Local Development Chain with Protocol Contracts

How to set up a Celo development blockchain that includes all of thecore protocol contracts , including identity and stability contracts.

What to expect

At the end of this tutorial, you will have a local Celo development blockchain running exposed athttp://localhost:7545 and will be able to connect to it like any other local node. We will also go over how to inspect the development blockchain using theCelo CLI tool and theContractKit .

Running the development Celo blockchain is helpful because it greatly speeds up development time. You will start with 10 accounts pre-funded with CELO and all transactions on the network are virtually instant.

You can run the development Celo blockchain in several ways:

Use Ganache to fork Mainnet or Alfajores

You can start a local EVM development blockchain withGanache . Once Ganache is installed you can fork Mainnet or Alfajores with a single command. Read more about it in theGanache docs here .

ganache --fork.url https://forno.celo.org This allows you to interact with contracts and accounts that were deployed at the specified fork point, including DeFi applications and protocol contracts like theReserve ,cUSD ,cEUR ,Accounts , Governance andother contracts .

You also get 10 accounts funded with fake CELO to deploy your contracts and run tests in this simulated mainnet environment.

Be aware that you will not be able to pay transaction fees in stable coins since this is an Ethereum enviroment. It behaves similarly to the Celo EVM but not exactly.

Use the celo-devchain NPM package

The easiest is to use a "pre-generated" devchain from thecelo-devchain NPM package. For that all you have to do is:

npm

install --save-dev @terminal-fi/celo-devchain

npx celo-devchain --port

7545

or

yarn

add

--dev @terminal-fi/celo-devchain

yarn run celo-devchain --port

7545

Initialize your own devchain from the monorepo

If you prefer, you can initialize your own devchain and build it from scratch. To start, download the Celo monorepohere or with the following command.

git clone https://github.com/celo-org/celo-monorepo.git Seethis page for instructions on how to build the monorepo.

Once the monorepo is built, move into thecontractkit directory.

cd packages/sdk/contractkit From thecontractkit directory, run

yarn test:reset && yarn test:livechain This will start the development Celo blockchain. It will take at least a few minutes to start. The contract migrations will be printed in the terminal as they are deployed.

The process will finish and printGanache started . Leave this terminal window open to leave the development chain running.

Interacting with the chain

Inspecting the chain

Now that we have a Celo development chain running, we probably want to know what accounts we have access to, how much cGLD and cUSD they have as well as the addresses of the deployed protocol contracts.

We can use theCelo CLI tool for this, or we can use theContractKit npm package in a node script.

Celo CLI

You can install the CLI using npm by runningnpm install -g @celo/celocli . You can see thepackage details here . Once it is installed, you should be able to access the tool from the terminal by running celocli . Try celocli help .

The CLI will connect to the node athttp://localhost:8545 by default. To connect to port 7545 you can run celocli config:set -n http://localhost:7545 , and then check the connection by running celocli node:get .

You can see the accounts available on the Celo development chain by running celocli account:list . You should see something like:

['0x5409ED021D9299bf6814279A6A1411A7e866A631', '0x6Ecbe1DB9EF729CBe972C83Fb886247691Fb6beb', '0xE36Ea790bc9d7AB70C55260C66D52b1eca985f84', '0xE834EC434DABA538cd1b9Fe1582052B880BD7e63', '0x78dc5D2D739606d31509C31d654056A45185ECb6', '0xA8dDa8d7F5310E4A9E24F8eBA77E091Ac264f872', '0x06cEf8E666768cC40Cc78CF93d9611019dDcB628', '0x4404ac8bd8F9618D27Ad2f1485AA1B2cFD82482D', '0x7457d5E02197480Db681D3fdF256c7acA21bDc12', '0x91c987bf62D25945dB517BDAa840A6c661374402'] If you try to check the balance of the first account with celocli account:balance 0x5409ED021D9299bf6814279A6A1411A7e866A631 you might encounter an error saying that the node is not currently synced. You can silence this by adding this environment variable to the terminal export NO_SYNCCHECK=true . Running the command again will print:

All balances expressed in units of 10^-18. gold: 9.9999999693185872e+25 lockedGold: 0 usd: 5e+22 total: 1.0000497304369128770379157 5e+26 pending: 0

ContractKit + Node.js

You can also use theContractKit to access the local node in a node.js script.

As an example, try runningthis script in an npm project with contractkit installed.

The linked gist is called getInfo.js. Run it with node getInfo.js This will print some of the Celo blockchain information.

You are now prepared to start developing, transacting and deploying contracts on your own Celo development blockchain!

Using Ethereum developers tools with Celo

You can connect the development chain to a tool likeTruffle orRemix to begin interacting with it. Keep in mind that these tools are built primarily for Ethereum development and are compatible with Celo because Celo is similar to Ethereum. The two blockchains have similar block architectures and both run the Ethereum Virtual Machine (EVM) for executing smart contracts.

The main difference between Celo and Ethereum that dapp developers need to keep in mind is that Celo has a slightly different transaction object than Ethereum. Celo requires one additional field in a transaction object, afeeCurrency . When Truffle and Remix are connected to a locally running Celo node, the local node will fill these fields with default values (if the fields are empty). The node will sign the Celo transaction and broadcast it to the network. Edit this page Previous Setup Testnet Wallets Next Deploy on Celo