Running a Local Development Environment

This tutorial teaches you how to setup and run a local development environment of the entire OP Mainnet system.

What is this?

A development environment is a local installation of the entire OP Mainnet system. Our default development environment includes both L1 and L2 development nodes. Running the OP Mainnet environment locally is a great way to test your code and see how your contracts will behave on OP Mainnet before you graduate to a testnet deployment (and eventually a mainnet deployment).

Alternatively, you can get a hosted development node fromany of these providers .

Do I need this?

We generally recommend using the local development environment if your application falls into one of the following categories:

1. You're building contracts on both OP Mainnet and Ethereum that need to interact with one another.
2. The local development environment is a great way to quickly test interactions between L1 and L2. The OP Mainnet and test networks have a communication delay between L1 and L2 that can make testing slow during the early stages of development.
3. You're building an application that might be subject to one of the fewdifferences between Ethereum and OP Mainnet
4. .
5. Although OP Mainnet isEVM equivalent(opens in a new tab)
6. , it's not exactly the same as Ethereum. If you're building an application that might be subject to one of these differences, you should use the local development environment to double check that everything is running as expected. You might otherwise have unexpected issues when you move to testnet. We strongly recommend reviewing these differences carefully to see if you might fall into this category.

However, not everyone will need to use the local development environment. OP Mainnet isEVM equivalent(opens in a new tab) , which means that OP Mainnet looks almost exactly like Ethereum under the hood. If you don't fall into one of the above categories, you can probably get away with simply relying on existing testing tools like Foundry or Hardhat. If you don't know whether or not you should be using the development environment, feel free to hop into thedeveloper support forum(opens in a new tab) . Someone nice will help you out!

Before You Begin

The Optimism monorepo includesa devnode setup you can use(opens in a new tab).

Installation

First, make sure these components are installed. Note that the command line directions were verified under Ubuntu 22.04 LTS. Other OSes or versions may use different tools.

Install the command line utilitiesmake

andjq

sudo

apt

install

-y

make

jq

InstallGo programming language(opens in a new tab)

sudo

apt

update wget

https://go.dev/dl/go1.21.5.linux-amd64.tar.gz tar

xvzf

go1.21.5.linux-amd64.tar.gz sudo

cp

go/bin/go

/usr/bin/go sudo

mv

go

/usr/lib echo

export

GOROOT=/usr/lib/go

~/.bashrc .

~/.bashrc

Clone the Optimism monorepo

git

clone

https://github.com/ethereum-optimism/optimism.git cd

optimism

Install Foundry using monorepo tooling

The following command will install Foundry if you don't already have it on your system.

pnpm

install:foundry If you do already have Foundry installed, make sure to run the following to verify your installed version will work with the monorepo tooling as expected.

pnpm

check:foundry If you need to, you can run the following to update Foundry to the expected version.

pnpm

update:foundry

Operations

- To start, run (in the root directory of the monorepo)make devnet-up
- .
- The first time it runs it will be relatively slow because it needs to download the images, after that it will be faster.
- To stop, run (in the root directory of the monorepo)make devnet-down
- .
- To clean everything, run (in the root directory of the monorepo)make devnet-clean
- .

Additional Info

- The monorepo includes the L1 contract addresses(opens in a new tab)
- .
- The L2 contract addresses are, of course, the standard ones.
- There are some differences between the development node and the real world (a.k.a. Ethereum mainnet and OP

Mainnet):
- Parameter
- Real-world
- Devnode
- L1 chain ID
- 1
- 900
- L2 chain ID
- 10
- 901
- Time between L1 blocks (in seconds)
- 12
- 3
- Pre-Funded Dev Accounts
- By default, the same 10 dev accounts that are pre-funded using Anvil or HardHat are also pre-funded on both L1 and L2:
-
- These private keys are common knowledge, you shouldnot
- use them on any network other than this dev network.
- Using these private keys on mainnet, or even a testnet, will most likely resultin a loss of funds
- .
- Address
- Private Key
- Balance
- 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
- 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80
- 10,000 ETH
- 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
- 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d
- 10,000 ETH
- 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
- 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a
- 10,000 ETH
- 0x90F79bf6EB2c4f870365E785982E1f101E93b906
- 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6
- 10,000 ETH
- 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65
- 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a
- 10,000 ETH
- 0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc
- 0x8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba
- 10,000 ETH
- 0x976EA74026E726554dB657fA54763abd0C3a0aa9
- 0x92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e
- 10,000 ETH
- 0x14dC79964da2C08b23698B3D3cc7Ca32193d9955
- 0x4bbbf85ce3377467afe5d46f804f221813b2bb87f24d81f60f1fcdbf7cbf4356
- 10,000 ETH
- 0x23618e81E3f5cdF7f54C3d65f7FBc0aBf5B21E8f
- 0xdbda1821b80551c9d65939329250298aa3472ba22feea921c0cf5d620ea67b97
- 10,000 ETH
- 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720
- 0x2a871d0798f97d79848a013d4936a73bf4cc922c825d33c1cf7073dff6d409c6
- 10,000 ETH
-
    - Mnemonic:test test test test test test test test test test test junk
-
    - Derivation path:m/44'/60'/0'/0/