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 bowser***

***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.***

1. 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:

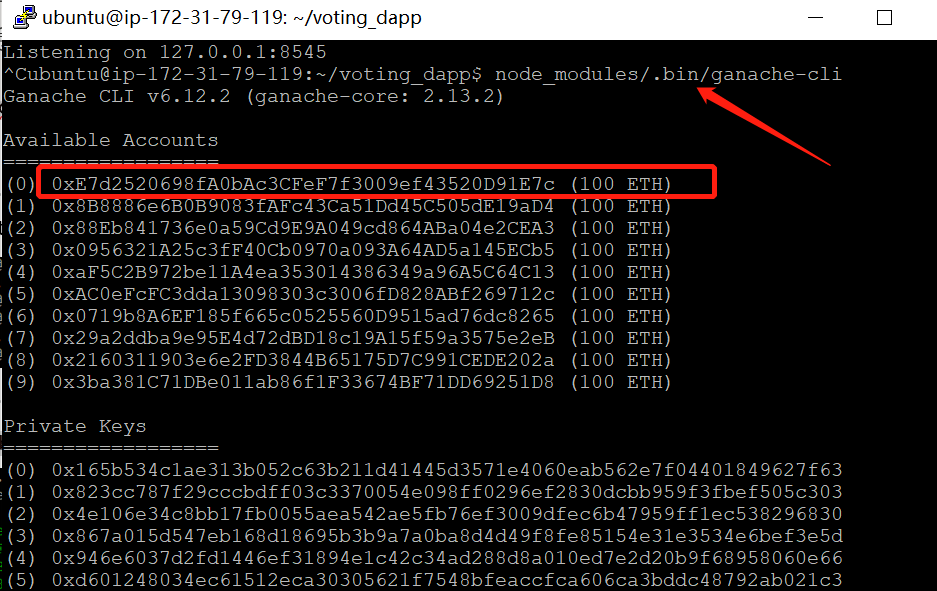
1. The return of running

node\_modules/.bin/ganache-cli

…

My screenshots:

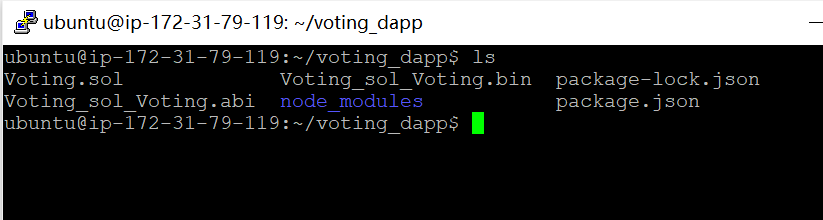
Start a blockchain automatically, show the accounts details and their keys.



1. 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).



1. 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.

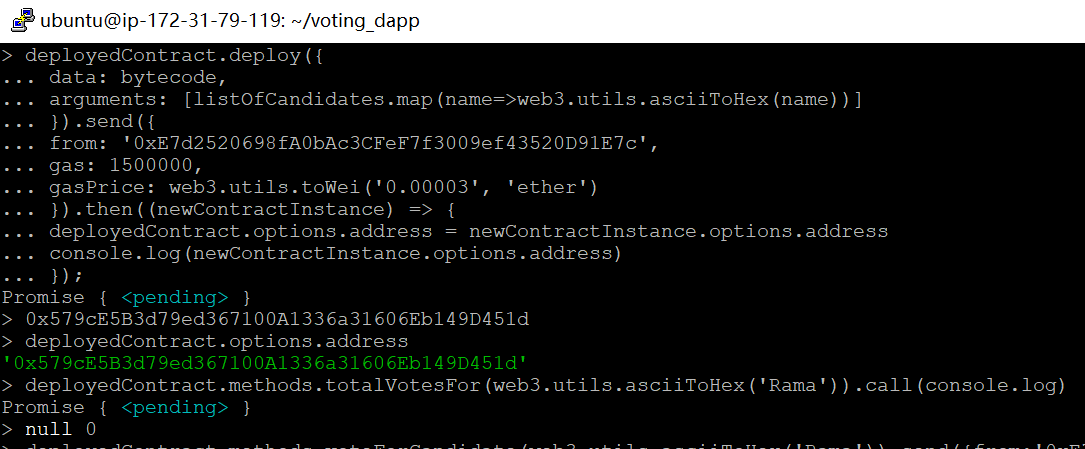
Generate the bytecode.



1. 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.

