**Hyperledger Fabric Network Setup using ‘Static IPs - extra\_hosts’**

[konda.kalyan@gmail.com](mailto:konda.kalyan@gmail.com)

Github Repo: <https://github.com/konda-kalyan/HLF-Multi-Host-Extra-Hosts>

I have used AWS as reference environment. If you are using other cloud or physical machines, you need to update AWS references accordingly.

Reference sites:

<https://medium.com/@kctheservant/demo-of-three-node-two-channel-setup-in-hyperledger-fabric-54ba8a9c461f> \*\*

<https://medium.com/@kctheservant/another-multi-node-setup-of-a-fabric-network-8d88e9b8d7c5>

**Network topology and different components:**

* Network name is ‘**hlf\_multi\_host\_network**’
* Fabric network with **3 organizations** installed in 3 physical machines (or 3 VMs)
* **Raft** Orderer type based ordering service with 3 Orderers (one per each organization)
* **One Fabric CA** per organization
* One **CLI –** default point toOrg1 – Peer0
* **Couchdb** as world state
* **One Channel** called ‘mychannel’
* **One Chaincode** named ‘couchdb\_java\_simp’ installed in the channel. Chaincode written in ‘java’ language with CouchDB rich queries

#### **STEP 1 - Initial preparations on VMs**

1. **Create 3 VMs**

**Do all steps (mentioned in this section) on all 3 VMs**

**If your deployment environment is AWS, then it is recommended that VM instance type should be minimum ‘t2.small’. Also, make sure that required ports are opened.**

Note: I have tested on Linux - Ubuntu VMs with 16.04 version.

**Caution**: Better set PS1 (in .bashrc) accordingly to make sure that we are doing steps on right machine. It avoids confusion. Example:

*export PS1="[ Org1 - :- \u@\h:\w]$ "*

*sudo apt update*

1. **Install Docker and Docker Compose**

**Do all steps (mentioned in this section) on all 3 VMs**

To install Docker

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>

*sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu* ***xenial*** *stable"*

Based on Ubuntu version (18.04 or 16.04), change bionic or xenial. Currently, I am setting up on 16.04 machine and hence I have used **xenial.**

**Add ‘User’ to the Docker group so that you no need to run docker commands with ‘sudo’**

To install docker-compose

<https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-ubuntu-16-04>

Install docker-compose Command-line completion (Note that command-line completion is available for docker when you install docker)

*sudo curl -L https://raw.githubusercontent.com/docker/compose/1.25.5/contrib/completion/bash/docker-compose -o /etc/bash\_completion.d/docker-compose*

Make sure that installations are succeeded

*docker version*

*docker-compose version*

1. **Make sure Org1 Node can SSH and SCP to other 2 Org’s Nodes**

**Run this step only on Org1 node**

Ref link: <https://www.youtube.com/watch?v=OwptfrdgzDA>

In ‘~/.ssh’ directory, create a file called ‘id\_rsa’ and copy the contents of .pem file (basically private key) which you have downloaded while you launched AWS VM.

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/.ssh$] vim id\_rsa*

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/.ssh$] chmod 600 id\_rsa*

Just check whether you can ssh and scp to Worker nodes

1. **‘AWS specific’ docker daemon configuration**

If you are setting up this environment on AWS, then do below step. Skip this step otherwise.

**Do all steps (mentioned in this section) on all 3 VMs**

**Follow steps explained in this link:** <https://success.docker.com/article/why-do-my-services-stay-pending-when-trying-to-schedule-them-with-placement-contraints>

1. Add labels to the daemon by editing the ‘/etc/docker/daemon.json’ file. Of course, region is the region and zone need to update accordingly (based on VPC region/zone your VMs are running)

**Note: VMs might have created on different regions and zones. Just watch and configure accordingly.**

*sudo vim /etc/docker/daemon.json*

{

"labels": [

"aws.region=us-east-2",

"aws.zone=a"

]

}

1. Then restart the docker daemon

***sudo service docker restart***

#### **STEP 2 - Create Fabric Environment**

1. **Clone the repository**

**Do below step on all 3 VMs**

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/$] git clone* <https://github.com/konda-kalyan/HLF-Multi-Host-Extra-Hosts>

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/$] cd HLF-Multi-Host-Extra-Hosts/network/*

1. **Configure Fabric versions and VM hostnames/IP-addresses**
   * + - 1. **Update Versions and VM’s hostnames in environment file**

In ‘~/HLF-Multi-Host-Extra-Hosts/network/.env’ file, update following variables:

* VM’s ip addresses
* Fabric version number
* AWS region info

*OS\_ARCH=amd64*

*FABRIC\_JUST\_VERSION=1.4.4*

*FABRIC\_VERSION=$OS\_ARCH-1.4.4*

*FABRIC\_CA\_VERSION=$OS\_ARCH-1.4.4*

*COUCHDB\_KAFKA\_ZOOKEEPER\_IMAGE\_VERSION=$OS\_ARCH-0.4.18*

# VERY IMPORTANT NOTE: If deployment environment is AWS then these IP addresses should be hostnames (as mentioned in below, mention ONLY private addresses and that too in the same format.

TO DO: Need to check why public addresses and private addresses in *172.31.13.137* is not working.

