# Quickstart: Indexing Arbitrum custom data via Flair

Flair , Real-time and historical custom data indexing for any evm chain.

Flair offers reusableindexing primitives (such as fault-tolerant RPC ingestors, custom processors, re-org aware database integrations) to make it easy to receive, transform, store and access your on-chain data.

## Why Flair?

Compared to other alternatives the main reasons are:

- Adoptingparallel and distributed processing
- paradigm means high scalability and resiliency for your indexing stack. Instead of constrained sequential processing (e.g Subgraph).
- Focused onprimitives
- , which means on the left you plug-in an RPC and on the right you output the data to any destination database.
- Nativereal-time stream processing
- for certain data workload (such as aggregations, rollups) for things like total volume per pool, or total portfolio per user wallet.
- ☁Managed
- cloud services avoid DevOps and irrelevant engineering costs for dApp developers.
- Avoid decentralizationoverhead
- (consensus, network hops, etc) since we believe to enable best UX for dApps reading data must be as close to the developers as possible.

### Features

- ✅ Listen toany EVM chain
- with just an RPC URL.* Free managed RPC URLs for +8 popular chains already included.
- - Works with both websocket and https-only RPCs.
- ✅ Track and ingestany contract
- forany event topic.
- - Auto-track new contracts deployed from factory contracts.
- ✅Custom processor scripts
- with Javascript runtime (withTypescript
- support)* Make external API or Webhook calls to third-party or your backend.
- - Get current or historical USD value of any ERC20 token amount of any contract address on any chain.
- - Use any external NPM library.
- ✅Stream
- any stored data to your destination database (Postgres, MongoDB, MySQL, Kafka, Elasticsearch, Timescale, etc).

## Getting Started

1️⃣Clone thestarter boilerplate template and follow the instructions

git clone https://github.com/flair-sdk/starter-boilerplate.git

# ... follow instructions in README.md

info Boilerplate instructions will create anew cluster , generatean API Key , and set up a manifest.yml to index yourfirst contract withsample custom processor scripts.

Learn more about thestructure of manifest.yml .2️⃣Configure Arbitrum RPC nodes

Set a unique namespace, Arbitrum chainId and RPC endpoint in your config. Remember that you can add up to 10 RPC endpoints for resiliency.

{ 'cluster' :

'dev' , 'namespace' :

'my-awesome-arbitrum-indexing-dev' , 'indexers' : [ { 'chainId' :

42161 , 'enabled' :

true , 'ingestionFilterGroup' :

'default' , 'processingFilterGroup' :

'default' , 'sources' :

[

# Highly-recommended to have at least 1 websocket endpoint

'wss : //arbitrum - one.publicnode.com' ,

# You can put multiple endpoints for failover

'https : //arbitrum.llamarpc.com' , ] , } , ] , }3Sync some historical data using backfill command . Remember thatenabled: true flag in yourconfig enabled your indexer to capture data in real-time already.

# backfill certain contracts or block ranges

pnpm flair backfill --chain 42161 --address 0x22dc069183f85a8473553e32b59efc9fec506baf -d backward --max-blocks 10000

# backfill for a specific block number, if you have certain events you wanna test with

pnpm flair backfill --chain 42161 -b 132763420

# backfill for the recent data in the last X minute

pnpm flair backfill --chain 42161 --min-timestamp = "30 mins ago" -d backward4Query your custom indexed data.

5Stream the data to yourown database .

## Examples

Explore real-world usage of Flair indexing primitives for various use-cases.

### DeFi

- Aggregate protocol fees in USD across multiple chains
- Calculate "Health Factor" of positions with contract factory tracking
- Index Uniswap v2 swaps with USD price for all addresses

### NFT

- Index ERC721 and ERC1155 NFTs on any EVM chain with an RPC URL

## Need help?

Our engineers are available to help you at any stage.Edit this page Last updatedonMar 7, 2024 Previous Envio Next PARSIQ