Nitro database snapshots

Nitro stores the chain state and data in a database in the local filesystem. When starting Nitro for the first time, it will initialize an empty database by default and start processing transactions from Genesis. It takes a long time for the node to sync from Genesis, so starting from a database snapshot is advisable instead. Moreover, for the Arb1 chain, you must start from a snapshot because Nitro cannot process transactions from the Classic Arbitrum node.

Supply the snapshot URL to Nitro

There are multiple ways to supply Nitro with the database snapshot. The most straightforward way is to provide the configuration, so Nitro downloads the snapshot by itself. It is also possible to download the database manually and supply it to Nitro.

Downloading the latest snapshot

Nitro has a CLI configuration for downloading the latest snapshot from a remote server. Set the flag--init.latest to eitherarchive ,pruned , orgenesis , and Nitro will download the preferred snapshot. You may also change the--init.latest-base flag to set the base URL when searching for the latest snapshot.

How it works

When searching for the latest snapshot, Nitro uses the chain name provided in--chain.name . Make sure to set it correctly; otherwise, Nitro might be unable to find the snapshot. Nitro will look for a remote file in//latest-.txt , where is the option supplied to--init.latest . This file should contain either the path or the full URL to the snapshot; if it only contains the path, Nitro will use the as the base URL.

After finding the latest snapshot URL, Nitro will download the archive and temporarily store it in the directory specified in--init.download-path . Nitro looks for a SHA256 checksum on the remote server and verifies the checksum of the snapshot after finishing the download. (It is possible to disable this feature by setting--init.validate-checksum to false.)

The snapshot can be a single archive file or a series of parts. Nitro first tries to download the snapshot as a single archive. In this case, Nitro will look for a checksum file in.sha256 . If the remote server returns not found (404 status code), Nitro will proceed to download the snapshot in parts. When downloading in parts, Nitro will look for a manifest file in.manifest.txt containing each part's name and checksum. In this case, Nitro will download each part in the manifest file and concatenate them into a single archive.

Finally, Nitro decompresses and extracts the snapshot archive, placing it in the database directory. Nitro will delete the archive after extracting it, so if you need to set up multiple nodes with the same snapshot, consider downloading it manually, as explained below.

Downloading the snapshot from a URL

Instead of letting Nitro search for the latest snapshot, you can provide a specific URL to download by setting the flag--init.url with the snapshot URL. If the URL points to a remote server, it should start with thehttps:// protocol definition. Given the URL, Nitro will download the snapshot as described in the "Downloading the Latest Snapshot" section.

Nitro also supports importing files from the local file system. In this case, you should provide the file path to--init.url starting with the prefixfile:// followed by the file path. Beware that when running Nitro inside a Docker container, you must mount a volume containing the provided snapshot using the docker flag-v (seeDocker documentation). Otherwise, the Nitro container running inside Docker won’t be able to find the snapshot in your local filesystem.

Downloading the snapshot manually

It is possible to download the snapshot manually and supply the archive instead of having Nitro download it.

The first step is downloading the snapshot. The command below illustrates how to do that on the command line using wget. The-c flag tells the wget to continue the download from where it left off, which is helpful because snapshots can be huge files, and the download can fail mid-way. The-P flag tells wget to place the snapshot on the temporary dir.

wget

-c

-P /tmp " SNAPSHOT_URL " After downloading the snapshot, make sure to verify whether the checksum matches the one provided by the remote server. To fetch the checksum, you may run the command below.

wget

-q

-O

" SNAPSHOT_URL " .sha256 Once you know the expected snapshot checksum, run the command below to compute the checksum of the downloaded snapshot. Then, compare both and see if they are the same. If they are not the same, consider redownloading the snapshot. You must provide a valid snapshot to Nitro; otherwise, it won't work properly.

sha256sum PATH_TO_SNAPSHOT Finally, you can provide a path to the downloaded snapshot archive to Nitro using the--init.url flag, as described in the "Download the Snapshot from a URL" section.

Downloading snapshot parts

If the snapshot is divided into parts, you should first download the manifest file in.manifest.txt . This manifest contains the names and checksums of each part. For instance, the snippet below shows how the manifest file should look. You may use the commands described previously to download each part of the snapshot and verify their checksums.

a938e029605b81e03cd4b9a916c52d96d74c985ac264e2f298b90495c619af74 archive.tar.part0
9e095ce82e70fa62bb6e7b4421e7f2c04b2cd9e21d2bc62cbbaaeb877408357b archive.tar.part1
e92172d6eaf770a76c7477e6768f742fc51555a5050de606bd0f837e59c7a61d archive.tar.part2
d1b6fb9aeeb23903cdbb2a7cca8e6909bff4ee8e51c8a5acac2a142b3e3a5437 archive.tar.part3
f37e4552453202f2044e58b307bab7e466205bd280426abbc84f8646c6430cfa archive.tar.part4
972c5f513faca6ac4fadd22c70bea97707c6d38e9a646432bc311f0ca10497ed archive.tar.part5 After downloading all the parts and verifying their checksums, you may use the command below to join them into a single archive.

