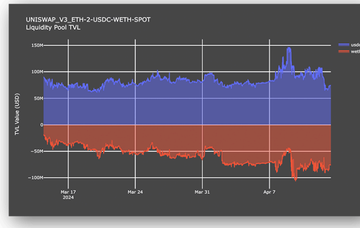
COINMETRICS
CM LABS: DEFI DATA & ATLAS
>>> LIQUIDITY POOL TVL DEMO



Automated Market Makers (AMMs) are an innovative new class of smart contracts introduced in decentralized exchange protocols like Uniswap, allowing users to permissionlessly provision liquidity for digital asset trading activity without needing a traditional central order book. However, an important aspect of providing users of these pools is understanding how the total USD value of the assets allocated to the contract can fluctuate over time, leading to impermanent loss for liquidity providers. In this notebook, we explore how Coin Metrics DEX market metadata can be combined with Reference Rates and ATLAS search engine capabilities to construct a timeseries representation pool TVL, allowing market participants to make more informed decisions about DEX market making and trading.

Resources

This notebook demonstrates basic functionality offered by the Coin Metrics Python API Client, ATLAS blockchain search engine, and DEX Market Data.

Coin Metrics offers a vast assortment of data for hundreds of cryptoassets. The Python API Client allows for easy access to this data using Python without needing to create your own wrappers using `requests` and other such libraries.

To understand the data that Coin Metrics offers, feel free to peruse the resources below.

- The Coin Metrics API v4 website contains the full set of endpoints and data offered by Coin Metrics.
- The Coin Metrics Knowledge Base gives detailed, conceptual explanations of the data that Coin Metrics offers.
- The API Spec contains a full list of functions.

Notebook Setup

In [1]:
```python
from os import environ
import sys
import pandas as pd
import numpy as np
import logging
from datetime import date, datetime, timedelta
from coinmetrics.api_client import CoinMetricsClient
import plotly.graph_objs as go
import logging
from pytz import timezone as timezone_conv
from datetime import timezone as timezone_info
from dateutil.relativedelta import relativedelta
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```python
logging.basicConfig(
    format='%(asctime)s %(levelname)-8s %(message)s',
    level=logging.INFO,
    datefmt='%Y-%m-%d %H:%M:%S'
)
```

In [3]:
```python
# We recommend privately storing your API key in your local environment.
try:
    api_key = environ["CM_API_KEY"]
    logging.info("Using API key found in environment")
except KeyError:
    api_key = ""
    logging.info("API key not found. Using community client")

client = CoinMetricsClient(api_key)
```

```
2024-04-13 10:32:37 INFO     Using API key found in environment
```

DEX Market Reference Data

The *reference-data/markets* endpoint returns a list of available markets meeting specified criteria. Users can pass in a list of markets, exchanges, or market types (spot, futures, options). For DEX markets, the endpoint also returns key liquidity pool metadata, such as fee tier and pool contract address.

In [4]:
```python
uni_v3_markets = client.reference_data_markets(
    exchange = 'uniswap_v3_eth'
).to_dataframe()
```

In [5]:
```python
uni_v3_markets
```

Out[5]:

