QUANTCONNECT DOCUMENTATION

Learn to use QuantConnect and Explore Our Features

LEAN CLI

An easy to use, python shell wrapper on LEAN

LEAN CLI provides notebooks, backtesting, optimization and live trading with a simple to use API, deploying to the cloud or on premise.

Table of Content

Key Concepts

Key Concepts

Getting Started

Introduction

The Lean CLI is a cross-platform CLI which makes it easier to develop with the LEAN engine locally and in the cloud.

Prerequisites

The Lean CLI is distributed as a Python package, so it requires `pip` to be installed. See Installing pip to learn how to install pip on your operating system. Note that the Python distribution from the Microsoft Store is not supported, we recommend using the Anaconda distribution instead.

The commands which run the LEAN engine locally also depend on Docker being installed and running. See Install Docker to learn how to install Docker on your operating system.

To use the CLI, you must be a member in an organization on a paid tier.

Installation

Run `pip install lean` in a terminal to install the latest version of the CLI.

After installing the CLI, open a terminal in an empty directory and run `lean login` to log in to your QuantConnect account and then run `lean init` to create your first organization workspace. The `lean init` command downloads the latest configuration file and sample data from the QuantConnect/Lean repository. We recommend running all Lean CLI commands in your organization workspace directory.

```
$ lean init
Downloading latest sample data from the Lean repository...
The following objects have been created:
- lean.json contains the configuration used when running the LEAN engine locally
- data/ contains the data that is used when running the LEAN engine locally
...
```

If you are running Docker on Windows using the legacy Hyper-V backend instead of the new WSL 2 backend, you need to enable file sharing for your temporary directories and for your organization workspace. To do so, open your Docker settings, go to **Resources > File Sharing** and add **C: / Users / <username> / AppData / Local / Temp** and your organization workspace path to the list. Click **Apply & Restart** after making the required changes.

Basic Usage

The CLI contains a lot of commands to make working on LEAN algorithms easier and more productive. Below we list some of the most common tasks, see the pages in the sidebar and the API reference for a complete overview of the supported features.

Authenticate With the Cloud

Before using the CLI to perform tasks in the cloud it is required to log in with your QuantConnect API credentials. Run `lean login` to open an interactive wizard which asks you for your user Id and API token. Request these credentials and we'll email them to you.

```
$ lean login
Your user Id and API token are needed to make authenticated requests to the QuantConnect API
You can request these credentials on https://www.quantconnect.com/account
Both will be saved in /home/<username>/.lean/credentials
User id: <user id>
API token: <api token>
Successfully logged in
```

Pull Projects From the Cloud

Run `lean cloud pull` to pull your QuantConnect projects to your local drive. This command pulls all your cloud projects to your local drive while preserving your QuantConnect directory structure. If you have a lot of projects and only want to work locally on a few of them you can run this command with the `--project "<projectName>"` option, which makes the command pull a single project instead.

```
$ lean cloud pull
[1/3] Pulling 'Creative Red Mule'
Successfully pulled 'Creative Red Mule/main.py'
[2/3] Pulling 'Determined Yellow-Green Duck'
Successfully pulled 'Determined Yellow-Green Duck/main.py'
Successfully pulled 'Determined Yellow-Green Duck/research.ipynb'
[3/3] Pulling 'Halloween Strategy'
Successfully pulled 'Halloween Strategy/benchmark.py'
Successfully pulled 'Halloween Strategy/main.py'
Successfully pulled 'Halloween Strategy/research.ipynb'
```

Source Data

Run `lean data generate --start 20180101 --symbol-count 100` to generate realistic fake market data to test with. You can also choose to download data from QuantConnect Datasets or convert your own data into LEAN-compatible data.

```
$ lean data generate --start 20180101 --symbol-count 100
Begin data generation of 100 randomly generated Equity assets...
...
```

Run a Local Backtest

Run `lean backtest "<projectName>"` to run a local backtest for the specified project. This command runs a backtest in a Docker container containing the same packages as the ones used on QuantConnect.com, but with your own data.

```
$ lean backtest "Project Name"
20210308 23:58:35.354 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode: DEBUG
(64bit)
20210308 23:58:35.360 TRACE:: Engine.Main(): Started 11:58 PM
...
```

Push Local Changes to the Cloud

Run `lean cloud push` to push local changes to the QuantConnect. This command pushes all your local projects to the cloud and creates new cloud projects when necessary. If you only want to push a single project you can run this command with the `--project "<projectName>"` option.

```
$ lean cloud push
[1/3] Pushing 'Creative Red Mule'
Successfully updated cloud file 'Creative Red Mule/main.py'
[2/3] Pushing 'Determined Yellow-Green Duck'
[3/3] Pushing 'Halloween Strategy'
```

