Setting up a Celestia full consensus node

This guide covers how to set up a full consensus node or a validator node on Celestia. Full consensus nodes allow you to sync blockchain history in the Celestia consensus layer.

Hardware requirements

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

- Memory:8 GB RAM
- CPU:Quad-Core
- Disk:250 GB SSD Storage
- Bandwidth:1 Gbps for Download/1 Gbps for Upload

Running a full consensus node requires significant storage capacity to store the entire blockchain history. As of the latest recommendation, it is advisable to have at least 250 GB of SSD storage for a Celestia full consensus node if you are using pruning. If you are not using pruning, you are running an archive node, and it is recommended to have 500 GB of SSD storage. Please ensure that your storage meets this requirement to ensure smooth syncing and operation of the node.

Setting up a full consensus node

The following tutorial is done on an Ubuntu Linux 20.04 (LTS) x64 instance machine.

Setup the dependencies

Follow the instructions oninstalling dependencies .

Install celestia-app

Follow the tutorial oninstallingcelestia-app .

Setup the P2P networks

Now we will setup the P2P Networks by cloning the networks repository:

sh cd HOME rm

-rf

networks git

clone

https://github.com/celestiaorg/networks.git cd HOME rm

-rf

networks git

clone

https://github.com/celestiaorg/networks.git To initialize the network pick a "node-name" that describes your node. Keep in mind that this might change if a new testnet is deployed.

Mainnet Beta

Mocha

Arabica bash celestia-appd

init

"node-name"

--chain-id

celestia celestia-appd

init

"node-name"

--chain-id

celestia bash celestia-appd

init

"node-name"

--chain-id

mocha-4 celestia-appd

init

"node-name"

--chain-id

mocha-4 bash celestia-appd

init

"node-name"

--chain-id

arabica-11 celestia-appd

init

"node-name"

--chain-id

arabica-11 Download thegenesis.json file:

Mainnet Beta

Mocha

Arabica bash celestia-appd

download-genesis

celestia celestia-appd

download-genesis

celestia bash celestia-appd

download-genesis

mocha-4 celestia-appd

download-genesis

mocha-4 bash celestia-appd

download-genesis

arabica-11 celestia-appd

download-genesis

arabica-11 Set seeds in theHOME/.celestia-app/config/config.toml file:

Mainnet Beta

Mocha

Arabica bash SEEDS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/celestia/seeds.txt |

tr '\n' ',') echo SEEDS sed

-i.bak

-e

"s/^seeds =./seeds = \" SEEDS \" /" HOME /.celestia-app/config/config.toml SEEDS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/celestia/seeds.txt |

tr '\n' ',') echo SEEDS sed

-i.bak

-e

"s/^seeds =./seeds = \" SEEDS \" /" HOME /.celestia-app/config/config.toml bash SEEDS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/mocha-4/seeds.txt |

tr '\n' ',') echo SEEDS sed

-i.bak

-e

"s/^seeds =./seeds = \" SEEDS \" /" HOME /.celestia-app/config/config.toml SEEDS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/mocha-4/seeds.txt |

tr '\n' ',') echo SEEDS sed

-i.bak

-e

"s/^seeds =./seeds = \" SEEDS \" /" HOME /.celestia-app/config/config.toml bash

For Arabica, you can set seeds manually in the

HOME/.celestia-app/config/config.toml file:

Comma separated list of seed nodes to connect to

seeds

=

""

For Arabica, you can set seeds manually in the

HOME/.celestia-app/config/config.toml file:

Comma separated list of seed nodes to connect to

seeds

=

"" Optionally , you can set persistent peers in yourconfig.toml file. You can get the persistent peers from the networks repository with the following commands:

Setting persistent peers is advised only if you are running a sentry node.

