Create a wallet with celestia-node

This tutorial will go over using thecel-key utility to generate a wallet on celestia-node.

While this tutorial will go over installation process ofcel-key , it is recommended that you complete the following prerequisites first:

- Setting up your environment

Once you completed the prerequisite, you can proceed with this tutorial.

Using thecel-key

utility

Inside the celestia-node repository is a utility calledcel-key that uses the key utility provided by Cosmos-SDK under the hood. The utility can be used toadd ,delete , and manage keys for any DA node type(bridge || full || light) , or just keys in general.

Installation

You need to first pull down the celestia-node repository:

sh git

clone

https://github.com/celestiaorg/celestia-node.git cd

celestia-node/ git

clone

https://github.com/celestiaorg/celestia-node.git cd

celestia-node/ It can be built using either of the following commands:

sh

dumps binary in current working directory, accessible via ./cel-key

make

cel-key

dumps binary in current working directory, accessible via ./cel-key

make

cel-key or

sh

installs binary in GOBIN path, accessible via cel-key

make

install-key

installs binary in GOBIN path, accessible via cel-key

make

install-key For the purpose of this guide, we will use themake cel-key command.

Steps for generating node keys

To generate a key for a Celestia node, select the tab for your node type:

TIP

You do not need to declare a network for Mainnet Beta. Refer tothe chain ID section on the troubleshooting page for more information Bridge

Full

Light bash ./cel-key

add

< key-nam e

--keyring-backend

test

--node.type

bridge

\ --p2p.network

< networ k

```
    ./cel-key
```

add

< key-nam e

--keyring-backend

test

--node.type

bridge

\ --p2p.network

< networ k

```
    bash ./cel-key
```

add

< key-nam e

--keyring-backend

test

--node.type

full

\ --p2p.network

< networ k

```
    ./cel-key
```

add

< key-nam e

--keyring-backend

test

--node.type

full

\ --p2p.network

< networ k

bash ./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

This will load the key into the directory of the node.

Further flags you can use to customize your key are the following:

- --node.store
- : Specifies a different directory you can use to save your node data and configurations. Expects a path to a directory.
- --p2p.network
- : Specifies which network you want the key for. Values arearabica
- andmocha
- . Please note the default network will bemocha
- .

Keep in mind that your celestia-node will only pick up keys that are inside thenode.store directory under/keys so you should make sure to pointcel-key utility to the correct directory via thenode.store orp2p.network flags if you have specified a custom directory or network other than Mocha.

Also keep in mind that if you do not specify a network with--p2p.network , the default one will always bemocha .

Steps for exporting node keys

You can export a private key from the local keyring in encrypted and ASCII-armored format.

Bridge

Full

```
Light bash ./cel-key
export
< key-nam e
--keyring-backend
test
--node.type
bridge
\ --p2p.network
< networ k
     ./cel-key
export
< key-nam e
--keyring-backend
test
--node.type
bridge
\ --p2p.network
< networ k
     bash ./cel-key
export
< key-nam e
--keyring-backend
test
--node.type
full
\ --p2p.network
< networ k
     ./cel-key
export
< key-nam e
--keyring-backend
test
--node.type
full
\ --p2p.network
< networ k
     bash ./cel-key
```

export

< key-nam e

--keyring-backend

test

--node.type

light

\ --p2p.network

< networ k

    ./cel-key

export

< key-nam e

--keyring-backend

test

--node.type

light

\ --p2p.network

< networ k

Steps for importing node keys

To import from a mnemonic, use the following command, then enter your bip39 mnemonic:

Bridge

Full

Light bash ./cel-key

add

< key-nam e

--recover

--keyring-backend

test

\ --node.type

bridge

--p2p.network

< networ k

    ./cel-key

add

< key-nam e

--recover

--keyring-backend

test

```
\ --node.type
bridge
--p2p.network
< networ k
        bash ./cel-key
add
< key-nam e
--recover
--keyring-backend
test
\ --node.type
full
--p2p.network
< networ k
        ./cel-key
add
< key-nam e
--recover
--keyring-backend
test
\ --node.type
full
--p2p.network
< networ k
        bash ./cel-key
add
< key-nam e
--recover
--keyring-backend
test
\ --node.type
light
--p2p.network
< networ k
        ./cel-key
add
< key-nam e
```

--recover

--keyring-backend

test

\ --node.type

light

--p2p.network

< networ k

View all options forcel-key

sh ./cel-key

--help ./cel-key

--help

Docker andcel-key

Prerequisites

- Docker installed on your machine
- Understanding of theguide on how to run celestia-node with Docker
- .

