
title: Quick Start

Let's use mev-inspect to find the same arbitrage as [MEV Alpha Leak](#)

Install

mev-inspect-py is built to run on kubernetes locally and in production

Install dependencies

First, setup a local kubernetes deployment - we use [Docker](#) and [kind](#)

If using kind, create a new cluster with:

```
sh kind create cluster
```

Next, install the kubernetes CLI [kubect1](#)

Then, install [helm](#) - helm is a package manager for kubernetes

Lastly, setup [Tilt](#) which manages running and updating kubernetes resources locally

Start up

Set an environment variable `RPC_URL` to an RPC for fetching blocks Example:

```
sh export RPC_URL="http://111.111.111.111:8546"
```

:::note

mev-inspect-py currently requires an RPC with support for OpenEthereum / Erigon traces (not geth 😊)

:::

Next, start all services with:

```
sh tilt up
```

Press "space" to see a browser of the services starting up

On first startup, you'll need to apply database migrations. Apply with:

```
sh ./mev exec alembic upgrade head
```

Tear down

First stop the running tilt window with `Ctrl+C`

Then run

```
sh tilt down
```

Inspect a block

Using the [linked etherscan transaction](#), we can see the block number is 12914944.

To inspect this block, run

```
sh ./mev inspect 12914944
```

Connect to Postgres

We'll connect to the Postgres database to see the data inspect found in that block

Let's start up a client container connected to the DB:

```
sh ./mev db
```

When you see the prompt

```
sh mev_inspect=#
```

You're ready to query!

To make the data display nice, switch into "Expanded display" mode by running

```
sh \x
```

Query for arbitrage data

Let's find that arbitrage by querying the `arbitrages` table:

```
sql SELECT * FROM arbitrages WHERE block_number = 12914944 AND transaction_hash =
'0xfcf4558f6432689ea57737fe63124a5ec39fd6ba6aaf198df13a825dd599bffc'
```

You should see output like this:

```
txt id | ff2deb13-c2c1-4ef5-a6ff-0ca813a07d6b created_at | 2021-09-27 15:26:58.193263 account_address |
0x0000fee6275dab194ab538a01dd8b18b02b20000 profit_token_address | 0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2
block_number | 12914944 transaction_hash | 0xfcf4558f6432689ea57737fe63124a5ec39fd6ba6aaf198df13a825dd599bffc
start_amount | 70287643212620210176 end_amount | 123848351154563483804 profit_amount | 53560707941943273628
```

We can see this matches the original tweet description!

The `profit_token_address` is the address for WETH, our `start_amount` is 70 WETH (assuming 18 decimals), and our `end_amount` is 123 WETH

Query for arbitrage swaps

We can learn about the swaps involed in this arbitrage by joining against the `arbitrage_swaps` and `swaps` tables

:::note You'll need to switch in the id you got in the first query for `arbitrage_id` :::

```
sql SELECT s.* FROM swaps s JOIN arbitrage_swaps arb_swaps ON s.transaction_hash =
arb_swaps.swap_transaction_hash AND s.trace_address = arb_swaps.swap_trace_address WHERE arb_swaps.arbitrage_id
= 'ff2deb13-c2c1-4ef5-a6ff-0ca813a07d6b';
```

You should see output like this:

```
txt -[ RECORD 1 ]-----+----- created_at | 2021-
09-27 15:26:58.180131 abi_name | UniswapV3Pool transaction_hash |
0xfcf4558f6432689ea57737fe63124a5ec39fd6ba6aaf198df13a825dd599bffc block_number | 12914944 protocol |
pool_address | 0x88e6a0c2ddd26feeb64f039a2c41296fcb3f5640 from_address |
0x7ec0b75a98997c927ace6d87958147a105147ea0 to_address | 0x0000fee6275dab194ab538a01dd8b18b02b20000
token_in_address | 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48 token_in_amount | 283588902010 token_out_address |
0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2 token_out_amount | 123848351154563483804 trace_address | {0,0} error
| -[ RECORD 2 ]-----+----- ...
```

Query for miner payment

Lastly, we can see how much was paid to the miner for this transaction by querying by the transaction hash:

```
sql SELECT * FROM miner_payments WHERE transaction_hash =
'0xfcf4558f6432689ea57737fe63124a5ec39fd6ba6aaf198df13a825dd599bffc';
```

You should see results like this:

```
txt created_at | 2021-09-27 15:26:58.245444 block_number | 12914944 transaction_hash |
0xfcf4558f6432689ea57737fe63124a5ec39fd6ba6aaf198df13a825dd599bffc transaction_index | 1 miner_address |
0x5A0b54D5dc17e0AadC383d2db43B0a0D3E029c4c coinbase_transfer | 48204637147748941824 base_fee_per_gas | 0
gas_price | 0 gas_price_with_coinbase_transfer | 200463421638605 gas_used | 240466 transaction_to_address |
```

0x00000000454a11ca3a574738c0aab442b62d5d45 transaction_from_address | 0xd80276cd0348e9b3c5d017e1f7529f0a785fec3a

gas_price is the gas price paid directly as gas (it includes the EIP-1559 base fee)

coinbase_transfer is the amount of ETH paid directly as a transfer to the validator's address

gas_price_with_coinbase_transfer is the gas price including both original gas and coinbase transfers

So in total, this searcher paid 48.2 ETH to make 53.5 ETH for a net profit of 5.3 ETH

Next steps

To see what other data is available for querying, check out the [data](#) section

To learn about inspecting blocks in bulk or listening for new blocks as they come in, check out the [inspecting](#) section