NEAR Lake indexer basic tutorial

Source code for the tutorial near-examples/near-lake-raw-printer-js : source code for the tutorial on how to create an indexer that prints block height and number of shards Recently we havepublished a JavaScript version of the NEAR Lake Framework on npmjs.org

We want to empower you with a basic tutorial on how to use the JavaScript Library. Let's get started!

Requirements

Before we get started, please, ensure you have:

- nodejs
- installed

Create a project

Create an indexer project:

mkdir near-lake-raw-printer-js && cd near-lake-raw-printer-js Now we're going to callnpm init , we can continue with the default values pressing Enter on every question in the interactive mode:

npm init version: (1.0.0) description: entry point: (index.js) test command: git repository: keywords: author: license: (ISC) About to write to /Users/near/projects/near-lake-raw-printer-js/package.json:

{ "name": "near-lake-raw-printer-js", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "author": "", "license": "ISC" }

Is this OK? (yes) package.json is ready. Let's installnear-lake-framework

Install dependencies

Installnear-lake-framework

npm install near-lake-framework --save Installtypescript as dev dependency

npm install typescript --save-dev

Setup TypeScript

Now we can createtsconfig.json for TypeScript settings:

touch tsconfig.json Paste the content to the file:

{ "compilerOptions": { "lib": ["ES2018", "dom"] } } Now let's add thescripts section to thepackage.json

"scripts": { "start": "tsc && node index.js" } After that yourpackage.json should look similar to:

{ "name": "near-lake-raw-printer", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "start": "tsc && node index.js" }, "dependencies": { "near-lake-framework": "^1.0.1" }, "devDependencies": { "typescript": "^4.6.4" } } Now let's createindex.ts

touch index.ts Openindex.ts in your favorite editor to start coding.

Importnear-lake-framework

In theindex.ts file let's import the necessary dependencies:

import

{ startStream , types }

from

'near-lake-framework' ; We've imported the main functionstartStream which will be called to actually run the indexer, andtypes that hold theLakeConfig type we need to contruct.

Create a config

To get indexer running we need to start it with a config. We need to create an instance ofLakeConfig

const lakeConfig : types . LakeConfig

=

{ s3BucketName :

"near-lake-data-mainnet" , s3RegionName :

"eu-central-1" , startBlockHeight :

63804051 , } ;

Create indexer handler

Indexer will be streaming theStreamerMessage instances we need to handle according to our needs.

Innear-lake-framework JS library the handler have to be presented as a callback function. This function have to:

- be asynchronous
- accept an argument of typeStreamerMessage
- return nothing (void
-)

Creating the callback:

async

function

handleStreamerMessage (streamerMessage : types . StreamerMessage) :

Promise < void

{ // } For this tutorial our requirement is to log the block height and the numer of shards. That's simple:

async

function

handleStreamerMessage (streamerMessage : types . StreamerMessage) :

Promise < void

{ console . log (Block # { streamerMessage . block . header . height } Shards: { streamerMessage . shards . length }) ; }

Starting the stream

And the last thing to write is the call tostartStream with the config and pass the callback function.

(async

( )

=>

{ await

startStream (lakeConfig , handleStreamerMessage) ; }) ( ) ; That's it. Now we can compile the code and run it

Compile and run

Credentials To be able to access the data fromNEAR Lake you need to provide credentials. Please, see theCredentials article We've added thestart command to thescripts , so the compilation and run should as easy as

npm run start You should see something like the following:

Block #63804051 Shards: 4 Block #63804052 Shards: 4 Block #63804053 Shards: 4 Block #63804054 Shards: 4 Block #63804055 Shards: 4 Block #63804056 Shards: 4 Block #63804057 Shards: 4 Block #63804058 Shards: 4 Block #63804059 Shards: 4 Block #63804060 Shards: 4 You can stop the indexer by pressing CTRL+C

What's next?

You can play around and change the content of the callback functionhandleStreamerMessage to handle the data differently.

You can find thesource code for this tutorial on the GitHub . Edit this page Last updatedonNov 17, 2023 byDamian Parrino Was this page helpful? Yes No

Previous Lake Primitive Types Next Python tutorial