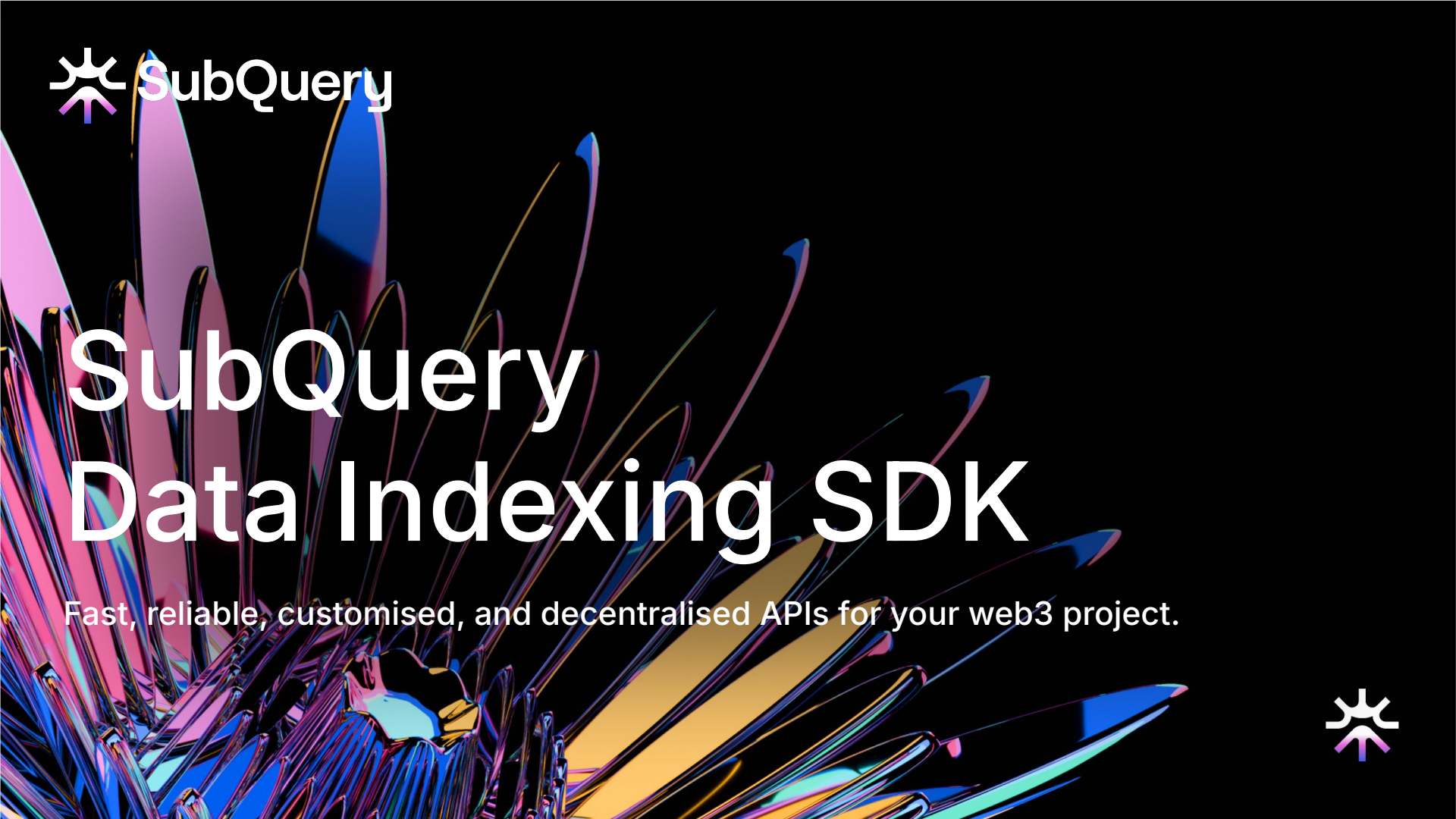
