# Connecting Two Instances of Ethereum in a BlockChain

**By Aneesh Bhatnagar and Vivek Singh**

When connecting two instances of Ethereum in a single BlockChain, you have various ways where these instances of Ethereum are running. It can be two different terminal windows on the same computer, two different computers on the same LAN network, two different computers anywhere in the world or even a virtual machine running on top of a host OS, both capable of running ethereum. The commands that you execute for all these situations will change by a bit only, keeping most of the things still common.

Let’s get started with this step-by-step guide and see how we can get two instances of ethereum connected to each other.

### 1. Installing Ethereum (skip if already done):

In order to do anything with ethereum, you obviously need to install ethereum. Below are the commands if you are running on an Ubuntu machine but if you’re on a different operating system, you will need to look for similar commands for your operating system on the Ethereum website.

Now, enter the following commands one by one into your terminal and wait for ethereum to get installed.

sudo apt install software-properties-common

sudo add-apt-repository -y ppa:ethereum/ethereum

sudo apt update

sudo apt install ethereum

### 2. Creating a new Geth Account on your machine(skip if already done):

Geth is the command line interface for using ethereum on your machine. We will use it to execute all our commands and then sometimes use the RPC protocol or a console to use some other commands or to transact. Before we can get started with geth, we need to create a new user account on our computer. If you are using different installations of ethereum to get the blockchain going, like on two different machines or virtualbox-host combination, you will need to do this command on all those installations but if you are using two-terminal approach, you need to do this only once since we are using the same ethereum installation.

In your terminal, type : ***geth account new*** this will create a new account. When asked for a passphrase, enter a passphrase to create this account. You must remember the passphrase if you want to ever unlock this account. If you are just doing it for fun, it doesn’t matter if you forget your passphrase.

### 3. Creating Required Directories to perform actions:

It is always a good idea to make yourself a working directory whenever working with anything in the field of programming and computers. For the purpose of the blockchain using ethereum, we need to make a data directory because when we initialize our blockchain, it will create a few folders and initialize a few files. Make a directory and call it anything you want. If you are using terminal, then just type **mkdir <directory\_name>** to create a directory. I called the directory testChain, so I executed **mkdir testChain** to create a directory on my computer. If you are using the two-terminal method, create two directories and call them say testChain1 and testChain2 so it makes it easier for you to remember and execute the future commands.

### 4. Creating a Custom Genesis File:

A Genesis block is like the zero block of the blockchain and when two nodes wish to be the part of the same blockchain, they need to have the same zero block initialized for them. Thus, we need to create the same genesis block file twice, depending on which two places you are running these instances. So, type **vi testChain/customGenesis.json** to create the customGenesis.json file in the testChain folder we created above. You will need to exeute this command twice, once for each folder that you’ve created in the previous step.

### 5. Adding Content to the Custom Genesis File:

Copy the content written below and in the terminal, paste it using Ctrl + Shift + V. Then press escape and then Shift + ZZ in order to save the file and exit from the Vim editor.

File content:

{

"config": {

"chainId": 13,

"homesteadBlock": 0,

"eip155Block": 0,

"eip158Block": 0

},

"alloc": {},

"coinbase" : "0x0000000000000000000000000000000000000000",

"difficulty" : "0x20000",

"extraData" : "",

"gasLimit" : "0x2fefd8",

"nonce" : "0x0000000000000042",

"mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"timestamp" : "0x00"

}

### 6. Initializing the blockchain with Custom Genesis Block:

After you’ve saved the custom genesis file, we need to initialize our blockchain with that genesis block. You can do it by executing **geth init testChain/customGenesis.json** in the terminal. You will need to initialize all the instances of the customGenesis file that you’ve created in the previous step, be it on the same machine different folder or different machines. Execute the same command for all those instances.

### 7. Starting Geth and console

Now, we have done everything and we just need to execute our two instances of the ethereum chain. Simply execute the following command below by changing the datadir flag to the directory that you’ve created for that instance of the blockchain. Also, if you are running it on the same computer in two different consoles, make sure that you change the port and rpcport value. For example, you can change 30301 to 30302 and 8101 to 8102. If you keep both of them same, you will get an error saying port already in use.

geth --datadir="testChain/" --nodiscover --maxpeers 4 --networkid 13 --ipcdisable --rpc --rpccorsdomain "\*" --port 30301 --rpcport 8101 console

We are using a variety of flags when starting our blockchain. This is because we do not need the blockchain visible to other networks outside our own network and we don’t want random people to join the blockchain without our permission. If you want to read more about these flags, go through the official ethereum website and you’ll find out what each of these flag means.

### 8. Connecting to the other instance

Now that you’ve started both your instances, you want to connect them together to be part of the same chain. To do so, we need to just do one more simple step and we are then good to perform our other transactions of the blockchain.

After you started the instance with the command in the 7th step, towards the end of the messages that ethereum logs on your screen, you will see something like “**self="enode://d8dec0201db5349ce0fa038747c0ef063883808e041abd751790fb8dcea3b9ea79bfcad0d03a6232440492f0df8585538a037969f5724b9f42be050ed08cfbd2@[::]:30302?discport=0"**” where the enode value and the port would be different. Note this down in a separate place, say a text editor for now.

The enode value that we’ve just got is the identification for our current instance of the geth and we need to use this identity to connect this to another instance. The command to do that is: **admin.addPeer(<enode here>)**.Now we need to edit the enode string with the following things, based on the type of connection you’re trying to make.

* Same computer two-terminals: No change needed, just use the string starting from enode://, all the way to the end of your port number. eg: **enode://d8dec0201db5349ce0fa038747c0ef063883808e041abd751790fb8dcea3b9ea79bfcad0d03a6232440492f0df8585538a037969f5724b9f42be050ed08cfbd2@[::]:30302**
* Two machines on the same network or virtual machine-host combination: We need to change the IP Address in the enode string to the respective IP on the network of that machine. To find your ip address, simply type “ifconfig” in a terminal window and you’ll see the response there. If my IP Address is 192.168.1.179, I will change the enode string as follows: enode://d8dec0201db5349ce0fa038747c0ef063883808e041abd751790fb8dcea3b9ea79bfcad0d03a6232440492f0df8585538a037969f5724b9f42be050ed08cfbd2@**192.168.1.179**:30302
* If you are connecting to two computers on the internet, not on the same network, you can do the similar thing as above, just replace your external IP Address with the value. So do a google search for: “What’s my IP Address” and then whatever you get replace it in place of 192.168.1.179 in the above example.

After you have created your enode string for the current instance of your ethereum, switch to the other instance, i.e. another machine, virtual machine or another terminal and in the console, execute the following command. Remember to change the enode string in the command below. This is for my computer’s one instance.

**admin.addPeer(“enode://d8dec0201db5349ce0fa038747c0ef063883808e041abd751790fb8dcea3b9ea79bfcad0d03a6232440492f0df8585538a037969f5724b9f42be050ed08cfbd2@[::]:30302”).**

Well now you have successfully connected to the other instance of your ethereum. If you want to check, execute one of the following 2 commands and you’ll see the result.

* net.peerCount - If this gives you 1, it means that you are connected to 1 other peer in the network. If it gives you an output of 0, it means something went wrong.
* admin.peers - If this gives you a detailed information about a peer, it means you are connected to that peer. If it gives you “[ ]” as the output, it means that something went wrong.