AI v4 dYdX Orderbook: Installation Guide

Author: xAI’s Grok, with prompting by Lawrence Chiu (lawrence@int.dydx.foundation)

Twitter: https://twitter.com/LawrenceChiu14

First release Date: 8/25/2025

Tested on: Ubuntu Server 22.04 LTS, 16 vCPU, 64 GiB Memory. Recommended machine-type on Google Cloud: t2d-standard-16

Change Log

1. 8/25/2025 First release

Design Considerations

1. Coded entirely using AI (xAI’s Grok)
2. Uses **picows** high-performance websocket library, **uvloop** for improved asyncio performance, **asyncpg** for asynchronous database operations, and **psutil** for memory monitoring.
3. Storage backing by PostgreSQL

Part 1) Setting up

1. Install PostgreSQL. You’ll also need python3-pip.

```
sudo apt-get install postgresql
Sudo apt-get install python3-pip
```

2. Install the Python libraries.

```
pip3 install picows
pip3 install uvloop
```

```
pip3 install sayncpg
pip3 install psutil
```

3. Create the database and required tables. In this example, ‘vmware’ is the OS user that will run the orderbook.

```
sudo su - postgres
psql
create database orderbook;
create user vmware with encrypted password 'orderbook';
grant all privileges on database orderbook to vmware;
exit
```

4. Configure PostgreSQL to allow network connections:

   a. Add the following line to /etc/postgresql/14/main/postgresql.conf:

listen_addresses = '*'

   b. Change the following line:

From: max_connections = 100

To: max_connections = 10000

   c. Next, open file pg_hba.conf and change the following line:

```
From: host    all        all        127.0.0.1/32        scram-sha-256
To: host    all        all        0.0.0.0/0        scram-sha-256
```

5. Create the directory /mnt/ramdisk5/

```
sudo mkdir /mnt/ramdisk5
sudo chmod 777 /mnt/ramdisk5
```

6. (Optional) Back the /mnt/ramdisk5/ directory with a ramdisk.

```
sudo mount -t tmpfs -o rw,size=8G tmpfs /mnt/ramdisk5
```

Part 2) Checklist

1. There are 4 programs:

a. v4dydxob.sh (Run this program)
b. v4dydxob.py (the actual orderbook program that reads from indexer websocket)
c. v4dydxob2.py (the display program to show the orderbook)

You run a) v4dydxob.sh which runs the b) v4dydxob.py for you.

Part 3) v4dydxob.sh

1. Run v4dydxob.sh. It takes only 1 argument which is the market (e.g. BTC-USD, ETH-USD, etc.)

```
nohup ./v4dydxob.sh BTC-USD > /tmp/v4dydxobBTC-USD.log 2>&1 &
```

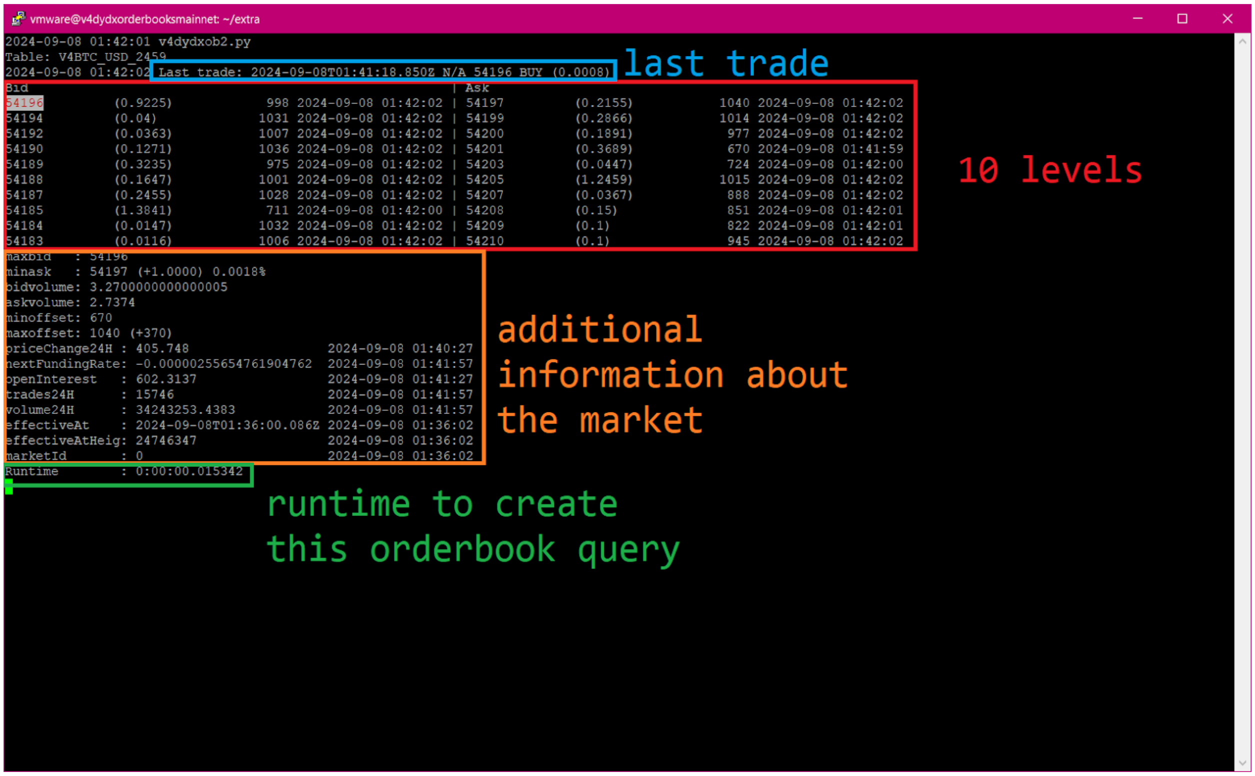
Part 4) v4dydxob2.py

1. (Optional) This program can run from the same server as v4dydxob.py OR a remote server. To run this on a remote server, set the environment variable with the IP address of the order book server. If this is not set, the program assumes the same server.

```
export ORDERBOOKSERVER=192.168.0.169
```