Run a Cloud Backtest

Run `lean cloud backtest "<projectName>"` to run a cloud backtest for the specified project. By default, a summary of the results and a link to the full results are shown in the terminal. Running this command with the `--open` flag automatically opens the full results in the browser once the backtest is finished. Additionally, you can run this command with the `--push` flag to push all local changes to the project to the cloud before running the backtest.

```
$ lean cloud backtest "Project Name"
Started compiling project 'Project Name'
Detected parameters (2):
- main.py:19 :: 1 Order Event parameter detected near "SetHoldings(self.spy, 1)".
- main.py:21 :: 1 Order Event parameter detected near "SetHoldings(self.spy, 0)".
Build Request Successful for Project ID: 4882833, with CompileID: eaf9b677c91cfadd0a9032eb95918beb-
c3b92b55d26a6d610e9b792ce561a687, Lean Version: 2.5.0.0.11058
Successfully compiled project 'Project Name'
Started backtest named 'Swimming Orange Lemur' for project 'Project Name'
...
```

LEAN vs LEAN CLI

LEAN is the open-source algorithmic trading engine. LEAN CLI is the way we recommend you run LEAN on your local machine. The LEAN CLI can do almost everything that LEAN can do. There are just some programs in the ToolBox that the LEAN CLI can't currently run. The `lean data generate` is a wrapper for the random data generator in the ToolBox. However, if you need any of the other programs in the ToolBox, you'll have to run LEAN manually and move the downloaded/parsed data to the CLI's **data** directory.

Key Concepts

Troubleshooting

Introduction

You might occassionally receive an error indicating that something went wrong. We try to provide accurate error descriptions in the CLI, but in some cases, those might not be enough. This page lists common errors with their possible cause and a way to fix them. In case you still need help after that this page also contains instructions on how to report issues to our engineers in a way that makes it easy for us to help you with your issue.

Common Errors

The following table describes errors you may see when using the CLI:

Error Message	Possible Cause and Fix
No such command '<name>' No such option: <option>	The command you tried to run does not exist or it doesn't support the option you provided. If the documentation says it is available you are probably using an outdated version of the CLI. Run `pip install --upgrade lean` to update to the latest version.
Invalid credentials, please log in using `lean login`	You are trying to use a command which communicates with the QuantConnect API and you haven't authenticated yourself yet. Run `lean login` to log in with your API credentials.
Please make sure Docker is installed and running	You are trying to use a command which needs to run the LEAN engine locally, which always happens in a Docker container. Make sure Docker is running if you installed it already, or visit Install Docker if you haven't.
This command requires a Lean configuration file, run `lean init` in an empty directory to create one, or specify the file to use with --lean-config	The command you are trying to run requires a Lean configuration file. The CLI automatically tries to find this file by recursively looking in the current directory and all of the parent directories for a **lean.json** file. This error is shown if no such file can be found. It can be fixed by running the command in your organization workspace directory (which generates the **lean.json** file), or by specifying the path to the **lean.json** file with the `--lean-config` option.
We couldn't find you account in the given organization, ORG: <32-char-hash>	The organization Id found in the **lean.json** is incorrect. You need to re-install Lean CLI running `lean init` in an empty directory.

Error Message	Possible Cause and Fix
Invalid value for 'PROJECT': Path '<path>' does not exist.	You are trying to run an action on a project but specified an invalid project path. Make sure you are running the command from your organization workspace directory and make sure `./<path>` points to an existing directory.
'<path>' is not a valid path	You provided a path that is not valid for your operating system. This error is most likely to appear on Windows, where the following characters are not allowed in path components: `<` , `>` , `:` , `"` , `|` , `?` , and `*` . On top of those characters the following names are not allowed (case-insensitive): `CON` , `PRN` , `AUX` , `NUL` , `COM1` until `COM9` , and `LPT1` until `LPT9` . Last but not least, path components cannot start or end with a space or end with a period on Windows.
invalid mount config for type "bind": bind source path does not exist: **/ var / folders / <path> / config.json** Mounts denied: The path **/ Users / <path> / data** is not shared from the host and is not known to Docker	Your Mac's Docker file sharing settings do not permit binding one or more directories that we need to share with the container. Go to Docker's **Settings > Resources > File Sharing** and add **/ private / var / folders** and either **/ Users** to share your entire **/ Users** directory, or **/ Users / <path>** where **<path>** is the path to your QuantConnect directory, which should have a **data** child directory, and child directories for your individual projects.
Docker wants access to <path>	You are running Docker on Windows using the legacy Hyper-V backend and haven't configured file sharing correctly. You need to enable file sharing for your temporary directories and for your organization workspace directory . To do so, open your Docker settings, go to **Resources > File Sharing** and add **C: / Users / <username> / AppData / Local / Temp** and your organization workspace directory to the list. Click **Apply & Restart** after making the required changes.

Report Issues

If with the information on this page and the error message shown by the CLI you're still unable to solve your issues you are welcome to contact our engineers by opening an issue in the QuantConnect/lean-cli repository on GitHub. Before doing so, please run the command that's giving issues with the `--verbose` flag and copy and paste the output into the issue (feel free to mask sensitive information). The `--verbose` flag enables debug messages to be printed, which makes it easier for us to help you.

Installation

Installation

Installing pip

Introduction

The Lean CLI is distributed as a Python package, so it requires `pip` to be installed. Because `pip` is distributed as a part of Python, you must install Python before you can install the CLI.

