Kaiko

MIGRATION GUIDE

DEX Trades & OHLCV

Amberdata → Kaiko

2026

www.kaiko.com

1 At a glance

What this guide covers. The three Amberdata endpoints that read swaps on decentralised exchange pools and the candles built from them, and the Kaiko endpoint that replaces each one. Kaiko serves them through two endpoints: one for swaps, one for open-high-low-close-volume candles of a pool.

What changes for you

- Three Amberdata paths become two Kaiko endpoints on `https://api.kaiko.io`. The pool in the Amberdata OHLCV path becomes the `pool_address` query parameter, and the latest candle is the first row of the same candle endpoint (§7).
- A swap is one pair of a base and a quote token with one price, quote per base, and `taker_side_sell` says which side the account took. It replaces the bought and sold legs with their two prices (§5, §6).
- A swap read needs a pool, a token, or both: send `pool_address` or `assets` (§6).
- Candles are keyed objects with ISO 8601 times, labelled by the bucket's open as on Amberdata. `timeInterval` becomes `interval` : `1m` , `1h` or `1d` (§7).
- Pool and token descriptions (symbols, decimals, fee) leave the candle row; Kaiko reference data serves them per pool (§4).

Endpoint map

Amberdata endpoint	Kaiko endpoint	How to call it	§
`defi/dex/trades`	`/defi/trades`	`pool_address` or `assets`, and a window	6
`market/defi/ohlcv/{pool}/historical`	`/defi/trades/ohlcv`	`pool_address` + `interval` + a window	7
`market/defi/ohlcv/{pool}/latest`	`/defi/trades/ohlcv`	`pool_address` + `interval=1m` + `page_size=1`, no window	7

Coverage & distribution

Data type: Swaps on DEX pools, one row per swap; OHLCV candles per pool at one minute, one hour or one day

Blockchains: ethereum, arbitrum, avalanche, base, bsc, optimism, polygon, celo, linea, hyperevm, bob, plasma, robinhood and solana (§4)

Distribution: REST API, `https://api.kaiko.io`

Availability: Live; response version `2026-10-01`

Not covered by this guide

The Amberdata pool and exchange listings, `defi/dex/information` , `defi/dex/ohlcv/information` , `market/defi/dex/exchanges` and `market/defi/dex/pairs` : they describe pools, not swaps or candles, and do not move to these two endpoints. Pools of a DEX outside the protocols of §4 return 404 `unknown_pool` .

IDENTICAL no change RENAMED same idea, new name or format NEW Kaiko only REMOVED no equivalent

2 Endpoint & authentication

These apply to both endpoints in this guide. Each endpoint section adds its own path and parameters.

	Amberdata (today)	Kaiko (new)
Method	GET	GET
Base URL	`https://api.amberdata.com`	`https://api.kaiko.io`
Path	`/defi/dex/trades` `/market/defi/ohlcv/{pool}/...`	`/defi/trades` `/defi/trades/ohlcv`
Auth	Header `x-api-key`	Header `X-Api-Key`, your Kaiko key. A request without it returns 401 `missing_api_key`.
Blockchain	Query `blockchain` on trades (`ethereum-mainnet`); implied by `exchange` on OHLCV	Query `blockchain`, required on both endpoints, short code (§4)
Version	Not returned on these endpoints	Optional query `api_version=YYYY-MM-DD`; current version `2026-10-01`, echoed in `meta.api_version`
Parameters		A parameter the endpoint does not declare returns 400 `invalid_parameter` naming it. Amberdata names such as `liquidityPoolAddress`, `startDate`, `timeInterval` and `exchange` are refused this way.

3 Response envelope & metadata

The fields around the data. The three Amberdata endpoints place their records in three shapes under `payload`. Kaiko always returns them as an array of objects under `data`.

