Prerequisite

1. Operating systems: Ubuntu Linux 14.04 / 16.04/18.04 LTS (both 64-bit), or macOS
2. cURL tool: The latest version
3. git
4. Docker engine: Version 17.06.2-ce or greater
5. Docker-compose: Version 1.14 or greater
6. Go: Version 1.13.x
7. Node: Version 10.21(Node.js version 10 is supported from 10.15.3 and higher)
8. npm: Version 6.14.4
9. Python: 2.7.x

Installation

1. sudo apt-get install curl
2. sudo apt-get install nodejs
3. sudo apt-get install npm
4. sudo apt-get install python
5. Install and Upgrade Docker & Docker Compose
    a. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
    b. sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
    c. sudo apt-get update
    d. apt-cache policy docker-ce
    e. sudo apt-get install -y docker-ce
    f. Docker Compose
        i. sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
        ii. sudo chmod +x /usr/local/bin/docker-compose

iii. sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
iv. docker-compose --version ————> 1.26

g. sudo apt-get upgrade
h. Run hello-world docker image
i. docker run hello-world
ii. If you are getting permission error, follow the below steps
1. sudo groupadd docker
2. sudo usermod -aG docker ${USER}
3. Open new command line or restart same one
4. Docker run hello-world
5. If above image run successfully, we are good to go for next step

6. Go Lang Installation
a. wget https://dl.google.com/go/go1.13.12.linux-amd64.tar.gz
b. tar -xzvf go1.13.12.linux-amd64.tar.gz
c. sudo mv go/ /usr/local
d. Add following into bash rc file
i. export GOPATH=/usr/local/go
ii. export PATH=$PATH:$GOPATH/bin
e. curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
f. sudo apt-get install -y nodejs

1. Verify All Versions
a. curl -V
b. npm -v
c. docker version
d. docker-compose version
e. go version
f. python -V
g. node -v

Install Fabric-Sample, Binaries & Docker Images

1. Latest Images
a. curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.0.0 1.4.7
2. Version Specific
a. curl -sSL https://bit.ly/2ysbOFE | bash -s -- <fabric_version> <fabric-ca_version> <thirdparty_version>
b. **curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.0.1 1.4.6 0.4.18**

Important Note:

Add following binary path to bashrc file

- export PATH=$PATH:/home/ubuntu/fabric-samples/bin

Run test network in fabric sample

a. **./network.sh createChannel -ca -s couchdb**

b. **./network.sh deployCC**

Configure remote ssh extension in Vs code

1. **Install 'remote ssh' extension by Microsoft.**
   To connect - ssh ubuntu@<public IP>

Docker Swarm Network

1. Note down all VM IP Address as below
   a. vm1 - Org1 - 35.193.123.34
   b. vm2 - Org2 - 35.239.12.98
   c. vm3 - Org3 - 130.211.215.60
   d. vm4 - Ord.Org - 34.123.163.5
2. Init docker swam network
   a. VM1
      i. docker swarm init --advertise-addr 35.193.123.34
      ii. docker swarm join-token manager
3. Note down all four command as below and execute on specific VM as per IP address
   a. VM2
      i. docker swarm join --token SWMTKN-1-4xf5nhty82m284zl31×0kcdlhnxtdg9lztebnpdd99t0fg3ka9-1clsa8wtr37iq483aexpvubb4 35.193.123.34:2377 --advertise-addr 35.239.12.98
   b. VM3
      i. docker swarm join --token SWMTKN-1-4xf5nhty82m284zl31×0kcdlhnxtdg9lztebnpdd99t0fg3ka9-1clsa8wtr37iq483aexpvubb4 35.193.123.34:2377 --advertise-addr 130.211.215.60
   c. VM4
      i. docker swarm join --token SWMTKN-1-4xf5nhty82m284zl31×0kcdlhnxtdg9lztebnpdd99t0fg3ka9-1clsa8wtr37iq483aexpvubb4 35.193.123.34:2377 --advertise-addr 34.123.163.5

CLI Execution:

---------- CLI Execution ----------

```
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/channel/crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/channel/crypto-
config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=peer0.org1.example.com:7051
export CHANNEL_NAME="mychannel"
export CC_NAME="fabcar"
export ORDERER_CA=/etc/hyperledger/channel/crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem
export VERSION="1"
```

```
 peer lifecycle chaincode commit -o orderer.example.com:7050 --ordererTLSHostnameOverride
orderer.example.com \
        --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
        --channelID $CHANNEL_NAME --name ${CC_NAME} \
        --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt \
        --peerAddresses peer0.org3.example.com:11051 --tlsRootCertFiles
/etc/hyperledger/channel/crypto-
config/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt \
        --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt \
        --version ${VERSION} --sequence ${VERSION} --init-required
```

```
peer chaincode invoke -o orderer.example.com:7050 \
        --ordererTLSHostnameOverride orderer.example.com \
        --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
        -C $CHANNEL_NAME -n ${CC_NAME} \
        --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt \
        --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt  \
         --peerAddresses peer0.org3.example.com:11051 --tlsRootCertFiles
/etc/hyperledger/channel/crypto-
config/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt  \
        --isInit -c '{"Args":[]}'
```

```
peer chaincode invoke -o orderer.example.com:7050 \
        --ordererTLSHostnameOverride orderer.example.com \
```

```
        --tls \
        --cafile /etc/hyperledger/channel/crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem \
        -C mychannel -n fabcar \
        --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt \
        --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /etc/hyperledger/channel/crypto-
config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt  \
         --peerAddresses peer0.org3.example.com:11051 --tlsRootCertFiles
/etc/hyperledger/channel/crypto-
config/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt  \
        -c '{"function": "createCar","Args":["666666", "Audi", "R8", "Red", "Sandip"]}'
```

```
peer chaincode query -C $CHANNEL_NAME -n ${CC_NAME} -c '{"function": "queryCar","Args":["666666"]}'
```

Flow Diagrams:

Just Click on below URL

```
https://viewer.diagrams.net/?highlight=0000ff&layers=1&nav=1&title=Udemy%3A%20Multihost%20Deployment#Uh
ttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1xRl4FoMvh1scKQCmAUIVZc0YLu-9PBRZ%26export%3Ddownload
```

Postman Collection

```
https://www.getpostman.com/collections/cdcf6c996890a111531d
```