Setting up a Celestia validator node

This tutorial will guide you through setting up a validator node on Celestia. Validator nodes allow you to participate in consensus in the Celestia network.

Hardware requirements

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

- Memory:16 GB RAM
- CPU:8 cores
- Disk:2 TB SSD Storage
- Bandwidth:1 Gbps for Download/1 Gbps for Upload

Setting up a validator node

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

First, follow the instructions onsetting up a consensus node .

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. Using an RPC of your own, or one fromMainnet Beta ,Mocha testnet orArabica devnet , initialize your node.

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

If you are running celestia-app v1.x.x:

sh celestia-appd

start celestia-appd

start If you are running celestia-app >= v2.0.0: then you'll want to start the node with a--v2-upgrade-height that is dependent on the network. The--v2-upgrade-height flag is only needed during the v2 upgrade height so after your node has executed the upgrade (e.g. you see the logupgraded from app version 1 to 2), you don't need to provide this flag for futurecelestia-appd start invocations.

Mainnet Beta

Mocha

Arabica sh celestia-appd

start

--v2-upgrade-height

2371495 celestia-appd

start

--v2-upgrade-height

2371495 sh celestia-appd

start

--v2-upgrade-height

2585031 celestia-appd

start

--v2-upgrade-height

2585031 sh celestia-appd

start

--v2-upgrade-height

1751707 celestia-appd

start

--v2-upgrade-height

1751707 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 .

Optional: Transaction indexer configuration options

Follow the instructions undertransaction indexer configuration options to configure yourconfig.toml file to select which transactions to index.

Additional resources

For additional resources, refer tothe extra resources for consensus nodessection of the consensus node page.

FAQ

+2/3 committed an invalid block: wrong Block.Header.Version

If you encounter an error like:

bash 2024-04-25

14 :48:24

6 :48PM

ERR

CONSENSUS

FAILURE!!!

err="+2/3 committed an invalid block: wrong Block.Header.Version. Expected {11 1}, got {11 2}"

module=consensus

stack="goroutine 214 [running]:\nruntime/debug.Stack()\n\t/usr/local/go/src/runtime/debug/stack.go:24 +0x64\ngithub.com/tendermint/tendermint/consensus.(*State).receiveRoutine.func2()\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:746 +0x44\npanic({0x1b91180?, 0x400153b240?})\n\t/usr/local/go/src/runtime/panic.go:770 +0x124\ngithub.com/tendermint/tendermint/consensus.(*State).finalizeCommit(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:1637 +0xd30\ngithub.com/tendermint/tendermint/consensus.(*State).tryFinalizeCommit(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:1606 +0x26c\ngithub.com/tendermint/tendermint/consensus.(*State).handleCompleteProposal(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:2001 +0x2d8\ngithub.com/tendermint/tendermint/consensus.(*State).handleMsg(0x400065ea88, {{0x2b30a00, 0x400143e048}, {0x40002a61b0, 0x28}})\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:856 +0x1c8\ngithub.com/tendermint/tendermint/consensus.(*State).receiveRoutine(0x400065ea88, 0x0)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:782 +0x2c4\ncreated by github.com/tendermint/tendermint/consensus.(*State).OnStart in goroutine 169\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:391 +0x110\n" 2024-04-25

14 :48:24

6 :48PM

ERR

CONSENSUS

FAILURE!!!

err="+2/3 committed an invalid block: wrong Block.Header.Version. Expected {11 1}, got {11 2}"

module=consensus

stack="goroutine 214 [running]:\nruntime/debug.Stack()\n\t/usr/local/go/src/runtime/debug/stack.go:24 +0x64\ngithub.com/tendermint/tendermint/consensus.(*State).receiveRoutine.func2()\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:746 +0x44\npanic({0x1b91180?, 0x400153b240?})\n\t/usr/local/go/src/runtime/panic.go:770 +0x124\ngithub.com/tendermint/tendermint/consensus.(State).finalizeCommit(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:1637 +0xd30\ngithub.com/tendermint/tendermint/consensus.(*State).tryFinalizeCommit(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:1606 +0x26c\ngithub.com/tendermint/tendermint/consensus.(State).handleCompleteProposal(0x400065ea88, 0x3)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:2001 +0x2d8\ngithub.com/tendermint/tendermint/consensus.(*State).handleMsg(0x400065ea88, {{0x2b30a00, 0x400143e048}, {0x40002a61b0, 0x28}})\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:856 +0x1c8\ngithub.com/tendermint/tendermint/consensus.(State).receiveRoutine(0x400065ea88, 0x0)\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:782 +0x2c4\ncreated by github.com/tendermint/tendermint/consensus.(*State).OnStart in goroutine 169\n\t/go/pkg/mod/github.com/celestiaorg/ /consensus/state.go:391 +0x110\n" then it is likely that the network has upgraded to a new app version but your consensus node was not prepared for the upgrade. To fix this, you'll need to update your binary to the latest version and restart your node with the relevant--v2-upgrade-height for the network you're running on. If your node still can't sync to the tip of the chain after the above steps, consider acelestia-appd tendermint reset-state to reset your node and start syncing from the genesis block.

1. [Optional] Back up your validator keys.
2. [Optional] Back up thedata/priv_validator_state.json
3. inside your CELESTIA_HOME directory.
4. Remove DBs from your CELESTIA_HOME directory via:celestia-appd tendermint reset-state
5. .
6. Remove thedata/application.db
7. inside your CELESTIA_HOME directory.
8. Download the latest binary for your network.
9. Restart your consensus node with the relevant--v2-upgrade-height
10. for the network you're running on. [[Edit this page on GitHub] Last updated: Previous page Consensus node Next page IBC relaying guide []