*MANAGER\_NODE\_HOSTNAME=ip-172-31-13-137*

*WORKER\_NODE1\_HOSTNAME=ip-172-31-12-5*

*WORKER\_NODE2\_HOSTNAME=ip-172-31-20-131*

*AWS\_REGION=us-east-2*

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  *vim .env*

FROM HERE ONWARDS, everything can be run by SCRIPTS (no manual modification/update of any files). There is a ‘**MASTER\_SCRIPT\_TO\_RUN\_ALL.sh**’ script that does **ALL** tasks from STEP2 (except .env changes that are just right above) to STEP5. In fact, if you just run this one master script, you are DONE. It does, EVERYTHING for you (starting from fabric environment creation to chaincode invocation).

Again, there is one MAIN scripts (**STEP….sh** scripts) at/for each step. Individual step scripts do all tasks mentioned in that whole step.

BUT… I would **strongly** recommend you go one task at time (not even at STEP level), run individual task level script (example: ./scp\_env\_file\_to\_worker\_nodes.sh) and understand what is going and see outputs/work doing by that particular task.

**ALL** (yes, you read it correctly 😉) the scripts need to be run JUST **ONLY** on Org1 node and **not** required to run other two nodes. Those commands which need to be run on Org2 and Org3 nodes are running from Org1 Node itself (by ssh to other 2 nodes in helper scripts).

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*

***./MASTER\_SCRIPT\_TO\_RUN\_ALL.sh***

(it asks for confirmation (yes/no) when connecting to other 2 systems, so say ‘yes’). From second run onwards, it won’t ask for confirmation

**If you see ‘1000’ as a result from Chaincode query, then ALL GOOD.**

-------------------------------------------------------------------------

\*\*\*\* Just run below one script which does rest of the tasks mentioned in step 2. If you want to do/refer step by step, go one task at time so that you understand what is going on

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]* **./*STEP2\_pull\_fabric\_images\_binaries\_on\_all\_machines.sh***

---------------------------------------------------------

* + - * 1. **Move ‘.env’ file to Other 2 Org’s nodes**

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*

*./scp\_env\_file\_to\_other\_orgs\_nodes.sh*

* + - * 1. **Pull fabric images & binaries (on all 3 machines)**

\*\* Below scripts take quite amount of time

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  ./*pull-fabric-images.sh*

1. **Clean the environment (from previous runs) (just safety step)**

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  ./*bring\_down\_whole\_network\_on\_all\_machines.sh*

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  ./*clean\_artifacts\_on\_all\_machines.sh*

#### **STEP 3 - Generate Artifacts (Channel and Crypto (Certs & Keys)) and move them**

\*\*\*\* Just run below one script which does all the tasks mentioned in step 3. If you want to do/refer step by step, go one step at time so that you understand what is going on

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]* **./*STEP3\_generate\_artifacts\_and\_copy\_to\_worker\_nodes.sh***

---------------------------------------------------------

1. **Generate *artifacts (Channel and Crypto (Certs & Keys))***

*[Org1 - ubuntu@ip-172-31-18-197 ~/hlf-multi-node-setup-docker-swarm/network$]*  *./generate\_crypto.sh*

1. **Update VM’s hostnames and fabric image versions in all docker-compose files.**

*[Org1 - ubuntu@ip-172-31-18-197 ~/hlf-multi-node-setup-docker-swarm/network$]*  *./populate\_hostname\_and\_ca\_sk\_files.sh*

1. **Copy all artifact files to one common directory (/var/mynetwork)**

*[Org1 - ubuntu@ip-172-31-18-197 ~/hlf-multi-node-setup-docker-swarm/network$]*   *./copy\_crypto.sh*

1. **Copy artifacts to other nodes as well**

*[Org1 - ubuntu@ip-172-31-18-197 ~/hlf-multi-node-setup-docker-swarm/network$]*   *./scp\_artifacts\_to\_other\_nodes\_and\_copy\_to\_common\_dir.sh*

#### **STEP 4 - Bring up ALL containers**

\*\*\*\* Just run below one script which does all the tasks mentioned in step 4. If you want to do/refer step by step, go one step at time so that you understand what is going on

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]* ***./STEP4\_bringup\_fabric\_network.sh***

---------------------------------------------------------

1. **Bring up the network - Deploy the containers**

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  *./start\_all.sh*

Just in case, you want to start one org containers at a time and see whether containers are getting up or not, then, run one org script at a time. Below is sample for Org1

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  *./scripts/network/deploy\_services\_org1.sh*

Do same for other 2 orgs

Docker container logs gets stored at

*/var/lib/docker/containers/<container id>/<container id>-json.log*

**MAKE SURE THAT NETWORK IS UP AND RUNNING ON ALL 3 NODES. CHECK DOCKER NETWORK, NODES, SERVICES AND CONTAINERS ARE RUNNING AS EXPECTED**

#### **STEP 5 - Create Channel, join Peers and Chaincodes installation, instantiation and invocation**

\*\*\*\* Just run below one script which does all the tasks mentioned in step 5. If you want to do/refer step by step, go one step at time so that you understand what is going on

*[Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]* ***./STEP5\_create\_channel\_then\_do\_chaincode\_operations.sh***

---------------------------------------------------------

* + 1. **Create Channel and get Peers join the Channel**

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  *./scripts/create\_join\_channel.sh*

* + 1. **Install, instantiate and invoke Chaincode**

*[****Org1*** *- ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Extra-Hosts/network$]*  *./scripts/install\_instantiate\_invoke\_chaincode.sh*

**If you see ‘1000’ as a result from Chaincode query, then ALL GOOD.**