This page contains installation instructions for Anaconda , which is a Python distribution containing a lot of packages that are also available when running the LEAN engine. Having these packages installed locally makes it possible for your editor to provide autocomplete for them.

The Python distribution from the Microsoft Store is not supported. Docker doesn't support Python 3.10 , so you must have 3.9.x or lower.

Install on Windows

Follow these steps to install Anaconda on your computer:

1. Download the latest 64-bit Graphical Installer for Windows.
2. Run the installer and click **Next** .
3. Read the licensing terms and click the **I Agree** check box.
4. Select the **Just Me** check box and click **Next** .
5. Select the destination folder to install Anaconda in (make sure this path does not contain spaces) and click **Next** .
6. In the Advanced Options, enable the **Add Anaconda3 to my PATH environment variable** and **Register Anaconda3 as my default Python 3.x** check boxes and then click **Install** .
7. After the installation has completed, restart your computer to make sure changes are propagated.
8. Open a terminal and run:

```
$ conda update --all
```

Install on macOS (Intel)

Follow these steps to install Anaconda on your Mac with an Intel chip:

1. Download the latest 64-bit Graphical Installer for macOS.

2. Click **Continue** on the Introduction, Read Me, and License pages.
3. Agree to the license by clicking **Agree** .
4. Click **Install** on the Installation Type page to install to start the installation.
5. After the installation has finished, click **Continue** on the PyCharm IDE page and **Close** on the Summary page to close the installer.

Install on macOS (Apple)

Anaconda does not support Apple chips. Follow these steps to install Miniforge, a minimal version of Anaconda, on your Mac with an Apple chip:

1. Download Miniforge3-MacOSX-arm64.sh .
2. Open a terminal in the directory containing the installer.
3. Run `bash Miniforge3-MacOSX-arm64.sh` to start the installer.
4. Press **Enter** to view the license terms, press **Q** to exit the license terms, and enter **Yes** to accept the license terms.
5. Specify where Miniforge should be installed or accept the default.
6. Enter **Yes** when the installer prompts whether you want the installer to initialize Miniforge.
7. Re-open the terminal window after the installation has finished.

Install on Linux

Follow these steps to install Anaconda on your computer:

1. Download the latest 64-bit (x86) Installer for Linux.
2. Open a terminal in the directory containing the installer.
3. Run `bash <fileName>` where `<fileName>` is the name of the installer.
4. Press **Enter** to view the license terms, press **Q** to exit the license terms, and enter **Yes** to accept the license terms.
5. Specify where Anaconda should be installed or accept the default.
6. Enter **Yes** when the installer prompts whether you want the installer to initialize Anaconda.
7. Re-open the terminal window after the installation has finished.

Installation

Installing Lean CLI

Introduction

The Lean CLI is distributed as a Python package, so it requires `pip` to be installed. To learn how to install `pip` on your operating system, see Installing pip .

The commands which run the LEAN engine locally also depend on Docker being installed and running.

Install Docker

If you run the LEAN engine locally with the CLI, LEAN executes in a Docker container. These Docker containers contain a minimal Linux-based operating system, the LEAN engine, and all the packages available to you on QuantConnect.com. It is therefore required to install Docker if you plan on using the CLI to run the LEAN engine locally.

Install on Windows

Windows systems must meet the following requirements to install Docker:

- A 64-bit processor
- 4 GB RAM or more
- Windows 10, version 1903 or higher (released May 2019)
- Hardware virtualization enabled in the BIOS
- 20 GB hard drive or more

Follow these steps to install Docker:

1. Follow the Install Docker Desktop on Windows tutorial in the Docker documentation.

   As you install docker, enable WSL 2 features.

2. Restart your computer.
3. If Docker prompts you that the WSL 2 installation is incomplete, follow the instructions in the dialog shown by Docker to finish the WSL 2 installation.
4. Open PowerShell with adminstrator privledges and run:

   ```
   $ wsl --update
   ```

By default, Docker doesn't automatically start when your computer starts. So, when you run the LEAN engine with the CLI for the first time after starting your computer, you must manually start Docker. To automatically start Docker, open the Docker Desktop application, click **Settings > General** , and then enable the **Start Docker**

Desktop when you log in check box.

Install on macOS

Mac systems must meet the following requirements to install Docker:

- Mac hardware from 2010 or newer with an Intel processor
- macOS 10.14 or newer (Mojave, Catalina, or Big Sur)
- 4 GB RAM or more
- 20 GB hard drive or more

To install Docker, see Install Docker Desktop on Mac in the Docker documentation.

Install on Linux

To install, see Install Docker Desktop on Linux in the Docker documentation.

Install LEAN CLI

Before you install the LEAN CLI, check if it's already installed.

```
$ lean --version
```

Follow these steps to install the LEAN CLI:

1. Install pip .
2. Install Docker .
3. If you are on a Windows machine, open PowerShell as the adminstrator for the following commands.
4. Install the LEAN CLI with pip.

   ```
   $ pip install --upgrade lean
   ```

Next Steps

Log in to your account and then set up your first organization workspace .

