Setting up a multi node Hyperledger Fabric network with Docker Swarm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Created: | Papan Das | | Date: | 31st Jan 2019 | |

Hyperledger Fabric is now getting very popular and the community is coming out with various deployment strategies for the production. One such implementation is the docker swarm method, mainly when you go with multi-node architecture which is likely to be the case for a proper blockchain network. In this article, we’ll be seeing how we can do the docker swarm setup for our Hyperledger Fabric network.

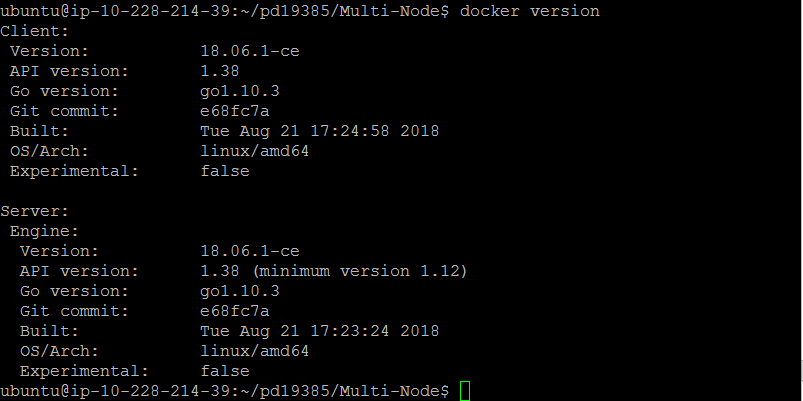
# Prerequisite

Before proceeding into the article, I want to make sure that you are much aware of how hyperledger fabric works and what architecture is behind it to understand this workflow much better and with less confusion.

* 4 AWS Machines/VM with Ubuntu 14.04

1. VM 1: IP 10.228.214.39
2. VM 2: IP 10.228.214.27
3. VM 3: IP 10.228.214.122
4. VM 4: IP 10.228.214.144

* Docker



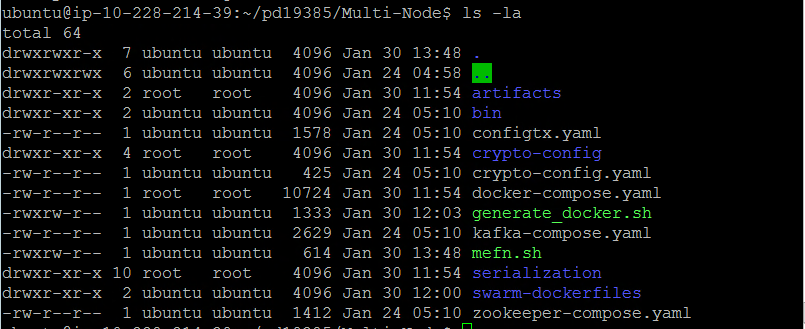
* fabric-ca v1.4
* fabric-tools v1.4
* fabric-ccenv v1.4
* fabric-orderer v1.4
* fabric-peer v1.4
* fabric-zookeeper v1.4
* fabric-kafka v1.4
* fabric-couchdb v1.4
* node version 6.4.1
* nvm version v8.12.0
* go version xgcc (Ubuntu 4.9.3-0ubuntu4) 4.9.3 linux/amd64
* package imported from following Github repository <https://github.com/zs-papandas/serialization>
* Basic understand of Hyperledger fabric and its working

# Port Details

Following are the ports used by the services.

|  |  |
| --- | --- |
| **PORT** | **USER BY** |
| 2181 | Zookeeper |
| 9092 | Kafka |
| 9100 | CouchDB |
| 7050 | Orderer |
| 7051, 7053 | Peer |
| 7054 | CA |

# Directory File Structure

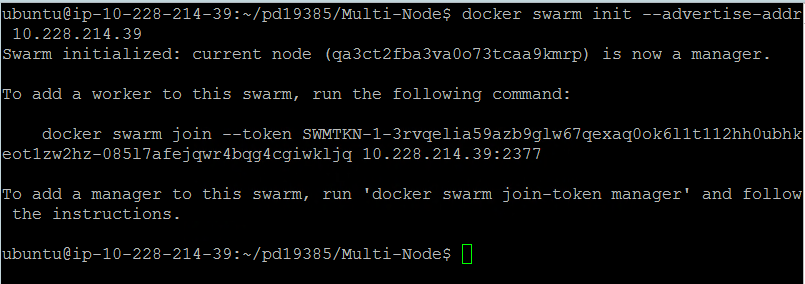


|  |  |  |
| --- | --- | --- |
| **FILE NAME** | **FILE TYPE** | **DESCRIPTION** |
| artifacts | Directory | Contains the Genesis Block |
| bin | Directory | Cryptographic files – Provided by Hyperledger |
| configtx.yaml | YAML | Config |
| crypto-config | Directory | Certificates |
| crypto-config.yaml | YAML | Configuration for Organization |
| docker-compose.yaml | YAML | Docker configuration for Docker Composer |
| generate\_docker.sh | SHELL FILE | To generate Docker Files |
| kafka-compose.yaml | YAML | Kafka Configuration for Docker Composer |
| serialization | DIRECTORY | Chaincode in Golang |
| swarm-dockerfiles | DIRECTORY | Docker file configuration |
| zookeeper-compose.yaml | YAML | Zookeeper configuration for the Docker Composer |
| terminate.sh | SHELL FILE | To end all docker containers running on docker swarm |

