Polygon Box¶

Table of contents generated with markdown-toc

This Truffle Polygon box provides you with the boilerplate structure necessary to start coding for Polygon's Ethereum L2 solution, the Polygon PoS chain (previously called the Matic PoS chain). For detailed information on how the Polygon PoS chain works, please see their 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 Polygon PoS without pushing developers to write any particular sort of application. With this box, you will be able to compile, migrate, and test Solidity code against several instances of Polygon PoS networks.

Polygon's L2 solution is fully compatible with the EVM. This means you will not need a new compiler to deploy Solidity contracts, and should be able to add your own Solidity contracts to this project. The main difference developers will encounter is in accessing and interacting with the Polygon PoS network. Additionally, Polygon offers multiple ways for dapp developers to implement communication between Ethereum ("Layer 1") and Polygon PoS. Further information about how to enable Ethereum-Polygon communication can be found in the Polygon documentationhere .

A note about naming: The Polygon ecosystem was previously called Matic Network. The chain to which we'll be deploying in this Truffle Box is now called the Polygon PoS chain. We have named this box the Polygon Box because we expect to include the ability to deploy to future Polygon chains in addition to what is presented here as an initial proof-of-concept, and developers using this Polygon Box may find themselves incorporating additional aspects of the Polygon ecosystem in their work.

Requirements¶

The Polygon box has the following requirements:

- Node.js
- 10.x or later
- NPM
- version 5.2 or later
- Windows, Linux or MacOS

Helpful, but optional:

- AnInfura
- account and Project ID
- AMetaMask
- account

Installation¶

truffle unbox polygon

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.polygon.js file expects aMNEMONIC value to exist in.env for running migrations on the networks listed intruffle-config.polygon.js .

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
2. in the command line to create a.env
3. file at the root of your project.
4. Open the.env
5. file in your preferred IDE
6. Add the following, filling in your own mnemonic and Infura project key:

MNEMONIC="" INFURA_PROJECT_ID="" 1. 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() 2. and refer to the variable you need withprocess.env['] 3. .

New Configuration File¶

A new configuration file exists in this project:truffle-config.polygon.js . This file contains a reference to the new file location of thecontracts_build_directory for Polygon PoS contracts and lists several networks that are running the Polygon PoS 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/polygon-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!

Polygon PoS Chain¶

Compiling¶

You do not need to add any new compilers or settings to compile your contracts for the Polygon PoS chain, as it is fully EVM compatible. Thetruffle-config.polygon.js configuration file indicates the contract and build paths for Polygon-destined contracts.

If you are compiling contracts specifically for the Polygon PoS network, use the following command, which indicates the appropriate configuration file:

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

Migrating¶

To migrate on the Polygon PoS network, runnpm run migrate:polygon --network=(polygon_infura_testnet | polygon_infura_mainnet) (remember to choose a network from these options!).

As you can see, you have several Polygon PoS L2 networks to choose from:

Infura networks . Infura is running a testnet node as well as a mainnet node for the Polygon PoS chain. Deployment to these networks requires that you sign up for an Infura account and initiate a project. See the Infura website fordetails . In the example network configuration, we expect you to have a public Infura project key, which you should indicate in your.env file. The following Infura networks are indicated in thetruffle-config.polygon.js file:

- polygon_infura_testnet
- : This is the Infura Polygon PoS testnet.
- polygon_infura_mainnet
- : This is the Infura Polygon PoS mainnet. Caution! If you deploy to this network using a connected wallet, the fees are charged in mainnet MATIC.

If you would like to migrate previously migrated contracts on the same network, you can runtruffle migrate --config truffle-config.polygon.js --network= (polygon_infura_testnet | polygon_infura_mainnet) and add the--reset flag.

Paying for Migrations¶

To pay for your deployments, you will need to have an account with ETH available to spend. You will need your mnemomic phrase (saved in the.env file or through some other secure method). The first account generated by the seed needs to have the ETH you need to deploy. For reference, the Polygon PoS testnets are based in goerli, so you should be able to use goerli ETH.

If you do not have a wallet with funds to deploy, you will need to connect a wallet to at least one of the networks above. For testing, this means you will want to connect a wallet to thepolygon_infura_testnet network. We recommend usingMetaMask .

Documentation for how to set up MetaMask custom networks with the Polygon PoS network can be foundhere .

Follow the steps in the documentation above using Infura's RPC endpoints ("https://polygon-mainnet.infura.io/v3/" + infuraProjectId and"https://polygon-mumbai.infura.io/v3/" + infuraProjectId). ThechainId values are the same as those in thetruffle-config.polygon.js networks entries.

To get testnet ETH to use, visit a faucet like https://goerli-faucet.slock.it/.

Basic Commands¶

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

To compile:

npm run compile:polygon To migrate:

npm run migrate:polygon --network=(polygon_infura_testnet | polygon_infura_mainnet) To test:

npm run test:polygon --network=(polygon_infura_testnet | polygon_infura_mainnet)

Testing¶

In order to run the test currently in the boilerplate, use the following command:npm run test:polygon --network=(polygon_infura_testnet | polygon_infura_mainnet) (remember to choose a network!). The current test file just has some boilerplate tests to get you started. You will likely want to add network-specific tests to ensure your contracts are behaving as expected.

Communication Between Ethereum and Polygon PoS Chains¶

The information above should allow you to deploy to the Polygon PoS Layer 2 chain. This is only the first step! Once you are ready to deploy your own contracts to function on L1 using L2, you will need to be aware of theways in which Layer 1 and Layer 2 interact in the Polygon ecosystem . Keep an eye out for additional Truffle tooling and examples that elucidate this second step to full Polygon PoS L2 integration!

Support¶

Support for this box is available via the Truffle communityhere .