Collin
Cusce
Computer Scientist, Futurist, Applied Philosopher. Central focus on decentralization.
Jun 22 · 16 min read

Using puppeth To Manually Create An Ethereum Proof Of Authority (Clique) Network On AWS

Creating a private blockchain is not a trivial task presently. It takes a lot of time and effort to get the network deployed. While services are coming out which makes this task much less complicated, there are some instances where your administrators want control over the private network and full ability to run and maintain it internally. To do this 100% by hand would be exceptionally time consuming, if it weren't for a handy tool by the Ethereum group called "puppeth".

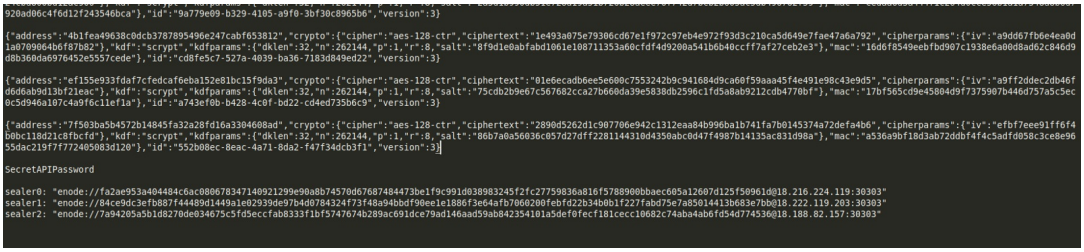
Puppeth makes deploying and managing a Proof of Authority network much less painful, but there is a lot of pre-work to getting the network up and running. It requires setting up your servers for your nodes and setting the security right. Then comes the steps of actually getting them to talk to each other. The puppeth tool makes the job go from a week's set up to a day to launch a full-scale Proof of Authority network on your own.

For the purpose of this article, I'm going to focus solely on AWS to launch the network. This is not a limitation of puppeth, it just makes writing this easier. You can launch your nodes anywhere, and in an ideal scenario, the nodes will exist across data centers and organizations. This article also focuses specifically on launching the network and the tools provided by puppeth. Once it's up, it's up to you to decide what to do with it.

I want to thank my good friend Corey Petty for helping me with my first installation. I would also like to thank Patrick Jang of Dynamo Technologies for working with me to iterate on our notes over and over to improve clarity. Without their assistance this tutorial would not be in the refined state that it is.

Pre-game Set-Up

Before you get started, you'll want to make sure you have a file for storing notes ready on your local machine. Keeping notes of certain values is absolutely essential for success. Throughout the tutorial I will tell you exactly what notes to record and why. In the end you will have a notes file that looks like this:



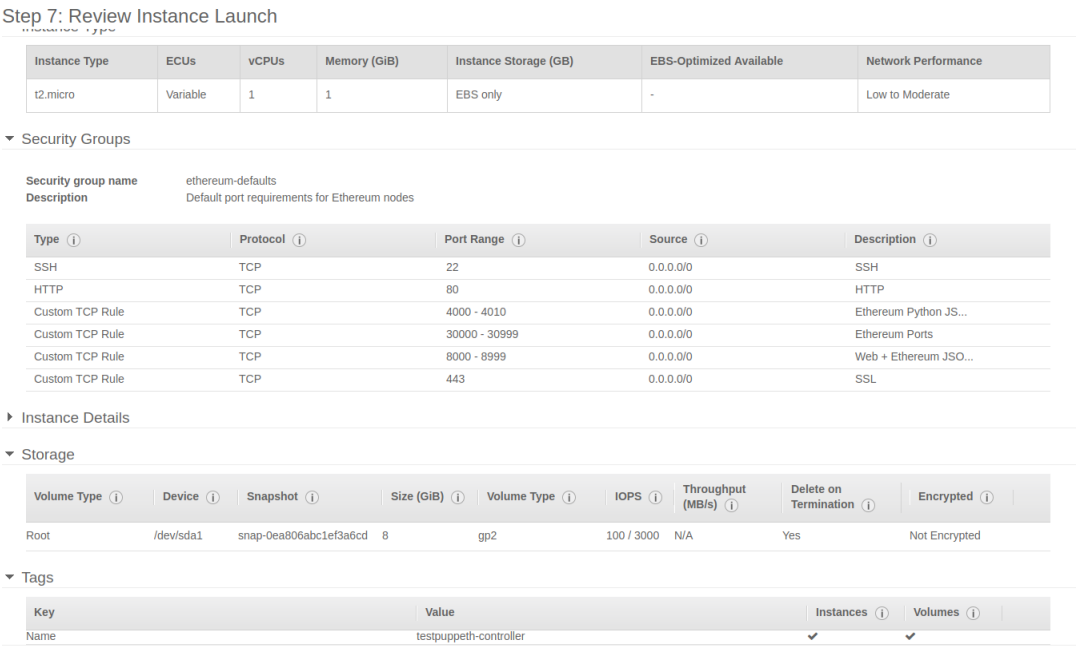
I'm going to assume you've already set up a keypair to log into your instances and named it "idchain.pem". It could be named whatever you set it, and can be another existing keypair. When I reference "idchain.pem", just mentally replace it with whatever keypair you need to log into that server. On your local machine, make sure to chmod the permissions on "idchain.pem" with `chmod 400 idchain.pem` before you use it. If you're getting permissions error on the file, that's likely why.

