Hyperledger - Besu

Hyperledger Besu is an Ethereum client designed to be enterprise-friendly for both public and private permissioned network use cases.

▼ How to Run a Besu Node

Prerequisites

Install Docker:

Docker Engine is an open source containerization technology for building and containerizing your applications. Docker Engine acts as a client-server application with:

A server with a long-running daemon process `dockerd`.

APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.

A command line interface (CLI) client `docker`.

OR

Install Besu

MacOS with Homebrew

To install Besu using Homebrew:

```
brew tap hyperledger/besu
brew install hyperledger/besu/besu
# To upgrade an existing Besu installation using Homebrew:
brew upgrade hyperledger/besu/besu
```

To display the Besu version and confirm installation:

```
besu --version
```

Linux/ Unix

Install from packaged binaries:

Download the Besu packaged binaries.

Unpack the downloaded files and change into the besu-[illegible]ory.

Display Besu command line help to confirm installation:

[illegible]

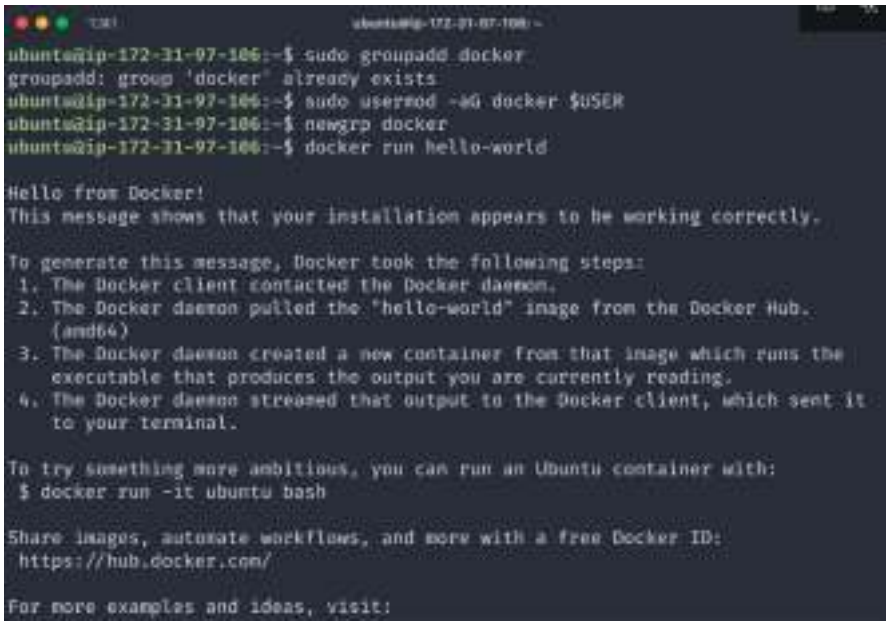
▼ Open up Docker and type

`$ docker --version`

Test whether docker works

[illegible]

The sample output is as follows:



```
ubuntu@ip-172-31-97-106:~$ sudo groupadd docker
groupadd: group 'docker' already exists
ubuntu@ip-172-31-97-106:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-97-106:~$ newgrp docker
ubuntu@ip-172-31-97-106:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
```

▼ Run Besu from a Docker image

Hyperledger Besu provides a Docker image to run a Besu node in a Docker container.

Use this Docker image to run a single Besu node without installing Besu.

Default node for Mainnet

To run a Besu node in a container connected to the Ethereum Mainnet: `docker run hyperledger/besu:latest`

To ensure your image is up to date, pull the latest version again using `docker pull hyperledger/besu:latest.`

Run Besu on local port

To run Besu exposing local ports for access:

```
docker run -p <localportJSON-RPC>:8545 -p <localportWS>:8546 -p <localportP2P>:30303 hyperledger/besu:latest --rpc-http-enabled --rpc-ws-enabled
```

Example:

To enable JSON-RPC HTTP calls to 127.0.0.1:8545 and P2P discovery on 127.0.0.1:13001: `docker run -p 8545:8545 -p 13001:30303 hyperledger/besu:latest --rpc-http-enabled`

Then, docker will run hyperledger besu on docker using local port. The sample output is as follows:

```
ubuntu@ip-172-31-97-106:~$ docker run -p 8545:8545 -p 8546:8546 -p 30303:30303 h
yperledger/besu:latest --rpc-http-enabled --rpc-ws-enabled
Unable to find image 'hyperledger/besu:latest' locally
latest: Pulling from hyperledger/besu
675920708c0b: Pull complete
21a4b5545b6d: Pull complete
10d7c0f64aad: Pull complete
Digest: sha256:ec7086cc797b4ec86cfc24df2ed6d4e598fd5c8ad687df4351a92c7d31214010
Status: Downloaded newer image for hyperledger/besu:latest
2022-09-29 18:04:37.111+00:00 | main | INFO  | Besu | Using LibEthPairings nativ
e alt bn128
2022-09-29 18:04:37.114+00:00 | main | INFO  | Besu | Using the native implement
ation of the signature algorithm
2022-09-29 18:04:37.119+00:00 | main | INFO  | Besu | Using the native implement
ation of the blake2bf algorithm
2022-09-29 18:04:37.126+00:00 | main | INFO  | Besu | Starting Besu version: bes
u/v22.7.4/linux-x86_64/openjdk-java-11
2022-09-29 18:04:37.594+00:00 | main | INFO  | Besu | Engine API authentication
enabled without key file. Expect ephemeral jwt.hex file in datadir
2022-09-29 18:04:37.596+00:00 | main | WARN  | Besu | --graphql-http-host has be
en ignored because --graphql-http-enabled was not defined on the command line.
2022-09-29 18:04:37.598+00:00 | main | INFO  | Besu | Static Nodes file = /opt/b
esu/static-nodes.json
2022-09-29 18:04:37.599+00:00 | main | INFO  | StaticNodesParser | StaticNodes f
ile /opt/besu/static-nodes.json does not exist, no static connections will be cr
```

