

The Chinese University of Hong Kong
Department of Information Engineering
FTEC5520 –Applied Blockchain & Cryptocurrency
Lab1 Report guidance

Ethereum Infrastructure Setup & Practice -Part2

1 Question Answering:

Answers for each question: there will be 3 questions, but you only need to finish the questions of **task6** and **peering**.

Q3: If you want to deploy the voting smart contract on the blockchain you created in tast1~5, what should you do? If you want to deploy your smart contract on other types of Eth networks like main net, Rinkeby, modern test net, etc, what should you do? (1 mark)

Answer:

deploy on the blockchain created in tast1~5:

1) Install nodejs, ganache-cli, Solidity compiler (solc) and web3

2) Write and compile the smart contract

for example, generate bytecode.

set Ether for gas.

3) Deployment script & Access to an Ethereum node

run your own or via an API, use '**deploy / send / then**' method.

(sending a transaction containing the code of the compiled smart contract)

deploy on other types of Eth networks:

1) install Meta-mask Chrome Extension, then run on the browser

2) Create a wallet at meta-mask

3) Select any one test network, e.g. mainnet/Rinkeby/modern test net, etc

4) Add some dummy Ethers in your wallet

5) Use editor remix to write the smart contract in Solidity

6) Create a .sol extension file

7) Deploy contract by pressing the deploy button in Remix window

Q4: Multi-Node Setup of a Private Ethereum on AWS with Contract Deployment. Please list the main steps.

- 1) How to setup a multi-node (**more than two nodes**) Ethereum blockchain network?

Reference: <https://blockgeeks.com/two-node-setup-of-a-private-ethereum/>

Answer:

multi-node (more than two nodes), say, N nodes here for illustration.

Step 1: Launch N EC2 instances.

all can be "t2.medium, 2 vCPU, 4 GB RAM, 8G SSD" configuration

all nodes are with the same security group

all allows TCP 30303 by default

Step 2: Install geth client on Node 1

Access to Node 1 with ssh & key

Step 3: Prepare the Genesis.json & Init the geth with Genesis.json

ensure all nodes share the same genesis block

Step 4: Launch geth now for Node 1

two screens in parallel: one for console, another for log

Step 5: Create an Account on Node 1 & Start Mining

personal.newAccount()

miner.start()

Step 6: Repeat Step 2-6 on other Nodes (N-1 remaining nodes)

the same Genesis.json is used when init on other Nodes (N-1 remaining nodes)

Step 7: Peering - peer all the nodes

"admin.peers" for every Node

"admin.nodeInfo.enode" Obtain enode information from a Node

Add this information to other Nodes (N-1 remaining nodes)

then all nodes are peering each other.

Step 8: Send Ethers Between Accounts

to verify if the setup is successful.

Do not forget Closing to save money.

- 2) How to deploy the sample Voting smart contract on it?

Reference: <https://blockgeeks.com/two-node-setup-of-a-private-ethereum-on-aws-with-contract-deployment-part-2/>

Answer:

multi-node (more than two nodes), say, N nodes here for illustration.

deploy the contract on Node 1

see how other Nodes (N-1 remaining nodes) accesses this contract.

Step 1: Install node.js and web3 on all N nodes.

command lines: "node -v", "npm -v"

to check whether the installation is successful.

check on all nodes.

Step 2: Access Blockchain through web3 in Node Console

all nodes:

use split screen: geth client & node console

"node", "admin.startRPC()" ...

command line: "web3.eth.accounts" to verify

Step 3: Deploy Contract on Node 1

set "abi, byteCode"

requires gas & a list of candidates.

execute the functions for verification.

"totalVoteFor", "voteForCandidate"

Step 4: Access Contract from other Nodes (N-1 remaining nodes)

obtain deployed address from Node 1.

for ABI, define VotingContract with the same abi on other Nodes (N-1 remaining nodes)

**for address, define the contract instance based on the VotingContract
the address obtained from Node 1**

Step 5: Verification

accounts on all nodes are acting on the same contract.

all nodes see the same result.

for example: vote count for Rama

Do not forget Closing to save money.