Provision Instances On AWS

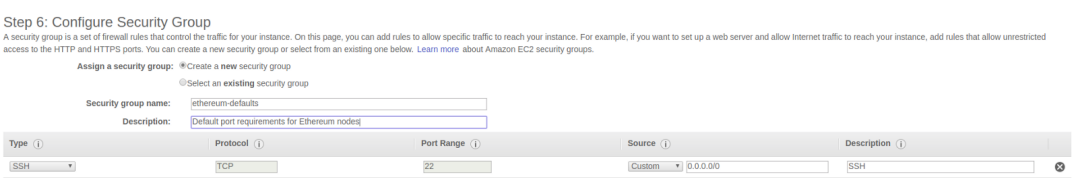
You're going to provision 4 instances on AWS for this tutorial: 3 nodes and 1 controller. The controller is where we're going to be doing a majority of your work. This will be where puppeth reside and it will launch the appropriate tools on the other instances once all dependencies are set up.

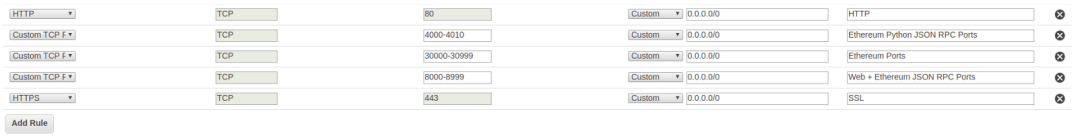
- 1 instance of t2.micro Ubuntu 16.04 for the controller
- 3 instances of t2.medium (10 gb) Ubuntu 16.04 for the nodes

The controller server should look as follows:



While creating this you need to create a security group and name it "ethereum-default". You will use this for all instances. It will have the following inbound port rules:





The node servers should be configured as follows:

Step 7: Review Instance Launch

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.medium	Variable	2	4	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-721f8618	ethereum-defaults	Default port requirements for Ethereum nodes

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	HTTP
SSH	TCP	22	0.0.0.0/0	SSH
Custom TCP Rule	TCP	30000 - 30999	0.0.0.0/0	Ethereum Ports
Custom TCP Rule	TCP	4000 - 4010	0.0.0.0/0	Ethereum Python JS...
Custom TCP Rule	TCP	8000 - 8999	0.0.0.0/0	Web + Ethereum JSO...
Custom TCP Rule	TCP	443	0.0.0.0/0	SSL

Instance Details

Storage

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-0ea806abc1ef3a6cd	10	gp2	100 / 3000	N/A	Yes	Not Encrypted

Note that every single instance has the same security group. This is for simplicity sake. The security rules are also fairly lenient, but for the sake of this tutorial they'll do well. You can tweak things after your first run through creating a network. For now, do everything as is in this article.

The instances should be named "testpuppeth-controller" for the controller server. The node servers should be named "testpuppeth000", "testpuppeth001", and "testpuppeth002" for this tutorial. It will get confusing if you change names, so at least in your first run, try to stick close to the tutorial. Trust me on this one.

Take down the IP addresses and name the instances in AWS. Record them in your notes file I asked you to create above. Example:

> *testpuppeth-controller: 18.221.99.213*
> *testpuppeth000: 18.216.224.119*
> *testpuppeth001: 18.222.119.203*
> *testpuppeth002: 18.188.82.157*

Enable Node Servers For Controller Authorized Keys

Now you should have each server launched, and the proper security group set up, with the same keypair on each node and the controller. Proceed with the following steps:

1. Log into controller instance using `ssh -i idchain.pem ubuntu@<ip of controller>`
2. Place the pem file for your security group onto the controller. You can just copy and paste the contents into a new file named "idchain.pem".

3. Run `chmod 400 idchain.pem`
4. To be sure, also run `sudo perl -pi -e ‘chomp if eof’ idchain.pem` in case a trailing newline is inserted.
5. Create controller’s rsa_id (I left password blank) and name it “id_rsa” by running `ssh-keygen`
6. Run the following for each node instance ip address to add the controller’s public key to their authorized keys: `ssh -i idchain.pem ubuntu@<ip of node> ‘cat >> ~/.ssh/authorized_keys’ < ~/.ssh/id_rsa.pub`
7. Test that it worked by logging into each node instance from the controller using `ssh ubuntu@<ip of node>`

This will enable puppeth to log into each sealer node server and do its magic. Without this, puppeth has no way of logging in yet. You need to make it pretty seamless and magical for puppeth to do its work.

Prepping Node Instances (Not Controller)

You’re going to need to fetch and install dependencies onto each node machine. Run all instructions in this second on each of the three node servers.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install linux-image-extra-4.4.0–59-generic linux-image-extra-virtual
sudo apt-get install \
 apt-transport-https \
 ca-certificates \
 curl \
 software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
 “deb [arch=amd64] https://download.docker.com/linux/ubuntu \
 $(lsb_release -cs) \
 stable”
sudo apt-get update
sudo apt-get install docker-ce
sudo apt-get install docker-compose
```

As a precaution, add user permissions to run docker and docker-compose. It’s probably done anyway, but just in case:

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

Log out, log in, and then run `docker ps` to make sure permissions are correct.

Prepare The Controller

You’re going to need to get the dependencies set for our controller instance. First, from your local machine, ssh into the controller instance:

-

```
ssh -i idchain.pem ubuntu@<ip of controller>
```

From the controller instance, run the following:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential bison
```

Install gvm, The Go Version Manager

The “geth” client needs to have the go language installed on the controller machine to compile from source. Since you’re going to be compiling from source in this tutorial, you’re going to use the gvm tool to install go.

```
bash < <(curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/binscripts/gvm-installer)
source ~/.gvm/scripts/gvm
```

Build The Newest Version of go Language

You have to install go version 1.4 and then transition to go version 1.10.

```
gvm install go1.4 --binary
gvm use go1.4
export GOROOT_BOOTSTRAP=$GOROOT
gvm install go1.10.1
gvm use go1.10.1
```

Clone And Build puppeth From go-ethereum

You’re going to use a local copy of puppeth, so you’ll clone from the go-ethereum git repo to build it directly.

```
git clone https://github.com/ethereum/go-ethereum.git
cd go-ethereum
make all
cd build/bin
```

Now execute `./puppeth` to make sure the program is working and then exit the program.

