Arbitrum Box¶

This Truffle Arbitrum Box provides you with the boilerplate structure necessary to start coding for Arbitrum's Ethereum Layer 2 solution. For detailed information on how Arbitrum 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 Arbitrum 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 a variety of Arbitrum test networks.

Arbitrum's Layer 2 solution is almost fully compatible with the EVM. You do not need a separate compiler to compile your Solidity contracts. The main difference between the EVM and the Arbitrum chain that developers will notice is that some opcodes are different and concepts such as time and gas are handled a little differently. Developers can use their regular Solidity compiler to compile contracts for Arbitrum. You can see the complete list of differences between the Arbitrum L2 chain and Ethereumhere .

Requirements¶

The Arbitrum 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/arbitrum-box). truffle unbox arbitrum

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.arbitrum.js file expects aMNEMONIC value to exist in.env for running commands on each of these networks, as well as a defaultMNEMONIC for the Arbitrum 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="jar deny prosper gasp flush glass core corn alarm treat leg smart" INFURA_KEY="" GOERLI_MNEMONIC="" MAINNET_MNEMONIC="" Note: the value for theMNEMONIC above is the one you should use, as it is expected within the local arbitrum 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.arbitrum.js . This file contains a reference to the new file location of thecontracts_build_directory andcontracts_directory for Arbitrum contracts and lists several networks for running the Arbitrum 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/arbitrum-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!

Arbitrum¶

Compiling¶

To compile your Arbitrum contracts, run the following in your terminal:

npm run compile:arbitrum This script lets Truffle know to use thetruffle-config.arbitrum.js configuration file, which tells Truffle where to store your build artifacts. When adding new contracts to compile, you may find some discrepancies and errors, so please remember to keep an ondifferences between ethereum and Arbitrum !

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

Migrating¶

To migrate on Arbitrum, run:

npm run migrate:arbitrum --network=(arbitrum_local | arbitrum_testnet | arbitrum_mainnet) (remember to choose a network from these options!).

You have several Arbitrum networks to choose from, prepackaged in this box (note: Layer 1 networks 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):

- arbitrum_local
- : This network is the default Layer 1/Layer 2 integration provided by Arbitrum for testing your Arbitrum-compatible code. Documentation about this setup can be foundhere
- .

Please note: Arbitrum is currently changing how running a local Arbitrum node works, and until the changes are complete the local Arbitrum network will not work. We will remove this note once the local node setup has been updated.

- 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 installLocalArbitrum
- . You should only need to run this initiation command once. It will create anarbitrum
- directory in this project that will house the repository you need. If at any point you want to update to the latest Arbitrum docker image, you can delete yourarbitrum
- directory and run this command again. If you'd rather house the Arbitrum local blockchain outside of this box, seethese notes
- for how to get started doing so.
- If you wish to use this network, follow these steps, in this order:
- 1) In a new terminal tab, enternpm run startLocalEthereum

- . 2) Wait for step #1 to complete. The Arbitrum Layer 2 blockchain depends on the existence of a Layer 1 for proper interoperability. 3) In another new terminal tab, enternpm run startLocalArbitrum
- . Wait a little while, and you will see the Arbitrum blockchain running and interacting with the Layer 1 simulation from step #1! You are ready to try out deploying your contracts! -arbitrum_testnet
- : Arbitrum has deployed a testnet to the Rinkeby network. The RPC endpoint is https://arbitrum-rinkeby.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 asRINKEBY_MNEMONIC
- . Using an.env
- file for the mnemonic is safer practice because it is listed in.gitignore
- and thus will not be committed. * Currently, we have the gasPrice for transactions on Arbitrum Rinkeby set to zero. You should be able to use this network as configured at this time. * In order to set up your MetaMask wallet to connect to the Arbitrum Rinkeby network, you will need to create a custom RPC network in your wallet. You can find detailed steps for this processhere
- . You will need the following information: - RPC Url:https://arbitrum-rinkeby.infura.io/v3/ +
-
    - chain id: 421611
- arbitrum_mainnet
- : This is the mainnet for Arbitrum's Layer 2 solution. You will need to connect your wallet to the Arbitrum mainnet RPC network, located at https://arbitrum-mainnet.infura.io/v3/

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. One example is an ERC20 contract that is deployed on an Arbitrum 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.arbitrum.js --network=(arbitrum_local | arbitrum_testnet | arbitrum_mainnet) and add the--reset flag.

Basic Commands¶

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

To compile:

npm run compile:arbitrum

To migrate:

npm run migrate:arbitrum --network=(arbitrum_local | arbitrum_testnet | arbitrum_mainnet)

To test:npm run test:arbitrum --network=(arbitrum_local | arbitrum_testnet | arbitrum_mainnet)

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:arbitrum --network=(arbitrum_local | arbitrum_testnet | arbitrum_mainnet) Remember that there are some differences between Arbitrum and Ethereum, and refer to the Arbitrum documentation if you run into test failures.

Communication Between Ethereum and Arbitrum Chains¶

The information above should allow you to deploy to an Arbitrum 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 Arbitrum ecosystem . Keep an eye out for additional Truffle tooling and examples that elucidate this second step to full Arbitrum integration!

Support¶

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