Amberdata field	Kaiko field	Change	Notes
`status`	HTTP status line	REMOVED	Read the HTTP status code instead.
`title`, `description`		REMOVED	"OK" label and free-text status.
`payload.records`	`data`	RENAMED	Swaps (§6).
`payload.data.{exchangeId}` (arrays)	`data`	RENAMED	Historical candles. One keyed object per bucket replaces the array under the factory address (§7).
`payload.{pairAddress}` (object)	`data`	RENAMED	Latest candle: one row, with `pool_address` on it (§7).
`payload.metadata.columns`		REMOVED	Kaiko rows are keyed objects; there is no column list to read.
`payload.metadata.startDate`, `.endDate`	`meta.url`	RENAMED	`meta.url` is the request as received, with `api_version` added.
	`meta.count`	NEW	Number of rows in `data`.

Amberdata field	Kaiko field	Change	Notes
	`meta.next_url`	NEW	Complete URL of the next page; `null` on the last page. Use it as returned.
	`meta.api_version`	NEW	Version the endpoint served.

- **Empty results.** HTTP 200 with `data: []` for a known pool or token with no swap in the window. On candles, a bucket with no swap is a row with `null` prices, `"0"` volumes and a `trade_count` of 0 (§7).
- **Unknown identifiers.** HTTP 404 `unsupported_blockchain` for a blockchain Kaiko does not serve here (`bitcoin`); 404 `unknown_pool` or `unknown_asset` for a pool or token with no swap on the blockchain. A malformed address or a blockchain such as `ethereum-mainnet` returns 400 `invalid_parameter`.
- **Missing values.** Every key of a row is always present. On swaps no value is `null`; on candles only the four prices of an empty bucket are.
- **Errors.** Returned as `application/problem+json` with `status`, `title`, `detail`, `instance`, a stable `code` such as `invalid_parameter`, `not_entitled` or `rate_limited`, and on a 400 an `errors` list naming each parameter. Branch on `code`, not on the text.

4 Blockchain, protocol & naming conventions

This guide's equivalent of exchange names: the identifiers you send and receive.

Blockchains and protocols

Both endpoints serve the same blockchains and protocols. The `protocol` code is on every swap row.

Amberdata	Kaiko code	Protocol codes on Kaiko
`ethereum-mainnet`	`ethereum`	usp2, usp3, usp4, sush, pks3, curv, crv2, blc2, blcr
`arbitrum-mainnet`	`arbitrum`	usp3, usp4, cml3, curv, blc2
`avalanche-mainnet`	`avalanche`	usp2, usp3, usp4, sush, curv, blc2, tjv1, tjv2, tj21
`bnb-mainnet`	`bsc`	usp2, usp3, usp4, sush, pks3, pksp
`optimism-mainnet`	`optimism`	usp3, usp4, curv
`polygon-mainnet`	`polygon`	usp2, usp3, usp4, sush, qsp2, qsp3, curv, blc2
	`base`	usp2, usp3, usp4, pks3, aerb, aerc, curv, blc2
	`celo`	usp3, usp4
	`linea`	pks3, etx3
	`hyperevm`	hsp3
	`bob`	usp3
	`plasma`	usp3

Amberdata	Kaiko code	Protocol codes on Kaiko
	robinhood	usp4
	solana	orca, raya, rayc, rayp, meta, metd, mev2, pmsp, humi, jupz, alpq, aqfr, arch, bisn, byrl, gnv2, hadr, quan, ript, scor, slv1, slv2, tesv, wstr, zero

The Amberdata form `ethereum-mainnet` is refused with 400 `invalid_parameter`; `bnb` returns 404 `unsupported_blockchain`: use `bsc`.

Protocol codes

Amberdata	Kaiko code	Protocol
uniswap_v2, uniswapv2	usp2	Uniswap V2
uniswap_v3, uniswapv3	usp3	Uniswap V3
uniswap_v4	usp4	Uniswap V4
	sush	Sushiswap
	pksp, pks3	Pancakeswap, Pancakeswap V3
	curv, crv2	Curve, Curve V2
	blcr, blc2	Balancer, Balancer v2
	aerb, aerc	Aerodrome basic, Aerodrome concentrated
	qsp2, qsp3	Quickswap V2, Quickswap V3
	tjv1, tjv2, tj21	Trader Joe v1, v2, v2.1
	cml3, hsp3, etx3	Camelot V3, HyperSwap V3, Etherex CL
	orca, raya, rayc, rayp	Orca CLMM, Raydium AMM, Raydium CLMM, Raydium CPMM
	meta, metd, mev2, pmsp	Meteora AMM, Meteora DLMM, Meteora DAMM V2, PumpSwap
	humi, jupz, alpq, aqfr, arch, bisn, byrl, gnv2, hadr, quan, ript, scor, slv1, slv2, tesv, wstr, zero	HumidiFi, JupiterZ, AlphaQ, Aquifer, Archer, BisonFi, Byreal, GoonFi V2, HadronFi, Quantum, Riptide, Scorch, SolFi V1, SolFi V2, Tessera V, WhaleStreet, ZeroFi

