Mainnet 0

Espresso Mainnet 0 release — October 2024

Mainnet 0 marks the production release of the Espresso Network. This release is an important step towards enabling Espresso's vision of an ecosystem of open, composable and permissionless applications.

Espresso's Global Confirmation Layer will support applications such as rollups with faster bridging, and lays the groundwork for chains to improve coordination amongst each other, enhancing cross-chain interoperability.

Underpinning the Espresso Network is HotShot, which will be run in a production setting for the first time. During the initial stages of Mainnet 0, HotShot will be run by a set of 20 node operators, running 100 nodes in total. An important upcoming milestone will be to open up participation to more node operators and increase economic security through proof-of-stake.

During the initial release, block size has conservatively been set to 1 MB, with plans to scale up to 5 MB in the short term if there is sufficient demand. We expect it to take around ~8 seconds for a transaction to be confirmed in the initial release. In the short term, an update to HotShot will implement ideas from HotStuff-2 to reduce this time to ~5-6 seconds. In the medium term, a VID update is in the works that will enable confirmation latency of ~3 seconds, even as we scale to thousands of consensus nodes. This will also enable us to increase the blocksize further without greatly impacting latency.

You can track activity on Mainnet in our block explorer, and you can interact with it via the public API endpoint.

Running a Mainnet 0 Node

This page provides the specific configuration used to run different types of nodes in Mainnet 0.

Note: during Mainnet 0 only a fixed set of preregistered operators can run a node. The Espresso Network will upgrade to proof-of-stake in a later release.

👊 TL;DR: For operator running Decaf nodes, the critical changes from that configuration are as follows:

- All URLs supplied by Espresso have changed
- The new container image version is 20241120-patch5
- The JSON-RPC endpoint specified by ESPRESSO_SEQUENCER_L1_PROVIDER should be an Ethereum mainnet endpoint instead of Sepolia

The container image to use for this deployment is

- ghcr.io/espressosystems/espresso-sequencer/sequencer:20241120-patch5 (if using Espresso's images)
- built off of the branch `20241120-patch5` (if building from source)

The configuration for all node types includes ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/mainnet.toml. This file is built into the

official Docker images. Operators building their own images will need to ensure this file is included and their nodes are pointed at it.

1. Regular Node

Command

Copy

sequencer -- http -- catchup -- status

Environment

Same for all nodes

Copy

ESPRESSO_SEQUENCER_ORCHESTRATOR_URL=https://orchestrator-kdrhoi6lwz.main.net.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.main.net.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.main.net.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/mainnet.toml

RUST_LOG="warn,libp2p=off"

RUST_LOG_FORMAT="json"

At least one state peer is required. The following URL provided by Espresso works.

Optionally, add endpoints for additional peers, separated by commas.

ESPRESSO_SEQUENCER_STATE_PEERS=https://query.main.net.espresso.network

Chosen by operators

Copy

JSON-RPC endpoint for Ethereum Mainnet

ESPRESSO_SEQUENCER_L1_PROVIDER # e.g. https://mainnet.infura.io/v3/<API-KEY>

Port on which to host metrics and healthchecks

ESPRESSO_SEQUENCER_API_PORT # e.g. 80

Path in container to store consensus state

ESPRESSO_SEQUENCER_STORAGE_PATH # e.g. /mount/sequencer/store/

Path in container to keystore

ESPRESSO_SEQUENCER_KEY_FILE # e.g. /mount/sequencer/keys/0.env

The address to bind Libp2p to in host:port form. Other nodes should be able to

access this; i.e. port must be open for UDP.

ESPRESSO_SEQUENCER_LIBP2P_BIND_ADDRESS

The address we should advertise to other nodes as being our Libp2p endpoint

(in host:port form). It should resolve a connection to the above bind address; i.e.

should use public IP address or hostname, and forward to the port given in the bind

address.

ESPRESSO_SEQUENCER_LIBP2P_ADVERTISE_ADDRESS

Volumes

- $ESPRESSO_SEQUENCER_STORAGE_PATH
- $ESPRESSO_SEQUENCER_KEY_FILE

2. DA Node

Requires operator to additionally run a Postgres server

Command

sequencer -- storage-sql -- http -- catchup -- status -- query

Environment

Same for all nodes

Copy

ESPRESSO_SEQUENCER_ORCHESTRATOR_URL=https://orchestrator-kdrhoi6lwz.main.net.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.main.net.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.main.net.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/mainnet.toml

ESPRESSO_SEQUENCER_POSTGRES_PRUNE="true"

ESPRESSO_SEQUENCER_IS_DA="true"

RUST_LOG="warn,libp2p=off"

RUST_LOG_FORMAT="json"

At least one state peer is required. The following URL provided by Espresso works.

