Setup and Installation

In this step, we're going to

1. Install prerequisites
2. Create a yarn project to house everything
3. Create a noir project for our Aztec contract
4. Create a hardhat project for our Ethereum contract(s)
5. Import all the Ethereum contracts we need
6. Create a yarn project that will interact with our contracts on L1 and the sandbox

We recommend going through this setup to fully understand where things live.

However if you'd rather skip this part, our dev-rels repo contains the starter code here.

Prerequisites

- node v18+
- docker
- Aztec sandbox
-
    - you should have this running before starting the tutorial
- Aztec CLI

/bin/sh -c " (curl -fsSL 'https://sandbox.aztec.network') "

Create the root project and packages

Our root project will house everything ✨

mkdir aztec-token-bridge cd aztec-token-bridge &&

mkdir packages We will hold our projects inside ofpackages to follow the design of the project in thisepo .

Create a noir project

Now insidepackages create a new directory calledaztec-contracts

Insideaztec-contracts , create the following file structure:

aztec-contracts └── token_bridge ├── Nargo.toml ├── src ├── main.nr InsideNargo.toml add the following content:

[package] name

=

"token_bridge" authors

=

[""] compiler_version

=

">=0.18.0" type

=

"contract"

[dependencies] aztec

=

{

git

"https://github.com/AztecProtocol/aztec-packages/" ,

tag

"aztec-packages-v0.28.1" ,

directory

"noir-projects/aztec-nr/aztec"

} token_portal_content_hash_lib

=

{

git

"https://github.com/AztecProtocol/aztec-packages/" ,

tag

"aztec-packages-v0.28.1" ,

directory

"noir-projects/noir-contracts/contracts/token_portal_content_hash_lib"

} We will also be writing some helper functions that should exist elsewhere so we don't overcomplicated our contract. Insrc create two more files - one calledutil.nr and one calledtoken_interface - so your dir structure should now look like this:

aztec-contracts └─── token_bridge ├─── Nargo.toml ├─── src ├─── main.nr ├─── token_interface.nr ├─── util.nr

Create a JS hardhat project

In thepackages dir, create a new directory calledl1-contracts and runyarn init -yp && npx hardhat init inside of it. Keep hitting enter so you get the default setup (Javascript project)

mkdir l1-contracts cd l1-contracts yarn init -yp npx hardhat init Once you have a hardhat project set up, delete the existing contracts, tests, and scripts, and create aTokenPortal.sol :

rm -rf contracts test scripts mkdir contracts &&

cd contracts touch TokenPortal.sol Now add dependencies that are required. These include interfaces to Aztec Inbox, Outbox and Registry smart contracts, OpenZeppelin contracts, and NomicFoundation.

yarn

add @aztec/foundation @aztec/l1-contracts @openzeppelin/contracts &&

yarn

add --dev @nomicfoundation/hardhat-network-helpers @nomicfoundation/hardhat-chai-matchers @nomiclabs/hardhat-ethers @nomiclabs/hardhat-etherscan @types/chai @types/mocha @typechain/ethers-v5 @typechain/hardhat chai@4.0.0 hardhat-gas-reporter solidity-coverage ts-node typechain typescript This is what yourl1-contracts should look like:

├─── README.md ├─── contracts ├─── hardhat.config.js ├─── node_modules └─── package.json We will need to ensure we are using the correct Solidity version. Inside yourhardhat.config.js updatesolidity version to this:

solidity :

"0.8.20" ,

Create src yarn project

In this directory, we will write TS code that will interact with our L1 and L2 contracts and run them against the sandbox.

We will useviem in this tutorial andjest for testing.

Inside thepackages directory, run

mkdir src &&

cd src &&

yarn init -yp yarn

add typescript @aztec/aztec.js @aztec/accounts @aztec/noir-contracts.js @aztec/types @aztec/foundation @aztec/l1-artifacts viem@1.21.4 "@types/node@^20.8.2" yarn

add -D jest @jest/globals ts-jest If you are going to track this repo using git, consider adding a.gitignore file to yoursrc directory and addingnode_modules to it.

Inpackage.json , add:

"type" :

"module" , "scripts" :

{ "test" :

"NODE_NO_WARNINGS=1 node --experimental-vm-modules (yarn bin jest)" } Yourpackage.json should look something like this (do not copy and paste):

{ "name" :

"src" , "version" :

"1.0.0" , "main" :

"index.js" , "license" :

"MIT" , "private" :

true , "type" :

"module" , "dependencies" :

{ "dep" :

"version" } , "devDependencies" :

{ "dep" :

"version" } , "scripts" :

{ "test" :

"NODE_NO_WARNINGS=1 node --experimental-vm-modules (yarn bin jest)" } } Create atsconfig.json and paste this:

{ "compilerOptions" :

{ "rootDir" :

"../" , "outDir" :

"./dest" , "target" :

"es2020" , "lib" :

["dom" ,

"esnext" ,

"es2017.object"] , "module" :

"NodeNext" , "moduleResolution" :

"NodeNext" , "strict" :

true , "declaration" :

true , "allowSyntheticDefaultImports" :

true , "esModuleInterop" :

true , "downlevelIteration" :

true , "inlineSourceMap" :

true , "declarationMap" :

true , "importHelpers" :

true , "resolveJsonModule" :

true , "composite" :

true , "skipLibCheck" :

true } , "include" :

["packages/src//**" , "**contracts**//*.json" , "packages/src//*" , "packages/aztec-contracts/token_bridge/target/*.json"] , "exclude" :

["node_modules" ,

"*/*.spec.ts" ,

"contracts/*/*.ts"] , "references" :

[ ] } The main thing this will allow us to do is to access TS artifacts that we generate later from our test.

Then create a jest config file:jest.config.json

{ "preset" :

"ts-jest/presets/default-esm" , "globals" :

{ "ts-jest" :

{ "useESM" :

true } } , "moduleNameMapper" :

{ "^(\.{1,2}/.*)\.js" :

"1" } , "testRegex" :

"./test/.*\.test\.ts" , "rootDir" :

"./test" } Finally, we will create a test file. Run this in thesrc directory.:

mkdir

test

&&

cd

test touch cross_chain_messaging.test.ts Yoursrc dir should look like:

src ├── node_modules └── test └── cross_chain_messaging.test.ts ├── jest.config.json ├── package.json ├── tsconfig.json In the next step, we'll start writing our L1 smart contract with some logic to deposit tokens to Aztec from L1. Edit this page

Previous Build a Token Bridge Next Depositing Tokens to Aztec