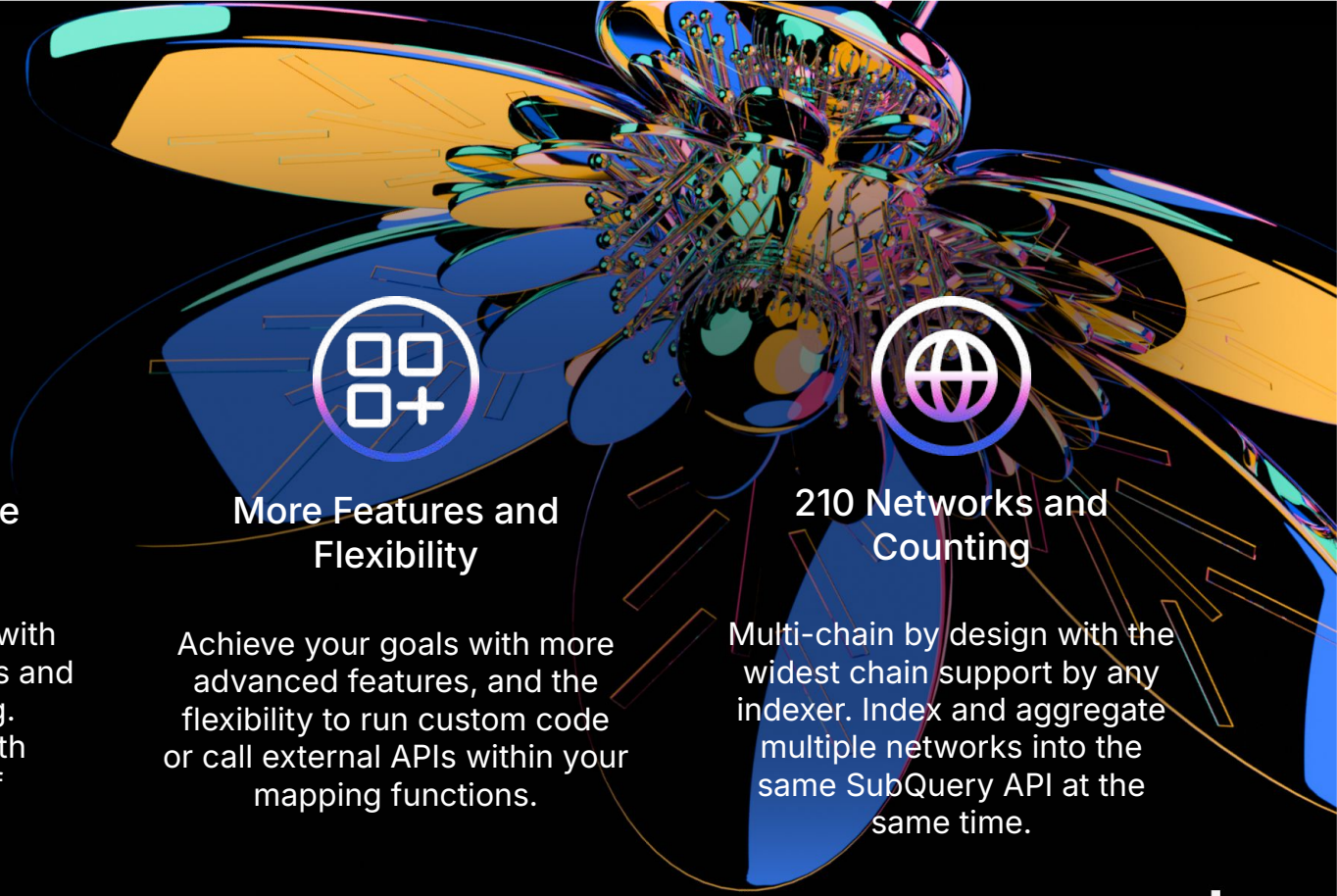

