Market Maker Keeper Bot Setup Guide

A Marketing Making Bot Set up Guide Level: Intermediate

Estimated Time: 60 minutes

Audience: Developers

Guide Agenda

1. Introduction

This guide is dedicated to showing you how to create your very own Market Maker Keeper Bot as well as educate the community about Market Maker Keeper bots and help both users and developers understand the value of this incredible software. We are proud to say that all of the code needed to get a Market Maker Keeper bot up and running is open-sourced.

List of current exchanges that Market Maker Keeper Bots can be built for

- OasisDEX (oasis-market-maker-keeper
-)
- EtherDelta (etherdelta-market-maker-keeper
-)
- RadarRelay and ERCdEX (0x-market-maker-keeper
-)
- Paradex (paradex-market-maker-keeper
-)
- DDEX (ddex-market-maker-keeper
-)
- Ethfinex (ethfinex-market-maker-keeper
-)
- GoPax (gopax-market-maker-keeper
-)
- OKEX (okex-market-maker-keeper
-)
- TheOcean (theocean-market-maker-keeper
-)
-

1. Prerequisite

2. Git

3. Python v3.6.6
4. virtualenv
5.
    - This project requiresvirtualenv

6.
    - to be installed if you want to use Maker's python tools. This helps with making sure that you are running the right version of python and checks that all of the pip packages that are installed in theinstall.sh
7.
    - are in the right place and have the right versions.
8. *
9. X-code
10. (for macs)
11.

1. Getting Started (Installation)

2. Clone the market-maker-keeper repository and switch into its directory:

```

Copy git clone git@github.com:makerdao/market-maker-keeper.git cd market-maker-keeper

```

1. Initializing the git submodules that will bring in both the pymaker and the pyexchange library:

```

Copy git submodule update --init --recursive

```

1. Set up the virtual env and activate it:

```

Copy python3 -m venv _virtualenv source _virtualenv/bin/activate

```

1. Check to make sure you have the correct version (Python 3.6.6) of Python by running:

```

Copy python3 -V

```

1. Install Requirements:

```

Copy pip3 install (cat requirements.txt (find lib -name requirements.txt | sort) | sort | uniq | sed 's/== /==/g')

```

- Note:
- This command is (used in place ofpip install -r requirements.txt
-) for iterating through all the dependencies in the lib directory to grab all of the requirements needed.
-

Potential errors that could arise:

- Needing to upgrade topip
- version 19.2.2.
-
    - Run:pip install --upgrade pip
-
    - to fix.
- *
- Installingjsonnet
- (if running macOS Mojave)
- To fix, run the following:
-
    - xcode-select --install
-

- open /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg
- pip3 install jsonnet==0.9.5
- Re-run:
- pip3 install (cat requirements.txt (find lib -name requirements.txt | sort) | sort | uniq | sed 's/== /==/g')
- *
-

OtherPotential Installation Issues:

Read the following document for other known Ubuntu and macOS issues (pymaker).

1. Testing

There is quite a lot of value in running all the unit tests to make suremarket-maker keeper has been installed properly. After the repository has been cloned and the installation has been completed, you can run unit tests by executing the following commands.

Firstly, the following command will install the libraries required to run unit tests:

```

Copy pip3 install -r requirements-dev.txt

```

To run the unit tests (py.test, etc..), use the following script:

```

Copy ./test.sh

