Setting up a Celestia light node

This tutorial will guide you through setting up a Celestia light node, which will allow you to perform data availability sampling (DAS) on Celestia's data availability (DA) network.

Overview of light nodes

Light nodes ensure data availability. This is the most common way to interact with Celestia networks.

Light nodes have the following behavior:

1. They listen forExtendedHeaders
2. , i.e. wrapped “raw” headers, that notify Celestia nodes of new block headers and relevant DA metadata.
3. They perform DAS on the received headers

Hardware requirements

The following minimum hardware requirements are recommended for running a light node:

- Memory:500 MB RAM (minimum)
- CPU:Single Core
- Disk:50 GB SSD Storage
- Bandwidth:56 Kbps for Download/56 Kbps for Upload

Setting up your light node

This tutorial was performed on an Ubuntu Linux 20.04 (LTS) x64 instance machine.

Set up dependencies on thesetting up environment page.

Install celestia-node

Install thecelestia binary bybuilding and installing celestia-node .

Initialize the light node

Run the following command:

Mainnet Beta

Mocha

Arabica sh celestia

light

init celestia

light

init sh celestia

light

init

--p2p.network

mocha celestia

light

init

--p2p.network

mocha sh celestia

light

init

--p2p.network

arabica celestia

light

init

--p2p.network

arabica The output in your terminal will show the location of your node store and config. It will also show confirmation that the node store has been initialized.

Start the light node

Start the light node with a connection to a validator node's gRPC endpoint (which is usually exposed on port 9090):

In order for access to the ability to get and submit state-related information, such as the ability to submitPayForBlobs transactions, or query for the node's account balance, a gRPC endpoint of a validator (core) node must be passed as directed below.

Refer to the ports section of the celestia-node troubleshooting page for information on which ports are required to be open on your machine.

To start the light node with a connection to a validator node's gRPC endpoint (which is usually exposed on port 9090):

sh celestia

light

start

--core.ip

< UR I

--p2p.network

< networ k

celestia

light

start

--core.ip

< UR I

--p2p.network

< networ k

TIP

You do not need to declare a network for Mainnet Beta. Refer to the chain ID section on the troubleshooting page for more information Using an RPC of your own, or one from the list on the Mocha testnet page or list on the Arabica devnet page , start your node.

For example, your command might look something like this for Mocha:

sh celestia

light

start

--core.ip

rpc-mocha.pops.one

--p2p.network

mocha celestia

light

start

--core.ip

rpc-mocha.pops.one

--p2p.network

mocha Or for Arabica:

sh celestia

light

start

--core.ip

validator-1.celestia-arabica-11.com

\ --p2p.network

arabica celestia

light

start

--core.ip

validator-1.celestia-arabica-11.com

\ --p2p.network

arabica

Keys and wallets

You can create your key for your node by running the following command with thecel-key utility in thecelestia-node directory:

sh ./cel-key

add

< key-nam e

--keyring-backend

test

\ --node.type

light

--p2p.network

< networ k

    ./cel-key

add

< key-nam e

--keyring-backend

test

\ --node.type

light

--p2p.network

< networ k

You can start your light node with the key created above by running the following command:

Mainnet Beta

Mocha

Arabica sh celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

consensus.lunaroasis.net celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

consensus.lunaroasis.net sh celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

rpc-mocha.pops.one

--p2p.network

mocha celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

rpc-mocha.pops.one

--p2p.network

mocha sh celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

validator-1.celestia-arabica-11.com

\ --p2p.network

arabica celestia

light

start

--keyring.accname

my_celes_key

\ --core.ip

validator-1.celestia-arabica-11.com

\ --p2p.network

arabica Once you start the light node, a wallet key will be generated for you. You will need to fund that address with testnet tokens to pay forPayForBlob transactions.

You canfind the address using the RPC CLI or by running the following command in thecelestia-node directory:

sh ./cel-key

list

--node.type

light

--keyring-backend

test

\ --p2p.network

< networ k

    ./cel-key

list

--node.type

light

--keyring-backend

test

\ --p2p.network

< networ k

Testnet tokens

You have two networks to get testnet tokens from:

- Arabica devnet
- Mocha testnet

You can request funds to your wallet address using the following command in Discord:

console request request Where is thecelestia1** address generated when you created the wallet.

Optional: run the light node with a custom key

In order to run a light node using a custom key:

1. The custom key must exist inside the celestia light node directory at the correct path (default:~/.celestia-light/keys/keyring-test
2.)
3. The name of the custom key must be passed uponstart
4. , like so:

Mainnet Beta

Arabica

Mocha sh celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ sh celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ --p2p.network

arabica celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ --p2p.network

arabica sh celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ --p2p.network

mocha celestia

light

start

--core.ip

< UR I

\ --keyring.accname

< name-of-custom-ke y

\ --p2p.network

mocha

Optional: start light node with SystemD

Followthe tutorial on setting up the light node as a background process with SystemD.

Data availability sampling

With your light node running, you can check outthis tutorial on submittingPayForBlob transactions . [][Edit this page on GitHub] Last updated: