

# 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

Updated: 8/26/2025

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
2. 8/26/2025 Added more content

## Design Considerations

1. Coded entirely using AI (xAI's Grok)
2. v4\_orderbook: Orderbook uses in-memory structure for high performance. Clients can retrieve this orderbook via curl or client-side program ([v4dydxob2.py](#)). Uses **aiohttp** to create httpd server.
3. v4\_trades, v4\_markets, v4\_subaccounts: Uses **picows** high-performance websocket library, **uvloop** for improved asyncio performance, **asyncpg** for asynchronous database operations, and **psutil** for memory monitoring. 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
Pip3 install aiohttp
```

## 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) Programs

1. There are 3 programs:
  - a. v4dydxob.py (the actual orderbook program that reads from indexer websocket and builds in-memory structure of orderbook)
  - b. v4dydxob.sh (Run this program)
  - c. v4dydxob2.py (the display program to show the orderbook)

### Part 2a) v4dydxob.py (orderbook server)

The parameter `--market <market>` is required (for example `--market BTC-USD`): This creates a http server on port `<n>` where `<n>` is `10000+clob_pair_id`. You can get the `clob_pair_id` from <https://indexer.dydx.trade/v4/perpetualMarkets>. For example, for BTC-USD, the `clob_pair_id` is 0 so the port is 10000. For ETH-USD, it is 1, so the port is 10001.

```
$ python3 v4dydxob.py --help
usage: v4dydxob.py [-h] [--market MARKET]
```

```
dYdX v4 Orderbook WebSocket Server
```

options:

```
-h, --help            show this help message and exit
--market MARKET      Market to subscribe to (e.g., BTC-USD, PEPE-USD) (default: BTC-USD)
```

### Part 2b) v4dydxob.sh (start script for orderbook server)

You run this program which runs [v4dydxob.py](#). It takes 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 2c) v4dydxob2.py (display program)

1. You don't need to use this. You can also get the orderbook by curl.

```
curl http://localhost:<port>/orderbook
```

Example for BTC-USD:

```
curl http://localhost:10000/orderbook
```

2. (Optional) The v4dydxob2.py 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 ORDERBOOKSERVER with the IP address of the order book server. If this is not set, the program assumes the same server. Note that you will use `-ip` parameter to also specify the remote server. To explain why you specify the IP address twice (once in the environment variable and again on the command line, the environment variable is used to query the `v4_trades` and `v4_markets` data in the postgres database and the `-ip` parameter is used to get the orderbook from v4dydxob.py)

```
export ORDERBOOKSERVER=192.168.0.169
```

2. The following parameters are supported. They are self-explanatory except for `-interval`. When `-interval` is set it will loop and refresh the display, however, this is not enabled unless you specify `OB2LOOP=x` environment variable.

```
$ python3 v4dydxob2.py --help
```

```
usage: v4dydxob2.py [-h] [--ip IP] [--market MARKET] [--depth DEPTH] [--interval INTERVAL]
```

dYdX v4 Orderbook Client

options:

<code>-h, --help</code>	show this help message and exit
<code>--ip IP</code>	Server IP address (default: localhost)
<code>--market MARKET</code>	Market to fetch orderbook for (e.g., BTC-USD, PEPE-USD) (default: BTC-USD)
<code>--depth DEPTH</code>	Number of orderbook rows to display (default: 10)
<code>--interval INTERVAL</code>	Interval between requests in seconds when looping (default: 1.0)

3. For example, to display 10 levels:

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

```
vmware@v4dydxorderbooksmainnet: ~/extra
Connecting to server at http://localhost:10000/orderbook
OrderBook for BTC-USD:
Last trade: 2025-08-26T14:57:10 0 110414.0 BUY 0.0025
  BidP | BidQ | AskP | AskQ
110410 | 0.0344 | 110414 | 0.0485
110409 | 0.0906 | 110419 | 0.1358
110408 | 0.1 | 110423 | 0.0369
110406 | 0.1533 | 110425 | 0.0089
110405 | 0.1 | 110430 | 0.2037
110402 | 0.1089 | 110432 | 0.1856
110399 | 0.0688 | 110433 | 0.0437
110398 | 0.1843 | 110436 | 0.0046
110396 | 0.0449 | 110437 | 0.2
110392 | 0.0407 | 110438 | 0.0906
MaxBid : 110410
MinAsk : 110414 (+4, 0.0036%)
BidVolume: 0.9259
AskVolume: 0.9583
priceChange24H : -1811.0501 2025-08-26T14:56:57
nextFundingRate : 0.00001964 2025-08-26T14:56:57
openInterest : 414.0668 2025-08-26T14:56:57
trades24H : 18705 2025-08-26T14:56:57
volume24H : 69383372.7067 2025-08-26T14:56:57
effectiveAt : 2025-08-26T14:56:27 2025-08-26T14:56:57
effectiveAtHeight : 53815177 2025-08-26T14:56:57
marketId : 0 2025-08-26T14:56:57
oraclePrice : 110390.7499 2025-08-26T14:56:57
Runtime : 0:00:00.024733
---
vmware@v4dydxorderbooksmainnet:~/extra$
```

last trade: (from v4\_trades websocket)  
timestamp  
createdatheight = 0  
price = 110414.0  
side = BUY  
size = 0.0025

10 levels by default (can change  
with --depth parameter)

spread

volume at this depth (not entire book)

additional information  
comes from v4\_markets websocket

runtime

## 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. v4trades<market1>\_usd (example: v4tradesbtc\_usd) - this contains all records from v4\_trades websocket channel
  - b. V4markets - this contains the data from the v4\_markets websocket channel

## 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 &
```