SubQuery
SubQuery
Data Indexing SDK
Fast, reliable, customised, and decentralised APIs for your web3 project.

Bringing web2 performance to web3

SubQuery APIs make your dApp lightning quick

By providing an indexed data layer, your dApps get richer data faster to allow you to build intuitive and immersive experiences for your users.



3.9x Faster than The Graph

Iterate and deliver faster with extremely quick sync times and parallelised processing. Reduce data latency with immediate indexing of unfinalised blocks.

More Features and Flexibility

Achieve your goals with more advanced features, and the flexibility to run custom code or call external APIs within your mapping functions.

210 Networks and Counting

Multi-chain by design with the widest chain support by any indexer. Index and aggregate multiple networks into the same SubQuery API at the same time.

Discover advanced features

- **Multi-chain Indexing Support**
  Index data from across different layer-1 networks into the same database behind a single GraphQL endpoint.
- **Non-deterministic Indexing**
  Import external libraries and make external API calls to enrich your data.
- **Real-time Indexing**
  Index unfinalised data and control required confirmations for faster indexing latency.
- **GraphQL Subscriptions**
  Allowing reactive front-end applications that subscribe to changes in your SubQuery project.

Discover advanced features

- **Historic State Indexing**
  Go back in time with the automatic query of the state of the SubQuery project at any block height.
- **Dynamic Data Sources**
  Index factory contracts, for example on DEXes or generative NFT projects.
- **EVM and Cosmos Project Scaffolding**
  Save time during SubQuery project creation by automatically generating typescript facades for EVM transactions, logs, and types.
- **Project Optimisation**
  Find out how to further optimise SubQuery projects to maximise performance.

How does the indexer work?

SubQuery is designed to make building your own custom API for your web3 project as easy and painless as possible. **It just takes three steps.**

1. Initialise

Run `subql init` command to create a new project from over 100 different example templates, or autogenerate code from your own EVM ABIs or Cosmos Protobufs.

How does the indexer work?

2. Build

All you need to do is edit three files, your GraphQL schema definition, your project manifest, and finally the mappings for your ETL pipelines.

2a. Define the shape of data you want to index with the Schema File

2b. Configure your endpoints & filters with the Manifest File

2c. Transform your data with the Mappings File

How does the indexer work?

2a. Define shape of your data

The `schema.graphql` file determines the shape of the resulting data that you are using SubQuery to index. These map to database table definitions and GraphQL schemas.

Learn more

```
type Transfer @entity {
  id: ID!
  from: String!
  to: String!
  tokenId: BigInt!
  blockNumber: BigInt!
  transactionHash: String!
  timestamp: BigInt!
  date: Date!
  boredApe: BoredApe
}

type Mint @entity {
  id: ID!
  minter: String!
  boredApe: BoredApe!
  timestamp: BigInt!
  date: Date!
}
```

```
type Gravatar @entity {
  id: ID!
  owner: Bytes!
  displayName: String!
  imageUrl: String!
  createdBlock: BigInt!
}
```

How does the indexer work?

2b. Configure your endpoints & filters

The Project Manifest (`project.ts`) file works as an entry point to your project. It defines the smart contract(s) that you index and the specific on chain transactions/logs/events that you want to index.

See our docs for detailed view of different manifest files across various ecosystems.

```
{
  dataSources: [
    {
      kind: EthereumDatasourceKind.Runtime,
      startBlock: 6175243,

      options: {
        // Must be a key of assets
        abi: "gravity",
        address: "0x2E645469f354BB4F5c8a05B3b30A929361cf77eC",
      },
      assets: new Map([["gravity", { file: "./abis/Gravity.json" }]]),
      mapping: {
        file: "./dist/index.js",
        handlers: [
          {
            kind: EthereumHandlerKind.Event,
            handler: "handleNewGravatar",
            filter: {
              topics: ["NewGravatar(uint256,address,string,string)"],
            },
          },
          {
            kind: EthereumHandlerKind.Event,
            handler: "handleUpdatedGravatar",
            filter: {
              topics: ["UpdatedGravatar(uint256,address,string,string)"],
            },
          },
        ],
      },
    },
  ],
}
```

How does the indexer work?

2c. Transform your data

Mapping functions define how chain data is transformed into the GraphQL entities that you previously defined in the `schema.graphql` file.

