State-synchronization (state-sync)

State-sync is a feature which allows you to quickly bootstrap a new node by allowing it to pull astate snapshot taken by other nodes.

⚠ The state-sync feature is only available fromfetchd v0.10.6 and later on. Prior versions needed to either synchronize from scratch or restore achain snapshot ↗ , which both could take hours before having the node fully synchronized. It now takes only a few minutes before having an operational node with state-sync,

Configuring the new node

If you want to instruct the node to sync itself using a state-sync snapshot, it need some configuration in the~/.fetchd/config/config.toml file. To configure the new node, open this file in an editor and lookup for thestatesync section.

By default, it should looks like this:

State Sync Configuration Options

[statesync]

State sync rapidly bootstraps a new node by discovering, fetching, and restoring a state machine

snapshot from peers instead of fetching and replaying historical blocks. Requires some peers in

the network to take and serve state machine snapshots. State sync is not attempted if the node

has any local state (LastBlockHeight > 0). The node will have a truncated block history,

starting from the height of the snapshot.

enable = false

RPC servers (comma-separated) for light client verification of the synced state machine and

retrieval of state data for node bootstrapping. Also needs a trusted height and corresponding

header hash obtained from a trusted source, and a period during which validators can be trusted.

For Cosmos SDK-based chains, trust_period should usually be about 2/3 of the unbonding time (~2

weeks) during which they can be financially punished (slashed) for misbehavior.

rpc_servers = "" trust_height = 0 trust_hash = "" trust_period = "168h0m0s"

Time to spend discovering snapshots before initiating a restore.

discovery_time = "15s"

Temporary directory for state sync snapshot chunks, defaults to the OS tempdir (typically /tmp).

Will create a new, randomly named directory within, and remove it when done.

temp_dir = ""

The timeout duration before re-requesting a chunk, possibly from a different

peer (default: 1 minute).

chunk_request_timeout = "10s"

The number of concurrent chunk fetchers to run (default: 1).

chunk_fetchers = "4" A few changes will be needed:

- First, you will need to setenable = true
- to activate the state-sync engine.
- Then, you need to provideat least 2
- rpc servers. A good place to find some is thecosmos chain registry ↗(opens in a new tab)
- . Servers must be comma separated without space (i.e.,rpc_servers = "https://rpc-fetchhub.fetch.ai:443,https://fetch-rpc.polkachu.com:443"
-). These servers will be used to verify the snapshots, so make sure you trust them enough for this.
- You will also need arecent
- trust_height
- andtrust_hash
- . Recent means it must be contained in thetrust_period
- (168 hours, or ~1 week old by default). These can be obtained from a RPC serveryou trust to provide you correct data
- (and the 2nd RPC server fromrpc_servers
- will be charged of confirming that the data are correct).
- And last, you need to setchunk_request_timeout
- to60s
- (the10s
- default value seems too short and can lead to "context deadline exceeded" timeout errors when verifying the hashes).

You can then retrieve the correct value for aFetch.ai RPC server , and the currentnetwork height , using:

curl

https://rpc-fetchhub.fetch.ai:443/block

|

jq

-r

'{"trusted_hash": .result.block_id.hash, "trusted_height": .result.block.header.height}' { "trusted_hash" :

"...some hash..." , "trusted_height" :

"...some height..." } and set thetrusted_hash andtrusted_height values in the config file.

Once this is set, you will need to make sure you have the correct genesis by downloading it from the RPC node:

curl

https://raw.githubusercontent.com/fetchai/genesis-fetchhub/fetchhub-4/fetchhub-4/data/genesis_migrated_5300200.json

--output

~/.fetchd/config/genesis.json You can then start the node using the seeds from the chain-registry:

fetchd

start

--p2p.seeds= "17693da418c15c95d629994a320e2c4f51a8069b@connect-fetchhub.fetch.ai:36456,a575c681c2861fe945f77cb3aba0357da294f1f2@connect-fetchhub.fetch.ai:36457,d7cda986c9f59ab9e05058a803c3d0300d15d8da@connect-fetchhub.fetch.ai:36458" After the node is initialized, it will start searching for available snapshots, and it should print log messages similar to the ones below:

8:22AM INF Discovered new snapshot format=1 hash="□/��\r□F#C(pD□<□□\x066□□\x1f□□\x1f<i□□" height=2000 module=statesync 8:22AM INF Discovered new snapshot format=1 hash="F□=\x05□Gh□{□| □□□□□,□Q'□=]\x1a□□b□Q" height=1900 module=statesync The node will select the one with theheight value the closest to the tip of the chain, and it will then start restoring the state, and finish synchronizing the few blocks remaining. If it fails to verify any blocks or hash when restoring, it will attempt to restore the next available snapshot, and, if no more are available, it will fallback in discovery mode until an usable snapshot is made available.

Configure an existing node to provide snapshots

You can configure existing nodes to create snapshot from which they can start on. This requires changes in the~/.fetchd/config/app.toml file, in thestate-sync section:

State Sync Configuration

State sync snapshots allow other nodes to rapidly join the network without replaying historical

blocks, instead downloading and applying a snapshot of the application state at a given height.

[state-sync]

snapshot-interval specifies the block interval at which local state sync snapshots are

taken (0 to disable). Must be a multiple of pruning-keep-every.

snapshot-interval = 0

snapshot-keep-recent specifies the number of recent snapshots to keep and serve (0 to keep all).

snapshot-keep-recent = 2 Here,snapshot-interval must be set to a number of blocks between each snapshot creation and it must be a multiple of your node pruning settings (default is 100, so valid values are 100, 1000, 700...).

The number of snapshots to keep can be set withsnapshot-keep-recent .

Was this page helpful?

How to use chain state snapshots How to set up a validator node