Private networks

This guide will walk you through usingKurtosisethereum-package to spin up a private, proof-of-stake (PoS) Ethereum devnet with three full Ethereum nodes locally over Docker. At the end of the guide, you will learn how to scale up your testnet on Kubernetes as well as enable optional services for your local testnet, such as network observability tools (e.g., Grafana, Prometheus) and Flashbot'smev-boost infrastructure to simulate MEV workflows.

Step 1: Prerequisites

Before you begin, ensure you haveKurtosis CLI andDocker installed. Theethereum-package is a Kurtosis environment definition known as apackage . For more info about Kurtosis, see theKurtosis docs .

Step 2: Configure your network

Next, in your home directory, create a file with the namenetwork_params.json and populate it with the following contents:

network_params.json { "participants" :

[{ "el_client_type" :

"nethermind" , "el_client_image" :

"nethermind/nethermind:latest" , "cl_client_type" :

"lighthouse" , "cl_client_image" :

"sigp/lighthouse:latest" , "count" :

1 } , { "el_client_type" :

"nethermind" , "el_client_image" :

"nethermind/nethermind:latest" , "cl_client_type" :

"teku" , "cl_client_image" :

"consensys/teku:latest" , "count" :

1 } , { "el_client_type" :

"nethermind" , "el_client_image" :

"nethermind/nethermind:latest" , "cl_client_type" :

"lodestar" , "cl_client_image" :

"chainsafe/lodestar:next" , "count" :

1 }] , "mev_type" :

"None" , "launch_additional_services" :

false } As you can see above, you have effectively created a network configuration file that Kurtosis will use to pass in the necessary parameters at runtime for your network. Notice that theparticipant key describes the execution and consensus client pairing desired for each full node and how many nodes of that type to instantiate.

There are many other configurations and flags you can use, including metrics and observability tools (e.g., Grafana, Prometheus, etc). For all supported options, see theethereum-package configuration .

Step 3: Deploy

Finally, once you have saved thenetwork_params.json file, it is time to deploy the private net:

kurtosis run github.com/kurtosis-tech/ethereum-package " (cat ~/network_params.json) " Kurtosis will use theethereum-package environment definition and your custom network configuration (defined innetwork_params.json) to spin up your network. Kurtosis will first spin up anenclave (i.e., an ephemeral, isolated environment) and begin to configure and instantiate the nodes in your network. In the end, Kurtosis will print the services running in your enclave that form your private testnet alongside all the container ports and files that were generated and used to start up the private testnet.