Screenshot of besu running on local port

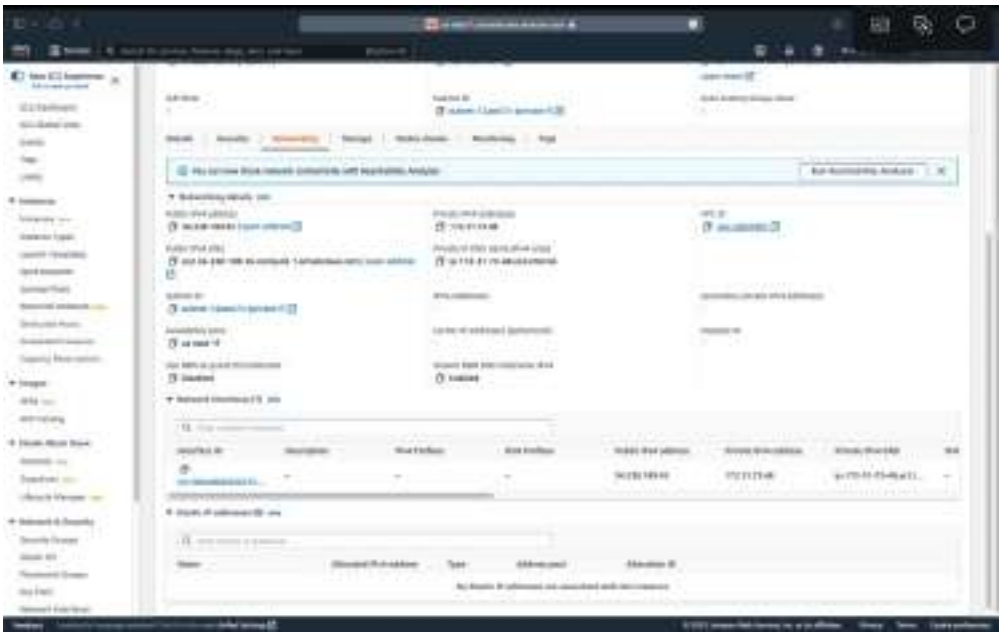
Then, add new environment variable to your computer.
`$ export PATH=/home/ubuntu/hyperledger-besu/besu-22.7.4/bin:$PATH` .

