Running a Local Development Environment

This guide is currently under active development. If you run into any issues, please open an issue onGithub(opens in a new tab) . This tutorial isdesigned for developers who want to learn about the OP Stack by spinning up a local OP Stack devnet. You'll perform the full deployment process, andyou'll end up with your very own OP Stack devnet .

It's useful to understand what each of these components does before you start deploying your chain. To learn about the different components please read thedeployment overview page .

You can use this devnet to experiment and perform tests, or you can choose to modify the chain to adapt it to your own needs.The OP Stack is free and open source software licensed entirely under the MIT license . You don't need permission from anyone to modify or deploy the stack in any configuration you want.

⚠ Modifications to the OP Stack may prevent a chain from being able to benefit from aspects of theOptimism Superchain . Make sure to check out theSuperchain Explainer to learn more.

Installing Dependencies

Dependency Version Version Check Commanddocker(opens in a new tab) ^27 docker --versionkurtosis(opens in a new tab) ^1.3.0 kurtosis version

Notes on Specific Dependencies

docker

We recommend using the latest version of Docker on Linux, orOrbStack(opens in a new tab) (a drop-in replacement for Docker Desktop) on OSX.

kurtosis

Kurtosis is a tool for packaging and deploying containerized services. It's used in this tutorial to automatically deploy your devnet in an isolated environment.

Configure your network

Now that you've installed all the necessary dependencies, you can start configuring your network. The Kurtosis package accepts a YAML file which configures how many network participants there are, what kind of software they're running, and the network's topology. An example YAML file is below:

optimism_package : chains :

you can define multiple L2s, which will be deployed against the same L1 as a single Superchain

- participants :

each participant is a node in the network. here we've defined two, one running op-geth and one running op-reth

- el_type :

op-geth

this node will be the sequencer since it's first in the list

- el_type :

op-reth network_params : name :

rollup-1

can be anything as long as it is unique

network_id :

12345

can be anything as long as it is unique

Save the above configuration to a file. For the rest of this tutorial, we'll assume you've saved it tonetwork-config.yaml .

Start your network

Now that you've configured your network, you can start it up using the Kurtosis CLI. Run the command below:

kurtosis

run

github.com/ethpandaops/optimism-package

--args-file

./network-config.yaml This command will start up your network and deploy the OP Stack based on the configuration you created. The command will produce a lot of output and will take about five minutes to complete. Once it's done, you'll see a message that looks like the one below:

INFO[2024-09-23T00:31:29-06:00] ==========
INFO[2024-09-23T00:31:29-06:00] || Created enclave: blue-marsh ||
INFO[2024-09-23T00:31:29-06:00] ==========
Name: blue-marsh
UUID: 91af529557cb
Status: RUNNING
Creation Time: Mon, 23 Sep 2024 00:29:58 MDT
Flags:

========== Files Artifacts ==========
UUID Name
a5824b041b28 1-lighthouse-geth-0-63-0
f7c0e13e9871 el_cl_genesis_data
bfa022049aea final-genesis-timestamp
0b5c53e3940f genesis-el-cl-env-file
46a78cc34966 genesis_validators_root
038ad1a753ed jwt_file
4fbc4bde03c2 keymanager_file
c36887606978 op-deployer-configs
d638c3222e56 op-deployer-fund-script
b02f20c287ac op_jwt_filerollup-1
655d57862785 prysm-password
28203054f5ec validator-ranges

========== User Services ==========
UUID Name Ports Status
29643e475cb7 cl-1-lighthouse-geth http: 4000/tcp -> http://127.0.0.1:33639 RUNNING metrics: 5054/tcp -> http://127.0.0.1:33640 tcp-discovery: 9000/tcp -> 127.0.0.1:33641 udp-discovery: 9000/udp -> 127.0.0.1:32920
e7dfdc2588ae el-1-geth-lighthouse engine-rpc: 8551/tcp -> 127.0.0.1:33636 RUNNING metrics: 9001/tcp -> http://127.0.0.1:33637 rpc: 8545/tcp -> 127.0.0.1:33634 tcp-discovery: 30303/tcp -> 127.0.0.1:33638 udp-discovery: 30303/udp -> 127.0.0.1:32919 ws: 8546/tcp -> 127.0.0.1:33635
5ff43094ccc3 op-batcher-rollup-1 http: 8548/tcp -> http://127.0.0.1:33650 RUNNING
aa30d376acc9 op-cl-1-op-node-op-geth-rollup-1 http: 8547/tcp -> http://127.0.0.1:33648 RUNNING tcp-discovery: 9003/tcp -> 127.0.0.1:33649 udp-discovery: 9003/udp -> 127.0.0.1:32922
af4abdbbe939 op-el-1-op-geth-op-node-rollup-1 engine-rpc: 8551/tcp -> 127.0.0.1:33645 RUNNING metrics: 9001/tcp -> 127.0.0.1:33646 rpc: 8545/tcp -> http://127.0.0.1:33643 tcp-discovery: 30303/tcp -> 127.0.0.1:33647 udp-discovery: 30303/udp -> 127.0.0.1:32921 ws: 8546/tcp -> 127.0.0.1:33644
578ee2b5bfe7 validator-key-generation-cl-validator-keystore RUNNING
aa69f73e96c1 vc-1-geth-lighthouse metrics: 8080/tcp -> http://127.0.0.1:33642 RUNNING
Also take note of the last log line above this message, which contains the address of the standard bridge. You'll need this address to deposit funds on your L2.

This might look complicated, but it's just a list of the services that were started up by Kurtosis. For each service, you can see:

- The enclave name, which identifies the services you just deployed within Kurtosis. The enclave is an isolated environment
- that runs your devnet.
- The service's name, which you can use with the Kurtosis CLI to view its logs and interact with it.
- The service's ports and addresses, which you can use to connect to the service.

At this point your chain is up and running. Let's move on to the next section to learn how to interact with it.

Interact with your network

You now have a fully functioning OP Stack Rollup. You can connect your wallet to this chain the same way you'd connect your wallet to any other EVM chain. You can find your node's RPC URL by runningkurtosis enclave inspect . Your enclave name is outputted at the end of thekurtosis run command above. The RPC url is therpc port name in any of the execution client services identified byop-el .

Depositing funds onto your network

Your network was configured to pre-fund development addresses using thetest test test test test test test test test test test junk mnemonic. To get ETH onto your L2, you import one of the private keys from that mnemonic into your favorite wallet or use a CLI tool likecast . For the purposes of this tutorial, we'll usecast and assume you want to use the first address generated by that mnemonic.

To move ETH onto your L2, run the following command. Make sure to replace the values in angle brackets with real values:

cast

send

--mnemonic

'test test test test test test test test test test test junk'

--mnemonic-path

"m/44'/60'/0'/0/0" \ --to

""

--amount

"eth"

--rpc-url

"http://127.0.0.1:" Wait ~30 seconds, then check your balance on L2 by running the following command:

export ETH_RPC_URL = "http://127.0.0.1:" export ADDRESS = "0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266" cast

balance

"ADDRESS" Your balance should match the amount you sent.

See Your Rollup in Action

You can interact with your Rollup the same way you'd interact with any other EVM chain. Send some transactions, deploy some contracts, and see what happens!

Next Steps

- Check out theprotocol specs(opens in a new tab)
- for more detail about the rollup protocol.
- If you run into any problems, please visit theChain Operators Troubleshooting Guide
- orfile an issue(opens in a new tab)
- for help.