	market	exchange	type	base	quote	pair	pool_config_id	contract_address	fee	bas
0	uniswap_v3_eth-1-1inch-dai-spot	uniswap_v3_eth	spot	1inch	dai	1inch-dai	1	063332bbf9f8385e4106919b5c6ae2e6a4f72228	0.01	111111111117dc0aa78b770fa6a7380
1	uniswap_v3_eth-1-1inch-usdc-spot	uniswap_v3_eth	spot	1inch	usdc	1inch-usdc	1	2ee7e6e459fffbbc655f09f2e1b3131abf98c397	0.01	111111111117dc0aa78b770fa6a7380
2	uniswap_v3_eth-1-1inch-weth-spot	uniswap_v3_eth	spot	1inch	weth	1inch-weth	1	1d1284e43da1de5ee8dd6acbb03f3624cfbd872c	0.01	111111111117dc0aa78b770fa6a7380
3	uniswap_v3_eth-1-ageur_eth-usdc-spot	uniswap_v3_eth	spot	ageur_eth	usdc	ageur_eth-usdc	1	735a26a57a0a0069dfabd41595a970faf5e1ee8b	0.01	1a7e4e63778b4f12a199c062f3efdd2
4	uniswap_v3_eth-1-ape-weth-spot	uniswap_v3_eth	spot	ape	weth	ape-weth	1	a82815da610e55e582dc3c433bb2a44923d63542	0.01	4d224452801aced8b2f0aebe155379bb
...	...	...	...	...	...	...	...	...	...	
1519	uniswap_v3_eth-agg-yfi-cvx-spot	uniswap_v3_eth	spot	yfi	cvx	yfi-cvx	-1	<NA>	<NA>	0bc529c00c6401aef6d220be8c6ea16
1520	uniswap_v3_eth-agg-yfi-link-spot	uniswap_v3_eth	spot	yfi	link	yfi-link	-1	<NA>	<NA>	0bc529c00c6401aef6d220be8c6ea16
1521	uniswap_v3_eth-agg-yfi-usdc-spot	uniswap_v3_eth	spot	yfi	usdc	yfi-usdc	-1	<NA>	<NA>	0bc529c00c6401aef6d220be8c6ea16
1522	uniswap_v3_eth-agg-yfi-wbtc-spot	uniswap_v3_eth	spot	yfi	wbtc	yfi-wbtc	-1	<NA>	<NA>	0bc529c00c6401aef6d220be8c6ea16
1523	uniswap_v3_eth-agg-yfi-weth-spot	uniswap_v3_eth	spot	yfi	weth	yfi-weth	-1	<NA>	<NA>	0bc529c00c6401aef6d220be8c6ea16

1524 rows × 12 columns

In [6]:
```
weth_usdc_markets = uni_v3_markets.loc[(uni_v3_markets['base']=='usdc') & (uni_v3_markets['quote']=='weth')]
```

In [7]:
```
weth_usdc_pools = weth_usdc_markets.dropna(subset=['contract_address'])
weth_usdc_pools
```

Out[7]:

	market	exchange	type	base	quote	pair	pool_config_id	contract_address	fee	base_address
55	uniswap_v3_eth-1-usdc-weth-spot	uniswap_v3_eth	spot	usdc	weth	usdc-weth	1	e0554a476a092703abdb3ef35c80e0d76d32939f	0.01	a0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
241	uniswap_v3_eth-2-usdc-weth-spot	uniswap_v3_eth	spot	usdc	weth	usdc-weth	2	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	0.05	a0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
568	uniswap_v3_eth-3-usdc-weth-spot	uniswap_v3_eth	spot	usdc	weth	usdc-weth	3	8ad599c3a0ff1de082011efddc58f1908eb6e6d8	0.3	a0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
926	uniswap_v3_eth-4-usdc-weth-spot	uniswap_v3_eth	spot	usdc	weth	usdc-weth	4	7bea39867e4169dbe237d55c8242a8f2fcdcc387	1.0	a0b86991c6218b36c1d19d4a2e9eb0ce3606eb48

Fetch contract balances over time with ATLAS

Now that we have a list of target liquidity pool contracts, we can use ATLAS blockchain search engine to query for balance updates in the pool for each asset.

