Optimism Box¶

Table of contents generated with markdown-toc

This Truffle Optimism Box provides you with the boilerplate structure necessary to start coding for Optimism's Ethereum Layer 2 solution. For detailed information on how Optimism works, please see the documentationhere .

As a starting point, this box contains only the SimpleStorage Solidity contract. Including minimal code was a conscious decision as this box is meant to provide the initial building blocks needed to get to work on Optimism without pushing developers to write any particular sort of application. With this box, you will be able to compile, migrate, and test Optimistic Solidity code against a variety of Optimism test networks. Check out how to build a NFT marketplace on Optimismhere .

Optimism's Layer 2 solution is almost fully compatible with the EVM, though it uses an "optimistic" EVM called the OVM. The main difference between the EVM and the OVM that developers will notice is that some opcodes are not available for contracts that are deployed to the OVM. You can see the complete list of differences between Optimism's fork of thesolc compiler and the originalhere .

Requirements¶

The Optimism Box has the following requirements:

- Node.js
- 10.x or later
- NPM
- version 5.2 or later
- docker
- , version 19.03.12 or later
- docker-compose
- , version 1.27.3 or later
- Recommended Docker memory allocation of >=8 GB.
- Windows, Linux or MacOS

Helpful, but optional: - AnInfura account and Project ID - AMetaMask account

Installation¶

Note that this installation command will only work once the box is published (in the interim you can usetruffle unbox https://github.com/truffle-box/optimism-box). truffle unbox optimism

Setup¶

Using the env File¶

You will need at least one mnemonic to use with the network. The.dotenv npm package has been installed for you, and you will need to create a.env file for storing your mnemonic and any other needed private information.

The.env file is ignored by git in this project, to help protect your private data. In general, it is good security practice to avoid committing information about your private keys to github. Thetruffle-config.ovm.js file expects aGANACHE_MNEMONIC and aGOERLI_MNEMONIC value to exist in.env for running commands on each of these networks, as well as a defaultMNEMONIC for the optimistic network we will run locally.

If you are unfamiliar with using.env for managing your mnemonics and other keys, the basic steps for doing so are below:

1) Usetouch .env in the command line to create a.env file at the root of your project. 2) Open the.env file in your preferred IDE 3) Add the following, filling in your own Infura project key and mnemonics:

MNEMONIC="candy maple cake sugar pudding cream honey rich smooth crumble sweet treat" INFURA_KEY="' GANACHE_MNEMONIC="" GOERLI_MNEMONIC="" Note: the value for theMNEMONIC above is the one you should use, as it is expected within the local optimistic ethereum network we will run in this Truffle Box.

4) As you develop your project, you can put any other sensitive information in this file. You can access it from other files withrequire('dotenv').config() and refer to the variable you need withprocess.env['] .

New Configuration File¶

A new configuration file exists in this project:truffle-config.ovm.js . This file contains a reference to the new file location of thecontracts_build_directory andcontracts_directory for Optimism contracts and lists several networks for running the Optimism Layer 2 network instance (seebelow).

Please note, the classictruffle-config.js configuration file is included here as well, because you will eventually want to deploy contracts to Ethereum as well. All normal truffle commands (truffle compile ,truffle migrate , etc.) will use this config file and save built files tobuild/ethereum-contracts . You can save Solidity contracts that you wish to deploy to Ethereum in thecontracts/ethereum folder.

New Directory Structure for Artifacts¶

When you compile or migrate, the resultingjson files will be atbuild/optimism-contracts/ . This is to distinguish them from any Ethereum contracts you build, which will live inbuild/ethereum-contracts . As we have included the appropriatecontracts_build_directory in each configuration file, Truffle will know which set of built files to reference!

Optimistic Ethereum¶

Compiling¶

To compile your code for Optimistic Ethereum, run the following in your terminal:

npm run compile:ovm This script lets Truffle know to use thetruffle-config.ovm.js configuration file, which references the directory in which we'll save your compiled contracts. When adding new contracts to compile, you may find some discrepancies and errors, so please remember to keep an eye ondifferences between solc and optimistic solc!

If you would like to recompile previously compiled contracts, you can manually run this command withtruffle compile --config truffle-config.ovm.js and add the--all flag.

Migrating¶

To migrate on an Optimistic Layer 2, run:

npm run migrate:ovm --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard) (remember to choose a network from these options!).