Mainnet Beta

Mocha

Arabica bash PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/celestia/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/celestia/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml bash

PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/mocha-4/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml
PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/mocha-4/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml bash
PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/arabica-11/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml
PERSISTENT_PEERS = (curl

-sL https://raw.githubusercontent.com/celestiaorg/networks/master/arabica-11/peers.txt |

tr '\n' ',') echo PERSISTENT_PEERS sed

-i.bak

-e

"s/^persistent_peers =./persistent_peers = \" PERSISTENT_PEERS \" /" HOME /.celestia-app/config/config.toml

Storage and pruning configurations

Connecting a consensus node to a bridge node

If your consensus node is being connected to a celestia-node bridge node, you will need to enable transaction indexing and retain all block data. This can be achieved with the following settings in yourconfig.toml .

Enable transaction indexing

toml indexer = "kv" indexer = "kv"

Retain all block data

And in yourapp.toml ,min-retain-blocks should remain as the default setting of0 :

toml min-retain-blocks = 0

retain all block data, this is default setting

min-retain-blocks = 0

retain all block data, this is default setting

Querying transactions by hash

If you want to query transactions using their hash, transaction indexing must be turned on. Set theindexer to"kv" in yourconfig.toml :

toml indexer = "kv" indexer = "kv"

Accessing historical state

If you want to query the historical state — for example, you might want to know the balance of a Celestia wallet at a given height in the past — you should run an archive node withpruning = "nothing" in yourapp.toml . Note that this configuration is resource-intensive and will require significant storage:

toml pruning = "nothing" pruning = "nothing"

Saving on storage requirements

If you want to save on storage requirements, consider usingpruning = "everything" in yourapp.toml to prune everything. If you select"everything" or"default" , but still want to keep the block data, you can do so by not changing the default value ofmin-retain-blocks = 0 in yourapp.toml . A value of0 formin-retain-blocks will keep all block data. This will prune snapshots of the state, but it will keep block data:

toml pruning = "everything" min-retain-blocks = 0

this is the default setting

pruning = "everything" min-retain-blocks = 0

this is the default setting

Syncing

By default, a consensus node will sync using block sync; that is request, validate and execute every block up to the head of the blockchain. This is the most secure mechanism yet the slowest (taking up to days depending on the height of the blockchain).

There are two alternatives for quicker syncing.

State sync

State sync uses light client verification to verify state snapshots from peers and then apply them. State sync relies on weak subjectivity; a trusted header (specifically the hash and height) must be provided. This can be found by querying a trusted RPC endpoint (/block). RPC endpoints are also required for retrieving light blocks. These can be found in the docs here under the respective networks or fromthe chain-registry .

InHOME/.celestia-app/config/config.toml , set

toml rpc_servers = "" trust_height = 0 trust_hash = "" rpc_servers = "" trust_height = 0 trust_hash = "" to their respective fields. At least two different rpc endpoints should be provided. The more, the greater the chance of detecting any fraudulent behavior.

Once setup, you should be ready to start the node as normal. In the logs, you should see:Discovering snapshots . This may take a few minutes before snapshots are found depending on the network topology.

Quick sync

Quick sync effectively downloads the entiredata directory from a third-party provider meaning the node has all the application and blockchain state as the node it was copied from.

Run the following command to quick-sync from a snapshot:

Mainnet Beta

Mocha

Arabica bash cd HOME rm

```
-rf
~/.celestia-app/data mkdir
-p
~/.celestia-app/data SNAP_NAME = ( curl
-s https://snaps.qubelabs.io/celestia/ |
\ egrep
-o ">celestia.*tar" |
tr
-d ">") aria2c
-x
16
-s
16
-o
celestia-snap.tar
"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar
xf
celestia-snap.tar
-C
~/.celestia-app/data/ cd HOME rm
-rf
~/.celestia-app/data mkdir
-p
~/.celestia-app/data SNAP_NAME = ( curl
-s https://snaps.qubelabs.io/celestia/ |
\ egrep
-o ">celestia.*tar" |
tr
-d ">") aria2c
-x
16
-s
16
-o
celestia-snap.tar
"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar
xf
```

