Hyperledger Fabric Implementation Notes:

Steps for Installation of Hyperledger Fabric on ARM:

OS: Raspbian stretch
Prerequisites need to be installed:
sudo apt-get install git curl gcc libc6-dev libltdl3-dev python-setuptools -y

Go programming language:
Sudo apt-get install golang
Go version - go1.7.4

Docker and Docker compose:
Sudo apt-get install docker.io docker-compose

In rpi: use this method mentioned below to install docker as repositories have outdated version:

Install Docker

curl -sSL https://get.docker.com | sh

Install Docker Compose

curl -s https://packagecloud.io/install/repositories/Hypriot/Schatzkiste/script.deb.sh | sudo bash

Install Required Libraries

sudo apt-get install git python-pip python-dev docker-compose build-essential libtool libltdl-dev libssl-dev libevent-dev libffi-dev

Install Python Libraries

sudo pip install --upgrade pip

sudo pip install --upgrade setuptools

sudo pip install behave nose docker-compose

sudo pip install -I flask==0.10.1 python-dateutil==2.2 pytz==2014.3 pyyaml==3.10 couchdb==1.0 flask-cors==2.0.1 requests==2.4.3 pyOpenSSL==16.2.0 pysha3==1.0b1 grpcio==1.0.4

sudo usermod -aG docker $USER #(to run docker without SUDO)

For Hyperledger fabric installation and bootstrapping in rpi3:

Clone repo:

git clone https://github.com/ArvsIndrarys/hyperledger-fabric-on-arm
Or
git clone https://github.com/anirudhkabi/hyperledger-fabric-on-arm.git

Run ./bootstrap.sh: Downloads all the images and sets up the rpi.

Docker swarm has to be created:
On one of the machines or rpis to be part of Blockchain network, initialise swarm

docker swarm init

On the same PC, Generate join-token as manager for others to be added:

docker swarm join-token manager/worker

On all other nodes, use the token generated similar to

"docker swarm join --token
SWMTKN-1-55m8eaiqmqyggiqeniy13iuh5l0682xfs5tl9y4zlubtkxlgi6-3uu0chuj0z1qp20h9eak25hbk 192.168.142.164:2377"

to join to the swarm network.

Now create a docker network: in one of the machines:
docker network create --attachable --driver overlay my-net

Docker swarm created.

==========

Going further with Hyperledger:

On each node

Add path to binaries in “hyperledger-fabric-on-arm” folder in .bashrc
Export PATH=$PATH:<PATH-TO-HL-Fabric-BINARIES>

Download the network config, chaincode, scripts etc from:

```
git clone https://github.com/anirudhkabi/HLF.git
cd HLF
```

NOTE: We generate keys and channel configuration on only one node and copy them to all nodes in blockchain network.

To change network configuration: Change the config in crypto-config.yaml and configtx.yaml.

Setting up experimental setup for 1 organisation (1 CA server),1 orderer, 3 peers, 1 client: The nodes have to be started in the same order in which I have mentioned below:

1. Generate keys and channel info into directories **crypto-config** and **channel-artifacts** respectively by running:
   `./bmhn.sh`
2. Copy directories crypto-config and channel-artifacts to all machines in the current directory from where the containers start.

NOTE: Every time we change network configuration and create new keys and channel info, we need to remove previously existing keys. Running command:

```
rm -rf crypto-config channel-artifacts
```

3. Start all the following components in the directory where you have (or have copied) channel-config and crypto-config directories.

Starting CA server

```
docker run --rm -itd --network="my-net" --name ca.example.com -p
7054:7054 -e FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server -e
FABRIC_CA_SERVER_CA_NAME=ca.example.com -e
FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-c
onfig/ca.org1.example.com-cert.pem -e
FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/<KEY-GENERATED> -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/ca/:/etc
/hyperledger/fabric-ca-server-config -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=hyp-net
hyperledger/fabric-ca sh -c 'fabric-ca-server start -b
admin:adminpw -d'
```

NOTE: <KEY-GENERATED> is in the path crypto-config/peerOrganizations/org1.example.com/ca . Has a Key like **7992f7c0de4e13f63f023b6ef5f615fd8d19300a9dfd2bddf6e5c76635f00d28_sk**

Starting orderer:

```
docker run --rm -it --network="my-net" --name
orderer.example.com -p 7050:7050 -e
ORDERER_GENERAL_LOGLEVEL=debug -e
ORDERER_GENERAL_LISTENADDRESS=0.0.0.0 -e
ORDERER_GENERAL_LISTENPORT=7050 -e
ORDERER_GENERAL_GENESISMETHOD=file -e
ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.gen
esis.block -e ORDERER_GENERAL_LOCALMSPID=OrdererMSP -e
ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp -e
ORDERER_GENERAL_TLS_ENABLED=false -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -v
$(pwd)/channel-artifacts/genesis.block:/var/hyperledger/orderer/
orderer.genesis.block -v
$(pwd)/crypto-config/ordererOrganizations/example.com/orderers/o
rderer.example.com/msp:/var/hyperledger/orderer/msp -w
/opt/gopath/src/github.com/hyperledger/fabric
hyperledger/fabric-orderer orderer
```

Starting Peer0:

```
docker run --rm -it --link
orderer.example.com:orderer.example.com --network="my-net"
--name peer0.org1.example.com -p 8051:7051 -p 8053:7053 -e
CORE_PEER_ADDRESSAUTODETECT=true -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e
CORE_LOGGING_LEVEL=DEBUG -e
CORE_PEER_NETWORKID=peer0.org1.example.com -e CORE_NEXT=true -e
CORE_PEER_ENDORSER_ENABLED=true -e
CORE_PEER_ID=peer0.org1.example.com -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:7050 -e
CORE_PEER_GOSSIP_IGNORESECURITY=true -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051 -e
CORE_PEER_TLS_ENABLED=false -e
CORE_PEER_GOSSIP_USELEADERELECTION=false -e
CORE_PEER_GOSSIP_ORGLEADER=true -e CORE_PEER_LOCALMSPID=Org1MSP
-v /var/run/:/host/var/run/ -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/pe
er0.org1.example.com/msp:/etc/hyperledger/fabric/msp -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
talium/fabric-peer peer node start
```

Starting peer1:

```
docker run --rm -it --network="my-net" --link
orderer.example.com:orderer.example.com --link
peer0.org1.example.com:peer0.org1.example.com --name
peer1.org1.example.com -p 9051:7051 -p 9053:7053 -e
CORE_PEER_ADDRESSAUTODETECT=true -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e
CORE_LOGGING_LEVEL=DEBUG -e
CORE_PEER_NETWORKID=peer1.org1.example.com -e CORE_NEXT=true -e
CORE_PEER_ENDORSER_ENABLED=true -e
CORE_PEER_ID=peer1.org1.example.com -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:7050 -e
CORE_PEER_GOSSIP_ORGLEADER=true -e
```

```
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:7051  -e
CORE_PEER_GOSSIP_IGNORESECURITY=true                           -e
CORE_PEER_LOCALMSPID=Org1MSP                                   -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net                   -e
CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051         -e
CORE_PEER_GOSSIP_USELEADERELECTION=false                       -e
CORE_PEER_TLS_ENABLED=false    -v    /var/run/:/host/var/run/    -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/pe
er1.org1.example.com/msp:/etc/hyperledger/fabric/msp           -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
talium/fabric-peer peer node start
```

Starting peer2:

```
docker     run     --rm     -it     --network="my-net"     --link
orderer.example.com:orderer.example.com                    --link
peer0.org1.example.com:peer0.org1.example.com              --link
peer1.org1.example.com:peer1.org1.example.com              --name
peer2.org1.example.com    -p    9051:7051    -p    9053:7053    -e
CORE_PEER_ADDRESSAUTODETECT=true                               -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock              -e
CORE_LOGGING_LEVEL=DEBUG                                       -e
CORE_PEER_NETWORKID=peer2.org1.example.com  -e  CORE_NEXT=true  -e
CORE_PEER_ENDORSER_ENABLED=true                                -e
CORE_PEER_ID=peer2.org1.example.com                            -e
CORE_PEER_PROFILE_ENABLED=true                                 -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:7050    -e
CORE_PEER_GOSSIP_ORGLEADER=true                                -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2.org1.example.com:7051  -e
CORE_PEER_GOSSIP_IGNORESECURITY=true                           -e
CORE_PEER_LOCALMSPID=Org1MSP                                   -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net                   -e
CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051         -e
CORE_PEER_GOSSIP_USELEADERELECTION=false                       -e
CORE_PEER_TLS_ENABLED=false    -v    /var/run/:/host/var/run/    -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/pe
er2.org1.example.com/msp:/etc/hyperledger/fabric/msp           -w
```

```
/opt/gopath/src/github.com/hyperledger/fabric/peer
talium/fabric-peer peer node start
```

Starting cli:

```
docker run --rm -it --network="my-net" --name cli --link
orderer.example.com:orderer.example.com --link
peer0.org1.example.com:peer0.org1.example.com --link
peer1.org1.example.com:peer1.org1.example.com --link
peer2.org1.example.com:peer2.org1.example.com -p 12051:7051 -p
12053:7053 -e GOPATH=/opt/gopath -e CORE_PEER_LOCALMSPID=Org1MSP
-e CORE_PEER_TLS_ENABLED=false -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e
CORE_LOGGING_LEVEL=DEBUG -e CORE_PEER_ID=cli -e
CORE_PEER_ADDRESS=peer0.org1.example.com:7051 -e
CORE_PEER_NETWORKID=cli -e
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org1.example.com/users/Admin
@org1.example.com/msp -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -v
/var/run/:/host/var/run/ -v
$(pwd)/chaincode/:/opt/gopath/src/github.com/hyperledger/fabric/
examples/chaincode/go -v
$(pwd)/crypto-config:/opt/gopath/src/github.com/hyperledger/fabr
ic/peer/crypto/ -v
$(pwd)/scripts:/opt/gopath/src/github.com/hyperledger/fabric/pee
r/scripts/ -v
$(pwd)/channel-artifacts:/opt/gopath/src/github.com/hyperledger/
fabric/peer/channel-artifacts -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
hyperledger/fabric-tools /bin/bash
```

From the cli go into scripts folder:

a) Setup environment by running command: source setclienv.sh
b) Setup channels by running: ./channel-setup.sh
c) Install chaincode by running ./install-chaincode.sh 1.0. The chaincode is present in chaincode_example02 folder which is mounted on cli

d) Instantiate chaincode by running ./instantiate-chaincode.sh 1.0
e) Details about other scripts is given in **README** in scripts folder

You can choose to install chaincode and instantiate chaincode in only few nodes and not all nodes.