Stay Up To Date

We regularly update the CLI to add new features and to fix issues. Therefore, it's important to keep both the CLI and the Docker images that the CLI uses up-to-date.

Keep the CLI Up-To-Date

To update the CLI to the latest version, run `pip install --upgrade lean` . The CLI automatically performs a version check once a day and warns you in case you are running an outdated version.

Keep the Docker Images Up-To-Date

Various commands like `lean backtest`, `lean optimize` and `lean research` run the LEAN engine in a Docker container to ensure all the required dependencies are available. When you run these commands for the first time the Docker image containing LEAN and its dependencies is downloaded and stored. Run these commands with the `--update` flag to update the images they use. Additionally, these commands automatically perform a version check once a week and update the image they use when you are using an outdated Docker image.

Uninstall

To uninstall the Lean CLI, run `pip uninstall lean`. To perform a full uninstallation, you must also delete configuration files that the CLI generates, which you can find in the following directories:

- The **.lean** directory in your user's home directory
- Your organization workspaces

Initialization

Initialization

Authentication

Introduction

Some of the Lean CLI commands need to communicate with the QuantConnect API. If you use any commands which interact with the cloud, you must log in using your QuantConnect account so the CLI can send authenticated API requests.

Log In

Follow these steps to log in to your QuantConnect account with the CLI:

1. Request your user-id and API token .
2. Retrieve your user-id and API token from the email you registered on QuantConnect.
3. Run `lean login` to log in to the CLI, and enter your user-id and token when prompted. This command opens an interactive wizard asking you for your user-id and API token.

```
$ lean login
Your user Id and API token are needed to make authenticated requests to the QuantConnect API
You can request these credentials on https://www.quantconnect.com/account
Both will be saved in C:\Users\john\.lean\credentials
User id: 123456
API token: ******************
Successfully logged in
```

Log Out

Run `lean logout` to log out. This command removes the global configuration file containing your user Id and API token.

Check Account Status

Run `lean whoami` to see the name and email address of the user who is currently logged in.

Initialization

Organization Workspaces

Introduction

After you install the CLI and log in to your account , you need to create an organization workspace. Your organization workspace connects a directory on your local machine with one of your organizations in QuantConnect Cloud. Your organization workspace includes the basic files and folders necessary to use LEAN, including a local data directory and a LEAN configuration file.

We recommend running all Lean CLI commands in your root of your organization workspace directory. Doing so ensures the directory structure is always kept consistent when synchronizing projects between the cloud and your local drive. It also makes it possible for the CLI to automatically find the Lean configuration file when you run the LEAN engine on your local machine.

Create Workspaces

To create an organization workspace for one of your organizations, open a terminal in an empty directory, run `lean init` and then select an organization to link with the organization workspace. This command scaffolds a standard directory structure containing a **data** directory and a Lean configuration file , both of which are required to run the LEAN engine locally.

If you are running Docker on Windows using the legacy Hyper-V backend instead of the new WSL 2 backend, you need to enable file sharing for your temporary directories and for your organization workspace. To do so, open your Docker settings, go to **Resources > File Sharing** and add **C: / Users / <username> / AppData / Local / Temp** and your organization workspace path to the list. Click **Apply & Restart** after making the required changes.

To set the default language of new projects in a new organization workspace, run `lean init --language <value>` where the `<value>` is `python` or `csharp` .

Directory Structure

The `lean init` commands creates the following structure:

```
.
├── data/
│   ├── alternative/
│   ├── crypto/
│   ├── equity/
│   ├── ...
│   ├── market-hours/
│   ├── option/
│   ├── symbol-properties/
│   └── readme.md
└── lean.json
```

These files contain the following content:

File/Directory	Description
data /	This directory contains the local data that LEAN uses to run locally. This directory is comes with sample data from the QuantConnect/Lean repository . As you download additional data from the dataset market, it's stored in this directory. Each organization workspace has its own **data** directory because each organization has its own data licenses.
lean.json	This file contains the Lean configuration that is used when running the LEAN engine locally.

When you create new projects or pull existing projects from the cloud, your organization workspace stores the project files.

Initialization

Configuration

Introduction

The CLI stores its persistent configuration in various places depending on the context of the configuration. We make the distinction between global configuration, Lean configuration, and project configuration, all of which store a specific subset of configurable properties.

Global Configuration

The global CLI configuration directory stores the CLI defaults and API credentials. The exact location of the directory depends on your operating system. The following table shows the path of each operating system:

Operating System	Path
Windows	**C: \ Users \ <username> \ .lean**
macOS	**/ Users / <username> / .lean**
Linux	**/ home / <username> / .lean**

The global CLI configuration directory contains three files and one directory. The following table describes the files and directory:

Name	Description
credentials	This file contains the API credentials given via `lean login` .
config	This file contains the CLI defaults, like the default project language used when running `lean project-create` .
cache	This file contains an internal cache that the CLI uses whenever it needs to persistently store data. One of its uses is to store the last time an update check has run to make sure we don't check for updates too often.
ssh	This directory contains the SSH keys that are needed to debug over SSH when debugging with Rider.

The global configuration files are always updated via the CLI and should not be updated or removed manually, unless when you are uninstalling the CLI.

Lean Configuration

The Lean configuration contains settings for locally running the LEAN engine. This configuration is created in the **lean.json** file when you run `lean init` in an empty directory. The configuration is stored as JSON, with support

for both single-line and multiline comments.

The Lean configuration file is based on the **Launcher / config.json** file from the Lean GitHub repository. When you run `lean init`, the latest version of this file is downloaded and stored in your organization workspace. Before the file is stored, some properties are automatically removed because the CLI automatically sets them.

The CLI commands can update most of the values of the **lean.json** file. The following table shows the configuration settings that you need to manually adjust in the **lean.json** file if you want to change their values:

Name	Description	Default
`show-missing-data-logs`	Log missing data files. This is useful for debugging.	true
`maximum-warmup-history-days-look-back`	The maximum number of days of data the history provider will provide during warm-up in live trading. The history provider expects older data to be on disk.	5
`maximum-data-points-per-chart-series`	The maximum number of data points you can add to a chart series in backtests.	4000

Project Configuration

For information about project configuration, see Projects > Configuration .

Datasets

Datasets

Format and Storage

Introduction

LEAN strives to use an open, human-readable format, so all data is stored in flat files (formatted as CSV or JSON). The data is compressed on disk using zip

Default Location

When you create an organization workspace in an empty directory, the CLI downloads the latest data directory from the LEAN repository . This directory contains a standard directory structure from which the LEAN engine reads. Once downloaded, the **data** directory tree looks like this:

```
data
├── alternative/
├── cfd/
├── crypto/
├── equity/
├── forex/
├── future/
├── futureoption/
├── index/
├── indexoption/
├── market-hours/
├── option/
├── symbol-properties/
└── readme.md
```

By default, the **data** directory contains a small amount of sample data for all asset types to demonstrate how data files must be formatted. Additionally, the **data** directory itself and most of its subdirectories contain **readme.md** files containing more documentation on the format of the data files of each asset type.

Change Location

You can configure the data directory to use in the `data-folder` property in your Lean configuration file . The path this property is set to is used as the **data** directory by all commands that run the LEAN engine locally. By default, this property points to the **data** directory inside your organization workspace . If this property is set to a relative path, it is resolved relative to the Lean configuration file's parent directory.

The **data** directory is the only local directory that is mounted into all Docker containers ran by the CLI, so it must contain all the local files you want to read from your algorithms. You can get the path to this directory in your

algorithm using the `Globals.DataFolder` variable.

Other Data Sources

If you already have data of your own you can convert it to a LEAN-compatible format yourself. In that case, we recommend that you read the **readme.md** files generated by the `lean init` command in the **data** directory, as these files contain up-to-date documentation on the expected format of the data files.

For development purposes, it is also possible to generate data using the CLI. This generator uses a Brownian motion model to generate realistic market data, which might be helpful when you're testing strategies locally but don't have access to real market data.

Datasets

Downloading Data

To locally run the LEAN engine, you need local data. We recommend you source local data from our Dataset Market , so you can use the same data that is available to your algorithm when you run it in the cloud. There are two methods of downloading data. You can download smaller discrete datasets at a low cost or download complete collections in bulk to avoid selection bias.

Download By Ticker

Low cost option for individual tickers

Download in Bulk

All tickers to avoid selection bias

See Also

Datasets

Downloading Data

Download By Ticker

Downloading data by the ticker is the ideal, low-cost option to acquiring local trading data if you don't need an entire universe. This technique is for smaller, discrete downloads.

Key Concepts

Costs

See Also

Download in Bulk

Download By Ticker

Key Concepts

Introduction

Downloading data by the ticker is the ideal, low-cost option to acquiring local trading data if you don't need an entire universe. This technique is for smaller, discrete downloads. You can download our ticker data through the CLI or LEAN in exchange for some QuantConnect Credit (QCC). Before the CLI or LEAN download new files, they check if your local machine already stores the files.

Using the CLI

You can download datasets with the CLI using the non-interactive mode or the interactive mode.

Non-Interactive Mode

Follow these steps to download datasets with the non-interactive mode of the CLI:

1. Open the listing page of the dataset that you want to download.
2. Click the **CLI** tab.

   If there is no **CLI** tab, you can't download the dataset.

3. Fill in the Command Line Generator form.
4. Select **Windows** or **Unix** .
5. Copy the CLI command that the form displays.
6. Open a terminal in your organization workspace and then run the command.
7. If you haven't already agreed to the CLI API Access and Data Agreement , in the browser window that opens, read the document and click **I Accept The Data Agreement** .

The CLI displays a summary of your purchase and the download progress.

```
Data that will be purchased and downloaded:
```

Dataset	Vendor	Details	File count	Price
Binance Crypto Price Data	CoinAPI	Data type: Trade Ticker: BTCBUSD Resolution: Second Start date: 2022-05-05 End date: 2022-06-05	32	800 QCC

```
Total price: 800 QCC
Organization balance: 1,000 QCC

Data Terms of Use has been signed previously.
Find full agreement at: https://www.quantconnect.com/terms/data/?organization=<organizationId>
================================================================================
CLI API Access Agreement: On 2022-04-12 22:34:26 You Agreed:
- Display or distribution of data obtained through CLI API Access is not permitted.
- Data and Third Party Data obtained via CLI API Access can only be used for individual or internal
employee's use.
- Data is provided in LEAN format can not be manipulated for transmission or use in other
applications.
- QuantConnect is not liable for the quality of data received and is not responsible for trading
losses.
================================================================================

 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% (32/32)
```

Interactive Mode

Follow these steps to download datasets with the interactive mode of the CLI:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in one of your organization workspace directories .
3. Run `lean data download` to start an interactive downloading wizard.
4. Go through the interactive wizard to configure the data you want to download.
5. After configuring the data you want to download, enter **N** when asked whether you want to download more data.

```
$ lean data download
Selected data:
...
Total price: 10 QCC
Organization balance: 10,400 QCC
Do you want to download more data? [y/N]: N
```

6. In the browser window that opens, read the CLI API Access and Data Agreement and click **I Accept The Data Agreement** .
7. Go back to the terminal and confirm the purchase to start downloading.

```
$ lean data download
You will be charged 10 QCC from your organization's QCC balance
After downloading all files your organization will have 10,400 QCC left
Continue? [y/N]: y
[1/1] Downloading equity/usa/daily/spy.zip (10 QCC)
```

Many of the datasets depend on the US Equity Security Master dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes. To check if a dataset depends the US Equity Security Master, see the listing in the Dataset Market . If you try to download a dataset through the CLI that depends on the US Equity Security Master and you don't have an active subscription to it, you'll get an error.

For example, follow these steps to download US Equity data from the Dataset Market:

1. Open a terminal in one of your organization workspace directories .
2. Run `lean data download` to start an interactive downloading wizard and then enter the dataset category number.

```
$ lean data download
Select a category:
1) Commerce Data (3 datasets)
2) Consumer Data (2 datasets)
3) Energy Data (2 datasets)
4) Environmental Data (1 dataset)
5) Financial Market Data (30 datasets)
6) Industry Data (1 dataset)
7) Legal Data (2 datasets)
8) News and Events (4 datasets)
9) Political Data (2 datasets)
10) Transport and Logistics Data (1 dataset)
11) Web Data (9 datasets)
Enter an option: 5
```

3. Enter the dataset number.

```
$ lean data download
Select a dataset:
1) Binance Crypto Future Margin Rate Data
2) Binance Crypto Future Price Data
3) Binance Crypto Price Data
4) Binance US Crypto Price Data
5) Bitcoin Metadata
6) Bitfinex Crypto Price Data
7) Brain Language Metrics on Company Filings
8) Brain ML Stock Ranking
9) CFD Data
10) Coinbase Crypto Price Data
11) Composite Factor Bundle
12) Cross Asset Model
13) FOREX Data
14) Insider Trading
15) Kraken Crypto Price Data
16) NFT Sales
17) Tactical
18) Template (Do not Edit)
19) US Equity Coarse Universe
20) US Congress Trading
21) US ETF Constituents
22) US Equities
23) US Equity Options
24) US Equity Security Master
25) US Federal Reserve (FRED)
26) US Futures Security Master
27) US SEC Filings
28) US Treasury Yield Curve
29) VIX Central Contango
30) VIX Daily Price
Enter an option: 22
```

If you don't have an active subscription to the US Equity Security Master, you'll get the following error message:

Your organization needs to have an active Security Master subscription to download data from the 'US Equities' dataset
You can add the subscription at https://www.quantconnect.com/datasets/quantconnect-security-master/pricing

4. Enter the data type.

```
$ lean data download
Data type:
1) Trade
2) Quote
3) Bulk
Enter an option: 1
```

5. Enter the ticker(s).

```
$ lean data download
Ticker(s) (comma-separated): SPY
```

6. Enter the resolution.

```
$ lean data download
Resolution:
1) Tick
2) Second
3) Minute
4) Hour
5) Daily
Enter an option: 3
```

7. If you selected tick, second, or minute resolution in the previous step, enter the start and end date.

```
$ lean data download
Start date (yyyyMMdd): 20230101
End date (yyyyMMdd): 20230105
```

8. Review your selected data and enter whether you would like to download more data.

```
$ lean data download
Selected data:
┌──────────────┬──────────┬──────────────────────────┬──────────────┬──────────┐
│ Dataset      │ Vendor   │ Details                  │ File count │ Price  │
├──────────────┼──────────┼──────────────────────────┼──────────────┼──────────┤
│ US Equities │ AlgoSeek │ Data type: Trade       │ 3          │ 15 QCC │
│              │          │ Ticker: SPY            │              │        │
│              │          │ Resolution: Minute     │              │        │
│              │          │ Start date: 2023-01-01 │              │        │
│              │          │ End date: 2023-01-05   │              │        │
└──────────────┴──────────┴──────────────────────────┴──────────────┴──────────┘
Total price: 15 QCC
Organization balance: 10,000 QCC
Do you want to download more data? [y/N]: n
```

9. If you haven't already agreed to the CLI API Access and Data Agreement , in the browser window that opens, read the document and click **I Accept The Data Agreement** .
10. Confirm your data purchase.

```
$ lean data download
Data Terms of Use has been signed previously.
Find full agreement at: https://www.quantconnect.com/terms/data/?organization=<organizationId>
================================================================================
CLI API Access Agreement: On 2022-04-12 22:34:26 You Agreed:
- Display or distribution of data obtained through CLI API Access is not permitted.
- Data and Third Party Data obtained via CLI API Access can only be used for individual or
internal employee's use.
- Data is provided in LEAN format can not be manipulated for transmission or use in other
applications.
- QuantConnect is not liable for the quality of data received and is not responsible for trading
losses.
================================================================================

You will be charged 15 QCC from your organization's QCC balance
After downloading all files your organization will have 9,985 QCC left
Continue? [y/N]: y
```

After you confirm your data purchsae, the CLI downloads the data and prints its progress.

```
$ lean data download
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% (3/3)
```

Using Lean

For more information about using LEAN to download datasets, see Deployment .

Supported Datasets

To view all of the supported datasets, see the Dataset Market .

Download By Ticker

Costs

Introduction

This page describes how to calculate the approximate cost of downloading local data for algorithms of each asset class. The prices reflect the data prices at the time of writing. To view the current prices of each dataset, open a dataset listing in the Dataset Market and then click the **Pricing** tab. To download the data, use the lean data download command or the ApiDataProvider .

US Equity

US Equity algorithms require the US Equity Security Master and some data from the US Equities dataset. The following table shows the cost of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

The US Equities dataset is available is several resolutions. The resolution you need depends on the US Equity subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Tick	One file per security per trading day per data format. Quote and trade data are separate files.	6 QCC = $0.06 USD
Second	One file per security per trading day per data format. Quote and trade data are separate files.	5 QCC = $0.05 USD
Minute	One file per security per trading day per data format. Quote and trade data are separate files.	5 QCC = $0.05 USD
Hour	One file per security.	300 QCC = $3 USD
Daily	One file per security.	100 QCC = $1 USD

If you add universes to your algorithm, the following table shows the additional datasets you need:

Universe Type	Required Dataset	File Format	Cost per file
Coarse or Dollar Volume	US Equity Coarse Universe	One file per day.	5 QCC = $0.05 USD
ETF Constituents	US ETF Constituents	One file per ETF per day.	50 QCC = $0.50 USD

For example, the following algorithm creates a dollar volume universe with 100 securities and then subscribes to minute resolution data for each US Equity in the universe:

```
class USEquityDataAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        self.AddUniverse(self.Universe.DollarVolume.Top(100))
```

The following table shows the data cost of the preceding algorithm on the Quant Researcher tier:

Dataset	Package	Initial Cost	Ongoing Cost
US Equity Security Master	Download On Premise	$600 USD	$600 USD/year
US Equity Coarse Universe	On Premise Download	252 trading days => 252 files 252 files @ 5 QCC/file => 252 * 5 QCC = 12,600 QCC = $126 USD	1 trading day => 1 file 1 file/day @ 5 QCC/file => 5 QCC/day = $0.05 USD/day
US Equity	Minute Download	100 securities over 252 trading days with 2 data formats => 100 * 252 * 2 files = 50,400 files 50,400 files @ 5 QCC/file => 50,400 * 5 QCC = 252,000 QCC = $2,520 USD	100 securities with 2 data formats => 100 * 2 files/day = 200 files/day 200 files/day @ 5 QCC/file => 200 * 5 QCC/day = 1,000 QCC/day = $10 USD/day

The preceding table assumes you download trade and quote data, but you can run backtests with only trade data.

Equity Options

Equity Option algorithms require the US Equity Security Master , some data from the US Equity Options dataset, and data for the underlying US Equity universes and assets. The following table shows the cost of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

The US Equity Options dataset is available is several resolutions. The resolution you need depends on the US Equity Option subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Minute	One file per Option per trading day per data format. Quote, trade, and open interest data are separate files.	15 QCC = $0.15 USD
Hour	One file per Option per year per data format. Trade and open interest data are separate files.	900 QCC = $9 USD
Daily	One file per Option per year. Trade and open interest data are separate files.	300 QCC = $3 USD

For example, the following algorithm subscribes to minute resolution data for an Equity Option and its underlying asset:

```py
class USEquityOptionsDataAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        underlying = self.AddEquity("GOOG").Symbol
        self.AddOption(underlying)
```

The following table shows the data cost of the preceding algorithm on the Quant Researcher tier:

Dataset	Package	Initial Cost	Ongoing Cost
US Equity Security Master	Download On Premise	$600 USD	$600 USD/year
US Equity	Minute Download	1 security over 252 trading days with 2 data formats => 1 * 252 * 2 files = 504 files 504 files @ 5 QCC/file => 504 * 5 QCC = 2,520 QCC = $25.20 USD	1 security with 2 data formats => 2 files/day 2 files/day @ 5 QCC/file => 2 * 5 QCC/day = 10 QCC/day = $0.10 USD/day
US Equity Options	Minute Download	1 Option over 252 trading days with 3 data formats => 1 * 252 * 3 files = 756 files 756 files @ 15 QCC/file => 756 * 15 QCC = 11,360 QCC = $113.60 USD	1 Option with 3 data formats => 3 files/day 3 files/day @ 15 QCC/file => 3 * 15 QCC/day = 45 QCC/day = $0.45 USD/day

The preceding table assumes you download trade, quote, and open interest data. However, you can run backtests with only trade data.