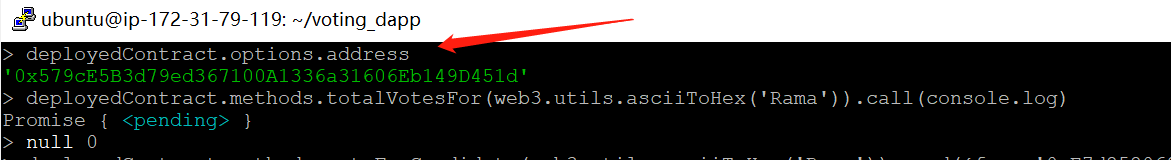


1. Use this command to check the contract address:

deployedContract.options.address

My screenshots:

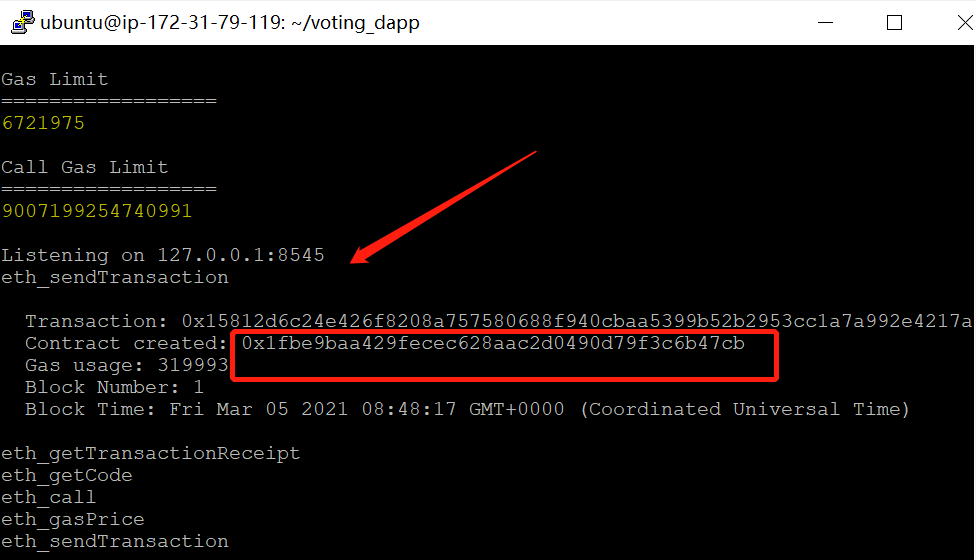
Show the HASH address to verify the smart contract is deployed successfully.



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

My screenshots:

Show the detailed information of the smart contract deployed.

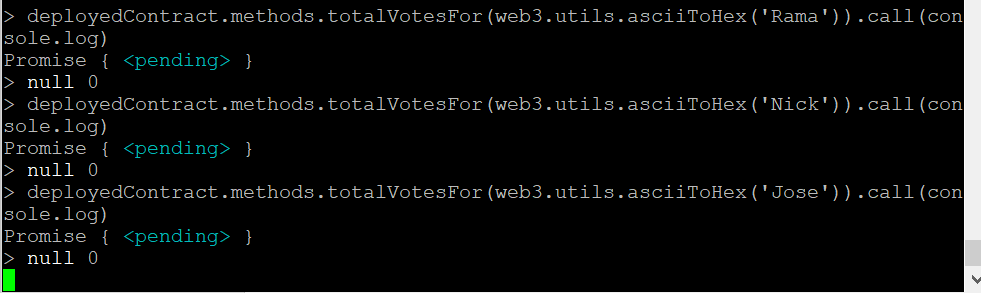


6. Check the voting result of all three candidates using this command:

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.



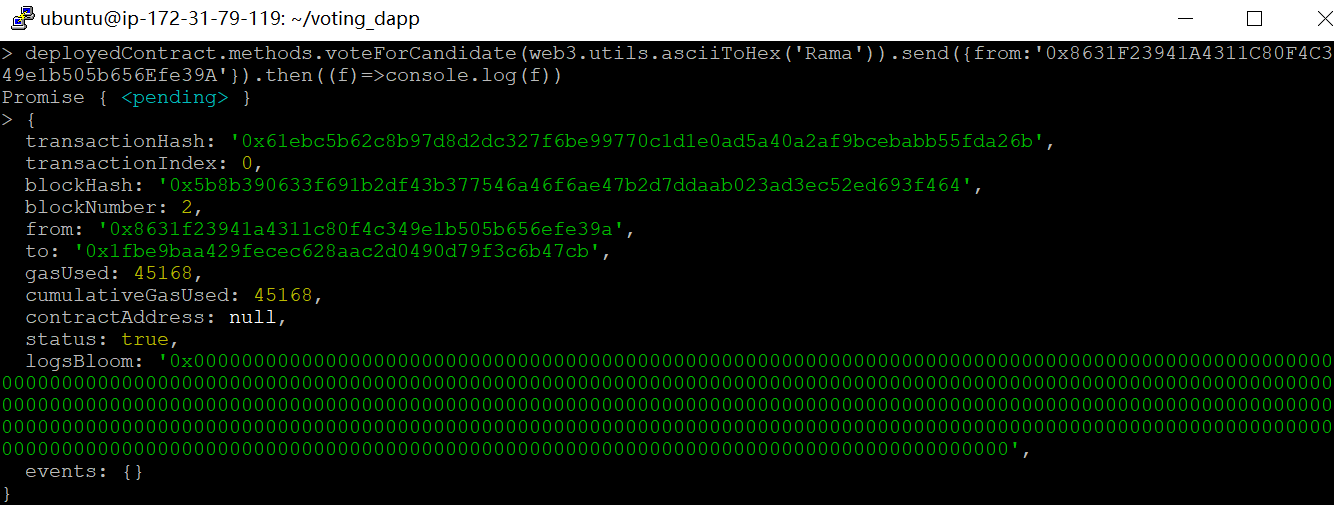
7. Vote for **Rama** using your 1st ganache account