Amberdata `project` (`uniswap`) has no separate field: the Kaiko code names the protocol and its version.

Conventions that change on both endpoints

- **Field names.** camelCase becomes snake_case: `blockNumber` → `block_number`, `quoteVolume` → `quote_volume`.
- **Pool identifiers.** `pool_address` takes the pool's address: 0x hex on EVM blockchains, matched in any letter case and returned lower case (a 32-byte pool id where the protocol has no per-pool contract); the base58 account on solana, matched exactly. A pair symbol such as `WETH_USDC` returns 400: find the pool address first.

- **Token identifiers.** `assets` takes one token contract address (base58 mint on solana), not a symbol. Rows carry token addresses only. Symbols, decimals, fee tier and the pool's tokens come from Kaiko reference data: `https://reference-data-api.kaiko.io/v1/pools?address={pool}`.
- **Time parameters.** `startDate` and `endDate` become `start_time` and `end_time`, ISO 8601 UTC with a time zone (`2026-11-01T01:00:00Z`), millisecond precision at most, start inclusive, end exclusive. A date alone or epoch milliseconds return 400. Without `end_time` the window ends now; without `start_time` it covers the 90 days before `end_time`. `timeFormat` is removed.
- **Timestamps.** Every time is an ISO 8601 UTC string with milliseconds (`1793494800000` → `2026-11-01T01:00:00.000Z`). Swap times are block times, in whole seconds.
- **Paging and order.** `page_size` sets rows per page: default 100, maximum 1000. A value above 1000 returns a 400 error. Follow `meta.next_url` until it is `null`. `sort` is `desc` (newest first, the default) or `asc`.
- **Numbers.** Prices, amounts and volumes are decimal strings in token units, decimals applied; parse them into a decimal type. Block numbers, log indices and trade counts are integers.

5 Will your values change?

Short answer

The same swap carries the same transaction, log index, block, time, pool, tokens and price on both APIs, and a candle matches Amberdata's on time, open, high, low, close and trade count. Three things need a code update: a swap is a base and quote pair with one price, the base token of a pool can be Amberdata's quote token, and amounts come in token units only.

- **One pair, one price per swap.** Kaiko orders each swap's pair as a base token (`base_token_address`) and a quote token (`quote_token_address`). `price` is in quote tokens per base token, `amount` counts base tokens, and `taker_side_sell` is `false` when the account bought the base token, `true` when it sold it. With `false`: Amberdata `assetBoughtAddress` is the base token, `assetBoughtAmount` is `amount`, `assetSoldAmount` is `amount` × `price`, `assetSoldPrice` is `price` and `assetBoughtPrice` is 1 / `price`. With `true`, swap bought and sold.
- **The base token is fixed per pair.** Read it from `base_token_address` on a swap of the pool. A candle row carries no token address: its prices are quote per base and `volume` counts base tokens of the same pair. When Amberdata names the other token as base, invert the prices (high becomes 1 / low) and swap `volume` with `quote_volume`.
- **Pools of more than two tokens.** A swap row names the pair that traded, so the rows of one pool can carry different pairs. Their candle mixes the prices of those pairs: read such pools through `/defi/trades`.
- **Candles are labelled by bucket open, as on Amberdata.** Buckets are aligned in UTC on the minute, the hour or midnight, and a row covers `[timestamp, timestamp + interval)`. The hourly row `2026-11-01T01:00:00.000Z` holds the swaps from 01:00:00 inclusive to 02:00:00 exclusive; Amberdata labels the same bucket `1793494800000`, the same instant (§7).
- **Every bucket has a row.** A bucket with no swap is returned with `null` open, high, low and close, `"0"` volumes and `trade_count` 0. Drop rows with `trade_count` 0 to keep traded buckets only.
- **Token units only.** `amount`, prices and volumes have the token decimals applied. `assetBoughtAmountRaw` and `assetSoldAmountRaw` have no field: multiply by 10^{decimals}, with decimals from reference data (§4).
- **Account and identifiers.** `user_address` is the account that sent the transaction, Amberdata's `transactionFromAddress`. Kaiko's `trade_id` is its own stable identifier; Amberdata's `tradeId` is `transaction_hash` + "_" + `log_index`.