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export PATH=/home/ubuntu/hyperledger-besu/besu-22.7.4/bin:$PATH
~
~
-- INSERT --                                          119,64        Bot
```

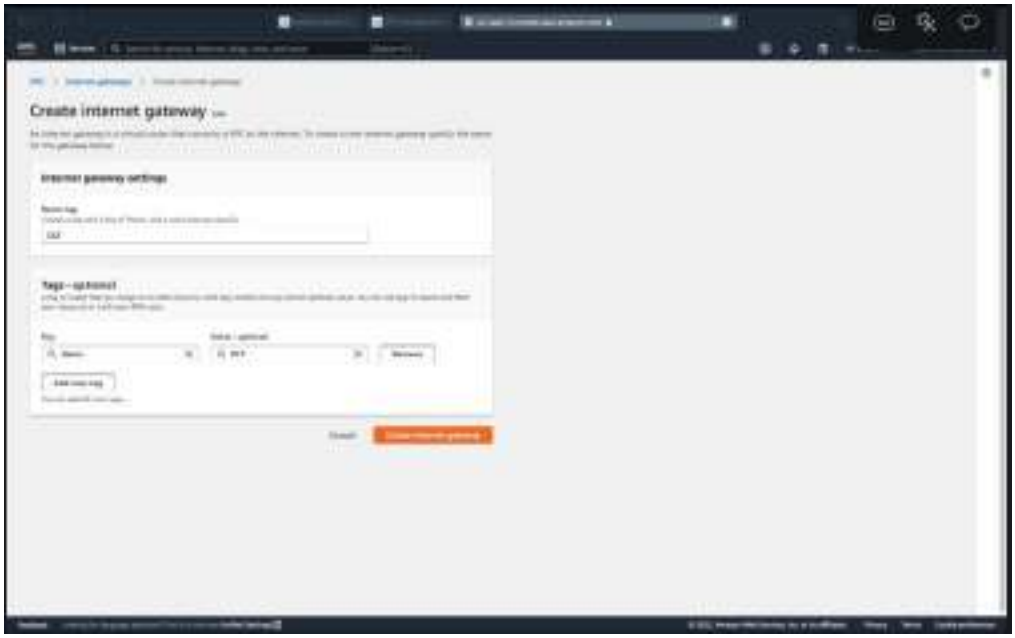
▼ Connecting to AWS

After setting up the EC2 instance at AWS, you can connect the hyperledger besu to AWS.

First, log into the EC2 instance:

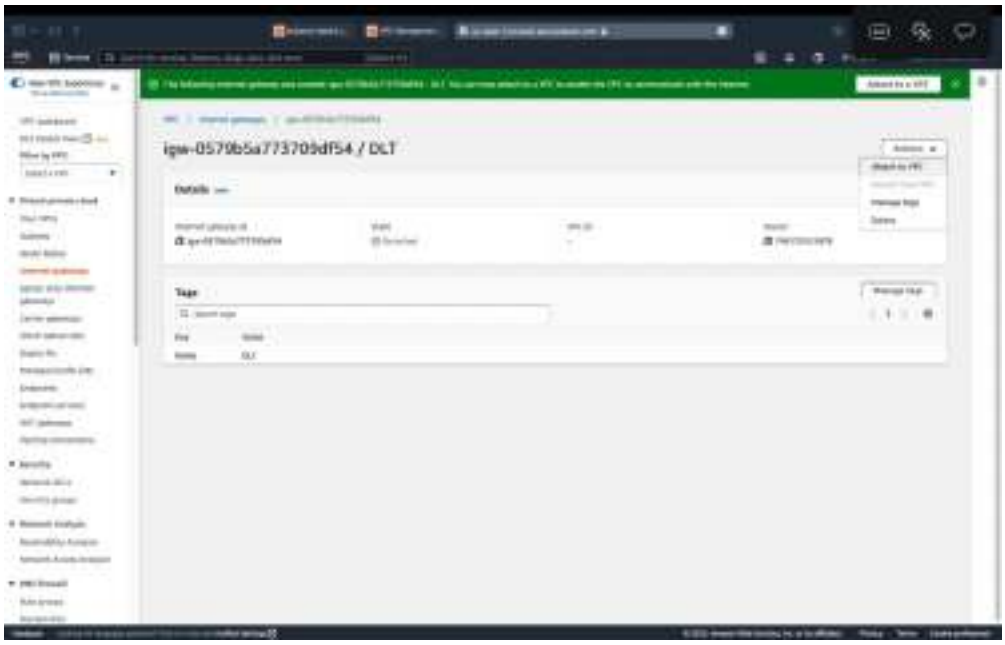


Then, create an internet gateway with name tag (DLT). Click on "Create Internet gateway".

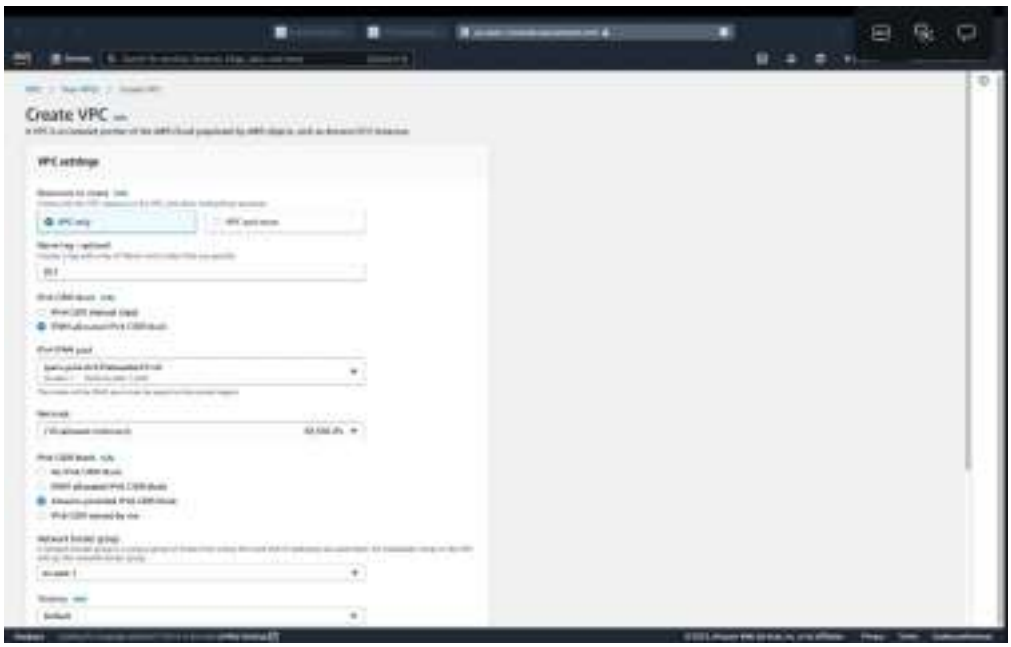


This is the information page after creating the internet gateway. Here, it includes gateway ID, owner, and name tage.

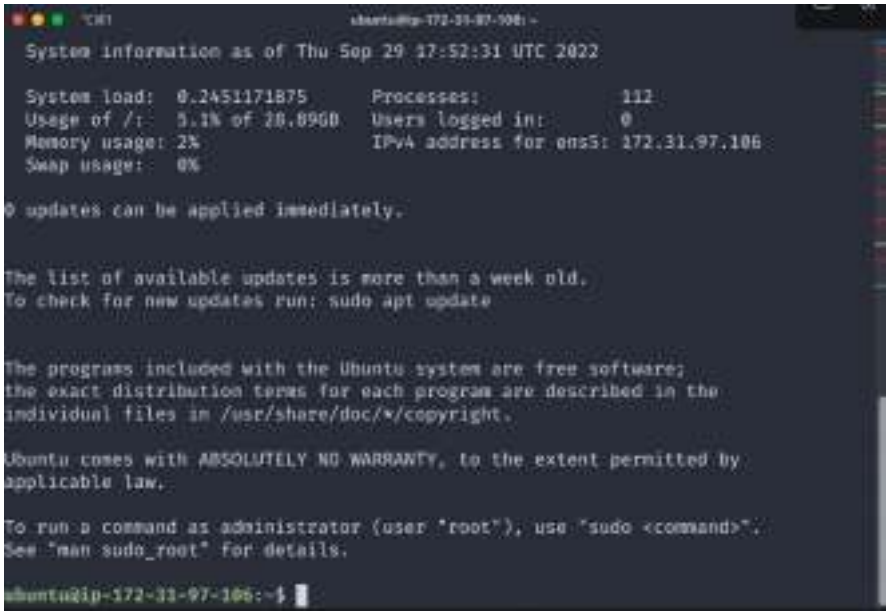
In the "Action" dropdown button, click on "Attch to VPC" button.



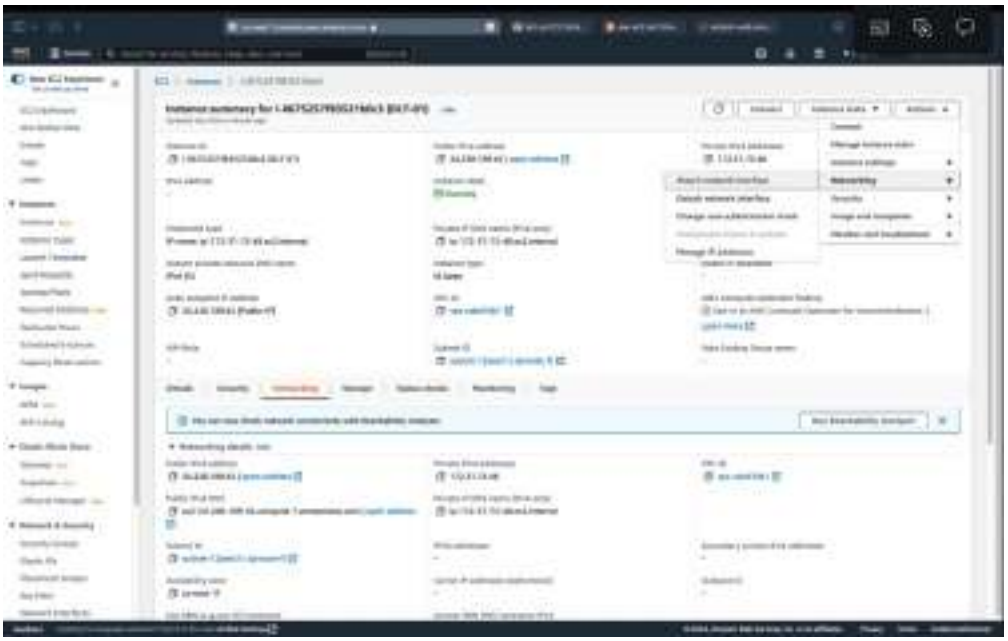
After clicking on "Attach to VPC", here is the page of creating VPC. Follow the options selected below.



After creating the VPC, it will generate system information as below.



Click on "Instance State" to the right corner, then click on "Networking", and "Attch network interface". You can now check network connectivity with reachability analyzer.



Instance Summary

Finally, you successfully connect to AWS instance! Pic above is the instance summary screenshot.

▼ How to run a sample test network

Clone the repository from the `besu-sample-networks` repository:

`git clone https://github.com/PegaSysEng/besu-sample-networks.git` Remember to checkout the **second last** commit.

Start the network

To build the docker images and run the containers, go to the `besu-sample-networks` directory and run:

```
./run.sh
```

The script builds the images, and runs the containers. It also scales the regular node container to four containers to simulate a network with enough peers to synchronize.

When the network proceeds, it will output as following:

Followed by a list of endpoints:

```
***************************************************************
JSON-RPC HTTP service endpoint      : http://localhost:8545
JSON-RPC WebSocket service endpoint : ws://localhost:8546
GraphQL HTTP service endpoint       : http://localhost:8547
Web block explorer address          : http://localhost:25000/
Prometheus address                  : http://localhost:9090/graph
Grafana address                     : http://localhost:3000/d/XE4V0WGZz/besu-overview?orgId=1&refresh=10s&from=now-30m&to=now&var-system
Kibana logs address                 : http://localhost:5601/app/kibana#/discover
***************************************************************
```

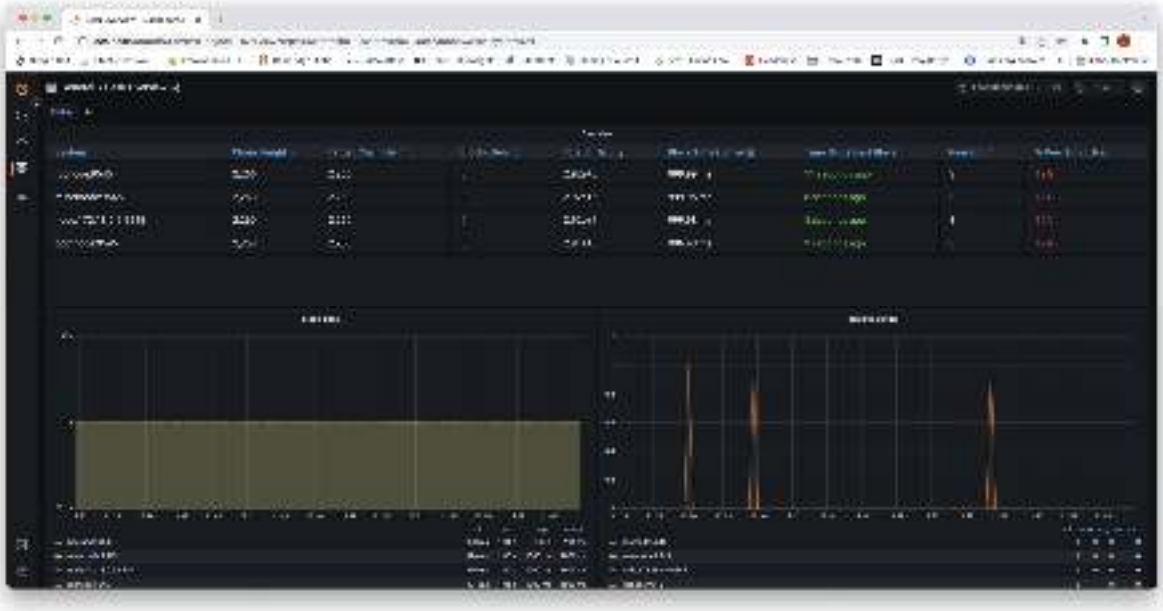
To display the list of endpoints again, run:

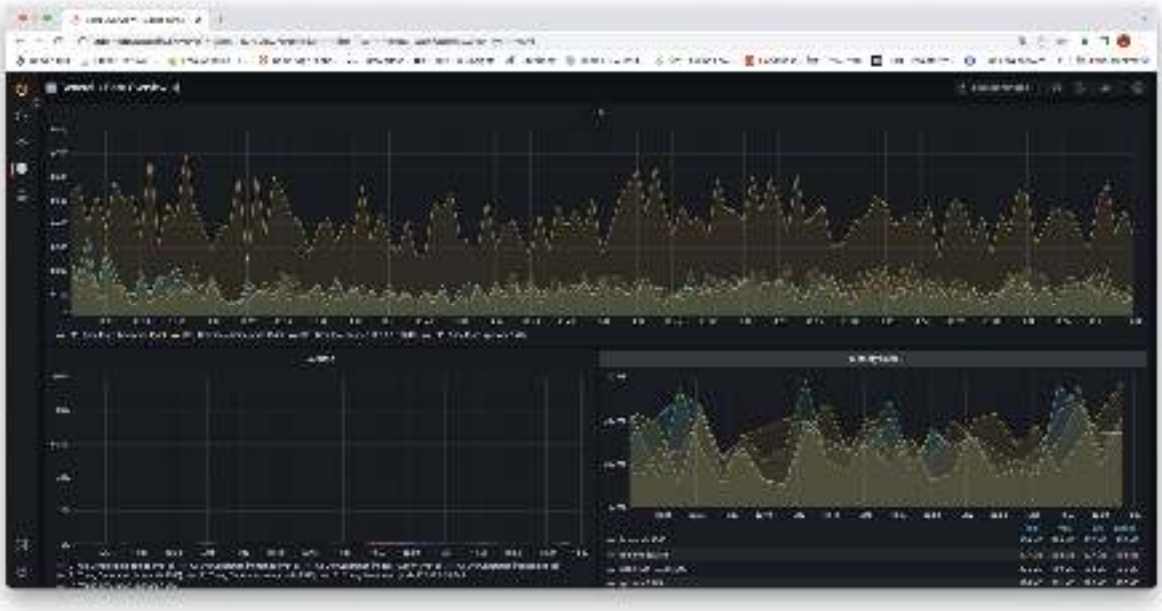
```
./list.sh
```

The whole process created **bootnode, minernode, normal node, rpcnode**. The other names such as grafana, exploreer, prometheus are for nodes monitoring purpose.

The sample network also includes Prometheus and Grafana monitoring tools to let you visualise node health and usage. You can directly access these tools from your browser at the addresses displayed in the endpoint list.

When you go to Grafana address, you will see the grafana dashboard:





You can check the status of running nodes through Docker Desktop as well.

Request the node version

Run the following command from the host shell:

```
curl -X POST --data '{"jsonrpc":"2.0","method":"net_peerCount","params":[],"id":1}' http://localhost:8545
```

Sample output is:

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "0x3"
}
```

Successfully calling this method shows that you can connect to the nodes using RPC.