2 Screen Capture of Main Steps:

Please replace the sample photos with your own results.

Your screenshots must include these parts at least and detailed description of each screenshots:

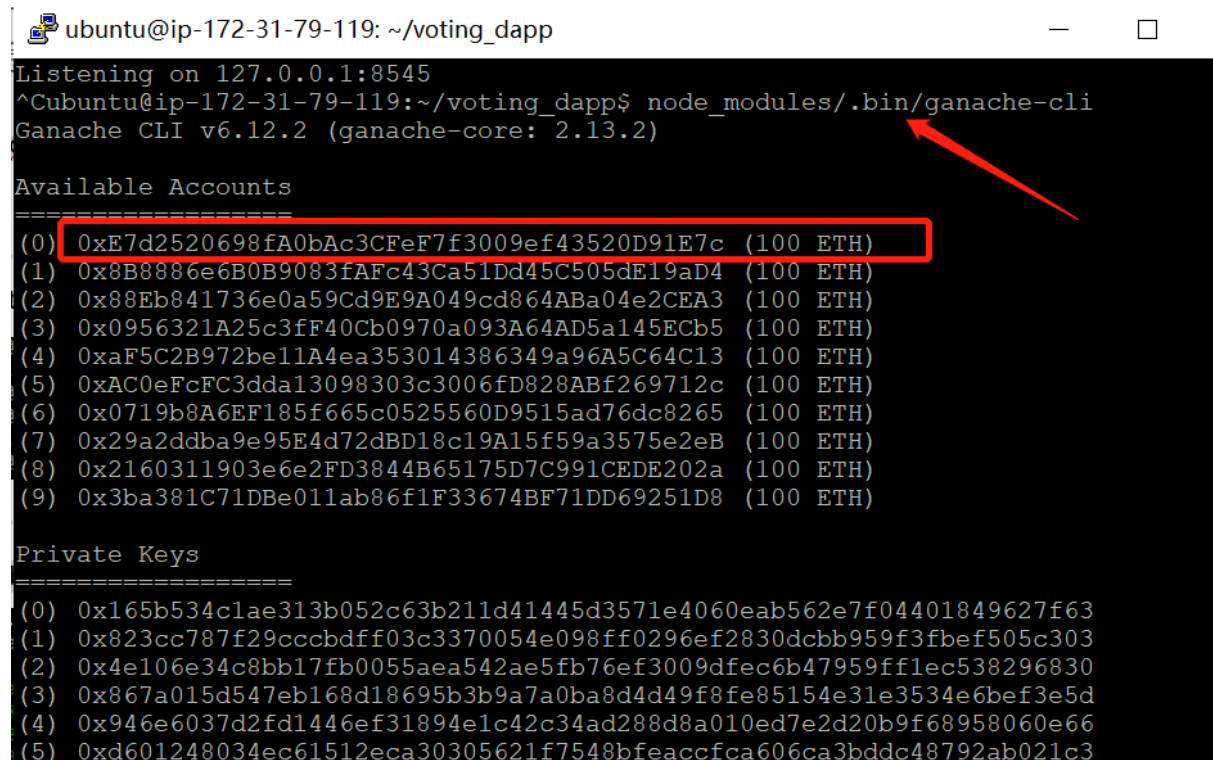
1 The return of running

```
node_modules/.bin/ganache-cli
```

...

My screenshots:

Start a blockchain automatically, show the accounts details and their keys.



```
ubuntu@ip-172-31-79-119: ~/voting_dapp
Listening on 127.0.0.1:8545
^Cubuntu@ip-172-31-79-119:~/voting_dapp$ node_modules/.bin/ganache-cli
Ganache CLI v6.12.2 (ganache-core: 2.13.2)

Available Accounts
=====
(0) 0xE7d2520698fA0bAc3CFeF7f3009ef43520D91E7c (100 ETH)
(1) 0x8B8886e6B0B9083fAFc43Ca51Dd45C505dE19aD4 (100 ETH)
(2) 0x88Eb841736e0a59Cd9E9A049cd864ABa04e2CEA3 (100 ETH)
(3) 0x0956321A25c3fF40Cb0970a093A64AD5a145ECb5 (100 ETH)
(4) 0xaF5C2B972be11A4ea353014386349a96A5C64C13 (100 ETH)
(5) 0xAC0eFcFC3dda13098303c3006fD828ABf269712c (100 ETH)
(6) 0x0719b8A6EF185f665c0525560D9515ad76dc8265 (100 ETH)
(7) 0x29a2ddba9e95E4d72dBD18c19A15f59a3575e2eB (100 ETH)
(8) 0x2160311903e6e2FD3844B65175D7C991CEDE202a (100 ETH)
(9) 0x3ba381C71DBe011ab86f1F33674BF71DD69251D8 (100 ETH)

Private Keys
=====
(0) 0x165b534c1ae313b052c63b211d41445d3571e4060eab562e7f04401849627f63
(1) 0x823cc787f29cccbdf03c3370054e098ff0296ef2830dccb959f3fbef505c303
(2) 0x4e106e34c8bb17fb0055aea542ae5fb76ef3009dfec6b47959ff1ec538296830
(3) 0x867a015d547eb168d18695b3b9a7a0ba8d4d49f8fe85154e31e3534e6bef3e5d
(4) 0x946e6037d2fd1446ef31894e1c42c34ad288d8a010ed7e2d20b9f68958060e66
(5) 0xd601248034ec61512eca30305621f7548bfeaccfca606ca3bddc48792ab021c3
```

2 Please use the ls command in the same file path of Voting.sol to show the compiled output of your Voting.sol

My screenshots:

Under this fold, there are 5 files and 1 folder (sub-folder).

```
ubuntu@ip-172-31-79-119: ~/voting_dapp
ubuntu@ip-172-31-79-119:~/voting_dapp$ ls
Voting.sol          Voting_sol_Voting.bin  package-lock.json
Voting_sol Voting.abi  node_modules          package.json
ubuntu@ip-172-31-79-119:~/voting_dapp$
```

3 Please show your ganache accounts in node console using this command:

```
web3.eth.getAccounts(console.log)
```

My screenshots:

Web3 object communicates with the blockchain, and queries the accounts.

Generate the bytecode.

```
ubuntu@ip-172-31-79-119: ~/voting_dapp
> web3.eth.getAccounts(console.log)
Promise{ <pending> }
> null [
  '0x611CA27E060077436cc8dF4d62f0cE7241bAD7BD',
  '0xa97A086E6c7983D1b12e359ca470853C75168B64',
  '0xF8D85318db384582C30DdDF0d835237eF4bcCDe8',
  '0x7F03fa77b348A9D0116657218542fa078Cc2d7Ad',
  '0x96C13090be591209baf66C95b00F75cF728ACB7D',
  '0x3beE839CCDd68a5E6202A4Ef8B86b669237e3e33',
  '0x01Ca23B8207Bd006EE5e2FFa459a3B590566cDeA',
  '0xbf423cA76183361722b0f8955614b4e264890c46',
  '0x38b2DcA6bCD688bFC07B48bE07d69c6e1Cb3d03D',
  '0xE0B5e0403Aa963e43dA8c462742A9dcC79C94CA2'
]
> bytecode = fs.readFileSync('Voting_sol_Voting.bin').toString()
'60806040523480156100115760006000fd5b506040516104c23803806104c28339818101604052
019080805160405193929190846401000000008211156100565760006000fd5b838201915060208
25186602082028301116401000000008211171561008b5760006000fd5b80835260208301925050
005b838110156100c35780820151818401525b6020810190506100a7565b5050505090500160405
190602001906100ed9291906100f5565b505b50610176565b828054828255906000526020600020
5b82811115610139578251826000509060001916905591602001919060010190610115565b5b509'
```

4 Please show the return contract address after you use deploy method with many parameters to deploy smart contract.

My screenshots:

Create a contract object, use it to deploy and initiate contracts in blockchain.