6 DEX trades

Replaces. `defi/dex/trades`.

What it gives you. One row per swap on a DEX pool of one blockchain: the pool and protocol, the transaction, block and time, the account, the pair, the price and the base amount. Filter by pool, by token, by account, or a combination; every filter sent applies.

	Amberdata (today)	Kaiko (new)
Path	`/defi/dex/trades`	`/defi/trades`

Request parameters

Amberdata param	Kaiko param	Change	Notes
`blockchain`	`blockchain`	RENAMED	Required, short code (§4).
`liquidityPoolAddress`	`pool_address`	RENAMED	One pool. Send `pool_address`, `assets` or both; neither returns 400.
`assetBoughtAddress`, `assetSoldAddress`	`assets`	RENAMED	One token address (two return 400), on either side of the swap. For one direction, keep the rows where the token is the base and `taker_side_sell` matches (§5).
`walletAddress`	`addresses`	RENAMED	One account: the swaps whose `user_address` it is. Send it with `pool_address` or `assets`.
`protocolName`		REMOVED	Filter on `protocol` in the rows, or name the pool.
`startDate` / `endDate`	`start_time` / `end_time`	RENAMED	End exclusive. Default: the 90 days before `end_time`, which defaults to now (§4).
	`start_block` / `end_block`	NEW	Block window on EVM blockchains: the time of `start_block` inclusive to the time of `end_block` exclusive. Combines with the time bounds.
	`sort`	NEW	By `timestamp`, `desc` by default; swaps of one block share a timestamp and come in `trade_id` order.
	`page_size`	NEW	Default 100, maximum 1000.
	`live`	NEW	`true` also returns the newest blocks, whose rows can change on a later call. Default `false`.

Response data fields

Amberdata field	Kaiko field	Type	Change	Notes
`tradeId`	`trade_id`	string	RENAMED	Kaiko's stable swap identifier; opaque, do not parse it (§5).

Amberdata field	Kaiko field	Type	Change	Notes
`blockchain`	`blockchain`	string	RENAMED	Short code.
`protocolName`, `project`	`protocol`	string	RENAMED	Kaiko protocol code (§4).
`liquidityPoolAddress`	`pool_address`	string	RENAMED	Lower case on EVM blockchains.
`blockNumber`	`block_number`	integer	RENAMED	The slot on solana.
`timestamp`	`timestamp`	string	IDENTICAL	Block time, ISO 8601 UTC with milliseconds.
`transactionHash`	`transaction_hash`	string	RENAMED	Lower case on EVM blockchains.
`logIndex`	`log_index`	integer	RENAMED	Position of the swap's event log in its block.
`transactionFromAddress`	`user_address`	string	RENAMED	Account that sent the swap.
`assetBoughtAddress`, `assetSoldAddress`	`base_token_address`, `quote_token_address`	string	RENAMED	Base and quote of the pair; which one was bought follows `taker_side_sell` (§5).
`assetSoldPrice`, `assetBoughtPrice`	`price`	string	RENAMED	Quote tokens per base token; the other price is 1 / `price`.
`assetBoughtAmount`, `assetSoldAmount`	`amount`	string	RENAMED	Base tokens swapped, decimals applied. The quote amount is `amount` × `price`.
	`taker_side_sell`	boolean	NEW	`true` when the account sold the base token, `false` when it bought it.
`assetPair`			REMOVED	Join the two token addresses.
`assetBoughtAmountRaw`, `assetSoldAmountRaw`			REMOVED	Amount × 10^{decimals} (§5).
`assetBoughtSymbol`, `assetSoldSymbol`, `assetBoughtDecimals`, `assetSoldDecimals`			REMOVED	From reference data per pool (§4).
`callerAddress`, `receiverAddress`, `transactionToAddress`			REMOVED	The contracts that called the pool and received its output.
`tradeFeeBought`, `tradeFeeSold`			REMOVED	The pool's fee tier is in reference data.