```

Example output:

```

Copy ==========================================  test session starts  ========================================== platform darwin -- Python 3.6.6, pytest-3.3.0, py-1.8.0, pluggy-0.6.0 rootdir: /Users/charlesst.louis/market-maker-keeper, inifile: plugins: timeout-1.2.1, mock-1.6.3, cov-2.5.1, asyncio-0.8.0 collected 97 items

tests/test_airswap_market_maker_keeper.py ..............  [ 16%] tests/test_band.py ......  [ 22%] tests/test_etherdelta_market_maker_keeper.py .....................  [ 49%] tests/test_feed.py .  [ 50%] tests/test_limit.py .......  [ 57%] tests/test_oasis_market_maker_cancel.py ...  [ 60%] tests/test_oasis_market_maker_keeper.py .....................  [ 80%] tests/test_price_feed.py .....  [ 92%] tests/test_reloadable_config.py .......  [100%]

---------- coverage: platform darwin, python 3.6.6-final-0 ----------- Name Stmts Miss Cover

---

market_maker_keeper/**init**.py 0 0 100% market_maker_keeper/airswap_market_maker_keeper.py 252 142 44% market_maker_keeper/band.py 260 37 86% market_maker_keeper/bibox_market_maker_keeper.py 93 93 0% market_maker_keeper/bitinka_market_maker_keeper.py 96 96 0% market_maker_keeper/bittrex_market_maker_keeper.py 96 96 0% market_maker_keeper/coinbase_market_maker_keeper.py 103 103 0% market_maker_keeper/coinbene_market_maker_keeper.py 95 95 0% market_maker_keeper/control_feed.py 7 5 29% market_maker_keeper/ddex_market_maker_keeper.py 126 126 0% market_maker_keeper/ercdex_market_maker_keeper.py 12 12 0% market_maker_keeper/etherdelta_market_maker_keeper.py 193 142 26% market_maker_keeper/ethfinex_market_maker_keeper.py 94 94 0% market_maker_keeper/feed.py 84 46 45% market_maker_keeper/gas.py 20 10 50% market_maker_keeper/gateio_market_maker_keeper.py 106 106 0% market_maker_keeper/gopax_market_maker_keeper.py 99 99 0% market_maker_keeper/hitbtc_market_maker_keeper.py 98 98 0% market_maker_keeper/idex_market_maker_keeper.py 193 193 0% market_maker_keeper/imtoken_pricing_server.py 51 51 0% market_maker_keeper/imtoken_utils.py 97 97 0% market_maker_keeper/kucoin_market_maker_keeper.py 108 108 0% market_maker_keeper/limit.py 46 0 100% market_maker_keeper/liquid_market_maker_keeper.py 97 97 0% market_maker_keeper/mpx_market_maker_keeper.py 137 137 0% market_maker_keeper/oasis_market_maker_cancel.py 38 22 42% market_maker_keeper/oasis_market_maker_keeper.py 133 94 29% market_maker_keeper/okex_market_maker_keeper.py 92 92 0% market_maker_keeper/order_book.py 219 188 14% market_maker_keeper/order_history_reporter.py 38 26 32% market_maker_keeper/paradex_market_maker_keeper.py 131 131 0% market_maker_keeper/price_feed.py 187 86 54%
```

market_maker_keeper/reloadable_config.py 67 3 96% market_maker_keeper/setzer.py 24 17 29% market_maker_keeper/spread_feed.py 7 5 29% market_maker_keeper/tethfinex_market_maker_keeper.py 149 149 0% market_maker_keeper/theocean_market_maker_keeper.py 129 129 0% market_maker_keeper/util.py 8 4 50% market_maker_keeper/zrx_market_maker_keeper.py 177 177 0% market_maker_keeper/zrxv2_market_maker_keeper.py 26 26 0%