By this point, the puppeth command is compiled, located in “~/go-ethereum/build/bin”, and the controller is able to operate all nodes.

Create Wallets

You’re going to need to set up some Ethereum wallets in order to create the network and have pre-funded accounts. The following steps go through how to do that.

and have pre-funded accounts. The following steps go through how to do that.

Create A Password

Create a password file named “passfile”, put a password (any password) in it, remove any possible trailing newlines, and chmod it 700.

```
cd ~
vim passfile
sudo perl -pi -e 'chomp if eof' passfile
chmod 700 passfile
```

Create Accounts

Type in the following command in the controller’s terminal.

```
./go-ethereum/build/bin/geth account new --password passfile
```

This will be your sealer account. Jot down the address in your notes file and label it “sealer0” as follows:

> *sealer0: 6c1766673883596ce01b0cfe3748081525d8a6dc*

This script will do it again 10 times:

```
for ((n=0;n<10;n++)); do ./go-ethereum/build/bin/geth account new --password ~/passfile; done
```

You can use as few or as many of these as you want for your purposes. When pre-loading ethereum, I will recommend you use at least three of these accounts.

Paste the wallet addresses generated into same text file and store them. Mark the first 2 added as sealer1 and sealer2. Mark the 3rd added as faucet.

> *sealer0: 6c1766673883596ce01b0cfe3748081525d8a6dc*
> *sealer1: 4b1fea49638c0dcb3787895496e247cabf653812*
> *sealer2: ef155e933fdaf7cfedcaf6eba152e81bc15f9da3*
> *faucet: 7f503ba5b4572b14845fa32a28fd16a3304608ad*
>
> *accounts:*
> *ea98c00832b70d3e20a7c14f3a7f9d6b11a8e5bd*
> *81b518fc09f1520190201fcef16c6e8e424c2516*
> *1fe4bbef8a2eb18ec6a7fe54282baa0b7019e892*
> *fc14badd2a8de765a9ca2726b4124071a82a2296*
> *0f84eb0db755d9503fe2fb0e40b5689aeabe1b97*
> *40a62173fcff9a3944999adba9e5dc8f0d003df2*
> *fc3626df3d120266586a993709d0b54ad985bca9*

What Is This?

In Proof of Work, every node is about the same. They all function as validators across

In Proof of Work, every node is about the same. They all function as validators across the network. In Proof of Authority, there's particular nodes which are assigned to validate blocks. These are known as "sealer" nodes. Each one needs its own account. You're going to reserve three of our accounts for that purpose. New authority nodes can be added but they need to be elected by the rest of the network through their geth console. For now, you're only concerned with setting up the initial authority network, so you'll stick with the three sealer nodes.

There will also need to be an account which can operate the "faucet" you'll install later. This will allow non-sealer nodes to request some eth for operational purposes. This faucet will come pre-loaded with some Ether on your network.

Use puppeth To Define the Ethereum Network

Equipped with our notes, accounts, and the tools, you will now run puppeth on the controller node to launch your Proof of Authority network and install the applications it uses.

Launch puppeth with:

```
./go-ethereum/build/bin/puppeth
```

The first prompt asks:

> *"Please specify a network name to administer (no spaces or hyphens, please)"*

This is the name of your network. Type "testpuppeth" and hit "Enter".

Configure Genesis

First step is to configure your genesis file. This is the definition of the network that will distribute to all nodes on the network. When puppeth launches it will ask what you would like to do. You're going to type "2" to select "2. Configure new genesis" and hit "Enter".

```
+-----------------------------------------------------------+
| Welcome to puppeth, your Ethereum private network manager |
|                                                           |
| This tool lets you create a new Ethereum network down to  |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|                                                           |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----------------------------------------------------------+

Please specify a network name to administer (no spaces or hyphens, please)
> testpuppeth

Sweet, you can set this via --network=testpuppeth next time!

INFO [06-22|22:09:58.603771] Administering Ethereum network           name=testpuppeth
WARN [06-22|22:09:58.603866] No previous configurations found         path=/home/ubuntu/.puppeth/testpuppeth

What would you like to do? (default = stats)
 1. Show network stats
 2. Configure new genesis
 3. Track new remote server
 4. Deploy network components
>
```

Next it's going to ask which consensus engine to use? You're going to type "2" to select "2. Clique - proof-of-authority"

2. Clique — proof-of-authority

It then asks how many seconds a block should take. The default is 15, which is approximately what you'd expect from the Ethereum mainnet. You're going to go a bit faster, though, and select 5 second block times. I have gone down as a low as 1 second and it works fine. The only issue is the growth rate of the blockchain increases with every block added, so the faster blocks are added, the faster the blockchain grows in size.