In [8]:
```
assets = ['usdc','weth']
pools = weth_usdc_pools['contract_address'].to_list()
pools_tvl = pd.DataFrame()
start = datetime.now() - timedelta(days=30)

for asset in assets:
    tvl = client.get_list_of_balance_updates_v2(
        asset=asset,
        accounts=pools,
        start_time = start
    ).parallel(max_workers=10,time_increment=relativedelta(days=1)).to_dataframe()

    # Add the asset name to a new 'asset' column
    tvl['asset'] = asset
    pools_tvl = pd.concat([pools_tvl, tvl], axis=0)
```

```
Exporting to dataframe type:  68%|██████▊   | 21/31 [00:22<00:08,  1.16it/s]2024-04-13 10:33:28 INFO     no data to export
Exporting to dataframe type: 100%|██████████| 31/31 [00:31<00:00,  1.03s/it]
Exporting to dataframe type: 100%|██████████| 31/31 [00:30<00:00,  1.03it/s]
```

In [9]:
```
# Create a mapping from contract_address to market
contract_to_market = weth_usdc_pools.set_index('contract_address')['market']
contract_to_market
```

Out[9]:
```
contract_address
e0554a476a092703abdb3ef35c80e0d76d32939f    uniswap_v3_eth-1-usdc-weth-spot
88e6a0c2ddd26feeb64f039a2c41296fcb3f5640    uniswap_v3_eth-2-usdc-weth-spot
8ad599c3a0ff1de082011efddc58f1908eb6e6d8    uniswap_v3_eth-3-usdc-weth-spot
7bea39867e4169dbe237d55c8242a8f2fcdcc387    uniswap_v3_eth-4-usdc-weth-spot
Name: market, dtype: string
```

In [10]:
```
pools_tvl['market'] = pools_tvl['account'].map(contract_to_market)
```

In [11]:
```
pools_tvl
```

Out[11]:

	chain_sequence_number	account	account_creation_height	change	previous_balance	new_balance	transaction_sequence_
0	83462733663567872	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	-53891.323631	90400786.260267	90346894.936636	
1	83462737958535168	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	-1825.310434	90346894.936636	90345069.626202	
2	83462737958535179	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	6710.02311	90345069.626202	90351779.649312	
3	83462737958535189	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	3511.911269	90351779.649312	90355291.560581	
4	83462742253502464	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	-397.849747	90355291.560581	90354893.710834	
...	...	...	...	...	...	...	
235162	84379262504665167	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	0.06	22993.783003	22993.843003	
235163	84379262504665199	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	0.641	22993.843003	22994.484003	
235164	84379266799632464	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	-0.461357	22994.484003	22994.022646	
235165	84379266799632469	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	0.641	22994.022646	22994.663646	
235166	84379271094599756	88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	12376729	-11.053992	22994.663646	22983.609655	

470671 rows × 21 columns

In [12]:

```
# Sort by 'consensus_time', 'market', and 'chain_sequence_number'
pools_tvl_sorted = pools_tvl.sort_values(by=['consensus_time', 'market', 'chain_sequence_number'], ascending=[True, True, False])

# Drop duplicates, keeping only the first occurrence (in this case, the highest 'chain_sequence_number')
pools_tvl_deduplicated = pools_tvl_sorted.drop_duplicates(subset=['consensus_time', 'market','asset'], keep='first')
```

In [13]:

```
pools_tvl_clean = pools_tvl_deduplicated[['market','consensus_time','asset','new_balance']].copy()
pools_tvl_clean['new_balance'] = pools_tvl_clean.apply(
    lambda row: -row['new_balance'] if row['asset'] == 'weth' else row['new_balance'],
    axis=1
)
pools_tvl_clean
```

Out[13]:

	market	consensus_time	asset	new_balance
0	uniswap_v3_eth-2-usdc-weth-spot	2024-03-14 10:33:23+00:00	weth	-4.648163e+03
0	uniswap_v3_eth-2-usdc-weth-spot	2024-03-14 10:33:23+00:00	usdc	9.034689e+07
3	uniswap_v3_eth-2-usdc-weth-spot	2024-03-14 10:33:35+00:00	weth	-4.646056e+03
3	uniswap_v3_eth-2-usdc-weth-spot	2024-03-14 10:33:35+00:00	usdc	9.035529e+07
5	uniswap_v3_eth-2-usdc-weth-spot	2024-03-14 10:33:47+00:00	weth	-4.650156e+03
...	...	...	...	...
235498	uniswap_v3_eth-2-usdc-weth-spot	2024-04-13 10:32:47+00:00	usdc	7.469295e+07
235163	uniswap_v3_eth-2-usdc-weth-spot	2024-04-13 10:32:59+00:00	weth	-2.299448e+04
235503	uniswap_v3_eth-2-usdc-weth-spot	2024-04-13 10:32:59+00:00	usdc	7.472405e+07
235165	uniswap_v3_eth-2-usdc-weth-spot	2024-04-13 10:33:11+00:00	weth	-2.299466e+04
235166	uniswap_v3_eth-2-usdc-weth-spot	2024-04-13 10:33:23+00:00	weth	-2.298361e+04

285676 rows × 4 columns

In [14]:

```
df = pd.DataFrame(pools_tvl_clean)
df['consensus_time'] = pd.to_datetime(df['consensus_time'])
# Split the DataFrame by market and store in a dictionary
market_dfs = {market: group.pivot(index='consensus_time', columns='asset', values='new_balance').ffill()
              for market, group in df.groupby('market')}
```

In [15]:

```
display(market_dfs)
```

```
{'uniswap_v3_eth-1-usdc-weth-spot': asset                              usdc      weth
 consensus_time
 2024-03-14 10:58:11+00:00  23461.368412 -3.525245
 2024-03-14 11:05:23+00:00  23662.238412 -3.474646
 2024-03-14 11:15:23+00:00  23652.198473 -3.477165
 2024-03-14 11:39:59+00:00  23457.101070 -3.526335
 2024-03-14 14:39:23+00:00  23181.148846 -3.596606
 ...                                 ...       ...
 2024-04-12 21:51:11+00:00  19870.578310 -3.065966
 2024-04-13 01:19:47+00:00  19870.255341 -3.066066
 2024-04-13 01:19:59+00:00  19867.032773 -3.067066
 2024-04-13 05:12:11+00:00  19878.305647 -3.063586
 2024-04-13 07:57:59+00:00  19878.305321 -3.063586

 [2771 rows x 2 columns],
 'uniswap_v3_eth-2-usdc-weth-spot': asset                              usdc          weth
 consensus_time
 2024-03-14 10:33:23+00:00  9.034689e+07  -4648.163465
 2024-03-14 10:33:35+00:00  9.035529e+07  -4646.055509
 2024-03-14 10:33:47+00:00  9.033898e+07  -4650.155509
 2024-03-14 10:33:59+00:00  9.032670e+07  -4653.243577
 2024-03-14 10:34:47+00:00  9.065965e+07  -4569.671932
 ...                                 ...           ...
 2024-04-13 10:32:35+00:00  7.469298e+07 -23003.986469
 2024-04-13 10:32:47+00:00  7.469295e+07 -23003.995469
 2024-04-13 10:32:59+00:00  7.472405e+07 -22994.484003
 2024-04-13 10:33:11+00:00  7.472405e+07 -22994.663646
 2024-04-13 10:33:23+00:00  7.472405e+07 -22983.609655

 [126067 rows x 2 columns],
 'uniswap_v3_eth-3-usdc-weth-spot': asset                              usdc          weth
 consensus_time
 2024-03-14 10:49:23+00:00  2.864290e+07  -5245.181793
 2024-03-14 10:50:47+00:00  2.857072e+07  -5263.348992
 2024-03-14 10:52:59+00:00  2.857051e+07  -5263.402223
 2024-03-14 10:55:59+00:00  2.849030e+07  -5283.611135
 2024-03-14 10:57:11+00:00  2.848734e+07  -5284.355245
 ...                                 ...           ...
 2024-04-13 09:42:59+00:00  3.393681e+07 -14697.377392
 2024-04-13 09:48:23+00:00  3.400722e+07 -14675.836169
 2024-04-13 09:49:23+00:00  3.413429e+07 -14636.980474
 2024-04-13 10:01:59+00:00  3.422011e+07 -14610.750432
 2024-04-13 10:21:23+00:00  3.422911e+07 -14608.000282

 [12957 rows x 2 columns],
 'uniswap_v3_eth-4-usdc-weth-spot': asset                              usdc        weth
 consensus_time
 2024-03-14 11:41:47+00:00  2.642002e+06 -283.532436
 2024-03-14 11:43:59+00:00  2.640952e+06 -283.798518
 2024-03-14 12:23:47+00:00  2.622886e+06 -288.384197
 2024-03-14 12:40:23+00:00  2.620937e+06 -288.879822
 2024-03-14 13:25:59+00:00  2.618804e+06 -289.422528
 ...                                 ...         ...
 2024-04-13 06:45:47+00:00  1.484304e+06 -515.811281
 2024-04-13 06:59:11+00:00  1.491348e+06 -513.655788
 2024-04-13 07:02:11+00:00  1.492328e+06 -513.355865
 2024-04-13 10:12:23+00:00  1.492931e+06 -513.171680
 2024-04-13 10:16:23+00:00  1.497931e+06 -511.643648

 [1149 rows x 2 columns]}
```

