Before you read

1. In lab 1 We have prepared a VMware image settled for attack demos. You can use it for Lab 2 as well.
2. If you are not in Lab room, You only need to install the VMware Workstation Player in your Windows OS computer, download my VM image, and load the VM image.
3. The computers in Lab room 604 A&B have already installed the VMware Workstation Player, and copied my VM image on desktop. If you would like to come to lab room 604 for tutorial, you are not required to install the VMware Player and download the VM image.

   When you boot up computers in PQ604 A&B, you need to choose the **COMP5566** device from the Boot Menu by tapping **2**. In such cases, you will enter the system with environments configured by me.

   *VM Account: **user***   *Passwd : **1234***

4. To simplify the steps and alleviating the need for computer performance, we use Docker containers to maintain the nodes separately

Proof-of-Work

1. Proof-of-Work (PoW) is a consensus mechanism used in blockchain networks where miners solve complex mathematical puzzles to validate transactions and create new blocks. It secures the network, but is energy-intensive.
2. Miners are rewarded with cryptocurrency for successfully adding blocks to the blockchain.
3. We will setup a private Ethereum network and try joining the network as a miner in this lab.

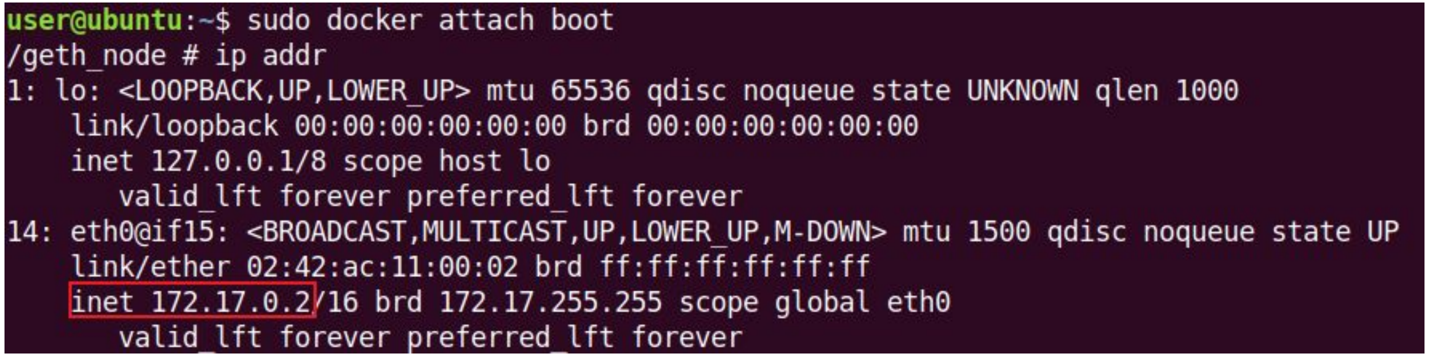
Preparation

1. Please download the attachment from blackboard. (In Lab\Tutorials - Lab2).
2. Copy the attachment to the COMP5566 VM and unzip it.
3. Prepare the bootnode image. Enter the bootnode dir, e.g., `cd ./bootnode`, and build the container, e.g., `docker build -t bootnode .`. Notice that there is a dot at the end of this cmd.
4. Prepare the node image. Change to node dir, e.g., `cd ../node`, and build the container, e.g., `docker build -t node .`.
5. You can check if your images are correctly built using `docker image ls`.

Stage 1: running the blockchain network

We will first launch Geth client in **boot** container, which first runs our Ethereum testnet.