Request the most recently mined block number

Call

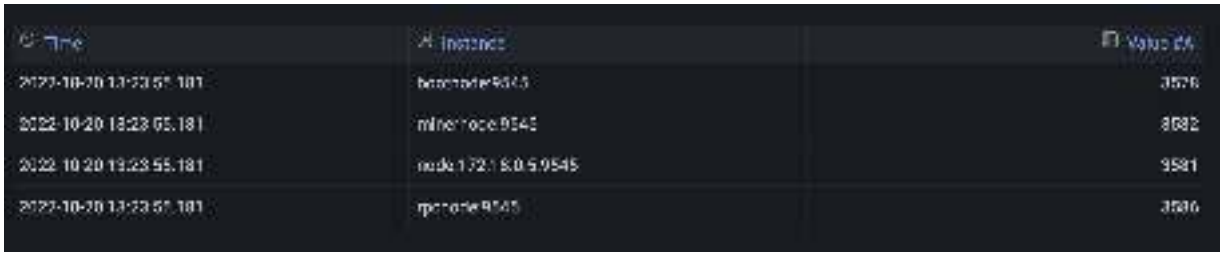
```
curl -X POST --data '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":1}' http://localhost:8545
```

`eth_blockNumber` to retrieve the number of the most recent block:

The result provides the most recently mined block:

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "0xd30"
}
```

The hexadecimal value `0xd30` translates to 3376 in decimal, the number of mined blocks so far. It is consistent with the grafana dashboard.



Check the miner account balance

Call `eth_getBalance` to retrieve the balance of the mining address (coinbase) defined in the miner node:

```
curl -X POST --data '{"jsonrpc":"2.0","method":"eth_getBalance","params":["0xfe3b557e8fb62b89f4916b721be55ceb828dbd73","latest"],"id":1}
```

The result specifies the miner account balance:

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "0x19c6e35ab16a3e00000"
}
```

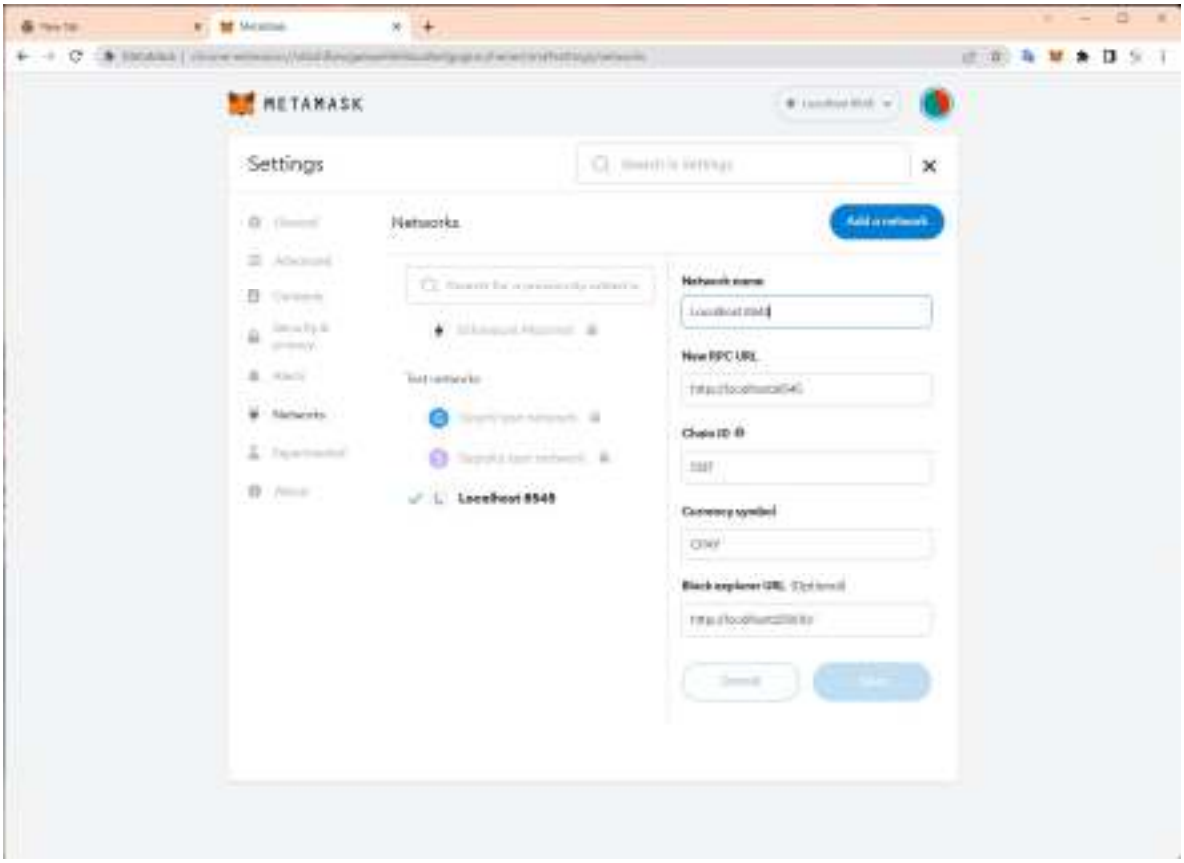
The miner account balance `0x19c6e35ab16a3e00000` = 7608 Ether. Wait a few seconds until there are new mined blocks then call `eth_getBalance` again. The balance increases, meaning the miner address successfully received the mining reward.