TOTAL 3988 3232 19%

========== 97 passed in 4.04 seconds ==========

```

1. Understanding Bands Configuration

The Bands configuration file is directly related to how your Market Maker Keeper will work. As mentioned in the introduction, these Keepers continuously monitor and adjust their positions in the order book, maintaining open buy and sell orders in multiple bands at the same time. For eachbuy andsell band, the Keepers aim to have open orders for at least theminAmount . In both cases, they will ensure the price of open orders stay within the range from the current price. When running, Keepers place orders for the average amounts (avgAmount ) in each band by using useavgMargin to calculate the order price.

As long as the price of orders stays within the set band(s) (i.e. it is in between the range from the current price), the Keepers keep them open/running. If some orders leave the band, they either enter another adjacent band or fall outside all bands. In the case of the latter, they would get immediately canceled. In case of the former, Keepers can keep these orders open as long as their amount is within the ranges for the band they just entered. If it is above the maximum, some of the open orders will get canceled and potentially a new one will be created to bring the total amount back within the range. If it is below the minimum, a new order gets created for the remaining amount so that the total amount of orders in this band is equal toavgAmount . The same process will happen if the total amount of open orders in a band falls below theminAmount as a result of other market participants taking these orders. In this case, a new order gets created for the remaining amount so the total amount of orders in this band is equal toavgAmount . There are some Keepers that will constantly use gas to cancel orders (ex: OasisDEX, EtherDelta and 0x) and create new ones (OasisDEX) as the price changes. Gas usage can be limited by setting the margin and amount ranges wide enough but also by making sure that the bands are always adjacent to each other and that their amount ranges overlap.

File format

The bands configuration file consists of two main sections:

1. buyBands
2. sellBands
3.

Note: Each section is an array containing one object per each band.

TheminMargin andmaxMargin fields in each band object represent the margin (spread) range of that band. These ranges may not overlap for bands of the same type (buy orsell ), and should be adjacent to each other for better Keeper performance (where fewer orders will get canceled if the bands are adjacent to each other). TheavgMargin represents the margin (spread) of newly created orders within a band.

Glossary

1. minAmount
2.
    - the
3. minimum amount for keeper engagement for a band.
4. avgAmount
5.
    - the target amount for keeper engagement for a band.
6. maxAmount
7.
    - the
8. maximum amount for keeper engagement for a band.
9. dustCutoff
10.
    - a
11. field for the minimum amount of every single order created in each individual band (expressed in buy tokens for buy bands and in sell tokens for sell bands).
12.
    - Setting this to a non-zero value prevents Keepers from creating a lot of very tiny orders, which can cost a lot of
```

gas. For example, in the case of OasisDEX, it can result in an order that is too small getting rejected by other exchanges.

13. *
14.

Setting up your own Bands Configuration File:

1. Creating your bands.json file

To start, take the sample configuration file below and copy-paste it to a.json file within the root directory of yourmarket-maker-keeper folder. For ease of use, we sugges to name itbands.json . This bands file will get configured as a command-line argument when we start up the Market Maker Keeper.

A Sample bands.json file containing two Bands:

This example shows bands for the ETH-DAI pair, where ETH represents the base currency and DAI as the quote currency:

```

Copy { "_buyToken": "DAI", "buyBands": [ { "minMargin": 0.005, "avgMargin": 0.01, "maxMargin": 0.02, "minAmount": 20.0, "avgAmount": 30.0, "maxAmount": 40.0, "dustCutoff": 0.0 }, { "minMargin": 0.02, "avgMargin": 0.025, "maxMargin": 0.03, "minAmount": 40.0, "avgAmount": 60.0, "maxAmount": 80.0, "dustCutoff": 0.0 } ], "buyLimits": [],

"_sellToken": "ETH", "sellBands": [ { "minMargin": 0.005, "avgMargin": 0.01, "maxMargin": 0.02, "minAmount": 2.5, "avgAmount": 5.0, "maxAmount": 7.5, "dustCutoff": 0.0 }, { "minMargin": 0.02, "avgMargin": 0.025, "maxMargin": 0.05, "minAmount": 4.0, "avgAmount": 6.0, "maxAmount": 8.0, "dustCutoff": 0.0 } ], "sellLimits": [] }

```

Thisbands.json file should be adequate enough for you to paste and run it as is. Of course, you are free to configure it however you would like.

Note: Since this example will be focused on getting a Market Maker Keeper set up on Kovan, you need to make certain that you have enough Kovan ETH (K-Eth) to get your Keeper up and running. To receive Kovan ETH, join the following Gitter Channel:https://gitter.im/kovan-testnet/faucet and post your ETH address from your MetaMask account to the main chat. The Kovan faucet will then populate your wallet with the test funds (note that this could take a couple of minutes or a couple of hours as it is done manually by the channel's administrator).

1. Setting Amounts

You will need to set this up if you are going to be trading small amounts to start. This will need to be set up such that those amounts are sufficiently meaningful on the exchange you want to work with (based on the rules of the exchanges that you want to work. For example, their specificly set dust limits).

As mentioned above, there is one parameter in the configuration file (dustCutoff) that will be used to determine theminAmount of trade quantity. ThedustCutoff will need to be sethigher than/or at theminAmount trade quantity of the specific exchange. This is to make sure you don't create trades that are less than an exchanges' minimum trade requirements. Note that in your configuration file, you can also lower the required quantities to make it easier on yourself. Reducing the K-ETH amounts will be helpful in reducing wait times as you likely won't want to wait long periods to get enough K-ETH to run your Keeper).