Running your node

Run the Docker image (in this example, we are using a light node on Mocha testnet):

bash docker

run

--name

celestia-node

-e

NODE_TYPE=light

-e

P2P_NETWORK=mocha

-p

26659 :26659

\ ghcr.io/celestiaorg/celestia-node:v0.13.1 celestia

light

start

\ --core.ip rpc-mocha.pops.one

--p2p.network

mocha docker

run

--name

celestia-node

-e

NODE_TYPE=light

-e

P2P_NETWORK=mocha

-p

26659 :26659

\ ghcr.io/celestiaorg/celestia-node:v0.13.1 celestia

light

start

\ --core.ip rpc-mocha.pops.one

--p2p.network

mocha TIP

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

You do not need to declare a network for Mainnet Beta. Refer tothe chain ID section on the troubleshooting page for more information List active containers in another window with:

bash docker

ps docker

ps The response will look like:

bash CONTAINER

ID

IMAGE

COMMAND

CREATED

STATUS

PORTS

NAMES < container-id

> celestia-node "/entrypoint.sh cele…" 22 seconds ago Up 21 seconds 2121/tcp docker-compose-test-celestia-1 CONTAINER

ID

IMAGE

COMMAND

CREATED

STATUS

PORTS

NAMES < container-id

> celestia-node "/entrypoint.sh cele…" 22 seconds ago Up 21 seconds 2121/tcp docker-compose-test-celestia-1

Interact with the container by replacing for the container ID:

bash docker

exec

-ti

< container-i d

/bin/bash docker

exec

-ti

< container-i d

/bin/bash Now, interact withcel-key to check for the key that was autogenerated when you started the node:

bash ./cel-key

list

--keyring-backend

test

--node.type

light ./cel-key

list

--keyring-backend

test

--node.type

light You can also export your key from the container. In the next section, you'll learn how to mount existing keys to the container.

Mounting existing keys to container

In this example, we'll be mounting an existing key to the container. We're also using an existing image called celestia-node. This will mount the entire/.celestia-light-/keys directory to your image, or on Mainnet Beta the/.celestia-light/keys directory.

Write adocker-compose.yml to accomplish this:

yaml version : "3.8" services : celestia : image : celestia-node environment : - NODE_TYPE=light command : celestia light start --core.ip rpc-mocha.pops.one --p2p.network mocha --keyring.accname my_celes_key volumes : - {PWD}/keys:/root/.celestia-light-mocha-4/keys ports : - 26659:26659 version : "3.8" services : celestia : image : celestia-node environment : - NODE_TYPE=light command : celestia light start --core.ip rpc-mocha.pops.one --p2p.network mocha --keyring.accname my_celes_key volumes : - {PWD}/keys:/root/.celestia-light-mocha-4/keys ports : - 26659:26659 Start the container by running the following command in the directory with yourdocker-compose.yml :

bash docker-compose

up docker-compose

up List active containers in another window with:

bash docker

ps docker

ps The response will look like:

bash CONTAINER

ID

IMAGE

COMMAND

CREATED

STATUS

PORTS

NAMES < container-id

    celestia-node "/entrypoint.sh cele…" 22 seconds ago Up 21 seconds 2121/tcp docker-compose-test-celestia-1
    CONTAINER

ID

IMAGE

COMMAND

CREATED

STATUS

PORTS

NAMES < container-id

    celestia-node "/entrypoint.sh cele…" 22 seconds ago Up 21 seconds 2121/tcp docker-compose-test-celestia-1
    Interact with the container by replacing for the container ID:

bash docker

exec

-ti

< container-i d

/bin/bash docker

exec

-ti

< container-i d

/bin/bash Now, interact withcel-key to check your address matches the address you expect with the key you mounted:

bash root@:/#

./cel-key

list

--keyring-backend

test

--node.type

light using

directory:

~/.celestia-light-mocha-4/keys

address:

celestia1wkhyhr7ngf0ayqlpnsnxg4d72hfs5453dvunm9 name:

my_celes_key pubkey:

'{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"A1/NsoY0RGL7Hqt4VWLg441GQKJsZ2fBUnZXipgns8oV"}' type : local root@:/#

./cel-key

list

--keyring-backend

test

--node.type

light using

directory:

~/.celestia-light-mocha-4/keys

address:

celestia1wkhyhr7ngf0ayqlpnsnxg4d72hfs5453dvunm9 name:

my_celes_key pubkey:

'{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"A1/NsoY0RGL7Hqt4VWLg441GQKJsZ2fBUnZXipgns8oV"}' type : local [][Edit this page on GitHub] Last updated: Previous page Rollkit Next page Integrating Wallets for developers []