:58:53.957 UTC [channelCmd] InitCmdFactory -> INFO 00f Endorser and orderer connections initialized
2019-10-25 11:58:54.161 UTC [cli/common] readBlock -> INFO 010 Got status: &{NOT_FOUND}
2019-10-25 11:58:54.170 UTC [channelCmd] InitCmdFactory -> INFO 011 Endorser and orderer connections initialized
2019-10-25 11:58:54.372 UTC [cli/common] readBlock -> INFO 012 Got status: &{NOT_FOUND}
2019-10-25 11:58:54.383 UTC [channelCmd] InitCmdFactory -> INFO 013 Endorser and orderer connections initialized
2019-10-25 11:58:54.585 UTC [cli/common] readBlock -> INFO 014 Got status: &{NOT_FOUND}
2019-10-25 11:58:54.594 UTC [channelCmd] InitCmdFactory -> INFO 015 Endorser and orderer connections initialized
2019-10-25 11:58:54.797 UTC [cli/common] readBlock -> INFO 016 Got status: &{NOT_FOUND}
2019-10-25 11:58:54.807 UTC [channelCmd] InitCmdFactory -> INFO 017 Endorser and orderer connections initialized
2019-10-25 11:58:55.013 UTC [cli/common] readBlock -> INFO 018 Received block: 0
[ec2-user@ip-10-0-175-133 ~]$
```

** If the channel creation times out, it's possible that the channel has still been created and you can get the block from the channel itself.
Executing the command below will read the channel config and save the genesis block in the same directory as mentioned above.

```
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_MSPCONFIGP
ATH=$MSP_PATH" cli peer channel fetch oldest /opt/home/fabric-sample
s/chaincode/hyperledger/fabric/peer/$CHANNEL.block -c $CHANNEL -o $O
RDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

4.4.Since this directory is mounted from the host Fabric client node, you can see the block file here.

```
ls -lt /home/ec2-user/fabric-samples/chaincode/hyperledger/fabric/pe
er
```

```
[ec2-user@ip-10-0-175-133 ~]$ ls -lt /home/ec2-user/fabric-samples/chaincode/hyperledger/fabric/peer
total 16
-rw-r--r-- 1 root root 14288 Oct 25 11:58 mychannel.block
[ec2-user@ip-10-0-175-133 ~]$
```

5. Join your peer node to the channel. (On the Fabric client node)

```
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 channel join -b $CHANNEL.block  -o $ORDERER --cafile $CAFILE --tls
```

You should see this:

```
2019-06-20 08:29:16.778 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-06-20 08:29:17.060 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
[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 channel join -b $CHANNEL.block  -o $ORDERER --cafile $CAFILE --tls
2019-10-25 12:01:10.007 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-10-25 12:01:10.440 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
[ec2-user@ip-10-0-175-133 ~]$
```

6. Install **chaincode** on your peer node.

```
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 install -n $CHAINCODENAME -v $CHAINCODEVERSION -p $CHAINCODEDIR
```

You should see this:

```
2019-06-20 08:32:06.263 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-06-20 08:32:06.264 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-06-20 08:32:07.091 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
[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 install -n $CHAINCODENAME -v $CHAINCODEVERSION -p $CHAINCODEDIR
2019-10-25 12:01:54.169 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-10-25 12:01:54.169 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-10-25 12:01:54.949 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
[ec2-user@ip-10-0-175-133 ~]$
```

7. Instantiate the **chaincode** on the channel.
   (The chaincode initializes two accounts:
   Account ‘a’ with balance of 100 and Account b with balance of 200)

```
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 instantiate -o $ORDERER -C $CHANNEL -n $CHAINCODENAME -v $CHAINCODEVERSION -c '{"Args":["init","a","100","b","200"]}' --cafile $CAFILE --tls
```

It takes some time, then you should see this:

```
2019-06-20 08:45:32.656 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-06-20 08:45:32.657 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
```