You can do any CRUD action here or make external API calls to augment the dataset.

See our docs

```
import {
  NewGravatarLog,
  UpdatedGravatarLog,
} from "../types/abi-interfaces/Gravity";
import { Gravatar } from "../types";
import assert from "assert";

export async function handleNewGravatar(log: NewGravatarLog): Promise<void> {
  logger.info("New Gravar at block " + log.blockNumber.toString());

  assert(log.args, "Require args on the logs");

  const gravatar = Gravatar.create({
    id: log.args.id.toHexString()!,
    owner: log.args.owner,
    displayName: log.args.displayName,
    imageUrl: log.args.imageUrl,
    createdBlock: BigInt(log.blockNumber),
  });

  await gravatar.save();
}
```

How does the indexer work?

3. Query

Run `yarn dev` to run your project in Docker (or deploy to our Network) and start querying against your indexed dataset.

Or connect directly to the database for more power.

[Find out more about data querying with GraphQL]

```
{
  gravatars(
   orderBy: CREATED_BLOCK_DESC
   filter: {owner: {equalTo: "ACCOUNT_ID"}}) {
   nodes {
    id
    owner
    displayName
    imageUrl
    createdBlock
   }
  }
}
```


```
{
  "data": {
    "gravatars": {
      "nodes": [
        {
          "id": "0x47",
          "owner": "\\xbc8dafeaca658ae0857c80d8aa6de4d487577c63",
          "displayName": "Victor",
          "imageUrl": "https://ucarecdn.com/295e4ba5-ad1c-48bf-9957-093424709881/
          "createdBlock": "6469958"
        },
        {
          "id": "0x46",
          "owner": "\\x0773cbc2c55cd6354a61b7bcbca52d9dccd56534",
          "displayName": "dgogel",
          "imageUrl": "https://ucarecdn.com/c44402a0-30f8-4c0c-bf1f-b13918903211/
          "createdBlock": "6460716"
```

Get inspired

Take advantage of our robust tech documentation and project marketplace

Find hundreds of example projects of some of developer favourites like Chainlink, Galxe, Uniswap, OpenSea and more.

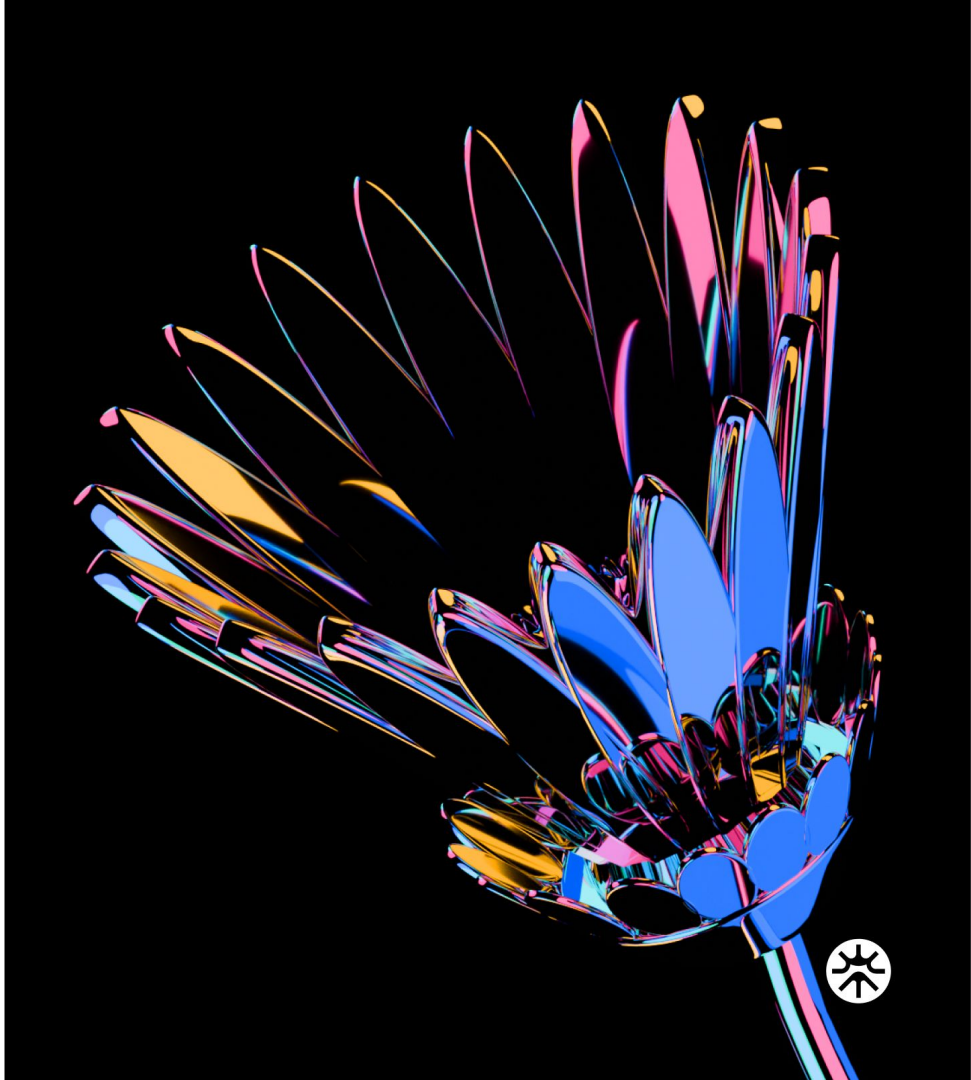
Learn how some of the biggest projects in web3 can be indexed using SubQuery's SDK, and use these projects as inspiration for your next SubQuery API.