celestia-snap.tar

-C

~/.celestia-app/data/ bash cd HOME rm

-rf

~/.celestia-app/data mkdir

-p

~/.celestia-app/data SNAP_NAME = (curl

-s https://snaps.qubelabs.io/celestia/ |

\ egrep

-o ">mocha-4.*tar" |

tr

-d ">") aria2c

-x

16

-s

16

-o

celestia-snap.tar

"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar

xf

celestia-snap.tar

-C

~/.celestia-app/data/ cd HOME rm

-rf

~/.celestia-app/data mkdir

-p

~/.celestia-app/data SNAP_NAME = (curl

-s https://snaps.qubelabs.io/celestia/ |

\ egrep

-o ">mocha-4.*tar" |

tr

-d ">") aria2c

-x

16

-s

16

-o

```
celestia-snap.tar
"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar
xf
celestia-snap.tar
-C
~/.celestia-app/data/ bash cd HOME rm
-rf
~/.celestia-app/data mkdir
-p
~/.celestia-app/data SNAP_NAME = ( curl
-s https://snaps.qubelabs.io/celestia/ |
\ egrep
-o ">arabica-11.*tar" |
tr
-d ">") aria2c
-x
16
-s
16
-o
celestia-snap.tar
"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar
xf
celestia-snap.tar
-C
~/.celestia-app/data/ cd HOME rm
-rf
~/.celestia-app/data mkdir
-p
~/.celestia-app/data SNAP_NAME = ( curl
-s https://snaps.qubelabs.io/celestia/ |
\ egrep
-o ">arabica-11.*tar" |
tr
-d ">") aria2c
-x
16
```

-s

16

-o

celestia-snap.tar

"https://snaps.qubelabs.io/celestia{ SNAP_NAME }" tar

xf

celestia-snap.tar

-C

~/.celestia-app/data/

Start the consensus node

In order to start your full consensus node, run the following:

sh celestia-appd

start celestia-appd

start Optional: If you would like celestia-app to run as a background process, you can follow theSystemD tutorial .

TIP

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

Optional: Setting up a validator

Setting up a Celestia validator node

Validator nodes allow you to participate in consensus in the Celestia network.

Validator hardware requirements

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

- Memory:8 GB RAM
- CPU:6 cores
- Disk:500 GB SSD Storage
- Bandwidth:1 Gbps for Download/1 Gbps for Upload

The following tutorial is done on an Ubuntu Linux 20.04 (LTS) x64 instance machine.

First, set up your full consensus node by following the instructions inthe previous section .

Wallet

Followthe tutorial on creating a wallet .

Delegate stake to a validator

Create an environment variable for the address:

bash VALIDATOR_WALLET =< validator-wallet-nam e

VALIDATOR_WALLET =< validator-wallet-nam e

If you want to delegate more stake to any validator, including your own you will need thecelesvaloper address of the validator in question. You can run the command below to get thecelesvaloper of your local validator wallet in case you want to delegate more to it:

bash celestia-appd

keys

show VALIDATOR_WALLET --bech

val

-a celestia-appd

keys

show VALIDATOR_WALLET --bech

val

-a After entering the wallet passphrase you should see a similar output:

bash Enter

keyring

passphrase: celesvaloper1q3v5cugc8cdpud87u4zwy0a74uxkk6u43cv6hd Enter

keyring

passphrase: celesvaloper1q3v5cugc8cdpud87u4zwy0a74uxkk6u43cv6hd To delegate tokens to thecelestiavaloper validator, as an example you can run:

bash celestia-appd

tx

staking

delegate

\ celestiavaloper1q3v5cugc8cdpud87u4zwy0a74uxkk6u4q4gx4p 1000000 utia

\ --from=VALIDATOR_WALLET --chain-id=mocha-4

\ --fees=21000utia celestia-appd

tx

staking

delegate

\ celestiavaloper1q3v5cugc8cdpud87u4zwy0a74uxkk6u4q4gx4p 1000000 utia

\ --from=VALIDATOR_WALLET --chain-id=mocha-4

\ --fees=21000utia If successful, you should see a similar output as:

console code: 0 codespace: "" data: "" gas_used: "0" gas_wanted: "0" height: "0" info: "" logs: [] raw_log: '[]' timestamp: "" tx: null txhash: code: 0 codespace: "" data: "" gas_used: "0" gas_wanted: "0" height: "0" info: "" logs: [] raw_log: '[]' timestamp: "" tx: null txhash: You can check if the TX hash went through using the block explorer by inputting thetxhash ID that was returned.

Optional: Deploy the celestia-node

Running a bridge node is critical to the Celestia network as it enables the data availability and consensus nodes to communicate with one another. It is recommended to support the data availability network, but is not required forcelestia-app .

If you are not running a bridge node, you can skip torun a validator node .

This section describes part 2 of Celestia validator node setup: running a Celestia bridge node daemon.

Install celestia-node

You canfollow the tutorial for installingcelestia-node

Initialize the bridge node

Run the following:

bash celestia

bridge

init

--core.ip

< UR I

celestia

bridge

init

--core.ip

< UR I

TIP

Refer tothe ports section of the celestia-node troubleshooting page for information on which ports are required to be open on your machine. If you need a list of RPC endpoints to connect to, you can find thelist on the Mocha testnet page orlist on the Arabica devnet page .

Run the bridge node

Run the following:

bash celestia

bridge

start celestia

bridge

start

Optional: start the bridge node with SystemD

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

You have successfully set up a bridge node that is syncing with the network.

Run the validator node

In order to start your validator node, run the following:

bash celestia-appd

start celestia-appd

start After completing all the necessary steps, you are now ready to run a validator! In order to create your validator onchain, follow the instructions below. Keep in mind that these steps are necessary ONLY if you want to participate in the consensus.

Pick amoniker name of your choice! This is the validator name that will show up on public dashboards and explorers.VALIDATOR_WALLET must be the same you defined previously. Parameter--min-self-delegation=1000000 defines the amount of tokens that are self delegated from your validator wallet.

Now, connect to the network of your choice.

You have the following option of connecting to list of networks shown below:

Continuing the validator tutorial, here are the steps to connect your validator to Mocha:

bash MONIKER = "your_moniker" VALIDATOR_WALLET = "validator"

celestia-appd

tx

staking

create-validator

\ --amount=1000000utia

\ --pubkey= (celestia-appd tendermint show-validator)

\ --moniker= MONIKER

\ --chain-id=mocha-4

\ --commission-rate=0.1

\ --commission-max-rate=0.2

\ --commission-max-change-rate=0.01

\ --min-self-delegation=1000000

\ --from= VALIDATOR_WALLET

\ --keyring-backend=test

\ --fees=21000utia

\ --gas=220000 MONIKER = "your_moniker" VALIDATOR_WALLET = "validator"

celestia-appd

tx

staking

create-validator

\ --amount=1000000utia

\ --pubkey= (celestia-appd tendermint show-validator)

\ --moniker= MONIKER

\ --chain-id=mocha-4

\ --commission-rate=0.1

\ --commission-max-rate=0.2

\ --commission-max-change-rate=0.01

\ --min-self-delegation=1000000

\ --from= VALIDATOR_WALLET

\ --keyring-backend=test

\ --fees=21000utia

\ --gas=220000 You will be prompted to confirm the transaction:

console confirm transaction before signing and broadcasting [y/N]: y confirm transaction before signing and broadcasting [y/N]: y Inputtingy should provide an output similar to:

console code: 0 codespace: "" data: "" gas_used: "0" gas_wanted: "0" height: "0" info: "" logs: [] raw_log: '[]' timestamp: "" tx: null txhash: code: 0 codespace: "" data: "" gas_used: "0" gas_wanted: "0" height: "0" info: "" logs: [] raw_log: '[]' timestamp: "" tx: null txhash: You should now be able to see your validator froma block explorer

Submit your validator information

After starting your node, please submit your node as a seed and peer to thenetworks repository .