You have several Optimistic Layer 2 networks to choose from, prepackaged in this box (note: Layer 1 networks associated with Optimism are included in the regulartruffle-config.js file, to aid you with further development. But here we'll just go through the Layer 2 deployment options available):

- optimistic_ethereum
- : This network is the default Layer 1/Layer 2 integration provided by Optimism for testing your Optimistic Ethereum code. Documentation about this setup can be foundhere
- .
- You will need to install the code for this network in this box in order to use the scripts associated with it. To install it, runnpm run installLocalOptimism
- . You should only need to run this initiation command once. It will create anoptimism
- directory in this project that will house the repository you need. If at any point you want to update to the latest optimism docker image, you can delete youroptimism
- directory and run this command again.
- If you wish to use this network, be sure to runnpm run startLocalOptimism
- so that the optimism test ecosystem docker image can be served. For our purposes, you should be able to compile, migrate, and test against this network once the docker image is fully running. Seedocumentation and updates
- about this docker container for additional information.
- Please note, after runningnpm run startLocalOptimism
- it can take several minutes for the test ecosystem to be up and running on your local machine. The first time you run this command, it will take a bit longer for everything to be set up. Future runs will be quicker!
- To stop the local docker container, usenpm run stopLocalOptimism

- in a new terminal tab to ensure graceful shutdown.
- ganache
- : This network uses an optimistic ganache instance for migrations. The usage is essentially identical to use of regular ganache.Note:
- This optimistic ganache instance is no longer actively maintained. We recommend usingoptimistic_ethereum
- for local testing. Stay tuned for additional ganache resources in the future!
- optimistic_goerli
- : Optimism has deployed a testnet to the Goerli network. The RPC endpoint is https://optimism-goerli.infura.io/v3/. In order to access this node for testing, you will need to connect a wallet (we suggestMetaMask
-). Save your seed phrase in a.env
- file asGOERLI_MNEMONIC
- . Using an.env
- file for the mnemonic is safer practice because it is listed in.gitignore
- and thus will not be committed.
- You will need Goerli ETH in an Optimistic Goerli wallet to deploy contracts using this network. In order to deploy to Optimistic Goerli, you will need to acquire Optimistic Goerli ETH. As of this writing, there is not an Optimistic Goerli ETH faucet. In order to get Optimistic Goerli ETH, follow these steps: 1) Acquire ETH for your Goerli wallet on MetaMask using aGoerli faucet
- . 2) Optimism's gateway still doesn't support Goerli, but if you transfer Goerli ETH to0x636Af16bf2f682dD3109e60102b8E1A089FedAa8
- (opens new window), you will get it on Optimistic Goerli.

Note: You may get an error about the block limit being exceeded. The Truffle team is working on this issue, but in the meantime you can add this line before the deployment in yourmigrations/1_deploy_contracts.js file:SimpleStorage.gasMultiplier = 0.9;

Layer 1 networks are included in thetruffle-config.js file, but it is not necessary to deploy your base contracts to Layer 1 right now. Eventually, you will likely have a Layer 2 contract that you want to connect with a Layer 1 contract (they do not have to be identical!). One example is an ERC20 contract that is deployed on an Optimistic Ethereum network. At some point the user will wish to withdraw their funds into Ethereum. There will need to be a contract deployed on Layer 1 that can receive the message from Layer 2 to mint the appropriate tokens on Layer 1 for the user. More information on this system can be foundhere .

If you would like to migrate previously migrated contracts on the same network, you can runtruffle migrate --config truffle-config.ovm.js --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard) and add the--reset flag.

Basic Commands¶

The code here will allow you to compile, migrate, and test your code against an Optimistic Ethereum instance. The following commands can be run (more details on each can be found in the next section):

To compile:

npm run compile:ovm

To migrate:

npm run migrate:ovm --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard)

To test:

npm run test:ovm --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard)

To run a script:

npm run exec:ovm script --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard) Usingtruffle exec gives your script access to the instance of web3 you have running, viaweb3 , and also includes your contracts as global objects when executing the script. For more information on this command, seehere .

Testing¶

Currently, this box supports testing via Javascript/TypeScript tests. In order to run the test currently in the boilerplate, use the following command:

npm run test:ovm --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard) Remember that there are some differences between the EVM and the OVM, and refer to the Optimism documentation if you run into test failures.

Running Scripts¶

You can write scripts that have access to your Truffle configuration and a web3 instance that is connected to the network

indicated usingtruffle exec using the following command:

npm run exec:ovm script --network=(ganache | optimistic_ethereum | optimistic_goerli | dashboard) Remember that there are some differences between the EVM and the OVM, and refer to the Optimism documentation if you run into failures.

Communication Between Ethereum and Optimism Chains¶

The information above should allow you to deploy to the Optimism Layer 2 chain. This is only the first step! Once you are ready to deploy your own contracts to function on Layer 1 using Layer 2, you will need to be aware of theways in which Layer 1 and Layer 2 interact in the Optimism ecosystem . We have anOptimism Bridge Box that shows you just how to do that!

Support¶

Support for this box is available via the Truffle community available here .