Amberdata field	Kaiko field	Type	Change	Notes
`transactionFee`, `transactionGasPrice`, `transactionGasUsed`, `event`			REMOVED	Transaction gas context and the raw event signature.

Before / after example

A TRAC/WETH swap of 18 June 2026 at 00:00:11 on Uniswap V3 pool `0xb1914469...85e5`; the Kaiko call reads the same pool and minute. Kaiko response live, 4 rows of which the first shown.

Amberdata

```
GET /defi/dex/trades
    ?blockchain=ethereum-mainnet
    &startDate=2026-06-18T00:00:00
    &endDate=2026-06-18T00:01:00
x-api-key: <your-key>
```

Kaiko

```
GET https://api.kaiko.io
    /defi/trades
    ?blockchain=ethereum
    &pool_address=0xb1914469...85e5
    &start_time=2026-06-18T00:00:00Z
    &end_time=2026-06-18T00:01:00Z
    &sort=asc
X-Api-Key: <your-key>
```

Amberdata

```
{ ...
  "payload": {
    "records": [ {
      "assetBoughtAddress":
        "0xaa7a9ca8...0a6f",
      "assetBoughtAmount":
        "7950.0221604798143381",
      "assetBoughtPrice":
        "4921.620536901789045496",
      "assetSoldAddress":
        "0xc02aaa39...6cc2",
      "assetSoldAmount":
        "1.6153261107538444",
      "assetSoldPrice":
        "0.000203185107933882",
      "blockNumber": 25340719,
      "blockchain": "ethereum-mainnet",
      "liquidityPoolAddress":
        "0xb1914469...85e5",
      "logIndex": 132,
      "protocolName": "uniswap_v3",
      "timestamp":
        "2026-06-18T00:00:11.000Z",
      "tradeId": "0xc260a450...0b37_132",
      "transactionFromAddress":
        "0x5b43453f...edd1",
      "transactionHash":
        "0xc260a450...0b37",
      ...
    }, ... ],
    "metadata": {} } }
```

Kaiko

```
{
  "data": [ {
    "trade_id": "ethereum-u3-0xc260...",
    "blockchain": "ethereum",
    "protocol": "usp3",
    "pool_address":
      "0xb1914469...85e5",
    "block_number": 25340719,
    "timestamp":
      "2026-06-18T00:00:11.000Z",
    "transaction_hash":
      "0xc260a450...0b37",
    "log_index": 132,
    "user_address":
      "0x5b43453f...edd1",
    "base_token_address":
      "0xaa7a9ca8...0a6f",
    "quote_token_address":
      "0xc02aaa39...6cc2",
    "price": "0.000203185107933882",
    "amount": "7950.022160479814",
    "taker_side_sell": false
  }, ... ],
  "meta": { "api_version": "2026-10-01",
    "url": "...", "count": 4,
    "next_url": null }
}
```

Transaction, log index, block, time and pool match. The account bought TRAC (`taker_side_sell` `false`), so the base token is Amberdata's bought asset and the quote token its sold asset; `price` equals `assetSoldPrice`, `protocol` `usp3` is `uniswap_v3`, and `user_address` is `transactionFromAddress`. Amberdata's `tradeId` is `transaction_hash` + "_132".

7 DEX OHLCV

Replaces. `market/defi/ohlcv/{pool}/historical` and `market/defi/ohlcv/{pool}/latest`.

What it gives you. One row per bucket of one pool: the open, high, low and close swap prices, the base and quote volumes and the swap count. A served candle is final: it is built once its bucket has closed.