It's now going to ask for your sealer accounts. You're going to paste in the first sealer, "sealer0", hit "Enter", then do the same for "sealer1" and "sealer2". When done, simply hit "Enter" again with no account on the line and it will continue.

```
Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2

How many seconds should blocks take? (default = 15)
> 5

Which accounts are allowed to seal? (mandatory at least one)
> 0x6c1766673883596ce01b0cfe3748081525d8a6dc
> 0x4b1fea49638c0dcb3787895496e247cabf653812
> 0xef155e933fdaf7cfedcaf6eba152e81bc15f9da3
> 0x
```

Next it will ask which accounts should be pre-funded. You're going to select the sealer accounts, the faucet account, and for good measure, three additional accounts. Paste them in one-at-a-time and hit "Enter" when you are done populating the list. Remember you can add as many from the accounts we created as you want.

```
Which accounts are allowed to seal? (mandatory at least one)
> 0x6c1766673883596ce01b0cfe3748081525d8a6dc
> 0x4b1fea49638c0dcb3787895496e247cabf653812
> 0xef155e933fdaf7cfedcaf6eba152e81bc15f9da3
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0x6c1766673883596ce01b0cfe3748081525d8a6dc
> 0x4b1fea49638c0dcb3787895496e247cabf653812
> 0xef155e933fdaf7cfedcaf6eba152e81bc15f9da3
> 0x7f503ba5b4572b14845fa32a28fd16a3304608ad
> 0xea98c00832b70d3e20a7c14f3a7f9d6b11a8e5bd
> 0x81b518fc09f1520190201fcef16c6e8e424c2516
> 0x1fe4bbef8a2eb18ec6a7fe54282baa0b7019e892
> 0x
```

It's going to ask you for your "chain/network ID" next. This is a number which specifies your chain on the network. This would distinguish your chain from others. I tend to go with random, but you can enter in your own value if you want.

At this point, you've created a new genesis file!

Now save the genesis file. At the prompt you see, hit "2" to "Manage existing genesis" and then "2" again to "Export genesis configuration". It will then ask you what to name the file. Go with the default. The genesis file is now in your home directory for future use.

Get Wallet Keyfiles

You're going to need to copy over the private keys for the sealer nodes so you can use them in puppeth in a bit. Exit puppeth and then run the following:

```
cd ~/.ethereum/keystore
ls
```

This will list all the key files in the controller machine. Find the key files for each of the sealer nodes and the faucet node. They should have their address at the end of the file. Then "cat" it to terminal and copy the contents into your notes for later use.

> *cat UTC—2018–06–22T21–49–10.683674340Z—6c1766673883596ce01b0cfe3748081525d8a6dc*

> *cat UTC—2018–06–22T21–51–14.962170613Z—4b1fea49638c0dcb3787895496e247cabf653812*

> *cat UTC—2018–06–22T21–51–15.849894157Z—ef155e933fdaf7cfedcaf6eba152e81bc15f9da3*

> *cat UTC—2018–06–22T21–51–16.736330036Z—7f503ba5b4572b14845fa32a28fd16a3304608ad*

The output of this will be JSON. Copy only the JSON into your notes so you have it later.

Use Puppeth To Deploy The Ethereum Network

Open up puppeth again:

```
cd ~/go-ethereum/build/bin
./puppeth
```

Type in the network name, "testpuppeth", and hit "Enter".

Create Ethstats

Going forward I'm going to abbreviate the commands into list format because I think you get the idea by now.

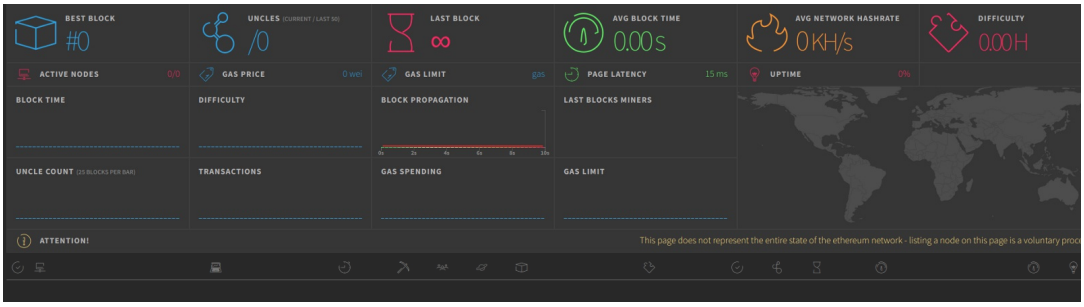
You need to install Ethstats first for it to monitor the network. Ethstats shows you the status of your ethereum network. You're going to install this (and other applications) on "testpuppeth000".

1. Type "4" for "4. Deploy network components"
2. Type "1" for "1. Ethstats—Network monitoring tool"
3. Type "1" for "1. Connect another server"

4. Paste in the IP address for “testpuppeth000”
5. Type in “8080” for the port Ethstats listens on
6. Type “n” to prevent port sharing with other services.
7. Type out a secret password for the API (be sure to either commit it to memory or store it in your notes)

```
Which server do you want to interact with?
 1. Connect another server
> 1

Please enter remote server's address:
> 18.216.224.119

The authenticity of host '18.216.224.119:22 (18.216.224.119:22)' can't be established.
SSH key fingerprint is 2e:88:55:36:03:15:b7:ce:69:39:6a:de:47:82:8a:55 [MD5]
Are you sure you want to continue connecting (yes/no)? yes

Which port should ethstats listen on? (default = 80)
> 8080

Allow sharing the port with other services (y/n)? (default = yes)
> n

What should be the secret password for the API? (must not be empty)
> SecretAPIPassword
Creating network "testpuppeth_default" with the default driver
Building ethstats
Step 1/2 : FROM puppeth/ethstats:latest
latest: Pulling from puppeth/ethstats
Digest: sha256:1728c03555d3327f68be924116eed9f9de56671949c21505f4b78518f06e687e
Status: Downloaded newer image for puppeth/ethstats:latest
 ---> fb62abe59cb2
Step 2/2 : RUN echo 'module.exports = {trusted: ["18.216.224.119"], banned: [], reserved: ["yournode"]};' > lib/utils/config.js
 ---> Running in d839c7085b0b
Removing intermediate container d839c7085b0b
 ---> ee0a5c1f9a0f
Successfully built ee0a5c1f9a0f
Successfully tagged testpuppeth/ethstats:latest
Creating testpuppeth_ethstats_1
INFO [06-22|22:47:02.739248] Starting remote server health-check      server=18.216.224.119
WARN [06-22|22:47:02.990354] Ethstats service seems unreachable       server=18.216.224.119 port=8080 err="dial tcp 18.216.224.119:8080: connect: connection refused"
+----------------+----------------+----------+----------------------+------------------+
|     SERVER     |    ADDRESS     | SERVICE  |        CONFIG        |       VALUE      |
+----------------+----------------+----------+----------------------+------------------+
| 18.216.224.119 | 18.216.224.119 | ethstats | Banned addresses     | []               |
|                |                |          | Login secret         | SecretAPIPassword |
|                |                |          | Website address      | 18.216.224.119   |
|                |                |          | Website listener port | 8080             |
+----------------+----------------+----------+----------------------+------------------+
```