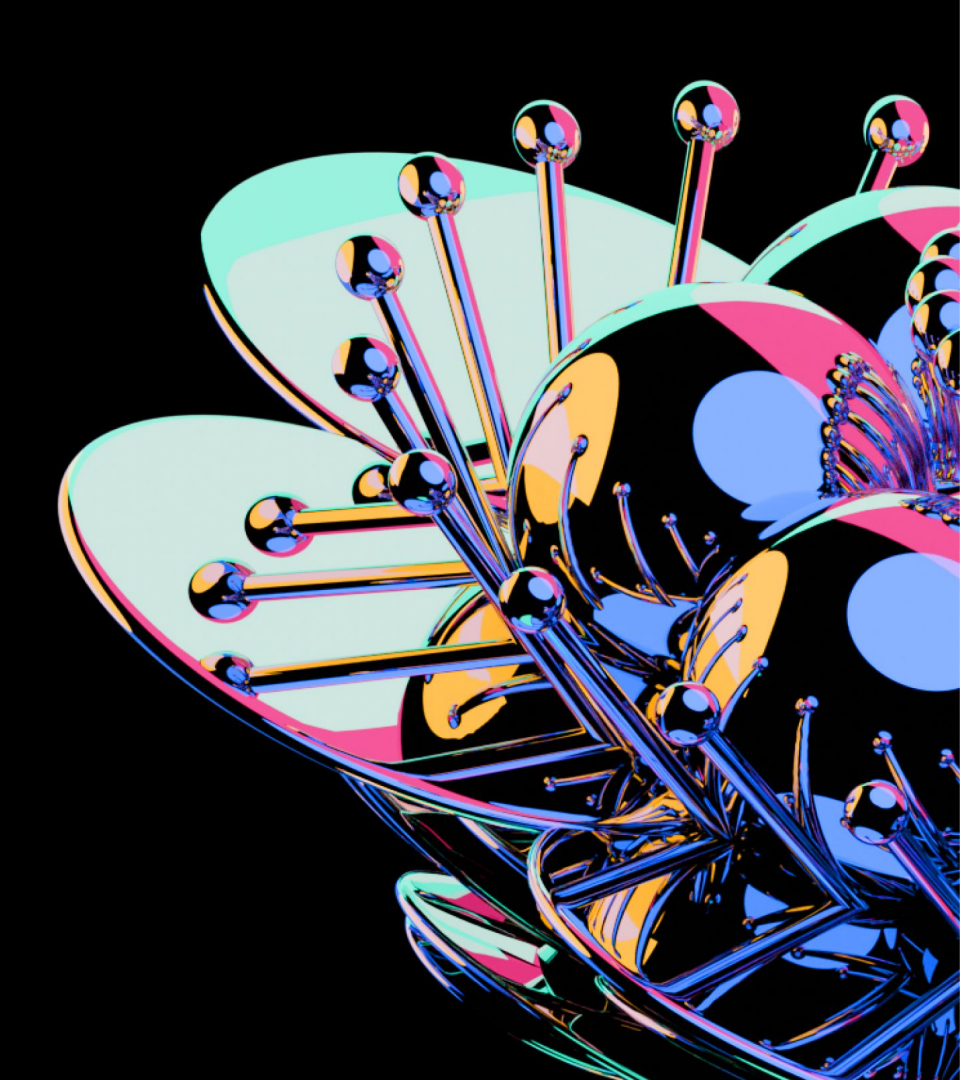
Your indexer, run your way

Running SubQuery Locally

SubQuery is open-source, meaning developers have the freedom to run SubQuery projects locally on their own computers or cloud providers as a local NodeJS + Postgres setup, or within a Docker container.

Learn more





Your indexer, run your way

SubQuery's Managed Service

With 100s of millions of daily requests and hundreds of active projects, SubQuery's Managed Service provides industry-leading hosting for our customer's production SubQuery projects with optional SLAs.

