Subsquid cheatsheet for EVM networks

v0.0-draft

Substrate cheatsheet link

Get a template squid up and running:

```
npm install --global @subsquid/cli
sqd init my-new-squid -t evm
cd my-new-squid
npm ci
sqd up
sqd process
```

Data ingestion is done, start an API with

```
sqd serve
```

Quickstart guide link

Requesting data (src/processor.ts):

```
const processor =
  new EvmBatchProcessor()
  .setDataSource({
    archive:
      lookupArchive('eth-mainnet'),
    chain: ETHEREUM_NODE_RPC
  })
  .addLog({
    address: [CONTRACT_ADDRESS],
    topic0:
      [abi.events.Transfer.topic],
    range: {
      from: 1_000_000,
      to: 2_000_000
    }
  })
  .setFields({
    log: {
      data: false,
      transactionHash: true
    }
  })
```

Transactions, execution traces and contract state diffs can be requested similarly.

Processor config docs link

Processing and saving data (src/main.ts):

```
processor.run(new TypeormDatabase(), async context => {
  // THIS function gets called
  // for each batch of 1-1000 blocks
  // DEFINE some buffers to hold the processed data
  for (let block of context.blocks) {
    for (let log of block.logs) {
      if (log.address === CONTRACT_ADDRESS &&
          log.topics[0] === abi.events.Transfer.topic) {
        // DO whatever you want with the log data!
        // QUERY chain state, use IPFS, HTTP APIs
        // SDK provides tools for decoding and
        // handling historical runtimes
    }}
    // PROCESS transactions, execution traces and
    // contract state diffs in the same way
  }
  // SAVE the results for the whole batch
  // Postgres or CSV/Parquet/JSON files supported
})
```

Batch context interface link

State queries link

Data decoding is done using TypeScript classes auto-generated from contract ABI. The generator tool can fetch the ABI from an Etherscan-like API by contract address. Examples:

- `npx squid-evm-typegen src/abi erc20.json`
- `npx squid-evm-typegen src/abi \ 0xBB9bc244D798123fDe783fCc1C72d3Bb8C189413 \ --etherscan-api https://api.bscscan.com/`

Usage:

```
import * as abi from './abi/erc20'

if (log.topics[0] ===
    abi.events.Transfer.topic) {

  let {from, to, value} =
    abi.events.Transfer.decode(log)
}
```

Typegen link

Example squids

Examples repo link

BAYC NFT indexer link

Thena squid link

Postgres & GraphQL share a data definition. Config at schema.graphql

```
type Account @entity {
  id: ID!
  transfersTo: [Transfer!] @derivedFrom(field: "to")
}
type Transfer @entity {
  id: ID!
  to: Account!
  hash: String! @index
}
```

creates a foreign key column `to_id` at the “transfer” table on the database side and allows queries like this:

```
query {
  transfers {
    to {
      transfersTo }}
  accounts {
    transfersTo }}
```

The hash column will be `@index`-ed for fast queries.
The GraphQL server can be extended with arbitrary queries.

Data definition

API extensions

It is possible to write to file-based datasets:

Formats:
- CSV
- Parquet

Destinations:
- Local
- S3

Persisting to files

The 'Aquarium' – an optional managed service for squids. Free tier forever.

How it works:
- Register at app.subsquid.io
- Authorize your Squid CLI
- Run `sqd deploy .` <- Mind the dot!

...and that's it!

Aquarium

Deployment docs

No time to code? Contract indexers can be autogenerated from ABIs!

```
sqd init my-new-squid -t abi
cd my-new-squid && npm ci
# save JSON ABI as abi/myAbi.json
sqd generate --name contractName \
  --address <contractAddress> \
  --archive <networkArchiveAlias> \
  --abi ./abi/myAbi.json \
  --function '*' \
  --event '*' # or a comma-separated list
```

Squid-gen quickstart

Squid-gen reference

SUBSQUID

Web

Docs

SDK overview

Dev chat

Discord