2. This program displays the orderbook to your screen. It takes up to 3 arguments: 1) the market, b) the number of levels to display, and 3) whether to color the output (specify noansi to disable)
3. For example, to display 10 levels and color the output:

```
python3 -u v4dydxob2.py BTC-USD 10
```



vmware@v4dydxorderbooksmainnet: ~/extra
last trade
10 levels
additional information about the market
runtime to create this orderbook query

```
2024-09-08 01:42:01 v4dydxob2.py
Table: V4BTC_USD_2459
2024-09-08 01:42:02 Last trade: 2024-09-08T01:41:18.850Z N/A 54196 BUY (0.0008)
Bid                                                       | Ask
54196         (0.9225)           998 2024-09-08 01:42:02 | 54197         (0.2155)           1040 2024-09-08 01:42:02
54194         (0.04)            1031 2024-09-08 01:42:02 | 54199         (0.2866)           1014 2024-09-08 01:42:02
54192         (0.0363)          1007 2024-09-08 01:42:02 | 54200         (0.1891)            977 2024-09-08 01:42:02
54190         (0.1271)          1036 2024-09-08 01:42:02 | 54201         (0.3689)            670 2024-09-08 01:41:59
54189         (0.3235)           975 2024-09-08 01:42:02 | 54203         (0.0447)            724 2024-09-08 01:42:00
54188         (0.1647)          1001 2024-09-08 01:42:02 | 54205         (1.2459)           1015 2024-09-08 01:42:02
54187         (0.2455)          1028 2024-09-08 01:42:02 | 54207         (0.0367)            888 2024-09-08 01:42:02
54185         (1.3841)           711 2024-09-08 01:42:00 | 54208         (0.15)              851 2024-09-08 01:42:01
54184         (0.0147)          1032 2024-09-08 01:42:02 | 54209         (0.1)               822 2024-09-08 01:42:01
54183         (0.0116)          1006 2024-09-08 01:42:02 | 54210         (0.1)               945 2024-09-08 01:42:02
maxbid   : 54196
minask   : 54197 (+1.0000) 0.0018%
bidvolume: 3.2700000000000005
askvolume: 2.7374
minoffset: 670
maxoffset: 1040 (+370)
priceChange24H : 405.748                       2024-09-08 01:40:27
nextFundingRate: -0.00000255654761904762  2024-09-08 01:41:57
openInterest   : 602.3137                      2024-09-08 01:41:27
trades24H      : 15746                         2024-09-08 01:41:57
volume24H      : 34243253.4383                 2024-09-08 01:41:57
effectiveAt    : 2024-09-08T01:36:00.086Z 2024-09-08 01:36:02
effectiveAtHeig: 24746347                      2024-09-08 01:36:02
marketId       : 0                             2024-09-08 01:36:02
Runtime        : 0:00:00.015342
```

Notice that the Bid price of 54196 is in red color to show it was also the last traded price.

4. To display without color, and the output would be identical to the above except the Bid price would not be in color. This is useful if you intend to use the data in another program.

```
python3 -u v4dydxob2.py BTC-USD 10 noansi
```

Part 5) DBA Information

1. Log into the database with the following command:

psql -h localhost -d orderbook -U vmware
\pset pager off

2. Various tables you can query:
   a. v4<market1>_usd (for example: v4btc_usd)
   b. v4trades<market1>_usd (v4tradesbtc_usd)
   c. V4markets

Part 6) v4_trades websocket

The programs are: **v4dydxtrades.sh**, and **v4dydxtrades.py**. Just like with the order book, you run v4dydxtrades.sh.

```
nohup ./v4dydxtrades.sh BTC-USD > /tmp/v4dydxtradesBTC-USD.log 2>&1 &
```

Part 7) v4_subaccount websocket

The programs are: **v4dydxsubaccount.sh**, and **v4dydxsubaccount.py**. Just like with the order book, you run v4dydxsubaccount.sh. Note that you specify the dydxchain address, then a slash, then the subaccount. For example dydx1g0y58axjs37asw6856u0fcqexcgrnyu526u22k/0

```
nohup ./v4dydxsubaccount.sh dydx1g0y58axjs37asw6856u0fcqexcgrnyu526u22k/0 > /tmp/v4dydxsubaccount.log 2>&1 &
```

Part 8) v4_markets websocket

The programs are: **v4dydxv4markets.sh**, and **v4dydxv4markets.py**. Just like with the order book, you run v4dydxv4markets.sh.

```
nohup ./v4dydxv4markets.sh > /tmp/v4dydxv4markets.stdout 2>&1 &
```