You now have installed Ethstats on “testpuppeth000”. Go to the http://<ip of testpuppeth000>:8080 and you should see an empty Ethstats!



Nothing is running because the network isn’t set up and running blocks yet, but things will filter in as you proceed.

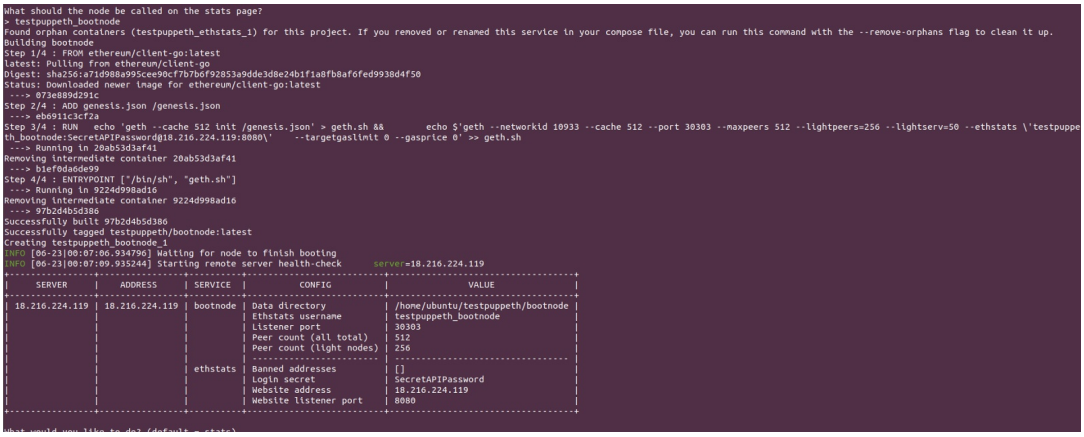
Create A Bootnode

The bootnode service helps bootstrap the Ethereum Proof of Authority network so all nodes can come along.

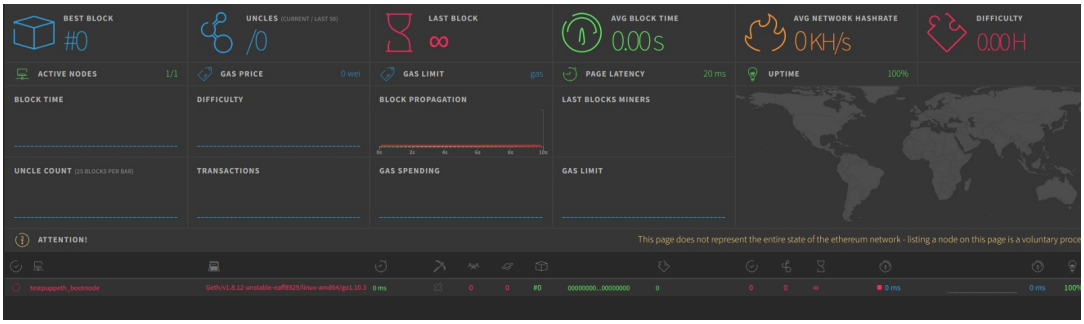
1. Type “4” for “4. Deploy network components”
2. Type “2” for “2. Deploy new network component”
3. Type “2” for “2. Bootnode—Entry point of the network”
4. Type “1” for “1. <ip of testpuppeth000>”
5. Store the data on the machine at “/home/ubuntu/testpuppeth/bootnode”
6. Type “30305” for the TCP/IP port
7. Hit “Enter” to select the default number of peer connections

7. Hit “Enter” to select the default number of peer connections
8. Hit “Enter” to select the default number of light peers
9. Type “testpuppeth_bootnode” for the name of the node on the stats page

You’ll see the bootnode being created in your terminal.



Once the bootnode is running you’ll see it appear on Ethstats by going to http://<ip of testpuppeth000>:8080 .



