External Node Configuration

#

External Node Configuration

This document outlines various configuration options for the EN. Currently, the EN requires the definition of numerous environment variables. To streamline this process, we provide prepared configs for the zkSync Era - for both mainnet and testnet. You can use these files as a starting point and modify only the necessary sections.

#

Database

The EN uses two databases: PostgreSQL and RocksDB.

PostgreSQL serves as the main source of truth in the EN, so all the API requests fetch the state from there. The PostgreSQL connection is configured by theDATABASE_URL . Additionally, theDATABASE_POOL_SIZE variable defines the size of the connection pool.

RocksDB is used in components where IO is a bottleneck, such as the State Keeper and the Merkle tree. If possible, it is recommended to use an NVME SSD for RocksDB. RocksDB requires two variables to be set:EN_STATE_CACHE_PATH andEN_MERKLE_TREE_PATH , which must point to different directories.

#

L1 Web3 client

EN requires a connection to an Ethereum node. The corresponding env variable isEN_ETH_CLIENT_URL . Make sure to set the URL corresponding to the correct L1 network (L1 mainnet for L2 mainnet and L1 sepolia for L2 testnet).

Note: Currently, the EN makes 2 requests to the L1 per L1 batch, so the Web3 client usage for a synced node should not be high. However, during the synchronization phase the new batches would be persisted on the EN quickly, so make sure that the L1 client won't exceed any limits (e.g. in case you use Infura).

#

Exposed ports

The dockerized version of the server exposes the following ports:

- HTTP JSON-RPC:3060
- WebSocket JSON-RPC:3061
- Prometheus listener:3322
- Healthcheck server:3081

While the configuration variables for them exist, you are not expected to change them unless you want to use the EN outside of provided docker environment (not supported at the time of writing).

Info

NOTE : if the Prometheus port is configured, it must bescrapedopen in new windowperiodically to avoid a memory leak due to abug in an external metrics libraryopen in new window. If you are not intending to use the metrics, leave this port not configured, and the metrics won't be collected.

#

API limits

There are variables that allow you to fine-tune the limits of the RPC servers, such as limits on the number of returned entries or the limit for the accepted transaction size. Provided files contain sane defaults that are recommended for use, but these can be edited, e.g. to make the EN more/less restrictive.

#

JSON-RPC API namespaces

There are 7 total supported API namespaces:eth ,net ,web3 ,debug - standard ones;zks - rollup-specific one;pubsub - a.k.a.eth_subscribe ;en - used by external nodes while syncing. You can configure what namespaces you want to enable usingEN_API_NAMESPACES and specifying namespace names in a comma-separated list. By default, all but thedebug namespace are enabled.

#

Logging and observability

MISC_LOG_FORMAT defines the format in which logs are shown:plain corresponds to the human-readable format, while the other option isjson (recommended for deployments).

RUST_LOG variable allows you to set up the logs granularity (e.g. make the EN emit fewer logs). You can read about the formathereopen in new window .

MISC_SENTRY_URL andMISC_OTLP_URL variables can be configured to set up Sentry and OpenTelemetry exporters.

If Sentry is configured, you also have to setEN_SENTRY_ENVIRONMENT variable to configure the environment in events reported to sentry.

[ ]Edit this pageopen in new windowLast update: Contributors: [[albicodes]]

Prev Component Breakdown Next Running Node