Bands Example

Here, we will be going over some example interactions using thebands.json file described above. These examples assume that it is denominated in DAI and the price of 10 DAI is 1 ETH.

Using Band 1

- If we look at the first buy band, the initial buy order will be 30 DAI (avgAmount
-) with the price of ->price - (price * avgMargin)
- ->0.1 - (0.1 * 0.01)
- ->0.099 ETH per Dai
- .
- If thebuy
- order listed above (30 DAI @ 0.099 ETH) gets partially filled (15 DAI are purchased), then we will have (15 DAI remaining in the order). However, this amount is below the band'sminAmount
- (20 DAI), therefore, another whole order of 15 DAI will be placed on the exchange at the same price of 0.099 ETH.
- In addition to thebuy
- orders, when the Market Maker Keeper starts up, twosell
- orders will also be placed.

-

Using Band 2

- For ease of explanation, let's assume we are selling ETH priced at 100.00 DAI (5 ETH @ 101 DAI and 6 ETH @ 102.5 DAI).
- Now imagine a situation where the price of ETH suddenly drops to 97.50 DAI, pushing the bands down. In this scenario, the second band will start working and will become responsible for both of thesell
- orders, as they fit in between the second band'sminMargin
- andmaxMargin
- .
-

The Market Maker Keeper will now reset it's bands by performing the following:

1. Creating an order inBand 1
2. (5 ETH @ 98.475 DAI) usingavgMargin
3. andavgAmount
4. .
5. Cancelling the second order (5 ETH @ 102.5 DAI) (which is now inBand 2
6.) becuasemaxMargin
7. has been breached (whenprice + (price * maxMargin) = orderPrice
8. ->97.5 + (97.5 * 0.05)
9. -> 102.375 > 102.5).
10. Keep the first order (5 ETH @ 101 DAI), which is now inBand 2
11. because it is withinminMargin
12. andmaxMargin
13. ofBand 2.
14. Creating an order inBand 2
15. (1 ETH @ 99.937 DAI) usingavgMargin
16. andavgAmount
17. .
18.

This results in a total of 3 orders:

- Band 1
- -> (5 ETH @ 98.475 DAI)
- Band 2
- -> (5 ETH @ 101 DAI)
- Band 2
- -> (1 ETH @ 99.837 DAI)
-

6.Order Rate Limitation

Next, we will want to add theRate Limitation to the configurations file. This will make sure that we don't constantly churn out old orders as well as help manage gas consumption. We do this because we want theperiod and theamount to be set to a low amount when we start out. This is done because we don't want new users' Market Maker Keeper bots to be frantically trading all of the time. The goal here is that we want to set up our initial states such that it is only placing an order every 5 min or so (or whatever time amount you decide on).

There are two (optional) limit sections (buyLimits andsendLimits) that can be used for limiting themaximum rate of orders created by Market Maker Keepers. They both use the same format.

Example of order rate limits:

```

Copy "buyLimits": [ { "period": "1h", "amount": 50.0 }, { "period": "1d", "amount": 200.0 } ]

```

- Theperiod
- defines the amount of time that the limit should be applied over.
- Theamount
- is the maximum amount of orders that should be placed during the setperiod
- amount.
-

In the example above, theperiod s are set to 1-hour and 1-day and theamount s are set to 50.0 orders and 200.0 orders.

This means that over the course of 1-hour, only 50.0 orders can be placed and over the course of 1-day, only 200.0 orders can be placed. The amounts will be expressed either in terms of thebuy or thesell token, this will depend on the section. Note that the above snippet imposes a limit of50.0 buy token within each 60-minute window. Additionally, a maximum of200.0 buy tokens within each 24-hour window. Note that the supported time units ares ,m ,h ,d , andw .

7.Data Templating Language

The Jsonnet data templating language that can be used for the configuration file.