Extra resources for consensus nodes

Optional: Reset network

This will delete all data folders so we can start fresh:

sh celestia-appd

tendermint

unsafe-reset-all

--home HOME /.celestia-app celestia-appd

tendermint

unsafe-reset-all

--home HOME /.celestia-app

Optional: Configuring an RPC endpoint

You can configure your full consensus node to be a public RPC endpoint. This allows it to accept connections from data availability nodes and serve requests for the data availability API.

Expose RPC

By default, the RPC service listens onlocalhost which means it can't be accessed from other machines. To make the RPC service available publicly, you need to bind it to a public IP or0.0.0.0 (which means listening on all available network interfaces).

You can do this by editing the config.toml file:

sh sed

-i

's#"tcp://127.0.0.1:26657"#"tcp://0.0.0.0:26657"#g'

~/.celestia-app/config/config.toml sed

-i

's#"tcp://127.0.0.1:26657"#"tcp://0.0.0.0:26657"#g'

~/.celestia-app/config/config.toml This command replaces thelocalhost IP address with0.0.0.0 , making the RPC service listen on all available network interfaces.

Note onexternal-address

Theexternal-address field in the configuration is used when your node is behind a NAT and you need to advertise a different address for peers to dial. Populating this field is not necessary for making the RPC endpoint public.

sh EXTERNAL-ADDRESS = (wget

-qO- eth0.me) sed

-i.bak

-e

"s/^external-address = ""/external-address = " EXTERNAL -ADDRESS:26656"/"

\ HOME /.celestia-app/config/config.toml EXTERNAL-ADDRESS = (wget

-qO- eth0.me) sed

-i.bak

-e

"s/^external-address = ""/external-address = " EXTERNAL -ADDRESS:26656"/"

\ HOME /.celestia-app/config/config.toml

Restart the node

After making these changes, restartcelestia-appd to load the new configurations.

Optional: Transaction indexer configuration options

This section guides you on how to configure yourconfig.toml file incelestia-app to select which transactions to index. Depending on the application's configuration, a node operator may decide which transactions to index.

The available options are:

1. null
2. : This option disables indexing. If you don't need to query transactions, you can choose this option to save space.
3. kv
4. (default): This is the simplest indexer, backed by key-value storage (defaults to levelDB; see DBBackend). Whenkv
5. is chosen,tx.height
6. andtx.hash
7. will always be indexed. This option is suitable for basic queries on transactions.
8. psql
9. : This indexer is backed by PostgreSQL. When psql is chosen,tx.height
10. andtx.hash
11. will always be indexed. This option is suitable for complex queries on transactions.

An example to set the value tokv inconfig.toml is:

toml indexer = "kv" indexer = "kv" Remember to restartcelestia-appd after making changes to the configuration to load the new settings.

Optional: Discard ABCI responses configuration

This section will guide you on how to configure yourconfig.toml file incelestia-app to manage the storage of ABCI responses. ABCI responses are the results of executing transactions and are used for/block_results RPC queries and to reindex events in the command-line tool.

Thediscard_abci_responses option allows you to control whether these responses are persisted in the state store:

- false
- (default): ABCI responses are stored in the state store. This ensures that ABCI responses are available for/block_results
- RPC queries and for reindexing events. However, it can consume a significant amount of disk space.
- true
- : ABCI responses are not stored in the state store. This can save a considerable amount of disk space, but/block_results
- RPC queries and event reindexing will not be available.

An example to set the value to false inconfig.toml is:

toml discard_abci_responses = false discard_abci_responses = false Remember to restartcelestia-appd after making changes to the configuration to load the new settings. [][[Edit this page on GitHub] Last updated: Previous page Bridge node Next page IBC relaying guide []]