# STEP 01 : Connect All the node together using Docker-Swarm

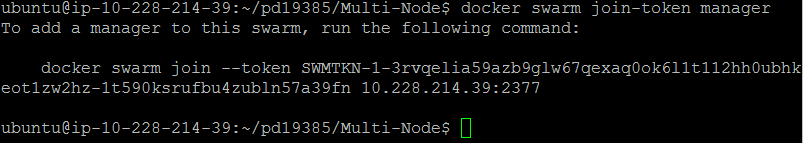
## Initialize docker swarm mode first (docker swarm documentation for more information)

|  |  |
| --- | --- |
| VM 1 | docker swarm init --advertise-addr 10.228.214.39 |



## And make all the other hosts join that swarm as “manager”

|  |  |
| --- | --- |
| VM 1 | docker swarm join-token manager |

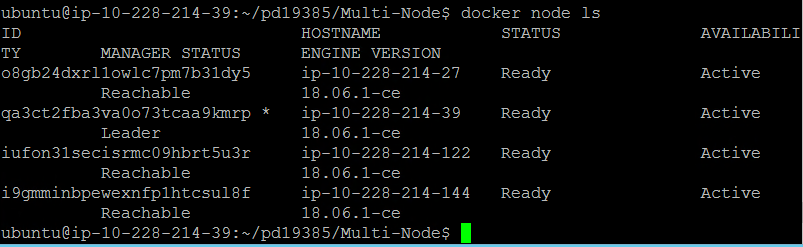


It will output something like this. We will copy it (the one on your terminal, not the one above) and execute it on VM2, VM3, VM4, terminal to make it join VM1

|  |  |
| --- | --- |
| VM 2 |  |
| VM 3 |  |
| VM 4 |  |

NOTE: To check then number of node joined (Works on all nodes)

|  |
| --- |
| docker node ls |



Create a service (“registry” in our case)**— VM1**

|  |
| --- |
| docker service create --name registry --publish 5000:5000 registry:2 |

Generate Docker containers

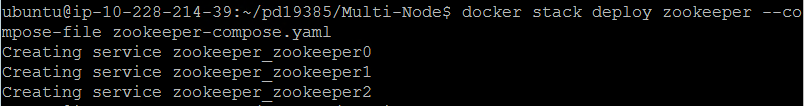
|  |
| --- |
| sudo ./generate\_docker.sh |

Docker network driver creates a distributed network among multiple Docker daemon hosts

|  |
| --- |
| docker network create --driver=overlay --attachable=true fabric-sample-nw |

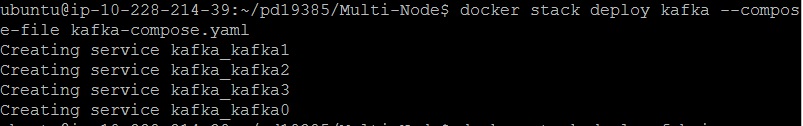
Deploy a new stack of Zookeeper

|  |
| --- |
| docker stack deploy zookeeper --compose-file zookeeper-compose.yaml |



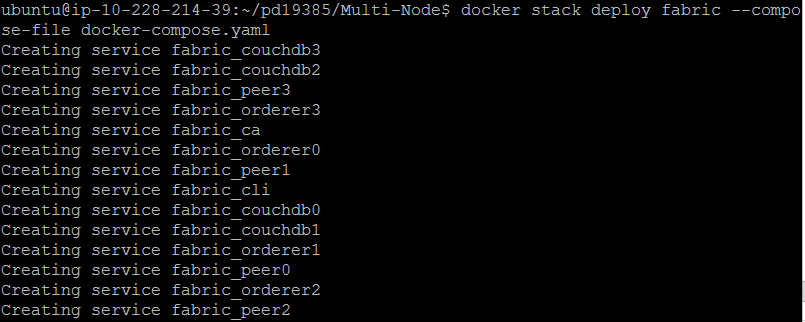
Deploy a new stack of Kafka

|  |
| --- |
| docker stack deploy kafka --compose-file kafka-compose.yaml |



Deploy a new stack of Fabric

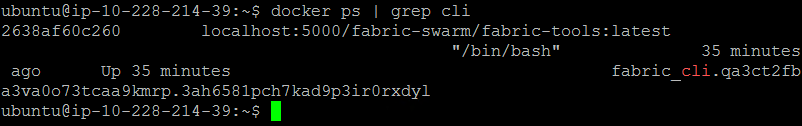
|  |
| --- |
| docker stack deploy fabric --compose-file docker-compose.yaml |