1. Run sudo docker attach boot in terminal, and we will enter into the **boot** container. (A container can be considered as a virtual machine).
2. Output the ip address of the **boot** container by running ip addr in terminal. The ip address will be used to compute the **enode url** of the **boot** node, and the **enode url** is used to connect nodes in Ethereum network.



```
user@ubuntu:~$ sudo docker attach boot
/geth_node # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
14: eth0@if15: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

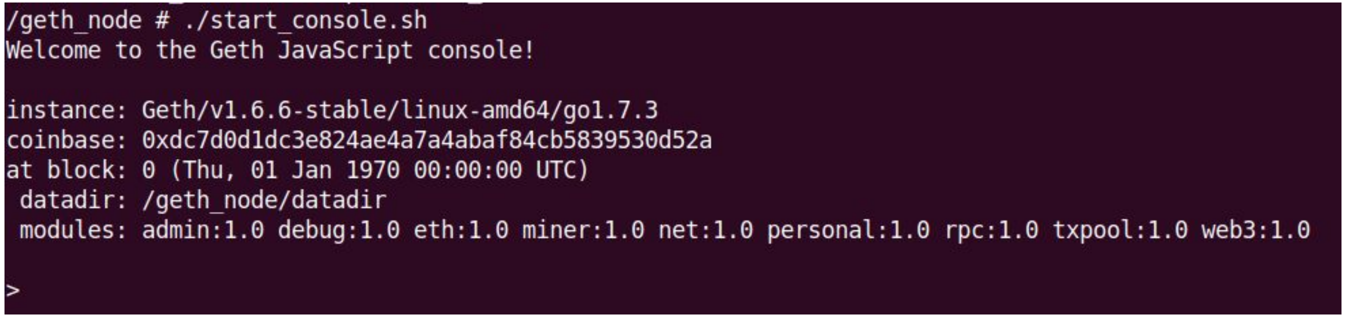
According to the contents displayed in terminal, we can find that the ip address of the **boot** container is **172.17.0.2**

NOTE: now you can start to follow my operations.

Stage 1: running the blockchain network

We will first launch Geth client in **boot** container, which first runs our Ethereum testnet.

3. Run ./start_console.sh in terminal to launch the Geth client in **boot** container (we name it as **boot** node).

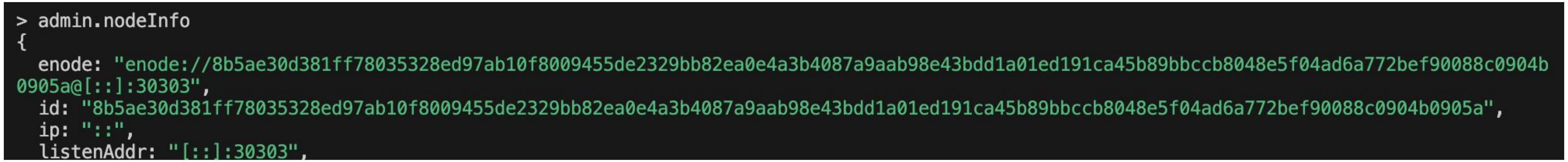


```
/geth_node # ./start_console.sh
Welcome to the Geth JavaScript console!

instance: Geth/v1.6.6-stable/linux-amd64/go1.7.3
coinbase: 0xdc7d0d1dc3e824ae4a7a4abaf84cb5839530d52a
at block: 0 (Thu, 01 Jan 1970 00:00:00 UTC)
 datadir: /geth_node/datadir
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

>
```

Stage 1: running the blockchain network

We will first launch Geth client in **boot** container, which first runs our Ethereum testnet.

4. Run admin.nodeInfo in terminal to check the node information of the **boot** node.



```
> admin.nodeInfo
{
  enode: "enode://8b5ae30d381ff78035328ed97ab10f8009455de2329bb82ea0e4a3b4087a9aab98e43bdd1a01ed191ca45b89bbccb8048e5f04ad6a772bef90088c0904b
0905a@[::]:30303",
  id: "8b5ae30d381ff78035328ed97ab10f8009455de2329bb82ea0e4a3b4087a9aab98e43bdd1a01ed191ca45b89bbccb8048e5f04ad6a772bef90088c0904b0905a",
  ip: "::",
  listenAddr: "[::]:30303",
```

According to the contents displayed in terminal, we can find that the enode information of the **boot** node is: enode://8b5ae30d381ff78035328ed97ab10f8009455de2329bb82ea0e4a3b4087a9aab98e43bdd1a01ed191ca45b89
bbccb8048e5f04ad6a772bef90088c0904b0905a@[::]:30303

5. TAP **Ctrl + P + Q** in terminal to exit the **boot** container with keeping it running.

Stage 1: running the blockchain network

We will first launch Geth client in **boot** container, which first runs our Ethereum testnet.

The format of **enode url** is **enode_id@ip_address:port**. Therefore, we can get the **enode url** of the boot node by replacing the **[::]** to its ip address (**172.17.0.2**) .

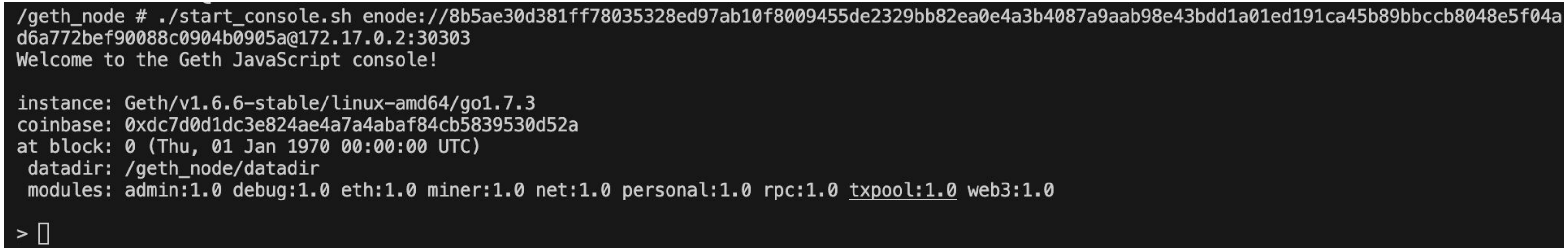
Finally, we get the **enode url** of **boot** node as:
enode://8b5ae30d381ff78035328ed97ab10f8009455de2329bb82ea0e4a3b4087a9aab98e43bdd1a01ed191ca45b89bbccb8048e5f04ad6a772bef90088c0904b0905a@**172.17.0.2**:30303

Besides, you can copy the enode url to a .txt file for easier copy later.

Stage 2: joining the network as miners

We will launch three miner nodes and create accounts for each node. Here we demonstrate the operations in node1. Repeat these steps for node2 and node3.

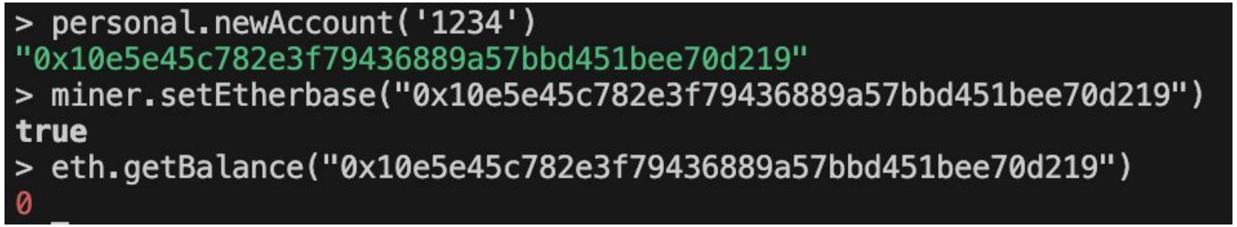
1. Attach node1 container and connect to boot node.