Here is a sample output:

```
INFO[2023-09-01T16:10:45-04:00] ================================================== INFO[2023-09-
01T16:10:45-04:00] || Created enclave: timid-knoll || INFO[2023-09-01T16:10:45-04:00]
================================================== Name: timid-knoll UUID: 939dfb5d59b0 Status:
RUNNING Creation Time: Fri, 01 Sep 2023 16:08:57 EDT

======================================== Files Artifacts ========================================
UUID Name a876b06035b7 1-lighthouse-nethermind-0-63 87955ef69845 2-teku-nethermind-64-127 4f77377da494 3-
lodestar-nethermind-128-191 9734313101e3 cl-genesis-data 4164ed5c594c el-genesis-data a49a3d2774b5 genesis-
generation-config-cl 16fcc4f96236 genesis-generation-config-el 5fc72346f646 geth-prefunded-keys 96ae153a0b51 prysm-
password

========================================= User Services
========================================= UUID Name Ports Status f369802ad2ae cl-1-lighthouse-nethermind
http: 4000/tcp -> http://127.0.0.1:49894 RUNNING metrics: 5054/tcp -> http://127.0.0.1:49892 tcp-discovery: 9000/tcp ->
127.0.0.1:49893 udp-discovery: 9000/udp -> 127.0.0.1:64949 5e14eb26ef45 cl-1-lighthouse-nethermind-validator http:
5042/tcp -> 127.0.0.1:49895 RUNNING metrics: 5064/tcp -> http://127.0.0.1:49896 fed533d0e143 cl-2-teku-nethermind http:
4000/tcp -> 127.0.0.1:49899 RUNNING metrics: 8008/tcp -> 127.0.0.1:49897 tcp-discovery: 9000/tcp -> 127.0.0.1:49898
udp-discovery: 9000/udp -> 127.0.0.1:55521 69cd832de246 cl-3-lodestar-nethermind http: 4000/tcp -> 127.0.0.1:49903
RUNNING metrics: 8008/tcp -> 127.0.0.1:49901 tcp-discovery: 9000/tcp -> 127.0.0.1:49902 udp-discovery: 9000/udp ->
127.0.0.1:50507 75e3eec0c7d1 cl-3-lodestar-nethermind-validator metrics: 8008/tcp -> 127.0.0.1:49904 RUNNING
e10c3f07e0e0 el-1-nethermind-lighthouse engine-rpc: 8551/tcp -> 127.0.0.1:49872 RUNNING rpc: 8545/tcp ->
127.0.0.1:49870 tcp-discovery: 30303/tcp -> 127.0.0.1:49869 udp-discovery: 30303/udp -> 127.0.0.1:64508 ws: 8546/tcp ->
127.0.0.1:49871 c6a28d3136fe el-2-nethermind-teku engine-rpc: 8551/tcp -> 127.0.0.1:49873 RUNNING rpc: 8545/tcp ->
127.0.0.1:49875 tcp-discovery: 30303/tcp -> 127.0.0.1:49874 udp-discovery: 30303/udp -> 127.0.0.1:52495 ws: 8546/tcp ->
127.0.0.1:49876 2fae3b3c41d3 el-3-nethermind-lodestar engine-rpc: 8551/tcp -> 127.0.0.1:49890 RUNNING rpc: 8545/tcp -
> 127.0.0.1:49888 tcp-discovery: 30303/tcp -> 127.0.0.1:49891 udp-discovery: 30303/udp -> 127.0.0.1:62119 ws: 8546/tcp -
> 127.0.0.1:49889 403cafe8416e prelaunch-data-generator-cl-genesis-data RUNNING ebea71008cf4 prelaunch-data-
generator-el-genesis-data RUNNING
```

And that is it! You now have a 3-node, private Ethereum devnet with Nethermind/Lodestar, Nethermind/Teku, and Nethermind/Lighthouse execution and consensus client combinations.

Notice how, at the end, Kurtosis will print out the contents of your enclave, which includes both the variousfiles artifacts and services that form your network. Kurtosis also maps the container ports to ephemeral local ports on your machine.

Genesis data was generated using thisgenesis-generator to bootstrap the execution and consensus clients for each node. The end result will be a private testnet with nodes deployed as Docker containers in an ephemeral, isolated environment on your machine called anenclave .

Kurtosis packages are modular, reproducible, and will work over Docker or Kubernetes. Read on to learn about additional services and configurations you may want to add to your private network.

Step 4: Optional workflows

This section briefly covers some optional configurations for your private devnet that are commonly used for validating and testing node-level behavior.

Simulating MEV workflows withmev-boost

Theethereum-package can simulate out-of-protocol Proposer Builder Separation (PBS) workflows using Flashbot'smev-boost infrastructure. With a single flag, you can configure your network's validators to be instantiated withmev-boost and be registered with a relayer to receive payloads from builders.

To enable this in yournetwork_params.json file, set"mev_type": to"full" or"mock" .

To learn more about how themev-boost infrastructure works with your private network, check outthis guide .

Observability tools

Theethereum-package comes out of the box with a few observability tools, including:

- A Grafana and Prometheus instance
- Abeacon metrics gazer service
- to collect network-wide participation metrics
- AJSON-RPC Snooper
- to log responses & requests between the execution engine API and the consensus client

To add your own custom Grafana dashboard template, fork theethereum-package repository and add your configuratiorhere .

Deploying on Kubernetes

As mentioned earlier, Kurtosis packages (i.e. environment definitions) are portable and will work the same way over Docker or on Kubernetes. Should you require a larger scale devnet, Kurtosis can deploy any package, including theethereum-package on Kubernetes, seeRunning Kurtosis in Kubernetes .

Questions and feedback

If you need help with your Nethermind full node in the private devnet, please don't hesitate to contact the Kurtosis team onGitHub orDiscord .