**Setting up a Private Ethereum Blockchain**

Step 1: create a new folder using the following command:

mkdir ethereumprivatenetwork

cd ethereumprivatenetwork/

After this,open the folder which you have created in any text editor of your choice.

Inside this folder create a file named init.json.

Inside of this file, you will define the different parameters for genesis block of a private network.

Step 2: After defined the parameters in init.json file go back to your terminal and just make sure that you are inside the project directory which you had created in the previous step.

Next, you need to issue the following command and press enter:

geth --datadir ./datadir init ./genesis.json

This command will create a private chain data for our blockchain.

When you go back to your code-editor you will see that there is a new folder inside your project directory called datadir which is where our chain data is going to be.

Step 3: Go back to your terminal and here you will start this particular network.

You need to type the following command and press enter:

geth --datadir ./datadir/ --networkid 1 console

This command is going to create a new blockchain network and it will start a console with which you can access the different APIs that are available for this private blockchain network you have created.

After you press enter the network will start with a network id of 1.

Step 4: The next step is to now create an account which can be created using personal API.

You need to type in the following command inside the console: personal.newAccount(‘1234’) Inside the parentheses, you need to type the password for this new account.

When you press enter it is going to create a new account for you and is going to list down the address of this new account. When you go inside of your project directory and inside the keystore folder there is a new keystore file which is the file for this new account which you have created.

Next using the same command create another account with the same password.

In your keystore folder, you will see that another file has been created

Step 5 :

The next step is to start the mining process.

The command for the same is miner.start() When you first start the mining process it’s going to create a new process called DAG generation.

It will only happen for the first time you start mining on your computer.

The process can take a considerable amount of time depending on your computer resources and processing power of your computer.

You will see that new blocks are now being mined on to your computer and all of these blocks have zero transactions.

To stop the mining process you need to issue the following command:

miner.stop()

The mining process is not going to stop unless the DAG creation process is completed. Once the DAG generation is complete mining process will stop automatically.

Install truffle in your system

sudo npm install truffle -g

Deploying smartcontract on private ethereum using truffle:

1.Initialize Trufflestart a new Truffle project by creating a new directory inside your computer. Enter the following command to create a new directory:

$ mkdir truffleproj  
$ cd truffleproj

$truffle init

//this will initialize truffle and create the necessary files and folders in the directory

**2. Create a Smart Contract.**Go to the “contracts” folder in the “truffleproj” directory, and create a new file called “KYC.sol”. You can create this file in Visual Studio, and cut paste the code of KYC.sol presen in Phase-2 folder.

**3. Configure the Truffle migrations folder to deploy the Solidity code.**  
Go to the “migrations” folder in the **“truffleproj”** folder, and create a new file called **2\_kyc\_migration.js**, with the following code.

var KYC = artifacts.require("KYC");

module.exports = function(deployer) {

// deployment steps

deployer.deploy(KYC);

};

4**. Update the truffle task runner with your account and network details. Update “truffle-config.js”**with the following code.

module.exports = {

  networks: {

    development: {

     host: "127.0.0.1",     // Localhost (default: none)

     port: 8545,            // Standard Ethereum port (default: none)

     network\_id: "1",       // Any network (default: none)

     },

     geth: {

    host: "127.0.0.1",

    port:  30303,

    network\_id: "1"

     }

  },

    mocha: {

    // timeout: 100000

  },

  // Configure your compilers

  compilers: {

    solc: {

      version: "^0.4.4",    // Fetch exact version from solc-bin (default: truffle's version)

      // docker: true,        // Use "0.5.1" you've installed locally with docker (default: false)

      // settings: {          // See the solidity docs for advice about optimization and evmVersion

      //  optimizer: {

      //    enabled: false,

      //    runs: 200

      //  },

      //  evmVersion: "byzantium"

      //}

    }

  }

}

**5. Compile the smart contract in Truffle, and deploy it to the private blockchain**

$ truffle compile  
$ truffle migrate

***Troubleshooting:***

* *Ensure that you are in the “truffle” folder to run the commands above.*
* *The deployment of the smart contract to the blockchain may take a few seconds.  
  Ensure that you are mining blocks at the time of smart contract deployment, so that the bytecode is added to the newly mined block of the blockchain. In case you stopped mining before, you can restart mining using the following command in the Geth Javascript console.*

> miner.start()

## Execute the Smart contract on the private Ethereum blockchain

In this step we will execute our “KYC” smart contract on the blockchain using the Truffle console.

Open a new Terminal window and navigate to the “truffleproj” folder. To launch Truffle console from the terminal shell,

$ truffle console