Running a Node With Docker

UsingDocker(opens in a new tab) is an easy way to run an OP Mainnet node. This tutorial will walk you through the process of usingsimple-optimism-node (opens in a new tab) to run an OP Mainnet or OP Sepolia node using Docker.simple-optimism-node also provides useful tools like a monitoring dashboard and health checking software. Although less flexible thanrunning a node from source or building your own Docker setup, this is a great way to quickly get started with OP Mainnet.

What's Included

simple-optimism-node includes all the basic components to run an OP Mainnet or OP Sepolia node. It also includes several additional services to monitor the health of your node and the health of the network you're connected to. See theWhat's Included(opens in a new tab) section of thesimple-optimism-node README for more details.

Dependencies

- Docker(opens in a new tab)
- Docker Compose(opens in a new tab)

Setup

Clone thesimple-optimism-node repository to get started.

git

clone

https://github.com/smartcontracts/simple-optimism-node.git cd

simple-optimism-node

Configuration

Configuration forsimple-optimism-node is handled through environment variables inside of an.env file.

Create an .env file

The repository includes a sample environment variable file located at.env.example that you can copy and modify to get started. Make a copy of this file and name it.env .

cp

.env.example

.env

Configure the .env file

Open the.env file in your favorite text editor. Set the variables inside of theREQUIRED (BEDROCK) section of the file according to the guide below:

NETWORK_NAME - Choose which Optimism network layer you want to operate on:

- op-mainnet
-
    - Optimism Mainnet
- op-sepolia
-
    - Optimism Sepolia (Testnet)

NODE_TYPE - Choose the type of node you want to run:

- full
- (Full node) - A Full node contains a few recent blocks without historical states.
- archive
- (Archive node) - An Archive node stores the complete history of the blockchain, including historical states.

OP_NODE__RPC_ENDPOINT - Specify the endpoint for the RPC of Layer 1 (e.g., Ethereum mainnet). For instance, you can use the free plan of Alchemy for the Ethereum mainnet.

OP_NODE__RPC_TYPE - Specify the service provider for the RPC endpoint you've chosen in the previous step. The available options are:

- alchemy
-
    - Alchemy
- quicknode
-
    - Quicknode (ETH only)
- erigon
-
    - Erigon
- basic
-
    - Other providers

HEALTHCHECK__REFERENCE_RPC_PROVIDER - Specify the public RPC endpoint for Layer 2 network you want to operate on for healthcheck. For instance:

- Optimism Mainnet
- -https://mainnet.optimism.io(opens in a new tab)
- Optimism Sepolia
- -https://sepolia.optimism.io(opens in a new tab)

You can also set the variables inside of theREQUIRED (LEGACY) section of the file if you wish to run an archival OP Mainnet node. Refer to theOP Mainnet only configurations(opens in a new tab)section ofsimple-optimism-node readme for more information about running legacy OP Mainnet software alongside your OP Mainnet node.

Run the Node

Once you've configured your.env file, you can run the node using Docker Compose. The following command will start the node in the background.

docker

compose

up

-d

--build

Upgrade the Node

Pull the latest updates from GitHub, Docker Hub and rebuild the container.

git

pull docker

compose

pull docker

compose

up

-d

--build

--force-recreate

Operating the Node

You can use Docker Compose to interact with the node and manage the various containers that you started. Refer to theOperating the Node(opens in a new tab) section of thesimple-optimism-node README for more information.

Checking Node Status

simple-optimism-node includes a monitoring dashboard that you can use to check the status of your node. You can access the dashboard by visitinghttp://localhost:3000 in your browser. Refer to theGrafana dashboard(opens in a new tab) section of thesimple-optimism-node README for more information.

Another way to check the node syncing progress is to run./progress.sh , which will print output showing the number of blocks per second and the estimated time until synchronization is completed.

./progress.sh Chain ID: 10 Please wait Blocks per minute: ... Hours until sync completed: ...

Overview Building a Node from Source