Create Sealer Nodes

Now it’s time to create the sealer nodes which validate the network. **You’re going to do these instructions 3 times for each node (testpuppeth000, testpuppeth001, and testpuppeth002).**

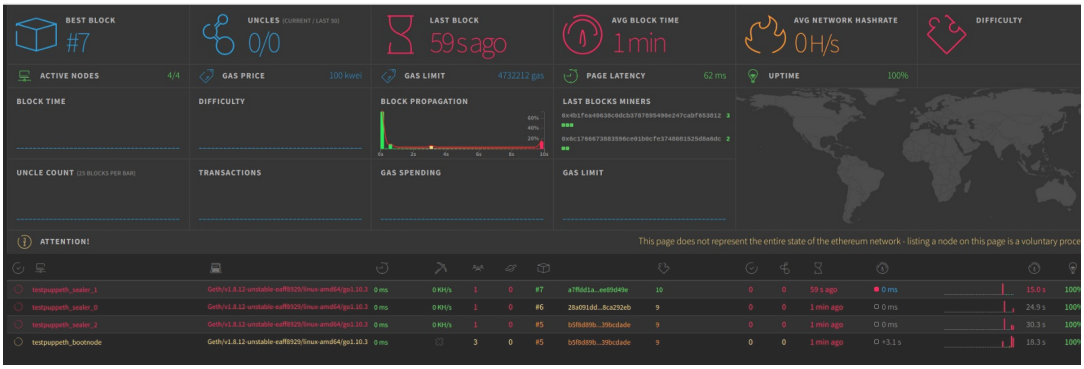
1. Type “4” for “4. Deploy network components”
2. Type “3” for “3. Deploy new network component” (this number increases each time)
3. Type “3” for “3. Sealer—Full node minting new blocks”
4. Type “1” for “1. <ip of testpuppeth000>” or type “2” for “2. Connect another server” to add testpuppeth001 and testpuppeth002 (this number increases each time)
5. If connecting another server, type in the IP address for one of the nodes each time you do this for testpuppeth001 or testpuppeth002
6. Store the data on the machine at “/home/ubuntu/testpuppeth/sealernode”
7. Hit “Enter” to select the default TCP/IP port
8. Hit “Enter” to select the default number of peer connections
9. Hit “Enter” to select the default number of light peers

9. Hit "Enter" to select the default number of light peers
10. Type “testpuppeth_sealer_0”, “testpuppeth_sealer_1”, or “testpuppeth_sealer_2” for the name of the node on the stats page, assigning a unique name for each sealer node
11. Paste in the JSON from the keyfile copied earlier into your notes that corresponds with the sealer node.
12. Type in the unlock password for this keyfile, stored in your “passfile” file created earlier (the text wont display as you type)
13. What gas limit should empty blocks target … make this what you want, but for this tutorial, we’re going to make it stupid high and set “94000000” MGas
14. What gas price should the signer require… make this what you want, but for this tutorial, we’re going to make it stupid low and set “0.0001” GWei

When you’re done with the final node, you should see terminal output as seen below.

```
Creating testpuppeth_sealnode_1
INFO [06-23|02:08:31.527501] Waiting for node to finish booting
INFO [06-23|02:08:34.527911] Starting remote server health-check      server=18.216.224.119
INFO [06-23|02:08:34.528203] Starting remote server health-check      server=18.222.119.203
INFO [06-23|02:08:34.528418] Starting remote server health-check      server=18.188.82.157
+----------------+----------------+----------+---------------------------+------------------------------------------+
|     SERVER     |    ADDRESS     | SERVICE  |          CONFIG           |                  VALUE                   |
+----------------+----------------+----------+---------------------------+------------------------------------------+
| 18.188.82.157  | 18.188.82.157  | sealnode | Data directory            | /home/ubuntu/testpuppeth/sealernode      |
|                |                |          | Ethstats username         | testpuppeth_sealer_2                     |
|                |                |          | Gas limit (baseline target) | 94000000.000 MGas                      |
|                |                |          | Gas price (minimum accepted) | 0.000 GWei                            |
|                |                |          | Listener port             | 30303                                    |
|                |                |          | Peer count (all total)    | 50                                       |
|                |                |          | Peer count (light nodes)  | 0                                        |
|                |                |          | Signer account            | 0xeF155E933FdAf7cfeDcaF6Eba152e81Bc15f9Da3 |
| -------------- | -------------- | -------- | ------------------------- | ---------------------------------------- |
| 18.216.224.119 | 18.216.224.119 | bootnode | Data directory            | /home/ubuntu/testpuppeth/bootnode        |
|                |                |          | Ethstats username         | testpuppeth_bootnode                     |
|                |                |          | Listener port             | 30305                                    |
|                |                |          | Peer count (all total)    | 512                                      |
|                |                |          | Peer count (light nodes)  | 256                                      |
|                |                |          | ------------------------- | ---------------------------------------- |
|                |                | ethstats | Banned addresses          | []                                       |
|                |                |          | Login secret              | SecretAPIPassword                        |
|                |                |          | Website address           | 18.216.224.119                           |
|                |                |          | Website listener port     | 8080                                     |
|                |                |          | ------------------------- | ---------------------------------------- |
|                |                | sealnode | Data directory            | /home/ubuntu/testpuppeth/sealernode      |
|                |                |          | Ethstats username         | testpuppeth_sealer_0                     |
|                |                |          | Gas limit (baseline target) | 94000000.000 MGas                      |
|                |                |          | Gas price (minimum accepted) | 0.000 GWei                            |
|                |                |          | Listener port             | 30303                                    |
|                |                |          | Peer count (all total)    | 50                                       |
|                |                |          | Peer count (light nodes)  | 0                                        |
|                |                |          | Signer account            | 0x6c1766673883596cE01B0CfE3748081525D8a6dc |
| -------------- | -------------- | -------- | ------------------------- | ---------------------------------------- |
| 18.222.119.203 | 18.222.119.203 | sealnode | Data directory            | /home/ubuntu/testpuppeth/sealernode      |
|                |                |          | Ethstats username         | testpuppeth_sealer_1                     |
|                |                |          | Gas limit (baseline target) | 94000000.000 MGas                      |
|                |                |          | Gas price (minimum accepted) | 0.000 GWei                            |
|                |                |          | Listener port             | 30303                                    |
|                |                |          | Peer count (all total)    | 50                                       |
|                |                |          | Peer count (light nodes)  | 0                                        |
|                |                |          | Signer account            | 0x4b1FEA49638C0DCb3787895496E247CabF653812 |
+----------------+----------------+----------+---------------------------+------------------------------------------+
```