ubuntu@ip-172-31-79-119: ~/voting_dapp

```
> deployedContract.deploy({
... data: bytecode,
... arguments: [listOfCandidates.map(name=>web3.utils.asciiToHex(name))]
... }).send({
... from: '0xE7d2520698fA0bAc3CF7f3009ef43520D91E7c',
... gas: 1500000,
... gasPrice: web3.utils.toWei('0.00003', 'ether')
... }).then((newContractInstance) => {
... deployedContract.options.address = newContractInstance.options.address
... console.log(newContractInstance.options.address)
... });
Promise { <pending> }
> 0x579cE5B3d79ed367100A1336a31606Eb149D451d
> deployedContract.options.address
'0x579cE5B3d79ed367100A1336a31606Eb149D451d'
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Rama')).call(console.log)
Promise { <pending> }
> null 0
> deployedContract.methods.totalVotesForCandidate(web3.utils.asciiToHex('Rama')).call(console.log)
```

5 Use this command to check the contract address:

```
deployedContract.options.address
```

My screenshots:

Show the HASH address to verify the smart contract is deployed successfully.

ubuntu@ip-172-31-79-119: ~/voting_dapp

```
> deployedContract.options.address
'0x579cE5B3d79ed367100A1336a31606Eb149D451d'
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Rama')).call(console.log)
Promise { <pending> }
> null 0
```

The new block information of this contract in ganache-cli Window:

My screenshots:

Show the detailed information of the smart contract deployed.

ubuntu@ip-172-31-79-119: ~/voting_dapp

```
Gas Limit
=====
6721975

Call Gas Limit
=====
9007199254740991

Listening on 127.0.0.1:8545
eth_sendTransaction

Transaction: 0x15812d6c24e426f8208a757580688f940cbaa5399b52b2953cc1a7a992e4217a
Contract created: 0x1f9e9baa429fecec628aac2d0490d79f3c6b47cb
Gas usage: 31993
Block Number: 1
Block Time: Fri Mar 05 2021 08:48:17 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_getCode
eth_call
eth_gasPrice
eth_sendTransaction
```

6. Check the voting result of all three candidates using this command:

```
deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Rama'))
).call(console.log)
```

My screenshots:

Show the voting results of these 3 candidates, all are 0. Because we have not voted.

```
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Rama')).call(console.log)
Promise { <pending> }
> null 0
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Nick')).call(console.log)
Promise { <pending> }
> null 0
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Jose')).call(console.log)
Promise { <pending> }
> null 0
```

7. Vote for **Rama** using your 1st ganache account

```
deployedContract.methods.voteForCandidate(web3.utils.asciiToHex('Ram  
a')).send({from: 'YOUR 1st ACCOUNT ADDRESS'}).then((f) =>  
console.log(f))
```

The transaction return in the node console:

My screenshots:

Vote for “Rama”, show the detailed information.

[illegible]

The transaction return in the ganache-cli window

My screenshots:

The returned transaction information: block 2. Because block 1 is created above.

```
eth_sendTransaction
Transaction: 0x61ebc5b62c8b97d8d2dc327f6be99770c1d1e0ad5a40a2af9bcebabb55fda26b
Gas usage: 45168
Block Number: 2
Block Time: Fri Mar 05 2021 08:48:52 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
eth_call
```

8. Repeat the last step again, but this time, please vote for **Nick** using the **2nd ganache account**.

My screenshots:

[illegible]

```
eth_getTransactionReceipt
eth_call
eth_gasPrice
eth_sendTransaction

Transaction: 0x14a63de6262691e2c59e233ebf47e006973a88d61248275228b2e3e6eadb1cd8
Gas usage: 47782
Block Number: 3
Block Time: Fri Mar 05 2021 08:54:04 GMT+0000 (Coordinated Universal Time)

eth_getTransactionReceipt
```

```

deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Rama')).call(console.log)
Promise { <pending> }
> null 1
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Nick')).call(console.log)
Promise { <pending> }
> null 1
> deployedContract.methods.totalVotesFor(web3.utils.asciiToHex('Jose')).call(console.log)
Promise { <pending> }
> null 0

```