▼ Create a transaction using MetaMask

1. Connect to the local network with MetaMask

   (by default it would detect the local network `Localhost 8545` but enter information otherwise)

   1. In the MetaMask network list, select **Custom RPC**.
   2. In the **New RPC URL** field, enter the JSON-RPC HTTP service endpoint displayed when you started the private network.

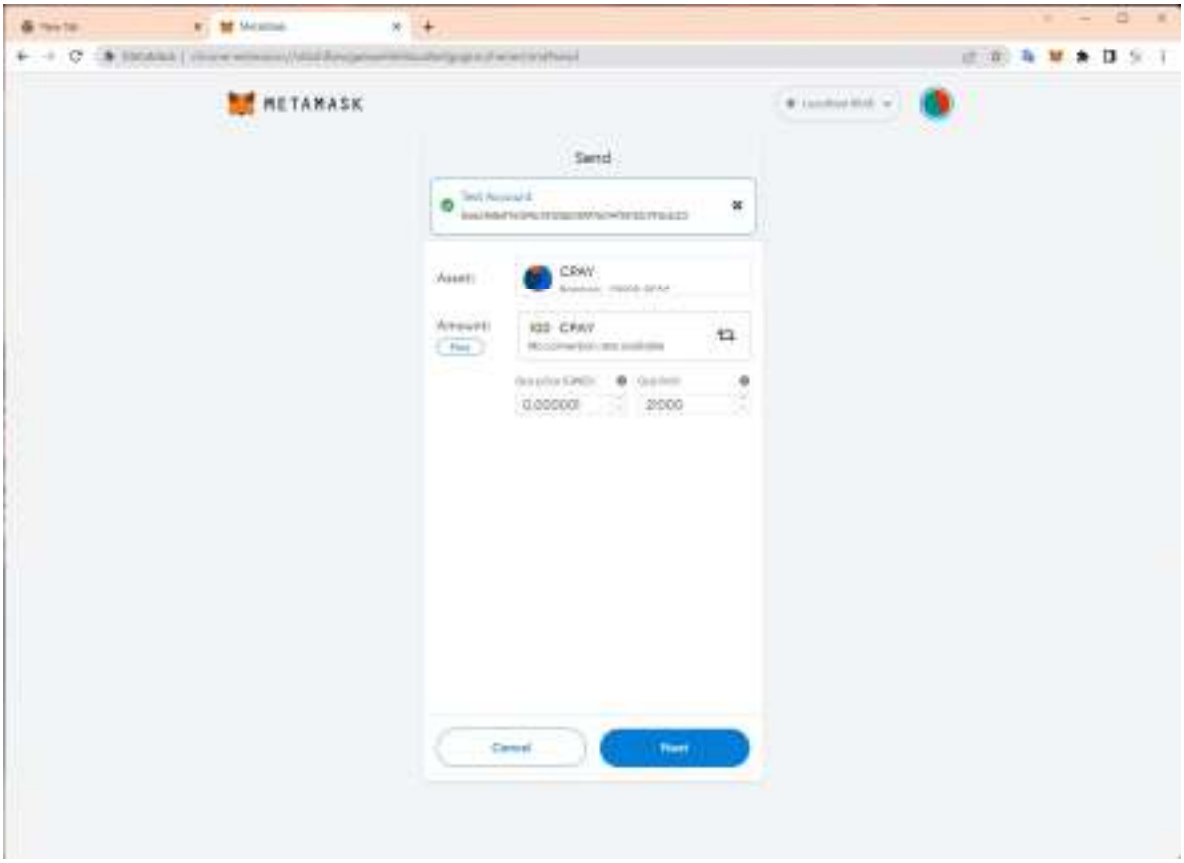


2. Use one of the following accounts with fund to login

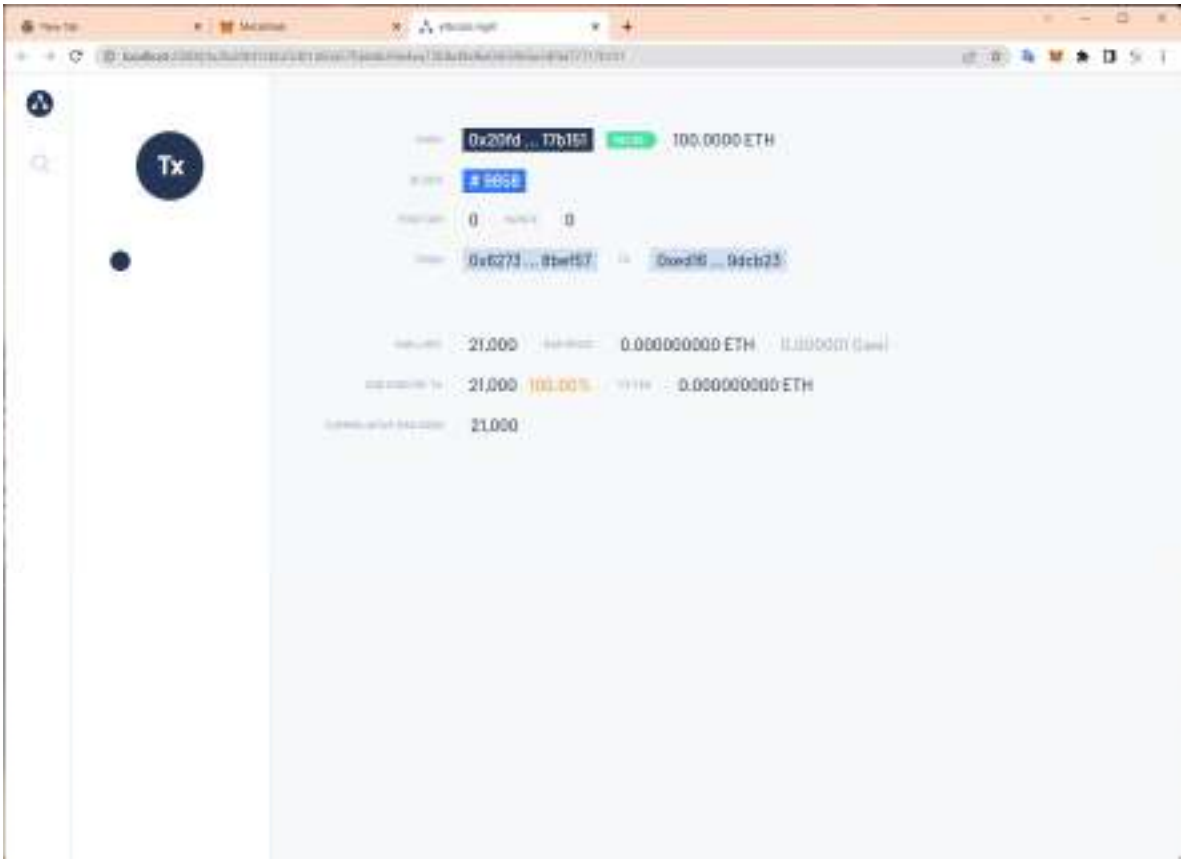
```
Account 1
    Address: 0xfe3b557e8fb62b89f4916b721be55ceb828dbd73
    Private key : 0x8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63
    Initial balance : 0xad78ebc5ac6200000 (200000000000000000000 in decimal)
Account 2
    Address: 0x627306090abaB3A6e1400e9345bC60c78a8BEf57
    Private key : 0xc87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
    Initial balance : 0x9000000000000000000000 (2785365088392105618523029504 in decimal)
Account 3
    Address: 0xf17f52151EbEF6C7334FAD080c5704D77216b732
    Private key : 0xae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
    Initial balance : 0x9000000000000000000000 (2785365088392105618523029504 in decimal)
```

3. Create another account to test transacting the fund to

4. Initiate the transaction



5. The transaction will be confirmed within seconds



The transaction is available on the explorer

▼ How to run a sample application with Hardhat

This is base on the tutorial and sample application that Hardhat provides.

Hardhat Boilerplate Project | Ethereum development environment for professionals by Nomic Foundatio

If you want to get started with your dApp quickly or see what this whole project looks like with a frontend, you can e

our boilerplate repo.