In the case of the data templating language, think of this like a pre-processing language for parsing the file. The whole purpose of the jsonnet is to set up a configuration file such that you can have it increment based on a price. Therefore, in addition to the price feed, you can also base percentages away from the market price. As you can see below, there is a hardcoded amount/price and then the amounts below it which are dependent on the price.

```

Copy { "_price": 10,

"_buyToken": "DAI", "buyBands": [ { "minMargin": 0.020, "avgMargin": 0.050, "maxMargin": 0.075, "minAmount": 0.05 * ._price, "avgAmount": 0.25 * ._price, "maxAmount": 0.35 * ._price, "dustCutoff": 0.1 } ],

"_sellToken": "ETH", "sellBands": [] }

```

Note: If you are working with a token that's price does not fluctuate wildly, you do not need to incorporate relative qualities for your amount. This is typically for people who want Market Maker Keepers open for months at a time and don't want to worry about having to change any of their configurations.

Another thing to note about these files is that the Market Maker Keeper reloads the configuration files automatically when it detects a change in them. This makes it easier as you don't have to constantly restart your Keeper bot when you change your band configurations. In short, this works by periodically taking a hash of the configuration file and comparing that hash with the current version. When it sees a change in that hash of the file, it will reload the configuration file and cancel orders as necessary to maintain the newly updated bands.

1. Price Feed Configuration

The price feed is one of the most important determining factors of success in a Market Maker Keeper. If you have the bands set up the way you want, the price feed will help make sure your bands are set at meaningful levels relative to the inside market. If you have wide bands and your strategy is to add liquidity to handle market imbalances, then the price feed is not as important. However, as you tighten up the spreads, the price feed is a crucial component to ensure that you are going to profit in the market.

Below, we list some of the existing public feeds. You can also use web sockets if you have your own price feed that you want to use. In short, it works by each Market Maker Keeper taking in a--price-feed command-line argument which then determines the price used for market-making.

As of today, these are the possible values of this argument that we list some of the existing public feeds:

- fixed:200
-
    - uses a fixed price, (1.56
- in this example). See below for a more in-depth example. When on mainnet, you typically won't use a fixed amount but it is an ideal example for this walkthrough as there aren't price feeds for Kovan.
- eth_dai
-
    - uses the price from the GDAX WebSocket ETH/USD price feed.
- eth_dai-setzer
-
    - uses the average of Kraken and Gemini ETH/USD prices.
- eth_dai-tub
-
    - uses the price feed fromTub
- (only works for keepers being able to access an Ethereum node).
- dai_eth
-
    - inverse of theeth_dai
- price feed.
- dai_eth-setzer
-
    - inverse of theeth_dai-setzer

- price feed.
- dai_eth-tub
-
    - inverse of theeth_dai-tub
- price feed.
- btc_dai
-
    - uses the price from the GDAX WebSocket BTC/USD price feed.
- dai_btc
-
    - inverse of thebtc_dai
- price feed.
- ws://...
- orwss://...
-
    - uses a price feed advertised over a WebSocket connection (custom protocol).
-

Additionally, we have a Uniswap price feed that can be used by Market Maker Keepers:https://github.com/makerdao/uniswap-price-feed .

Note: The--price-feed command line argument can also contain a comma-separated list of several different price feeds. In this case, if one of them becomes unavailable, the next one in the list will be used instead. All listed price feeds will be constantly running in the background, the second one and following ones ready to take over when the first one becomes unavailable.In the example below (in the Running Keepers section), you can see an example of how to use a fixed price amount.

1. Running Market Maker Keepers

Each Market Maker Keeper is a command-line tool which takes in generic command-line arguments (such as--config ,--price-feed ,--price-feed-expiry ,--debug , etc.) as well as some arguments which are specific to that particular Keeper (such as Ethereum node parameters, addresses, exchange API keys, etc.). All accepted command-line arguments are listed in the example section below. They can also be discovered by trying to start a Market Maker Keeper with the--help argument.

Example (Oasis Market Maker Keeper)

In order to runoasis-market-maker-keeper , you will need to go through the following process:

1. Firstly, you would deploy an Ethereum node (we recommend Parity).
2. Generate an account in it.
3. Permanently unlock that account.
4. Transfer some tokens to it.
5. Lastly, you can run the keeper (as seen below).
6.