	Amberdata (today)	Kaiko (new)
Path	`/market/defi/ohlcv/{pool}/historical` `/market/defi/ohlcv/{pool}/latest`	`/defi/trades/ohlcv`

How to call it

Amberdata endpoint	Kaiko query
`{pool}/historical?` `timeInterval=hours`	`blockchain=ethereum&pool_address={pool}&interval=1h` + `start_time` / `end_time`
`{pool}/latest`	`blockchain=ethereum&pool_address={pool}&interval=1m&page_size=1`: the first row is the most recent completed one-minute candle

Request parameters

Amberdata param	Kaiko param	Change	Notes
`pool` (path)	`pool_address`	RENAMED	Required query parameter, the pool's address. A pair symbol such as `DAI_USDC` returns 400.
	`blockchain`	NEW	Required. `ethereum` for the Amberdata pools (§4).
`exchange`		REMOVED	The pool address names the pool on its protocol.
`timeInterval`	`interval`	RENAMED	`minutes` → `1m`, `hours` → `1h`, `days` → `1d` (default, as on Amberdata). Lower case; any other value returns 400.
`startDate` / `endDate`	`start_time` / `end_time`	RENAMED	The buckets whose open is in `[start_time, end_time)`. Default: the 90 days before `end_time`, which defaults to now.
	`sort`	NEW	`desc` (newest bucket first, default) or `asc`.
	`page_size`	NEW	Buckets per page: default 100, maximum 1000. A day of `1m` candles is 1440 rows over two pages.
`timeFormat`		REMOVED	Always ISO 8601.

Response data fields

Historical rows are arrays in `columns` order; the latest candle is an object. Both land on one Kaiko row.

Amberdata field	Kaiko field	Type	Change	Notes
`timestamp`	`timestamp`	string	RENAMED	Open of the bucket, ISO 8601 instead of epoch milliseconds (§5).

Amberdata field	Kaiko field	Type	Change	Notes
	`blockchain`	string	NEW	As requested.
`pairAddress`	`pool_address`	string	RENAMED	Lower case on EVM blockchains.
`open`, `high`, `low`, `close`	`open`, `high`, `low`, `close`	string or null	IDENTICAL	Quote tokens per base token (§5). `null` when the bucket has no swap.
`volume`	`volume`	string	IDENTICAL	Base tokens swapped in the bucket.
`quoteVolume`	`quote_volume`	string	RENAMED	Quote tokens: each swap's amount times its price.
`trades`	`trade_count`	integer	RENAMED	Swaps in the bucket. Was a string on historical rows.
`baseAddress`, `quoteAddress`			REMOVED	`base_token_address` and `quote_token_address` of the pool's swaps (§5, §6).
`exchangeId`, `blockchainId`			REMOVED	Factory contract address and internal blockchain hash.
`exchangeName`			REMOVED	`protocol` on the pool's swaps, or in reference data.
`baseSymbol`, `quoteSymbol`, `baseDecimals`, `quoteDecimals`, `pair`, `fee`, `tokenSymbols`, `tokenAddresses`, `tokenDecimals`			REMOVED	From reference data per pool: `name`, `fee`, `protocol` and `tokens[]` with address, symbol and decimals (§4).
`baseName`, `quoteName`, `tokenNames`			REMOVED	Token display names.

Before / after example: historical candles

Uniswap V3 USDC/WETH 0.05% pool `0x88e6a0c2...5640`, the hour from 01:00 on 1 November 2026, on both APIs. USDC is the base token on both. Kaiko response live.

Amberdata

```
GET /market/defi/ohlcv/
    0x88e6a0c2...5640/historical/
    ?exchange=uniswapv3
    &timeInterval=hours
    &startDate=2026-11-01T01:00:00
    &endDate=2026-11-01T02:00:00
x-api-key: <your-key>
```

Kaiko

```
GET https://api.kaiko.io
    /defi/trades/ohlcv
    ?blockchain=ethereum
    &pool_address=0x88e6a0c2...5640
    &interval=1h
    &start_time=2026-11-01T01:00:00Z
    &end_time=2026-11-01T02:00:00Z
X-Api-Key: <your-key>
```