Retrieve Reference Rates to calculate the equivalent USD value for TVL

To normalize pool TVL into USD-denominated terms, we'll leverage the Coin Metrics Reference Rate, which represents a volume-weighted median price across a subset of the asset's most highly-liquid markets.

In [16]:
```
ref_rate = client.get_asset_metrics(
    assets=['usdc','weth'],
    metrics='ReferenceRateUSD',
    start_time=start,
    frequency='1m'
).parallel(max_workers=10,time_increment=relativedelta(days=1)).to_dataframe()
```

```
Exporting to dataframe type: 100%|██████████████| 62/62 [00:09<00:00,  6.80it/s]
```

In [17]:
```
ref_rate = ref_rate.pivot(index='time', columns='asset', values='ReferenceRateUSD')
ref_rate
```

Out[17]:

asset	usdc	weth
time		
2024-03-14 10:34:00+00:00	0.999897	3985.438079
2024-03-14 10:35:00+00:00	0.99994	3982.89304
2024-03-14 10:36:00+00:00	0.999912	3982.89304
2024-03-14 10:37:00+00:00	1.000045	3982.89304
2024-03-14 10:38:00+00:00	1.000003	3982.89304
...	...	...
2024-04-13 10:30:00+00:00	1.000031	3262.654799
2024-04-13 10:31:00+00:00	0.999591	3264.797214
2024-04-13 10:32:00+00:00	1.000048	3265.381434
2024-04-13 10:33:00+00:00	0.99999	3267.569238
2024-04-13 10:34:00+00:00	1.000103	3270.443579

43201 rows × 2 columns

In [18]:

```
# Iterate over each market DataFrame
for market, df in market_dfs.items():
    # Resample the DataFrame to 1-minute intervals
    df_resampled = df.resample('min').last().dropna()

    # Reindex the market DataFrame to the ref_rate DataFrame's index
    aligned_df = df_resampled.reindex(ref_rate.index, method='nearest')

    # Multiply the 'usdc' and 'weth' columns by the corresponding rate
    aligned_df['usdc'] = aligned_df['usdc'] * ref_rate['usdc']
    aligned_df['weth'] = aligned_df['weth'] * ref_rate['weth']

    # Replace the original DataFrame in the dictionary with the updated one
    market_dfs[market] = aligned_df
```

In [19]:

```
first_pool_key = list(market_dfs.keys())[0]
first_pool = market_dfs[first_pool_key]
first_pool
```