Run your project with us

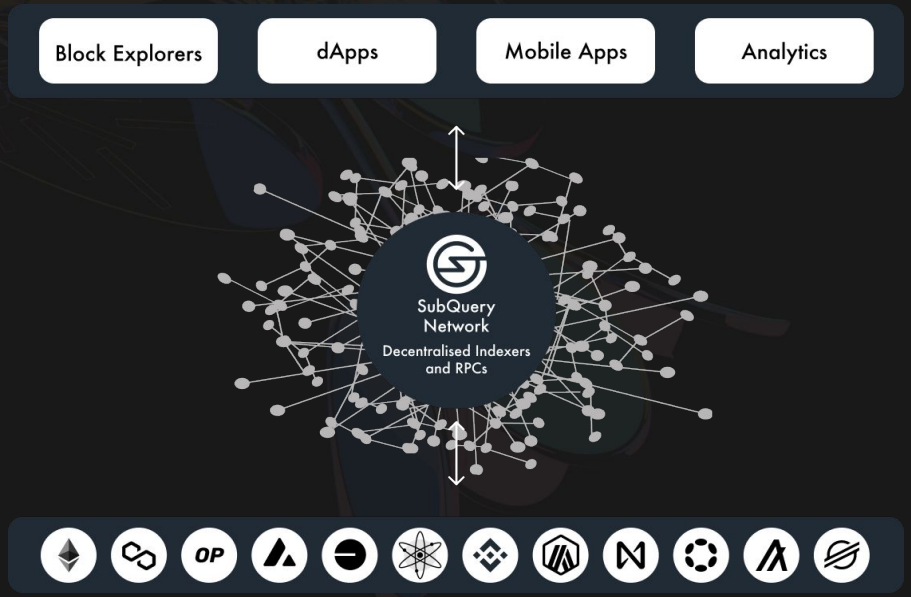
Your indexer, run your way

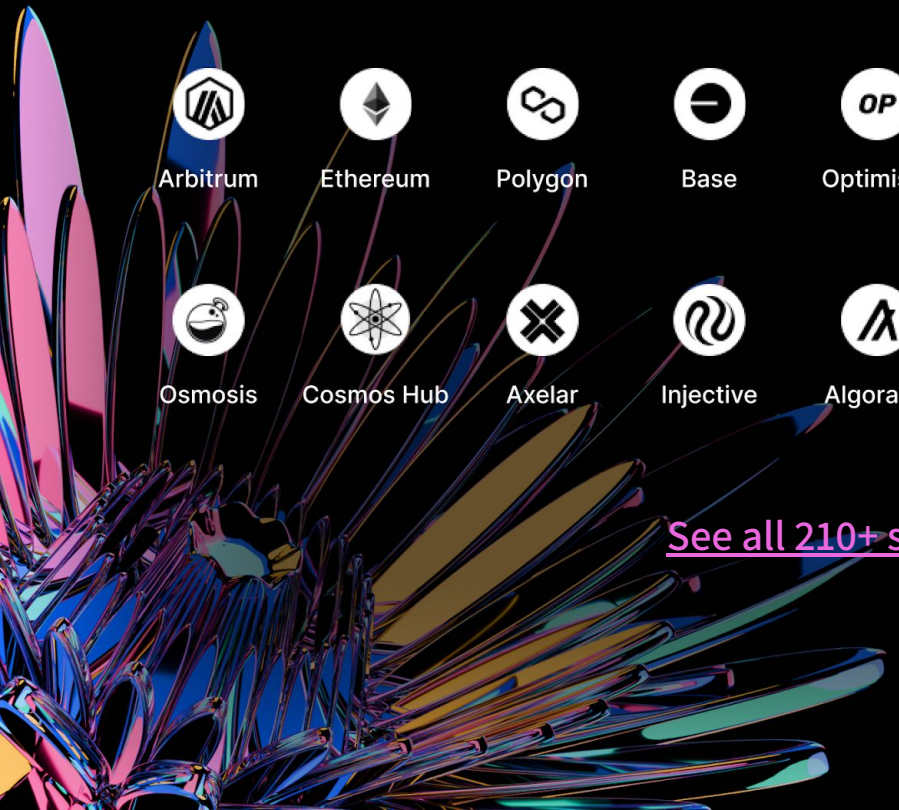
Decentralised SubQuery Network

Performant, reliable, and decentralised - the SubQuery Network allows you to completely decentralise your infrastructure stack. Indexers serve any data you need in a trustless and verifiable way.

View the network Publish your project

Our network decentralises both Indexers and RPCs





Arbitrum
Ethereum
Polygon
Base
Optimism
BNB
Fantom
Harmony
Avalanche
Near

Osmosis
Cosmos Hub
Axelar
Injective
Algorand
Stellar
Polkadot
Astar
Acala
