Testnets

Overview of current and past testnets

Espresso Systems has released and deployed several testnets since 2022:

- Decaf, our ongoing persistent testnet
- Cappuccino (May 2024)
- Gibraltar (January 2024)
- Cortado (September 2023)
- Doppio (July 2023)
- Americano (November 2022)

Decaf Testnet Release

Espresso Persistent Testnet (Decaf) — September 2024

In September of 2024, we launched Decaf, which will run as a persistent testnet of the Espresso system alongside the upcoming Mainnet release.

Decaf continues the process of decentralizing Espresso, by onboarding an additional 13 node operators, bringing the total operator set to 24. These 24 operators will run 100 geographically distributed nodes, all participating in HotShot consensus together. You can track activity on the Decaf testnet in our block explorer, and you can interact with Decaf via the public API endpoint.

Running a Node

Configuration for Decaf nodes

Decaf node operators are limited to a select group. If you are interested in running a node in a future release of Espresso, contact us.

This page give the configuration used to run different types of nodes in the Decaf testnet. For general information on running an Espresso node, see Running an Espresso Node.

All nodes in Decaf use the ghcr.io/espressosystems/espresso-sequencer/sequencer:20250228 Docker image, or an equivalent image built from source. Depending on the type of node, the configuration varies.

The configuration for all node types includes ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/decaf.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.

Regular Node

Command

sequencer -- http -- catchup -- status

Environment

Same for all nodes

Copy

ESPRESSO_SEQUENCER_ORCHESTRATOR_URL=https://orchestrator-UZAFTUIMZOT.decaf.testnet.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.decaf.testnet.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.decaf.testnet.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/decaf.toml

RUST_LOG="warn,hotshot_libp2p_networking=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.decaf.testnet.espresso.network

Chosen by operators

Copy

An HTTP JSON-RPC endpoint for Sepolia testnet. This is required

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

A `ws://` or `wss://` endpoint for Sepolia testnet. This is optional but

recommended since it decreases the load on your provider.

ESPRESSO_SEQUENCER_L1_WS_PROVIDER # e.g. wss://sepolia.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

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-UZAFTUIMZOT.decaf.testnet.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.decaf.testnet.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.decaf.testnet.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/decaf.toml

ESPRESSO_SEQUENCER_POSTGRES_PRUNE="true"

ESPRESSO_SEQUENCER_IS_DA="true"

RUST_LOG="warn,hotshot_libp2p_networking=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.decaf.testnet.espresso.network

ESPRESSO_SEQUENCER_API_PEERS=https://query.decaf.testnet.espresso.network

Chosen by operators

Copy

An HTTP JSON-RPC endpoint for Sepolia testnet. This is required

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

A `ws://` or `wss://` endpoint for Sepolia testnet. This is optional but

recommended since it decreases the load on your provider.

ESPRESSO_SEQUENCER_L1_WS_PROVIDER # e.g. wss://sepolia.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

Archival 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-UZAFTUIMZOT.decaf.testnet.espresso.network/

ESPRESSO_SEQUENCER_CDN_ENDPOINT=cdn.decaf.testnet.espresso.network:1737

ESPRESSO_STATE_RELAY_SERVER_URL=https://state-relay.decaf.testnet.espresso.network

ESPRESSO_SEQUENCER_GENESIS_FILE=/genesis/decaf.toml

ESPRESSO_SEQUENCER_IS_DA=true

ESPRESSO_SEQUENCER_ARCHIVE=true

RUST_LOG="warn,hotshot_libp2p_networking=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.decaf.testnet.espresso.network

ESPRESSO_SEQUENCER_API_PEERS=https://query.decaf.testnet.espresso.network

Chosen by operators

Copy

An HTTP JSON-RPC endpoint for Sepolia testnet. This is required

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

A `ws://` or `wss://` endpoint for Sepolia testnet. This is optional but

recommended since it decreases the load on your provider.

ESPRESSO_SEQUENCER_L1_WS_PROVIDER # e.g. wss://sepolia.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

Volumes

- $ESPRESSO_SEQUENCER_KEY_FILE

Contracts

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

- Light client on Sepolia testnet (for L2 testnets integrating Espresso): 0x303872bb82a191771321d4828888920100d0b3e4
- Light client on Arbitrum Sepolia (for Orbit L3 testnets integrating Espresso): 0x08d16cb8243b3e172dddcdf1a1a5dacca1cd7098