There is an error pop up

```
[ec2-user@ip-10-0-65-122 ngo-fabric]$ 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 instantiate -o $ORDERER -C $CHANNEL -n $CHAINCODENAME -v $CHAINCODEVERSION \
>     -c '{"Args":["init","a","100","b","200"]}' --cafile $CAFILE -tls
2019-08-13 08:37:01.880 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-08-13 08:37:01.880 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
Error: could not send: EOF
[ec2-user@ip-10-0-65-122 ngo-fabric]$
```

```
[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 instantiate -o $ORDERER -C $CHANNEL -n $CHAINCODENAME -v $CHAINCODEVERSION \
>     -c '{"Args":["init","a","100","b","200"]}' --cafile $CAFILE --tls
2019-10-25 12:02:35.400 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-10-25 12:02:35.400 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
[ec2-user@ip-10-0-175-133 ~]$
```

8. Query the **chaincode** on peer.

```
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 $CHANNEL -n $CHAINCODENAME
 -c '{"Args":["query","a"]}'
```

You should see the command with result:

```
2019-06-20 08:45:32.657 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
[ec2-user@ip-10-0-104-173 ngo-fabric]$ 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 $CHANNEL -n $CHAINCODENAME -c '{"Args":["query","a"]}'
100
```

There is an error pop up

```
[ec2-user@ip-10-0-65-122 ngo-fabric]$ 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 $CHANNEL -n $CHAINCODENAME -c '{"Args":["query","a"]}'
Error: endorsement failure during query. response: status:500 message:"make sure the chaincode mycc has been successfully instantiated and try again: getccdata mychannel/mycc responded with error: could not find chaincode with name 'mycc'"
[ec2-user@ip-10-0-65-122 ngo-fabric]$
```

```
[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 $CHANNEL -n $CHAINCODENAME -c '{"Args":["query","a"]}'
100
[ec2-user@ip-10-0-175-133 ~]$
```

9. Invoke a transaction.

   (The transaction will transfer 10 ‘dollars’ from account ‘a’ to account ‘b’)

```
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 -o $ORDERER -C $CHANNEL -n $
CHAINCODENAME -c '{"Args":["invoke","a","b","10"]}' --cafile $CAFILE
 --tls
```

You will see the command with result:

```
[ec2-user@ip-10-0-104-173 ngo-fabric]$ 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 -o $ORDERER -C $CHANNEL -n $CHAINCODENAME \
>     -c '{"Args":["invoke","a","b","10"]}' --cafile $CAFILE --tls
2019-06-20 08:55:38.940 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
```

There is an error pop up

```
[ec2-user@ip-10-0-65-122 ngo-fabric]$ 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 -o $ORDERER -C $CHANNEL -n $CHAINCODENAME \
>     -c '{"Args":["invoke","a","b","10"]}' --cafile $CAFILE --tls
Error: endorsement failure during invoke. chaincode result: <nil>
[ec2-user@ip-10-0-65-122 ngo-fabric]$
```

```
[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 invoke -o $ORDERER -C $CHANNEL -n $CHAINCODENAME \
>     -c '{"Args":["invoke","a","b","10"]}' --cafile $CAFILE --tls
2019-10-25 12:04:41.048 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
[ec2-user@ip-10-0-175-133 ~]$
```

(Optional – Query the balance of Account a again)

```
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 $CHANNEL -n $CHAINCODENAME
 -c '{"Args":["query","a"]}'
```

Updated result:

```
2019-06-20 08:55:38.940 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
[ec2-user@ip-10-0-104-173 ngo-fabric]$ 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 $CHANNEL -n $CHAINCODENAME -c '{"Args":["query","a"]}'
90
```

```
[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 $CHANNEL -n $CHAINCODENAME -c '{"Args":["query","a"]}'
90
[ec2-user@ip-10-0-175-133 ~]$
```