```
/geth_node # ./start_console.sh enode://8b5ae30d381ff78035328ed97ab10f8009455de2329bb82ea0e4a3b4087a9aab98e43bdd1a01ed191ca45b89bbccb8048e5f04a
d6a772bef90088c0904b0905a@172.17.0.2:30303
Welcome to the Geth JavaScript console!

instance: Geth/v1.6.6-stable/linux-amd64/go1.7.3
coinbase: 0xdc7d0d1dc3e824ae4a7a4abaf84cb5839530d52a
at block: 0 (Thu, 01 Jan 1970 00:00:00 UTC)
 datadir: /geth_node/datadir
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> 
```

Stage 2: joining the network as miners

2. Create new account and set etherbase for the node.
- \`personal.newAccount(“1234”)\` This cmd creates an account of which the passphrase is ‘1234’.
- Set etherbase for the node. Etherbase is an account that profits from mining.
- We can check that the newly created account has a balance of 0.



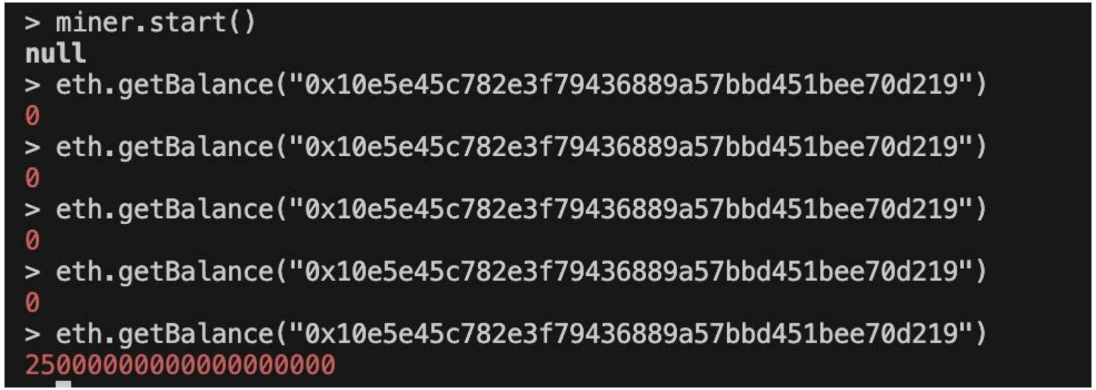
```
> personal.newAccount('1234')
"0x10e5e45c782e3f79436889a57bbd451bee70d219"
> miner.setEtherbase("0x10e5e45c782e3f79436889a57bbd451bee70d219")
true
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
0
```

Stage 2: joining the network as miners

2. Start mining.

- `miner.start()`. This cmd starts the mining process.
- `miner.stop()`. This cmd stops the mining process.

After startup, we can repeatedly check the balance to see if ether are rewarded to us.



```
> miner.start()
null
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
0
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
0
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
0
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
0
> eth.getBalance("0x10e5e45c782e3f79436889a57bbd451bee70d219")
25000000000000000000
```

You can repeat these steps for node2 and node3.

Thanks for Listening