Moonbeam

See all 210+ supported networks

Coming from the Graph?

Migrating is easy and should only take a few minutes.

https://academy.subquery.network/build/graph-migration.html

Need more info or inspiration?

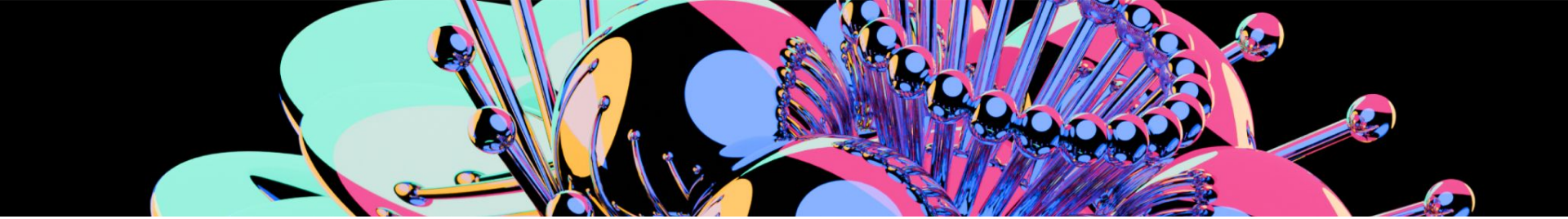
Check out our robust tech documentation.

https://academy.subquery.network/

Join Our Community

Connect with others and see what data they are indexing with SubQuery.

https://discord.com/invite/subquery



Book a demo

start@subquery.network

Visit our website

www.subquery.network

Follow us

Telegram | Twitter | LinkedIn | Medium | GitHub | Discord