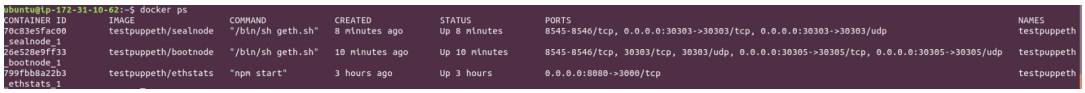
If you go to http://<ip of testpuppeth000>:8080 you should see the sealer nodes appear on Ethstats as well.



Start Mining (Sealing… Validating… Whatever…)

If you look at Ethstats, you'll see the network hasn't started mining yet. To make it mine, log into the bootnode server and look at the docker containers running.

```
ssh ubuntu@<ip of testpuppeth000>
docker ps
```



See the docker container for the sealer? Use that container's "Container ID" in the following command:

```
docker exec -it <container_id> geth attach ipc:/root/.ethereum/geth.ipc
```

This will open a geth console that is a javascript terminal into a local web3 interface for interacting with the node. In the geth console, type the following command:

```
admin.nodeInfo.enode
```

This prints out the enode of this node. This is an identifier used in the node discovery protocol. Copy that value into your notes, and then you'll want to replace the portion that goes [::] with the IP address of this server.

For example, with the enode:

```
> admin.nodeInfo.enode
"enode://fa2ae953a404484c6ac08067834714092129 9e90a8b74570d67687484473be1f9c991d038983245
f2fc27759836a816f5788900bbaec605a12607d125f50961d@[::]:30303"
```

Then put the following into your notes:

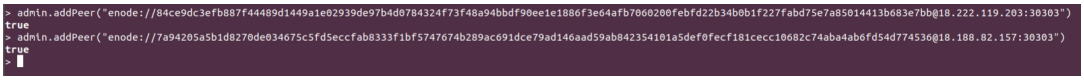
> *sealer0:*
> *"enode://fa2ae953a404484c6ac08067834714092129 9e90a8b74570d67687484473be1f9c991d038983245f2fc27759836a816f5788900bbaec605a12607d125f50961d@18.216.224.119:30303"*

Now log into the other two servers and use the same commands to attach to its geth as you did and take down their enodes in the same way. In the end you should have three enodes in your notes.

> *sealer0:*
> *“enode://fa2ae953a404484c6ac08067834714092129 9e90a8b74570d67687484473be1f9c991d038983245f2fc27759836a816f5788900bbaec605a12607d125f50961d@18.216.224.119:30303”*
> *sealer1:*
> *“enode://84ce9dc3efb887f44489d1449a1e02939de97b4d0784324f73f48a94bbdf90ee1e1886f3e64afb7060200febfd22b34b0b1f227fabd75e7a85014413b683e7bb@18.222.119.203:30303”*
> *sealer2:*
> *“enode://7a94205a5b1d8270de034675c5fd5eccfab8333f1bf5747674b289ac691dce79ad146aad59ab842354101a5def0fecf181cecc10682c74aba4ab6fd54d774536@18.188.82.157:30303”*

In the geth console on each node, use the following command twice, once for each OTHER node’s enode. This will add the other two nodes to this node’s peer list.

```
admin.addPeer("<enode from other server with IP inserted>")
```



Once you have done this same thing so that every node has the enode of the other two nodes added, your nodes should start mining! If you check http://<ip of testpuppeth000>:8080 you’ll see that the network is now producing blocks!

Adding The Rest!

This is pretty fantastic! But there’s still a few more tools you’ll need to get things going. The puppeth tool also comes with:

1. The MyCrypto wallet app
2. A faucet
3. A block explorer (net yet supported on PoA networks, sorry…)

3. ~~A block explorer (not yet supported on Pet Networks, sorry!)~~
4. A dashboard to tie it all together

To wrap up, I'll go through the tools you need, in order they should be installed. All of these will be installed on the same system that Ethstats is, testpuppeth000.

Add A Wallet App

1. Type "4" for "4. Deploy network components"
2. Type "6" for "6. Deploy new network component"
3. Type "5" for "5. Wallet—Browser wallet for quick sends"
4. Type the number corresponding with the IP address for "testpuppeth000"
5. Type in "8081" for the port the wallet listens on
6. Type "n" to prevent port sharing with other services.
7. Store the wallet at "/home/ubuntu/walletapp"
8. The backing node should listen on TCP/IP port 30301
9. The RPC API should listen on port 8544
10. The wallet should be called "testpuppeth_wallet" on the Ethstats page

Add A Faucet App

1. Type "4" for "4. Deploy network components"
2. Type "7" for "7. Deploy new network component"
3. Type "6" for "6. Faucet—Crypto faucet to give away funds"
4. Type the number corresponding with the IP address for "testpuppeth000"
5. Type in "8082" for the port the wallet listens on
6. Type "n" to prevent port sharing with other services.
7. Release 2 ether per request
8. Enforce 5 minute intervals between requests
9. Use the default funding tiers
10. We're going to disable reCaptcha, but the option is available to you.
11. Store the faucet at "/home/ubuntu/faucetapp"
12. The backing node should listen on TCP/IP port 30307
13. The faucet should be called "testpuppeth_faucet_0" on the Ethstats page
14. Paste the JSON of the faucet account's key pasted in to the notes much earlier.
15. Type in the unlock password for this keyfile, stored in your "passfile" file created earlier (the text wont display as you type)
16. Type "y" to permite non-authenticated funding requests.