https://hardhat.org/tutorial/boilerplate-proje

Some screenshorts

1. Install the hardhat package

   a. install node.js

```
sudo apt update
sudo apt install curl git
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
```

   b.

2. clone the sample application repo

3. Enter network information in the config file
   a. open `hardhat.config.js` in a text editor
   b. insert the besu network information in the `module.export` block

```
module.exports = {
  solidity: "0.8.9",
  networks: {
    hardhat: {
      chainId: 1337 // We set 1337 to make interacting with MetaMask simpler
    },
    besu:{
      url:"http://localhost:8545",
      accounts:[GOERLI_PRIVATE_KEY],
      chainId: 1337
    },
...
}
```

4. start local besu network
   a. start docker desktop
   b. use the local test network aforementioned

```
#1 go to the sample network directory
cd besu-sample-networks
#2 resume the network if used before (use Git Bash or Powershell to run)
./resume.sh
  # for the first time use:
  ./start.sh
```

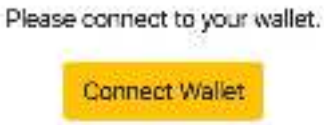
5. install dependencies and deploy the contract to besu
6. start and use the `frontend` web app
   a. go to `frontend` folder and start with command

```
npm run start
```

A browser will be opened, please make sure that you have Metamask installed.

Or go to `http://localhost:3000/`

   b. Connect to Metamask



c. get tokens from faucet


```
$ npx hardhat --network localhost faucet 0x15ed2c3b30251a128eb852c03871cecd09e8fae6
# Output:
Transferred 1 ETH and 100 tokens to 0x15ed2c3b30251a128eb852c03871cecd09e8fae6
```

d. transfer tokens to another wallet

7. Finished.

a. press `ctrl+c` to stop the web application

b. run `./stop.sh` in the `besu_sample_network` folder to stop the network