Out[19]:

asset	usdc	weth
time		
2024-03-14 10:34:00+00:00	23458.951944	-14049.646311
2024-03-14 10:35:00+00:00	23459.955449	-14040.674423
2024-03-14 10:36:00+00:00	23459.308652	-14040.674423
2024-03-14 10:37:00+00:00	23462.428113	-14040.674423
2024-03-14 10:38:00+00:00	23461.427278	-14040.674423
...	...	...
2024-04-13 10:30:00+00:00	19878.927927	-9995.422358
2024-04-13 10:31:00+00:00	19870.168805	-10001.985828
2024-04-13 10:32:00+00:00	19879.251575	-10003.775636
2024-04-13 10:33:00+00:00	19878.101981	-10010.478162
2024-04-13 10:34:00+00:00	19880.358821	-10019.283953

43201 rows × 2 columns

Plot USD-denominated TVL for target liquidity pools

In [20]:

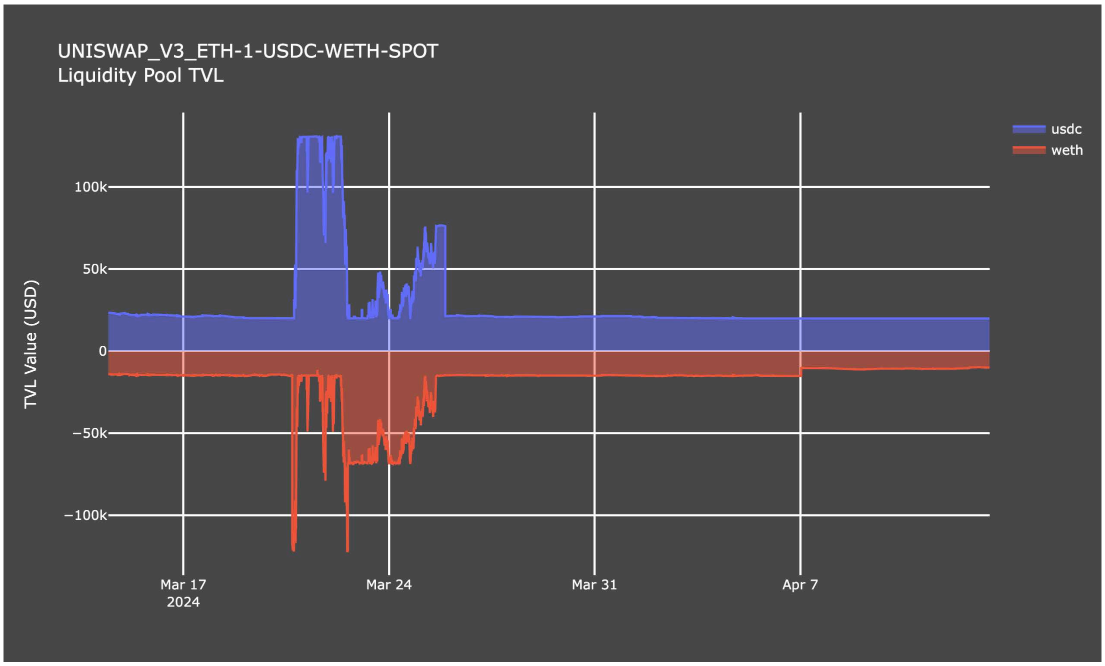
```
def generate_area_figure(df, layout, columns, diverging_colors=False):
    traces = []
    for series in columns:
        traces.append(
            go.Scatter(
                x=df.index,
                y=df[series],
                name=series,
                fill='tozeroy'  # Ensures filling to the zero line on the y-axis
            ))
    return go.Figure(data=traces, layout=layout)
```

In [21]:

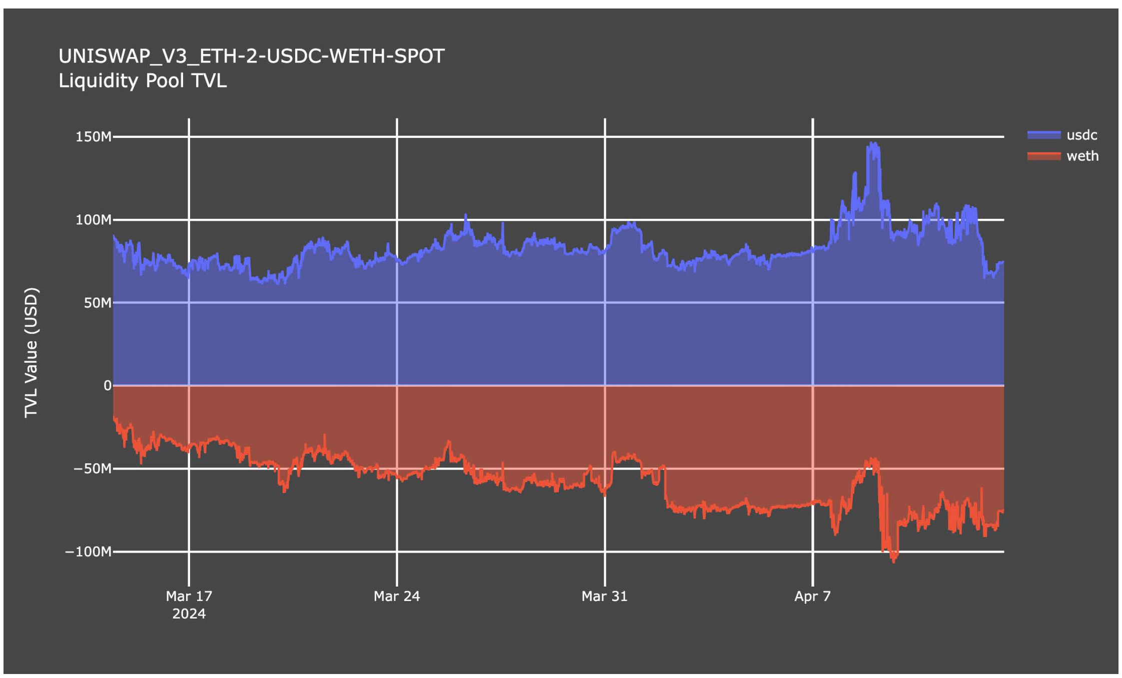
```
market_to_contract = contract_to_market.reset_index().set_index('market')
# Plotting for each market
for market, data in market_dfs.items():
    address = market_to_contract.loc[market, 'contract_address']
    print(f'{market}')
    print(f'{address}')
    print(f'USDC: https://atlas.coinmetrics.io/address-details?asset=usdc&address={address}')
    print(f'WETH: https://atlas.coinmetrics.io/address-details?asset=weth&address={address}')

    layout = go.Layout(
        title=f'{market.upper()}<br>Liquidity Pool TVL',
        xaxis=dict(
            title='',
            gridcolor='white',
            gridwidth=2,
            zerolinecolor='white',
            zerolinewidth=2,
```

```
            color='white'
        ),
        yaxis=dict(
            title='<br>TVL Value (USD)',
            gridcolor='white',
            gridwidth=2,
            zerolinecolor='white',
            zerolinewidth=2,
            color='white'
        ),
        showlegend=True,
        plot_bgcolor='#49494a',
        paper_bgcolor='#49494a',
        font=dict(color='white'),
        width=1000,
        height=600
    )

    fig = generate_area_figure(
        df=data,
        layout=layout,
        columns=data.columns,
        diverging_colors=True
    )

    fig.show()
```

