How to migrate state and history from a classic (pre-Nitro) node to a Nitro node

When running a Nitro node for the first time on a chain that produced classic blocks in the past (like Arbitrum One), you need to initialize its database to, at least, the state of the chain after executing the last classic block. The common, and recommended, way of doing that is to provide a database snapshot using the--init.url option (as mentioned in How to run a full node (Nitro)). In this how-to we show you an alternative way for doing that, migrating the state and history of the chain from a fully synced classic node.

Is this How-to for you? As mentioned, the recommended way of initializing a Nitro node is by using a pre-initialized database snapshot with the--init.url option. This guide is for those that are interested in re-creating the full state of the chain from the genesis block using their own classic node.

Keep in mind that this process only applies to Arbitrum One. Other Arbitrum chains didn't produce classic blocks in the past, they started as Nitro chains.

Prerequisites

To successfully migrate the state and history of the chain from a classic (pre-Nitro) node to a Nitro node, you'll need:

- A fully synced classic node: you can find instructions on how to run a classic node in this page
- .
- A clean, uninitialized Nitro node: you can find instructions on how to set up a Nitro node in this page
- .

Step 1: Enable export options in your classic node

Launch your classic node with the option--node.rpc.nitroexport.enable=true . All exported data will be written to directory "nitroexport" under the classic instance directory (e.g.{HOME}/.arbitrum/mainnet/nitroexport). Make sure the classic node has read the entire rollup state.

Caution Enabling the export options is only recommended for nodes with no public/external interfaces. Exported file contents are not deterministic Exporting the state of your own classic node should produce the same state as using files supplied by the Arbitrum Foundation (i.e. the same genesis blockhash). However, multiple exports of the same state will not necessarily create identical intermediate files. For example, state export is done in parallel, so the order of entries in the file is not deterministic.

Step 2: Export information from your classic node

Block & transaction history

These are block headers, transactions and receipts executed in the classic node. Nitro node uses the history to be able to answer simple requests, likeeth_getTransactionReceipt , from the classic history. The last block in the chain is the only one that affects the genesis block: timestamp is copied from the last block, and parentHash is taken from the last block's blockHash.

- Call the RPC methodarb_exportHistory
- with parameter"latest"
- to initiate history export. It will return immediately.
- Callingarb_exportHistoryStatus
- will return the latest block exported, or an error if the export failed.
- Data will be stored in the directorynitroexport/nitro/l2chaindata/ancient
- .

Rollup state

The rollup state is exported as a series of JSON files. State read from these JSON files will be added to Nitro's genesis block.

- Call the RPC methodarb_exportState
- with parameter"latest"
- to initiate state export. Unless disconnected, this will only return after the state export is done.
- Data will be stored in the directorynitroexport/state/
- .

Outbox messages (optional)

This data does not impact consensus and is optional. It allows a Nitro node to provide the information required when executing a withdrawal made on the classic rollup.

- Call the RPC methodarb_exportOutbox
- with parameter"0xffffffffffffffff"
- to initiate outbox export. It will return immediately.
- Callingarb_exportOutboxStatus
- will return the latest outbox batch exported, or an error if the export failed.
- Data will be stored in the directorynitroexport/nitro/classic-msg
- .

Step 3: Initialize your Nitro node importing the exported data

- Place thel2chaindata
- andclassic-msg
- (if exported) directories in Nitro's instance directory (e.g.{HOME}/.arbitrum/arb1-nitro/
-).
- Launch the Nitro node with the argument--init.import-file=/path/to/state/index.json

Caution This state import operation requires more resources than a regular run of a Nitro node.

Other useful Nitro options

Flag Description --init.accounts-per-sync Allows the node to make partial database writes to hard-disk during initialization, allowing memory to be freed. This should be used if memory load is very high. A reasonable initial value to try would be 100000. Systems with constrained memory might require a lower value. --init.then-quit Causes the node to quit after initialization is done. --init.force For an already-initialized node, forces the node to recalculate Nitro's genesis block. If the genesis blockhash doesn't match what's in the database, the node will panic.

See also

- How to run a full node (Nitro)
- How to run a full node (Classic, pre-Nitro)Edit this page Last updatedonMar 7, 2024 Previous How to build Nitro locally (Debian, Ubuntu, MacOS) Next Troubleshooting: Run a node