Quickstart

In this guide, you will

1. Set up the Aztec sandbox (local development environment) locally
2. Install the Aztec development kit
3. Use the CLI to deploy an example contract that comes with the sandbox
4. Use the CLI to interact with the contract you just deployed

... in less than 10 minutes.

Prerequisites

- Node.js >= v18 (recommend installing withnvm
-)

Install Docker

Seethis page of the Docker docs for instructions on how to install Docker Desktop for your operating system.

Once you have Docker installed, make sure it is running by opening the Docker Desktop application.

Note on Linux

If you are running Linux, you will need to set the context (because Docker Desktop runs in a VM by default). Seethis page for more information. You can do this by running:

docker context use default

Install the Sandbox

You can run the Sandbox using Docker.

To install the latest Sandbox version, run:

bash -i < (curl -s install.aztec.network) This will install the following:

- aztec
-
    - launches various infrastructure subsystems (sequencer, prover, pxe, etc).
- aztec-cli
-
    - a command line tool for interfacing and experimenting with infrastructure.
- aztec-nargo
-
    - aztec's build of nargo, the noir compiler toolchain.
- aztec-sandbox
-
    - a wrapper around docker-compose that launches services needed for sandbox testing.
- aztec-up
-
    - a tool to upgrade the aztec toolchain to the latest, or specific versions.

Once these have been installed, to start the sandbox, run:

aztec-sandbox This will attempt to run the Sandbox onlocalhost:8080 , so you will have to make sure nothing else is running on that port or change the port defined in./.aztec/docker-compose.yml . Running the installation again will overwrite any changes made to thedocker-compose.yml .

This command will also install the CLI if a node package version of the CLI isn't found locally.

Deploy a contract using the CLI

The sandbox is preloaded with multiple accounts. Let's assign them to shell variables. Run the following in your terminal, so we can refer to the accounts as ALICE and BOB from now on:

note The default accounts that come with sandbox will likely change over time. Save two of the "Initial accounts" that are printed in the terminal when you started the sandbox. declare-accounts ACCOUNTS = (aztec-cli get-accounts --json | jq -r '.[].address') ALICE = (echo

" ACCOUNTS "

|

sed -n 1p) BOB = (echo

" ACCOUNTS "

|

sed -n 2p) ALICE_PRIVATE_KEY = "0x2153536ff6628eee01cf4024889ff977a18d9fa61d0e414422f7681cf085c281"Source code: yarn-project/end-to-end/src/guides/up_quick_start.sh#L6-L11 Start by deploying a token contract. After it is deployed, we check that the deployment succeeded, and export the deployment address to use in future commands. For more detail on how the token contract works, see thetoken contract tutorial .

deploy CONTRACT = (aztec-cli deploy TokenContractArtifact --private-key ALICE_PRIVATE_KEY --salt 0 --args ALICE "TokenName"

"TKN"

18 --json | jq -r '.address') echo

"Deployed contract at CONTRACT " aztec-cli check-deploy --contract-address CONTRACTSource code: yarn-project/end-to-end/src/guides/up_quick_start.sh#L13-L17 note If you're not using the default port for the Sandbox, make sure to pass the--rpc-url parameter, e.g.:--rpc-url http://localhost:8000 . Note that the deployed contract address is exported, so we can use it asCONTRACT later on.

Call a contract with the CLI

Alice is set up as the contract admin and token minter in the_initialize function. Let's get Alice some private tokens.

We need to export theSECRET andSECRET_HASH values in order to privately mint tokens. Private tokens are claimable by anyone with the pre-image to a provided hash, see more about how the token contract works in thetoken contract tutorial . After the tokens have been minted, the notes will have to added to thePrivate Execution Environment (PXE) to be consumed by private functions. Once added, Alice can claim them with theredeem_shield function. After this, Alice should have 1000 tokens in their private balance.

mint-private SECRET = "0x29bf6afaf29f61cbcf2a4fa7da97be481fb418dc08bdab5338839974beb7b49f" SECRET_HASH = "0x0921759afa747c9073f75df9688a17d271cef0d6ec51eacf70e112402c4db6cd"

MINT_PRIVATE_OUTPUT

(aztec-cli send mint_private \ --args 1000 SECRET_HASH \ --contract-artifact TokenContractArtifact \ --contract-address CONTRACT \ --private-key ALICE_PRIVATE_KEY)

MINT_PRIVATE_TX_HASH

(echo

" MINT_PRIVATE_OUTPUT "

|

grep

"Transaction hash:"

|

awk

'{print NF}')

aztec-cli add-note \ ALICE

CONTRACT

5

84114971101151129711410111011678111116101

MINT_PRIVATE_TX_HASH

\ --note 1000

SECRET_HASH

aztec-cli send redeem_shield \ --args ALICE

1000

SECRET

\ --contract-artifact TokenContractArtifact \ --contract-address CONTRACT

\ --private-key ALICE_PRIVATE_KEYSource code: yarn-project/end-to-end/src/guides/up_quick_start.sh#L19-L40 We can have Alice privately transfer tokens to Bob. Only Alice and Bob will know what's happened. Here, we use Alice's private key to send a transaction to transfer tokens to Bob. Once they are transferred, we can verify that it worked as expected by checking Alice's and Bob's balances:

transfer aztec-cli send transfer \ --args ALICE

BOB

500

0

\ --contract-artifact TokenContractArtifact \ --contract-address CONTRACT

\ --private-key ALICE_PRIVATE_KEY

aztec-cli call balance_of_private \ --args ALICE

\ --contract-artifact TokenContractArtifact \ --contract-address CONTRACT

aztec-cli call balance_of_private \ --args BOB

\ --contract-artifact TokenContractArtifact \ --contract-address CONTRACTSource code: yarn-project/end-to-end/src/guides/up_quick_start.sh#L49-L65 Alice and Bob should have 500 tokens.

Congratulations! You are all set up with the Aztec sandbox!

What's next?

To deploy and interact with a contract using Aztec.js, go to the next page .

You can also dig more into the sandbox and CLI here .

To learn more about writing contracts, consider jumping to the tutorials section . Edit this page

Previous Getting Started Next Writing your first smart contract