The below file should be copy and pasted into a new file within the root directory of the repository (market-maker-keeper). This should be placed within the same folder where you put thebands.json file.

```

Copy

# !/bin/bash

bin/oasis-market-maker-keeper \ --rpc-host 127.0.0.1 \ --rpc-port \ --rpc-timeout 10 \ --eth-from [address of your generated Ethereum account] \ --eth-key {ACCOUNT_KEY} \ --tub-address 0x448a5065aebb8e423f0896e6c5d525c040f59af3 \ --oasis-address 0x14fbca95be7e99c15cc2996c6c9d841e54b79425 \ --price-feed fixed:200 \ --buy-token-address [address of the quote token, could be DAI] \ --sell-token-address [address of the base token, could be WETH] \ --config [path to the json bands configuration file, e.g bands.json] \ --smart-gas-price \ --min-eth-balance 0.001

```

- Make sure that you retrieve and paste the correct contract addresses when using the above snippet.
- --eth-key {ACCOUNT_KEY}
-
    - includes both the.json
- file (account.json) of your account and a.pass
- file (ex: account.pass) that contains your password in plaintext.
- If you do not have an account, you can useMyEtherWallet

- on Kovan and export the account details (by means of the Keystore file method). Make sure that you download the .json file to your local machine as this is what you will need to set up the account.
-

List of required Kovan Addresses for the above :

```

Copy V2_OASIS_SERVER1_ADDRESS= V2_OASIS_SERVER1_KEY="key_file=/home/ed/Projects/member-account.json,pass_file=/home/ed/Projects/member-account.pass"
TUB_ADDRESS=0xa71937147b55deb8a530c7229c442fd3f31b7db2 # tub-address
SAI_ADDRESS=0xc4375b7de8af5a38a93548eb8453a498222c4ff2 # buy-token-address
WETH_ADDRESS=0xd0a1e359811322d97991e03f863a0c30c2cf029c # sell-token-address
OASIS_ADDRESS_NEW=0x4a6bc4e803c62081ffebcc8d227b5a87a58f1f8f # oasis-address

```

General Notes:

- TheOASIS_SERVER1_KEY
- is simply your Kovan account private key (point this to your ETH accounts key file) and password file. If you do not have this, please set up a file with your password (in plain text).
- ETH From is the address location where the market-maker-keeper is going to get the tokens that it uses to participate and place orders.
-
    - Example:
-
    - Since Oasis is a decentralized exchange (dex), it is on-chain, so you need to provide all of the relevant addresses to the dex. Most DEX's are like this because when you are configuring with a dex you need to pass many addresses in, whereas, with a centralized exchange you are generally giving an API key, and username and password (see below for an example of how the process differs for centralized exchanges differ versus decentralized exchanges).
- *
- This Oasis example is currently forSingle Collateral DAI
- (SCD), where we configure the TUB_ADDRESS. However, as we move over toMulti-Collateral DAI (MCD)
- theTUB_ADDRESS
- will be changed to thePIP_ADDRESS
- for MCD.
-

Once completed, you can now run your Market Maker Keeper! Follow the next steps to get it running:

1. Open up your terminal
2. Runchmod +x
3. Run./
4. That's it, your Market Maker Keeper should now be running!
5.

Market Maker Keepers on Centralized Exchanges versus Decentralized Exchanges

In the situation where you want to use a centralized exchange vs. a decentralized exchange, the process differs a little:

1. You would need to have an existing account or create an account (on the exchange itself).
2. Get the set of API keys with trading permissions (will usually need to be generated as well).
3. Deposit tokens in your account on the exchange (as the keepers do not handle deposits and withdrawals themselves).
4. Run the Market Maker Keeper.
5.

1. Support

We are here to help! We welcome any questions about market making in the#keeper channel in the Maker Chat.

Previous Market Maker Keepers Next Cage Keeper Last updated4 years ago On this page *Guide Agenda * 1. Introduction * List of current exchanges that Market Maker Keeper Bots can be built for* 2. Prerequisite * 3. Getting Started (Installation) * 4. Testing * 5. Understanding Bands Configuration * File format * Setting up your own Bands Configuration File:* 6. Order Rate Limitation * 7. Data Templating Language * 8. Price Feed Configuration* 9. Running Market Maker Keepers * 10. Support

Export as PDF