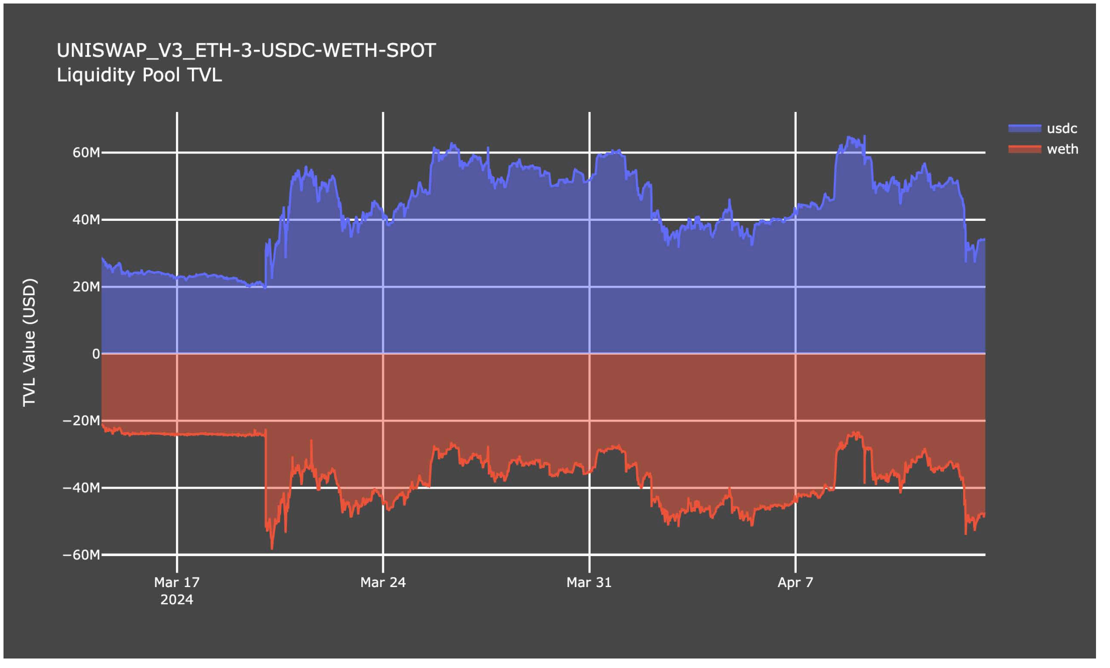
```
uniswap_v3_eth-1-usdc-weth-spot
e0554a476a092703abdb3ef35c80e0d76d32939f
USDC: https://atlas.coinmetrics.io/address-details?asset=usdc&address=e0554a476a092703abdb3ef35c80e0d76d32939f
WETH: https://atlas.coinmetrics.io/address-details?asset=weth&address=e0554a476a092703abdb3ef35c80e0d76d32939f
```




```
uniswap_v3_eth-2-usdc-weth-spot
88e6a0c2ddd26feeb64f039a2c41296fcb3f5640
USDC: https://atlas.coinmetrics.io/address-details?asset=usdc&address=88e6a0c2ddd26feeb64f039a2c41296fcb3f5640
WETH: https://atlas.coinmetrics.io/address-details?asset=weth&address=88e6a0c2ddd26feeb64f039a2c41296fcb3f5640
```



uniswap_v3_eth-3-usdc-weth-spot
8ad599c3a0ff1de082011efddc58f1908eb6e6d8
USDC: https://atlas.coinmetrics.io/address-details?asset=usdc&address=8ad599c3a0ff1de082011efddc58f1908eb6e6d8
WETH: https://atlas.coinmetrics.io/address-details?asset=weth&address=8ad599c3a0ff1de082011efddc58f1908eb6e6d8



uniswap_v3_eth-4-usdc-weth-spot
7bea39867e4169dbe237d55c8242a8f2fcdcc387
USDC: https://atlas.coinmetrics.io/address-details?asset=usdc&address=7bea39867e4169dbe237d55c8242a8f2fcdcc387
WETH: https://atlas.coinmetrics.io/address-details?asset=weth&address=7bea39867e4169dbe237d55c8242a8f2fcdcc387