Running NEAR Lake Indexer

info NEAR Lake is a blockchain indexer built on top ofNEAR Indexer microframework to watch the network and store all the events as JSON files on AWS S3.

How to start

The Lake Indexer setup consists of the following components:

- AWS S3 Bucket as a storage
- NEAR Lake binary that operates as a regular NEAR Protocol peer-to-peer node, so you will operate it as
- any otherRegular/RPC Node in NEAR

Prepare Development Environment

Before you proceed, make sure you have the following software installed:

- Rust compiler
- of the version that is mentioned inrust-toolchain
- file in the root ofnearcore
- project.
- Ensure you haveAWS Credentials configured
- From AWS Docs:
- For example, the files generated by the AWS CLI for a default profile configured with aws configure looks similar to the following.
- ~/.aws/credentials
- [default]
- aws_access_key_id=AKIAIOSFODNN7EXAMPLE
- aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Compile NEAR Lake

cargo build --release

Configure NEAR Lake

To connect NEAR Lake to the specific chain you need to have necessary configs, you can generate it as follows:

./target/release/near-lake --home ~/.near/testnet init --chain-id testnet --download-config --download-genesis The above code will download the official genesis config and generate necessary configs. You can replacetestnet in the command above to different network ID (betanet ,mainnet).

nearcore configuration According to changes innearcore config generation we don't fill all the necessary fields in the config file. Whilethis issue is open you need to download config you want and replace the generated one manually.

- testnet config.json
- betanet config.json
- mainnet config.json Configs for the specified network are in the--home provided folder. We need to ensure that NEAR Lake follows all the necessary shards, so"tracked_shards" parameters in~/.near/testnet/config.json needs to be configured properly. Currently,nearcore treats empty value for"tracked_shards" as "do not track any shard" andany value as "track all shards". For example, in order to track all shards, you just add the shard #0 to the list:

... "tracked_shards": [0], ...

Run NEAR Lake

Commands to run NEAR Lake, after./target/release/near-lake

Command Key/Subcommand Required/Default Responsible for --home Default ~/.near Tells the node where too look for necessary files: config.json

, genesis.json

, node_key.json

, and data

folder init

Tells the node to generate config files in--home-dir --chain-id Required

_localnet

_testnet

*mainnet Defines the chain to generate config files for --download-config Optional If provided tells the node to downloadconfig.json from the public URL. You can download them manually

-testnet config.json

-mainnet config.json --download-genesis Optional If provided tells the node to downloadgenesis.json from the public URL. You can download them manually

-testnet genesis.json

-mainnet genesis.json TODO: Otherneard keys run

Runs the node --bucket Required AWS S3 Bucket name --region Required AWS S3 Bucket region --fallback-region Default eu-central-1 AWS S3 Fallback region --endpoint Optional AWS S3 compatible API endpoint --stream-while-syncing Optional If provided Indexer streams blocks while they appear on the node instead of waiting the node to be fully synced --concurrency Default 1 Defines the concurrency for the process of saving block data to AWS S3 sync-from-latest One of thesync- subcommands is required Tells the node to start indexing from the latest block in the network sync-from-interruption One of thesync- subcommands is required Tells the node to start indexing from the block the node was interrupted on (if it is a first start it will fallback tosync-from-latest) sync-from-block --height N One of the sync-

subcommands is required Tells the node to start indexing from the specified block heightN (Ensure you node data has the block you want to start from) ./target/release/near-lake --home ~/.near/testnet run --stream-while-syncing --concurrency 50 sync-from-latest After the network is synced, you should see logs of every block height currently received by NEAR Lake.

Syncing

Whenever you run NEAR Lake for any network except localnet you'll need to sync with the network. This is required because it's a natural behavior ofnearcore node and NEAR Lake is a wrapper for the regularnearcore node. In order to work and index the data your node must be synced with the network. This process can take a while, so we suggest to download a fresh backup of thedata folder and put it in you--home-dir of your choice (by default it is~/.near)

tip Running your NEAR Lake node on top of a backup data will reduce the time of syncing process because your node will download only the data after the backup was cut and it takes reasonable amount time. All the backups can be downloaded from the public S3 bucket which contains latest daily snapshots:

You will needAWS CLI to be installed in order to download the backups.

Mainnet

aws s3 --no-sign-request cp s3://near-protocol-public/backups/mainnet/rpc/latest . LATEST=(cat latest) aws s3 --no-sign-request cp --no-sign-request --recursive s3://near-protocol-public/backups/mainnet/rpcLATEST ~/.near/data

Testnet

aws s3 --no-sign-request cp s3://near-protocol-public/backups/testnet/rpc/latest . LATEST=(cat latest) aws s3 --no-sign-request cp --no-sign-request --recursive s3://near-protocol-public/backups/testnet/rpcLATEST ~/.near/data

Running NEAR Lake as an archival node

It's not necessary but in order to index everything in the network it is better to do it from the genesis.nearcore node is running in non-archival mode by default. That means that the node keeps data only for5 last epochs . In order to index data from the genesis you need to turn the node in archival mode.

To do it you need to updateconfig.json located in--home-dir (by default it is~/.near).

Find next keys in the config and update them as following:

{ ... "archive": true, "tracked_shards": [0], ... } The syncing process in archival mode can take a lot of time, so it's better to download a backup provided by NEAR and put it in yourdata folder. After that your node will download only the data after the backup was cut and it takes reasonable amount time.

All the backups can be downloaded from the publicS3 bucket which contains the latest daily snapshots.

Seethis link for reference

Using the data

We write all the data to AWS S3 buckets:

- near-lake-data-testnet
- (eu-central-1
- region) for testnet
- near-lake-data-mainnet
- (eu-central-1
- region) for mainnet

Custom S3 storage

In case you want to run you ownnear-lake instance and store data in some S3 compatible storage (Minio orLocalstack as example) You can override default S3 API endpoint by using--endpoint option

- runminio

mkdir -p /data/near-lake-custom && minio server /data * runnear-lake

./target/release/near-lake --home ~/.near/testnet run --endpoint http://127.0.0.1:9000 --bucket near-lake-custom sync-from-latest

Data structure

The data structure we use is the following:

/ block.json shard_0.json shard_1.json ... shard_N.json ** is a 12-character-longu64 * string with leading zeros (e.g000042839521 *).See this issue for a reasoning * block_json * contains JSON-serializedBlockView * struct.Note: * this struct might change in the future, we will announce it * shard_N.json * whereN * isu64 * starting from0 * . Represents the index number of the shard. In order to find out the expected number of shards in the block you can look inblock.json * at.header.chunks_included

Access the data

All NEAR Lake AWS S3 buckets haveRequest Payer enabled. It means that anyone with their own AWS credentials can List and Read the bucket's content andbe charged for it by AWS . Connections to the bucket have to be done with AWS credentials provided. SeeNEAR Lake Framework for a reference.

NEAR Lake Framework

Once weset up the public access to the bucketsanyone will be able to build their own code to read it through.

For our own needs we are working onNEAR Lake Framework to have a simple way to create an indexer on top of the data stored by NEAR Lake itself.

note See the official NEAR Lake Frameworkannouncement on the NEAR Gov Forum . Edit this page Last updatedonDec 5, 2023 bygagdiez Was this page helpful? Yes No

Previous NFT indexer for Python Next Start options