**Hyperledger Fabric Multi Node Network Setup on AWS using Docker Swarm**

konda.kalyan@gmail.com

Repo: [**https://github.com/konda-kalyan/HLF-Multi-Host-Docker-Swarm**](https://github.com/konda-kalyan/HLF-Multi-Host-Docker-Swarm)

Reference site: [**https://www.skcript.com/svr/setting-up-a-multi-node-hyperledger-fabric-network-with-docker-swarm/**](https://www.skcript.com/svr/setting-up-a-multi-node-hyperledger-fabric-network-with-docker-swarm/)

**Network topology and different components:**

* Fabric network with **3 organizations** installed in 3 physical machines (or 3 VMs).
* **Kafka** based ordering service with 3 Orderers (one per each organization).
* **One Fabric CA** per organization
* **Couchdb** as world state in each and every peer.
* **One Channel** called ‘mychannel’
* **One Chaincode** named ‘simple’ installed in the channel. Simple Chaincode written in ‘go’ language

Used ‘**Docker Swarm**’ as container orchestration tool.

#### **Initial preparations**

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

1. **Create 3 VMs**

**If deployment environment is AWS, then recommends VM instance type should be minimum ‘t2.medium’. Make sure that required ports are opened.**

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

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

*export PS1="Machine111 - \u@\h \w$ "*

1. Install **Docker** and **Docker Compose**

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

To install docker-compose

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

Make sure that installations are succeeded

*docker version*

*docker-compose version*

#### **Clone the repository**

**Do below step on all 3 VMs**

1. **Clone the repository**

*git clone* [<https://github.com/konda-kalyan/HLF-Multi-Host-Docker-Swarm>*.git*](https://github.com/konda-kalyan/hlf-multi-node-setup-docker-swarm.git)

***cd HLF-Multi-Host-Docker-Swarm/network/***

1. **Clean the environment** (from previous runs)

(if any docker related things are exists/running) (in case, you are setting envi on existing system or doing from second time onwards)

*./network\_down\_and\_clean\_envi.sh*

*[****LeaderNode******Org1*** *- ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./network\_down\_and\_clean\_envi.sh*

*[****WorkerNode Org2*** *- ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./network\_down\_and\_clean\_envi.sh*

*[****WorkerNode Org3*** *- ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./network\_down\_and\_clean\_envi.sh*

#### **Configure Fabric versions and VM hostname/ip-addresses**

**Do below step on all 3 VMs**

* + - 1. **Update Versions and VM’s hostnames in environment file**

Update VM’s ip addresses in ‘~/HLF-Multi-Host-Docker-Swarm/network/.env’ file. Given examples below:

*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. (private addresses in the form of ‘hostname’)). Neither public addresses nor private addresses

*ORG1\_HOSTNAME=ip-172-31-13-137*

*ORG2\_HOSTNAME=ip-172-31-22-238*

*ORG3\_HOSTNAME=ip-172-31-20-131*

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

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *vim .env*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *vim .env*

* + - 1. **Pull required fabric images**

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

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./pull-fabric-images.sh*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./pull-fabric-images.sh*

* + - 1. **Get required fabric binaries**

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  ./*get-fabrc-binaries.sh*

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  ./*get-fabrc-binaries.sh*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  ./*get-fabrc-binaries.sh*

#### **AWS specific docker daemon configuration**

If you are setting up this environment is AWS, then do below extra step

**Follow steps mentioned 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 updated 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***

1. Update/configure AWS region info in ‘HLF-Multi-Host-Docker-Swarm/network/.env’ file based on the region your VMs are running

*AWS\_REGION=us-east-2*

Extra info, above REGION info will get updated in docker compose files when you run populate\_hostname.sh script

#### **Node configuration and moving artifacts (Channel and Crypto (Certs & Keys))**

Run below commands on all VMs

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

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./generate\_crypto.sh*

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./generate\_crypto.sh*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./generate\_crypto.sh*

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

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./populate\_hostname.sh*

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./populate\_hostname.sh*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./populate\_hostname.sh*

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

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *sudo ./copy\_crypto.sh*

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *sudo ./copy\_crypto.sh*

*[WorkerNode Org3 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *sudo ./copy\_crypto.sh*

#### **Create Docker-Swarm Network**

##### **Setup Swarm Network**

* + - 1. **Create a swarm network**

**Run below commands on a VM ONLY** (that you decided to choose as Leader node (Manager node))

*[****LeaderNode Org1*** *- ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$] docker swarm init*

Output from this command is…

Swarm initialized: current node (**ce67vmcbzz7x3bqbgk70lorz4**) is now a manager.

To add a worker to this swarm, run the following command:

**docker swarm join --token SWMTKN-1-2vijhxziu6oxy4n3013oxo7krlngz9mh4gdqqcdha0zxcz3u7y-aglktsqm6g1s92uk0zav04lbu 172.31.18.197:2377**

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions*.*

* + - 1. **Join Worker nodes to Swarm network**

Run below command on **Worker nodes** (basically from other two VMs)

*[WorkerNode Org2 - ubuntu@ip-172-31-22-238 ~/HLF-Multi-Host-Docker-Swarm/network$]* *docker swarm join --token SWMTKN-1-2vijhxziu6oxy4n3013oxo7krlngz9mh4gdqqcdha0zxcz3u7y-aglktsqm6g1s92uk0zav04lbu 172.31.18.197:2377*

*[WorkerNode Org3 - ubuntu@ip-172-31-20-131 ~/HLF-Multi-Host-Docker-Swarm/network$] docker swarm join --token SWMTKN-1-2vijhxziu6oxy4n3013oxo7krlngz9mh4gdqqcdha0zxcz3u7y-aglktsqm6g1s92uk0zav04lbu 172.31.18.197:2377*

Make sure that both nodes are joined Swarm network now. If you run below command on **Leader node** machine, it should show 3 nodes.

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *docker node ls*

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

ce67vmcbzz7x3bqbgk70lorz4 **\*** ip-172-31-18-197 Ready Active **Leader** 19.03.8

j7x9gtge075114a4m1v9d62pp ip-172-31-20-131 Ready Active 19.03.8

cemkwh2rmd6nj2yc8dio8mi09 ip-172-31-22-238 Ready Active 19.03.8

##### **Create Overlay Network**

**Run below commands on Leader Node VM ONLY**

Note that network name is ‘**Skcript**’

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *docker network create --driver overlay --subnet=10.200.1.0/24 --attachable skcript*

Make sure that ‘skcript’ swarm network is created. ‘*docker network ls*’ should show below output.

NETWORK ID NAME DRIVER SCOPE

abei8ujurmtz skcript overlay swarm

#### **Deploy the containers (Bring up the network)**

Run below commands on **Leader Node VM** **ONLY**

*[LeaderNode Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/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

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

Do same for other 2 orgs

Cross verify whether all the services are up and running.

*docker service ls | grep "0/1"*

The above command basically checks if there is any failed containers. If in case you find any failed containers, run the following commands to debug what went wrong.

*docker service ps --no-trunc <service id>*

*or*

*docker inspect <service id>*

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

#### **Create Channel**

Run below commands on **Leader Node VM** **ONLY**

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

#### **Install, instantiate and invoke Chaincode**

Run below commands on **Leader Node VM** **ONLY**

*[****LeaderNode*** *Org1 - ubuntu@ip-172-31-18-197 ~/HLF-Multi-Host-Docker-Swarm/network$]*  *./scripts/install\_chaincodes.sh*