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

The transaction return in the node console:

My screenshots:

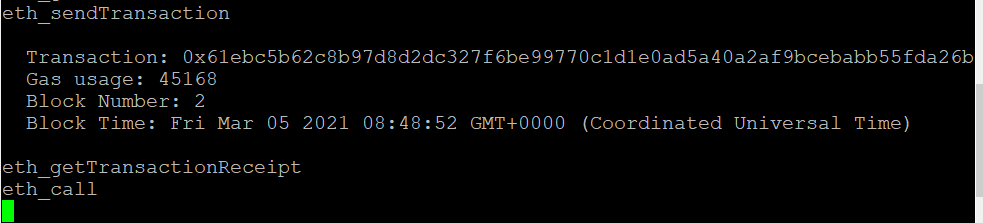
Vote for “Rama”, show the detailed information.



The transaction return in the ganache-cli window

My screenshots:

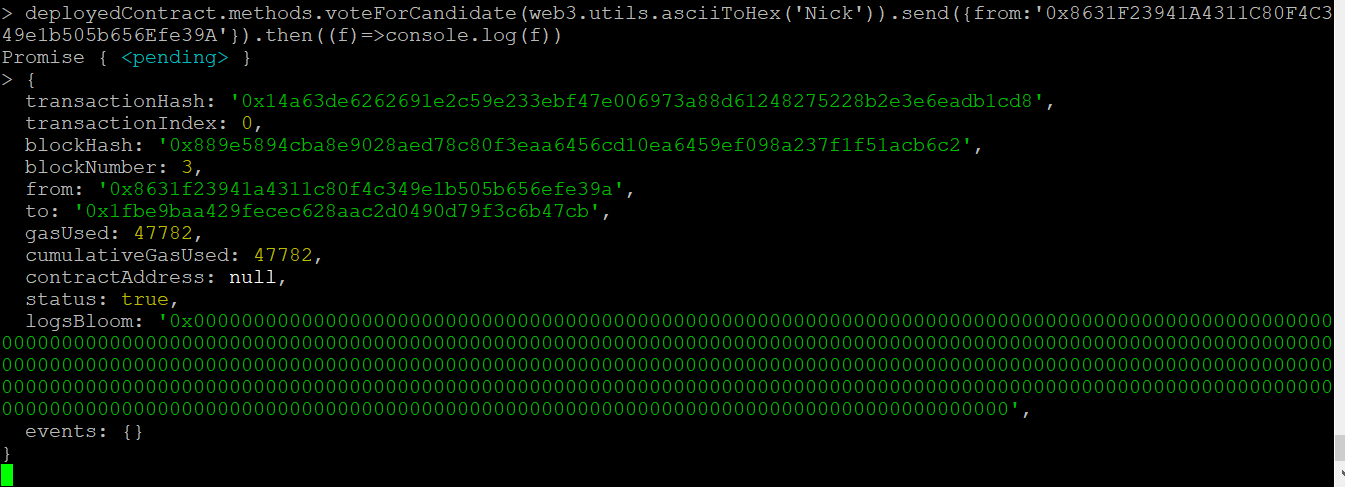
The returned transaction information: block 2. Because block 1 is created above.



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

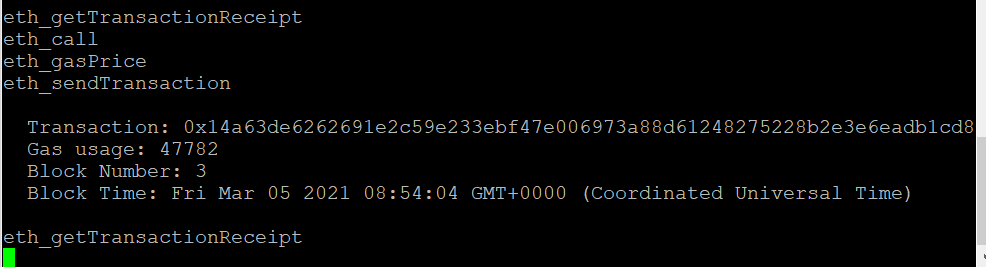
My screenshots:

Vote for “Nick”, show the detailed information.



My screenshots:

The returned transaction info: block 3. Because block 1 & 2 are created above.



9. Check the voting result of each candidates again.

Your photo should contain all three candidates’ voting results.

My screenshots:

After voting for “Rama” and “Nick”, both are 1 now. But “Jose” is 0, because no vote for “Jose”.