Optionally, add endpoints for additional peers, separated by commas.

ESPRESSO_SEQUENCER_STATE_PEERS=https://query.main.net.espresso.network

ESPRESSO_SEQUENCER_API_PEERS=https://query.main.net.espresso.network

Chosen by operators

Copy

JSON-RPC endpoint for Ethereum Mainnet

ESPRESSO_SEQUENCER_L1_PROVIDER # e.g. https://mainnet.infura.io/v3/<API-KEY>

Port on which to host metrics, healthchecks, and DA API

ESPRESSO_SEQUENCER_API_PORT # e.g. 80

Path in container to keystore

ESPRESSO_SEQUENCER_KEY_FILE # e.g. /mount/sequencer/keys/0.env

Connection to Postgres

ESPRESSO_SEQUENCER_POSTGRES_HOST

ESPRESSO_SEQUENCER_POSTGRES_USER

ESPRESSO_SEQUENCER_POSTGRES_PASSWORD

The address to bind Libp2p to in host:port form. Other nodes should be able to

access this; i.e. port must be open for UDP.

ESPRESSO_SEQUENCER_LIBP2P_BIND_ADDRESS

The address we should advertise to other nodes as being our Libp2p endpoint

(in host:port form). It should resolve a connection to the above bind address; i.e.

should use public IP address or hostname, and forward to the port given in the bind

address.

ESPRESSO_SEQUENCER_LIBP2P_ADVERTISE_ADDRESS

Volumes

- $ESPRESSO_SEQUENCER_KEY_FILE

Requires operator to additionally run a Postgres server

Command

sequencer -- storage-sql -- http -- catchup -- status -- query -- state

Environment

Same for all nodes

Copy

ESPRESSO_SEQUENCER_ORCHESTRATOR_URL=https://orchestrator-kdrhoi6lwz.main.net.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.main.net.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.main.net.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/mainnet.toml

ESPRESSO_SEQUENCER_IS_DA=true

ESPRESSO_SEQUENCER_ARCHIVE=true

RUST_LOG="warn,libp2p=off"

RUST_LOG_FORMAT="json"

At least one state peer is required. The following URL provided by Espresso works.

Optionally, add endpoints for additional peers, separated by commas.

ESPRESSO_SEQUENCER_STATE_PEERS=https://query.main.net.espresso.network

ESPRESSO_SEQUENCER_API_PEERS=https://query.main.net.espresso.network

Chosen by operators

Copy

JSON-RPC endpoint for Ethereum Mainnet

ESPRESSO_SEQUENCER_L1_PROVIDER # e.g. https://mainnet.infura.io/v3/<API-KEY>

Port on which to host metrics, healthchecks, and query API

ESPRESSO_SEQUENCER_API_PORT # e.g. 80

Path in container to keystore

ESPRESSO_SEQUENCER_KEY_FILE # e.g. /mount/sequencer/keys/0.env

Connection to Postgres

ESPRESSO_SEQUENCER_POSTGRES_HOST

ESPRESSO_SEQUENCER_POSTGRES_USER

ESPRESSO_SEQUENCER_POSTGRES_PASSWORD

The address to bind Libp2p to in host:port form. Other nodes should be able to

access this; i.e. port must be open for UDP.

ESPRESSO_SEQUENCER_LIBP2P_BIND_ADDRESS

The address we should advertise to other nodes as being our Libp2p endpoint

(in host:port form). It should resolve a connection to the above bind address; i.e.

should use public IP address or hostname, and forward to the port given in the bind

address.

ESPRESSO_SEQUENCER_LIBP2P_ADVERTISE_ADDRESS

Contracts

Espresso Mainnet maintains light client contracts that rollups and other applications integrating with Espresso can use to read the state of the Espresso network in a trust-minimized way.

- Light client on Ethereum (for L2s integrating Espresso): 0x95ca91cea73239b15e5d2e5a74d02d6b5e0ae458

- Light client on Arbitrum (for Orbit L3s integrating Espresso): 0x47495bb99cccbb1bda9f15b32b69093137f886db