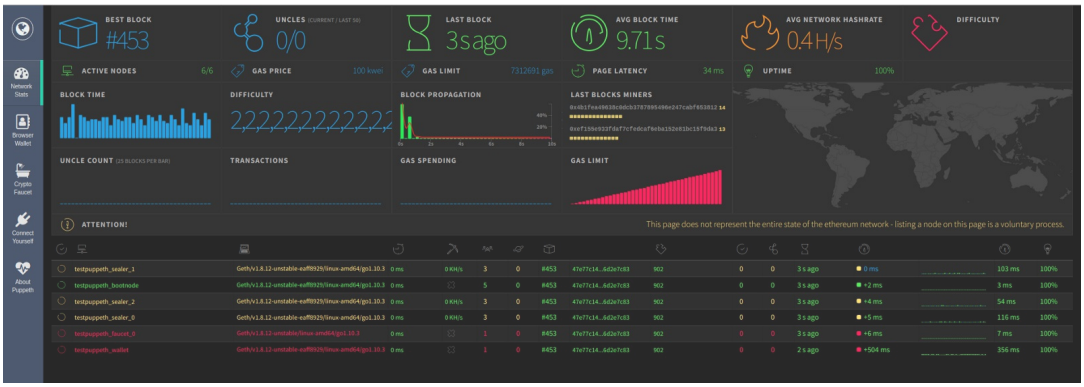
Create The Dashboard

Create The Dashboard

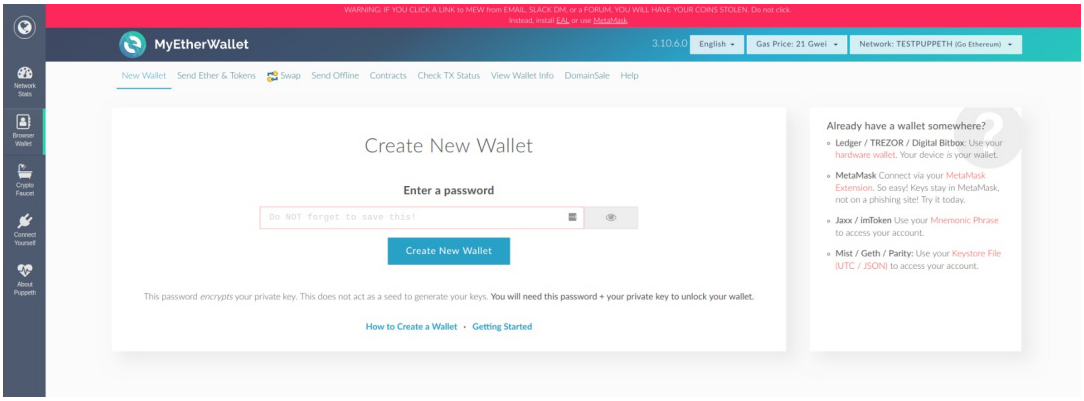
1. Type “4” for “4. Deploy network components”
2. Type “8” for “8. Deploy new network component”
3. Type “7” for “7. Dashboard — Website listing above web-services”
4. Type the number corresponding with the IP address for “testpuppeth000”
5. Type in “8000” for the port the dashboard listens on
6. Type “n” to prevent port sharing with other services.
7. Type “1” to list the local Ethstats service
8. Type “2” to opt-out of an explorer service
9. Type “1” to list the local wallet service
10. Type “1” to list the local faucet service
11. Go with the default for including the Ethstats secret on the dashboard

That’s A Wrap!

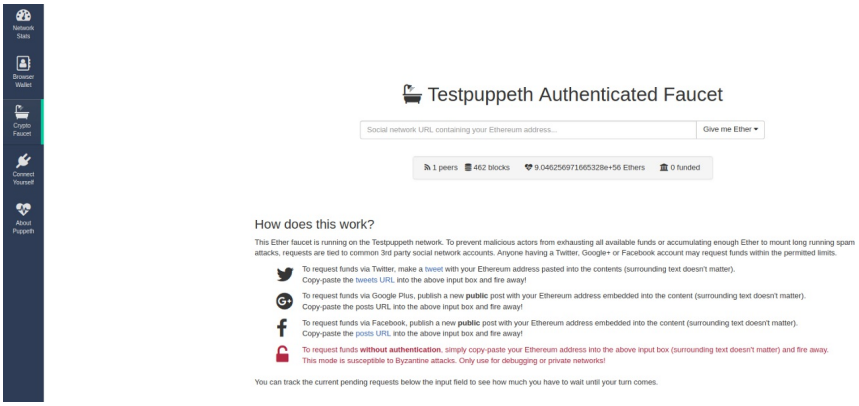
Now if you go to http://<ip of testpuppeth000>:8000 you’ll see your dashboard with Ethstats…



… you’ll be able to access your wallet app…



… and you’ll have access to your faucet app!



I hope this helps someone!

Deployment | Ethereum | Blockchain | Blockchain Technology | Tutorial

Like what you read? Give Collin Cusce a round of applause.

From a quick cheer to a standing ovation, clap to show how much you enjoyed this story.

148 3

Collin Cusce

Medium member since May 2018

Computer Scientist, Futurist, Applied Philosopher. Central focus on decentralization.

Follow

Coinmonks

Coinmonks is a technology focused publication embracing all technologies which have powers to shape our future. Education is our core value. Learn, Build and thrive.

Follow

Never miss a story from **Coinmonks**

GET UPDATES