Now that you have installed Geth on your system, it is now time to set-up a private blockchain network. This is a step by step guide of what you have learnt in the videos.

Step 1:

You first need to create a new folder using the following command:

mkdir ethereumprivatenetwork

Then you need to move to this path using the command:

cd ethereumprivatenetwork/

After this, you need to open the folder which you have created in any text editor of your choice. Inside this folder create a file named ***genesis.json***.

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

To define various parameters just follow the instructions in the video.

Step 2:

After you have defined the parameters in ***genesis.json*** file you need to 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:

Now you need to 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 2019 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 2019.

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('aakash')

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.

By using the command "*eth.coinbase"* you can check the coinbase for your private network. Coinbase is that account which will collect the ether rewards for mining on your

particular *geth* node. By default, the first account that you created on the node will become the coinbase account.

If you want to change the coinbase you can do so using the command

miner.setEtherbase('new account ID that is to be made new coinbase account').

Step 5:

To check ether balance in accounts you can use the following command:

eth.getBalance('account ID')

This will return you the balance in ***Wei*** and not in ***ether***.
You can also use the following command:

eth.getBalance(eth.coinbase)

This will return the balance of your coinbase in ***Wei***.

Step 6:

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.

Step 7:

Using the following command

eth.blockNumber

you can see what is the last block number on your private network.

Step 8:

To check the balance in ethers you use the following command:

web3.fromWei(eth.getBalance(eth.coinbase))

This will return to you the *ether* value instead of the *Wei* value.

Using *eth.accounts* command we can get the list of accounts on your network.

Step 9:

eth.sendTransaction({from: eth.accounts[1], to: eth.accounts[0], value: web3.toWei(10, "ethers")})

This command will send the specified amount of ethers which is 10 in this case from the 2nd account to the 1st account.
When you press enter you will get an error saying that you need a password to unlock this account.

Step 10:

The following command is used to unlock an account:

personal.unlockAccount('account ID', 'password')

After that just re-run the command which you tried to run in the previous step. Now you see that the transaction has been submitted.

When you try to check the balance of the 1st account you will see that it is still zero. This is because the transaction has still not been committed to the blockchain. You need to start the mining process again using *miner.start()* command. When you press enter you will see that one of the blocks has transaction as 1. Then stop the mining process again using the *miner.stop()* command.

After this if you check the balance of the 1st account you will get a value returned to you which will be in *Wei.*

Step 11:

Using the command *admin.peers* you can check which are the peers that this particular ethereum client is connected to.
When you run this command you will see that there are no peers that this ethereum client is connected to.

Step 12:

Open a new tab in your terminal. We will now be creating a new peer which means we will be starting a new *geth* client on the same computer.
Now in the new tab type in the following command and press enter:

geth --datadir ./datadir2 init ./genesis.json

The *datadir2* will be the chaincode directory for the second client that we are now creating. Inside your project folder, there are now two directories-one for the first peer and one for the second peer.

Step 13:

Now you need to start the console for the second peer. You need to run the following command and press enter:

geth --datadir ./datadir2 --networkid 2019 --port 30304 console

Also for this peer, you need to define the port number. By default, the port number is 30303 which in this case has been taken by the first client. So the port number has been kept as 30304.

When you run this command the second peer has joined the same network but at a different port number i.e. 30304.

Now if you go to the 1st tab and run the command *admin.peers* you will still see that there are no peers in the network. This is because the peers have to be connected manually in private networks.

Step 14:

Inside peer1 type the command *admin.nodeInfo.enode* and press enter. This will return you the *enode* ID for this particular node.

Step 15:

After getting the *enode ID* of the first peer, go back to the second peer and inside the console for the second peer, run the following command:

admin.addPeer("enode ID of the first peer").

When you paste the ID of the first peer make sure that the IP address is a local-host IP address. So if your IP address for your first peer is anything but 127.0.0.1 you need to change it to **“127.0.0.1”**. After you do this press enter.

Now if you run *admin.peers* command you will see that peer1 has been added inside peer2.

Now you have connected two peers to each other and they are part of the same private network that we have built as part of ethereum.