# STEP 03: INSTALL CHAINCODE

NOTE: To get CLI Container ID type. In this case 2638af60c260 is our container ID

|  |
| --- |
| docker ps | grep cli |



Follow the following commands to install the Chaincode

|  |
| --- |
| Create the channel |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp [CLI\_CONTAINER\_NAME] peer channel create -o orderer1:7050 -c fabric-serialization -f /etc/hyperledger/artifacts/channel.tx |

|  |
| --- |
| Copy channel genesis |
| docker exec [CLI\_CONTAINER\_NAME] cp /etc/hyperledger/artifacts/fabric-serialization.block /etc/hyperledger/genesis\_store/ |

|  |
| --- |
| Join **peer0** to the channel |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer0:7051 [CLI\_CONTAINER\_NAME] peer channel join -b /etc/hyperledger/artifacts/fabric-serialization.block |

|  |
| --- |
| Install chaincode to **peer0** |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer0:7051 [CLI\_CONTAINER\_NAME] peer chaincode install -n fabric -p github.com/zs-papandas/serialization -v 0.1 |

|  |
| --- |
| Join **peer1** to the channel |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer1:7051 [CLI\_CONTAINER\_NAME] peer channel join -b /etc/hyperledger/artifacts/fabric-serialization.block |

|  |
| --- |
| Install chaincode to **peer1** |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer1:7051 [CLI\_CONTAINER\_NAME] peer chaincode install -n fabric -p github.com/zs-papandas/serialization -v 0.1 |

|  |
| --- |
| Join **peer2** to the channel |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer2:7051 [CLI\_CONTAINER\_NAME] peer channel join -b /etc/hyperledger/artifacts/fabric-serialization.block |

|  |
| --- |
| Install chaincode to **peer2** |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer2:7051 [CLI\_CONTAINER\_NAME] peer chaincode install -n fabric -p github.com/zs-papandas/serialization -v 0.1 |

|  |
| --- |
| Join **peer3** to the channel |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer3:7051 [CLI\_CONTAINER\_NAME] peer channel join -b /etc/hyperledger/artifacts/fabric-serialization.block |

|  |
| --- |
| Install chaincode to **peer3** |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer3:7051 [CLI\_CONTAINER\_NAME] peer chaincode install -n fabric -p github.com/zs-papandas/serialization -v 0.1 |

|  |
| --- |
| Update anchor of **peer0** |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer0:7051 [CLI\_CONTAINER\_NAME] peer channel update -o orderer1:7050 -c fabric-serialization -f /etc/hyperledger/artifacts/Org1MSPanchors.tx |

|  |
| --- |
| Instantiate chaincode |
| docker exec -e CORE\_PEER\_LOCALMSPID=Org1MSP -e CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE\_PEER\_ADDRESS=peer0:7051 [CLI\_CONTAINER\_NAME] peer chaincode instantiate -o orderer1:7050 -C fabric-serialization -n fabric -v 0.1 -c '{"Args":[""]}' -P 'OR ('\''Org1MSP.member'\'')' |

To ADD a new user as Manufacturer:

|  |  |
| --- | --- |
|  |  |
| VM 1 | docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" -e "CORE\_PEER\_ADDRESS=peer0:7051" [CLI\_CONTAINER\_NAME] peer chaincode invoke -o orderer1:7050 -C fabric-serialization -n fabric -c '{"Args":["createAccount", "a", "Manufacturer","John","1987-11-14","manufacturer@zs.com","9641443962","ZS Associates India Pvt Ltd","manufacturer"]}' |

To GET a user detail:

|  |  |
| --- | --- |
|  | docker ps | grep cli |
| VM 1 | docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" -e "CORE\_PEER\_ADDRESS=peer0:7051" [CLI\_CONTAINER\_NAME] peer chaincode invoke -o orderer0:7050 -C fabric-serialization -n fabric -c '{"Args":["retrieveAccount", "a"]}' |
| VM 2 | docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" -e "CORE\_PEER\_ADDRESS=peer1:7051" [CLI\_CONTAINER\_NAME] peer chaincode invoke -o orderer1:7050 -C fabric-serialization -n fabric -c '{"Args":["retrieveAccount", "a"]}' |
| VM 3 | docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" -e "CORE\_PEER\_ADDRESS=peer2:7051" [CLI\_CONTAINER\_NAME] peer chaincode invoke -o orderer2:7050 -C fabric-serialization -n fabric -c '{"Args":["retrieveAccount", "a"]}' |
| VM 4 | docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" -e "CORE\_PEER\_ADDRESS=peer3:7051" [CLI\_CONTAINER\_NAME] peer chaincode invoke -o orderer3:7050 -C fabric-serialization -n fabric -c '{"Args":["retrieveAccount", "a"]}' |

# EXIT THE Docker Swarm

Run this command to exit

|  |
| --- |
| sudo ./terminate.sh |

To leave Docker Swarm

|  |
| --- |
| docker swarm leave --force |

# Troubleshooting and FAQs

|  |
| --- |
| Does the Chaincode has any logic? |
| Yes, the chaincode has two logics in place.   1. Once when manufacturer request for auto-generate of Pallet 2. Once the ownership changes, the inventory is also been controlled by the chaincode. |