Amberdata


```
{
  "status": 200,
  "title": "OK",
  "description": "Successful request",
  "payload": {
    "metadata": {
      "columns": [ "timestamp",
        "open", "high", "low",
        "close", "volume",
        "quoteVolume", "trades",
        "exchangeId", "pairAddress",
        "baseAddress",
        "quoteAddress" ],
      "startDate": 1793494800000,
      "endDate": 1793498400000
    },
    "data": {
      "0x1f98431c...f984": [ [
        1793494800000,
        "0.000442260699714294...",
        "0.000442785251102634...",
        "0.000440767116147998...",
        "0.000440785423331008...",
        "673237.09858",
        "...",
        "147",
        "0x1f98431c...f984",
        "0x88e6a0c2...5640",
        "0xa0b86991...eb48",
        "0xc02aaa39...6cc2"
      ] ]
    }
  }
}
```


Kaiko


```
{
  "data": [ {
    "timestamp":
      "2026-11-01T01:00:00.000Z",
    "blockchain": "ethereum",
    "pool_address":
      "0x88e6a0c2...5640",
    "open": "0.000442260699714294",
    "high": "0.000442785251102634",
    "low": "0.000440767116147998",
    "close": "0.000440785423331008",
    "volume": "673237.09858...",
    "trade_count": 147
  } ],
  "meta": {
    "api_version": "2026-10-01",
    "url": "...",
    "count": 1,
    "next_url": null
  }
}
```


Bucket open, open, high, low, close and trade count match: `1793494800000` is `2026-11-01T01:00:00.000Z`. Prices are WETH per USDC on both. Amberdata's `baseAddress` and `quoteAddress` are Kaiko's `base_token_address` and `quote_token_address` on this pool's swaps.

Before / after example: latest candle

The same pool. The Kaiko call names the minute of the Amberdata candle so the two can be compared. Drop `start_time` and `end_time` and keep `page_size=1` for the most recent candle.

Amberdata

```
GET /market/defi/ohlcv/
    0x88e6a0c2...5640/latest/
    ?exchange=uniswapv3
x-api-key: <your-key>
```

Kaiko

```
GET https://api.kaiko.io
    /defi/trades/ohlcv
    ?blockchain=ethereum
    &pool_address=0x88e6a0c2...5640
    &interval=1m
    &start_time=2026-11-02T14:48:00Z
    &end_time=2026-11-02T14:49:00Z
X-Api-Key: <your-key>
```

Amberdata


```
{
  "status": 200,
  "title": "OK",
  "description": "Successful request",
  "payload": {
    "0x88e6a0c2...5640": {
      "timestamp": 1793630880000,
      "open":
        "0.000436951476094221...",
      "high":
        "0.00043717148800915...",
      "low":
        "0.000436636384547435...",
      "close":
        "0.00043680468752080...",
      "volume": "59410.047875",
      "trades": 4,
      "blockchainId":
        "1c9c969065fcd1cf",
      "exchangeId":
        "0x1f98431c...f984",
      "exchangeName": "uniswapv3",
      "pairAddress":
        "0x88e6a0c2...5640",
      "baseAddress":
        "0xa0b86991...eb48",
      "quoteAddress":
        "0xc02aaa39...6cc2",
      "baseSymbol": "USDC",
      "quoteSymbol": "WETH",
      "baseDecimals": "6",
      "quoteDecimals": "18",
      "pair": "USDC_WETH",
      "fee": "0.0005",
      ...
    }
  }
}
```


Kaiko


```
{
  "data": [ {
    "timestamp":
      "2026-11-02T14:48:00.000Z",
    "blockchain": "ethereum",
    "pool_address":
      "0x88e6a0c2...5640",
    "open": "0.000436951476094221",
    "high": "0.00043717148800915...",
    "low": "0.000436636384547435",
    "close": "0.00043680468752080...",
    "volume": "59410.047875",
    "trade_count": 4
  } ],
  "meta": {
    "api_version": "2026-10-01",
    "url": "...",
    "count": 1,
    "next_url": null
  }
}
```


Bucket open (`1793630880000` is `2026-11-02T14:48:00.000Z`) is the same instant on both. The pool's description (name, fee, token symbols and decimals) is in reference data.