cat archive.tar.part*

archive.tar

Extracting the snapshot manually

It is also possible to extract the snapshot archive and place the files manually. First, you need to download the snapshot archive as described in "Manually Downloading the Snapshot". Then, create the directory where Nitro will look for its database. By default, Nitro stores the database onHOME/.arbitrumCHAIN/nitro . Move the archive to this directory and extract it. The commands below exemplify this process for the Arbitrum Sepolia chain.

export

CHAIN

sepolia-rollup export

ARCHIVE_PATH

/tmp/archive.tar.gz mkdir

-p

HOME /.arbitrum/ CHAIN /nitro cd

HOME /.arbitrum/ CHAIN /nitro tar zxfv ARCHIVE_PATH You should see the following subdirectories in this directory after extracting the archive.

arbitrumdata l2chaindata nodes

Creating a snapshot

To generate a snapshot for the Nitro database, you first need to stop the process gracefully. You must not generate the snapshot while Nitro runs because the database might be in an intermediary state. Nitro should print logs like the ones described below when stopping.

^CINFO [08-22 | 18 :10:55.015] shutting down because of sigint INFO [08-22 | 18 :10:55.016] delayed sequencer: context done

err

"context canceled" INFO [08-22 | 18 :10:55.016] rpc response method = eth_getBlockByNumber logId = 123

err

"context canceled"

result

null attempt = 0

args

"[\" 0x405661 \" , false]" INFO [08-22 | 18 :10:55.293] Writing cached state to disk block = 39988

hash

8bebf3 .. 939ab2 root = 4f7a22 .. 00c334 INFO [08-22 | 18 :10:55.297] Persisted trie from memory database nodes = 643

size

156 .31KiB time = 3 .673459ms gcnodes = 329

gcsize

102 .61KiB gctime = "248.708µs"

livenodes

2448

livesize

806 .00KiB INFO [08-22 | 18 :10:55.297] Writing cached state to disk block = 39987

hash

ddcd60 .. fe0fc3 root = d6973e .. 7b9265 INFO [08-22 | 18 :10:55.298] Persisted trie from memory database nodes = 34

size

11 .19KiB time = "283.875µs"

gcnodes

0

gcsize

0 .00B gctime = 0s livenodes = 2414

livesize

794 .81KiB INFO [08-22 | 18 :10:55.298] Writing cached state to disk block = 39861

hash

2a9dd3 .. f00ff0 root = 139d5a .. d6bf21 INFO [08-22 | 18 :10:55.298] Persisted trie from memory database nodes = 73

size

24 .88KiB time = "502.916µs"

gcnodes

0

gcsize

0 .00B gctime = 0s livenodes = 2341

livesize

769 .93KiB INFO [08-22 | 18 :10:55.299] Writing cached state to disk block = 39861

hash

2a9dd3 .. f00ff0 root = 139d5a .. d6bf21 INFO [08-22 | 18 :10:55.299] Persisted trie from memory database nodes = 0

size

0 .00B time = "1.417µs"

gcnodes

0

gcsize

0 .00B gctime = 0s livenodes = 2341

livesize

769 .93KiB INFO [08-22 | 18 :10:55.299] Writing snapshot state to disk root = bd18ce .. 3b0763 INFO [08-22 | 18 :10:55.299] Persisted trie from memory database nodes = 0

size

0 .00B time = "1.125µs"

gcnodes

0

gcsize

0 .00B gctime = 0s livenodes = 2341

livesize

769 .93KiB INFO [08-22 | 18 :10:55.304] Blockchain stopped After nitro stops, go to the database directory and generate an archive file for the directoriesarbitrumdata ,l2chaindata , andnodes . By default, the database directory for nitro isHOME/.arbitrumCHAIN/nitro . The commands below exemplify how to generate the snapshot for Nitro.

export

CHAIN

sepolia-rollup export

ARCHIVE_PATH

/tmp/archive.tar.gz cd

HOME /.arbitrum/ CHAIN /nitro tar zcfv ARCHIVE_PATH arbitrumdata l2chaindata nodes This command purposely omits thewasm directory from the snapshot archive. Thewasm contains native-code executables, so it might be a security concern for users downloading the snapshot. If the user downloading the snapshot trusts you, or if you are storing it for your own use, you may include thewasm directory in it.

Optional: divide it into parts

It is possible to divide the snapshot into smaller parts to facilitate its download. This is particularly useful for archive snapshots of heavily used chains, such as arb1. These kinds of snapshots can reach terabytes, so dividing them into smaller parts is helpful. The snippet below illustrates how to divide the snapshot into parts using the split command. The-b argument tells the split to divide the snapshot into 100 GB parts. The-d argument tells split to enumerate the parts using a numeric suffix instead of an alphabetic one.

split

-b 100g -d archive.tar.gz archive.tar.gz.part After dividing it into parts, you should generate the manifest file containing the parts' names and checksums. Nitro will use these files to know how many parts there are and to validate their checksum. The command below exemplifies how to do that. Edit this page Last updatedonJan 27, 2025 Previous Migrate to Nitro from Classic Next Troubleshooting