Crypto

Crypto algorithms require at least one of the CoinAPI datasets . The CoinAPI datasets are available is several resolutions. The resolution you need depends on the Crypto subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Tick	One file per security per trading day per brokerage per data format. Quote and trade data are separate files.	100 QCC = $1 USD
Second	One file per security per trading day per brokerage per data format. Quote and trade data are separate files.	25 QCC = $0.25 USD
Minute	One file per security per trading day per brokerage per data format. Quote and trade data are separate files.	5 QCC = $0.05 USD
Hour	One file per security per brokerage.	400 QCC = $4 USD
Daily	One file per security per brokerage.	100 QCC = $1 USD

If you add universes to your algorithm, you also need `CryptoCoarseFundamental` data. The file format of `CryptoCoarseFundamental` data is one file per day per brokerage and each file costs 100 QCC = $1 USD.

For example, the following algorithm creates a universe of 100 Cryptocurrencies and then subscribes to minute resolution data for each one in the universe:

```
class CryptoDataAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        self.AddUniverse(CryptoCoarseFundamentalUniverse(Market.GDAX, self.UniverseSettings,
self.universe_filter))

    def universe_filter(self, crypto_coarse: List[CryptoCoarseFundamental]) -> List[Symbol]:
        sorted_by_dollar_volume = sorted(crypto_coarse, key=lambda cf: cf.VolumeInUsd, reverse=True)
        return [cf.Symbol for cf in sorted_by_dollar_volume[:100]]
```

The following table shows the data cost of the preceding algorithm:

Dataset	Package	Initial Cost	Ongoing Cost
Coinbase Crypto Price Data	Universe Download	365 days => 365 files 365 files @ 100 QCC/file => 365 * 100 QCC = 36,500 QCC = $365 USD	1 file per day @ 100 QCC/file => 100 QCC/day = $1 USD/day
Coinbase Crypto Price Data	Minute Download	100 securities over 365 trading days with 2 data formats => 1 * 100 * 365 * 2 files = 73,000 files 73,000 files @ 5 QCC/file => 73,000 * 5 QCC = 365,000 QCC = $3,650 USD	100 securities with 2 data formats => 100 * 2 files/day = 200 files/day 200 files/day @ 5 QCC/file => 200 * 5 QCC/day = 1,000 QCC/day = $10 USD/day

The preceding table assumes you download trade and quote data, but you can run backtests with only trade data.

Forex

Forex algorithms require some data from the FOREX dataset. The FOREX dataset is available is several resolutions. The resolution you need depends on the Forex subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Second	One file per currency pair per trading day.	3 QCC = $0.03 USD
Minute	One file per currency pair per trading day.	3 QCC = $0.03 USD
Hour	One file per currency pair.	3 QCC = $0.03 USD
Daily	One file per currency pair.	3 QCC = $0.03 USD

For example, the following algorithm subscribes to minute resolution data for one Forex pair:

```
class ForexDataAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        self.AddForex("USDCAD")
```

The following table shows the data cost of the preceding algorithm:

Dataset	Package	Initial Cost	Ongoing Cost
FOREX Data	Minute Download	1 currency pair over 312 trading days => 312 files 312 files @ 3 QCC/file => 312 * 3 QCC = 936 QCC = $9.36 USD	1 currency pair/day => 1 file/day 1 file/day @ 3 QCC/file => 3 QCC/day = $0.03 USD/day

Futures

Futures algorithms require some data from the US Futures dataset. The US Futures dataset is available in several resolutions. The resolution you need depends on the US Future subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Tick	One file per ticker per trading day per data format. Trade, quote, and open interest data are separate files.	6 QCC = $0.06 USD
Second	One file per ticker per trading day per data format. Trade, quote, and open interest data are separate files.	5 QCC = $0.05 USD
Minute	One file per ticker per trading day per data format. Trade, quote, and open interest data are separate files.	5 QCC = $0.05 USD
Hour	One file per ticker per data format. Trade, quote, and open interest data are separate files.	300 QCC = $3 USD
Daily	One file per ticker per data format. Trade, quote, and open interest data are separate files.	100 QCC = $1 USD

If you want continuous contracts in your algorithm, you also need the US Futures Security Master dataset. The following table shows the cost of an annual subscription to the US Futures Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

For example, the following algorithm subscribes to minute resolution data for a universe of ES Futures contracts and creates a continuous contract:

```
class USFuturesDataAlgorithm(QCAlgorithm):
    def Initialize(self):
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        future = self.AddFuture(Futures.Indices.SP500EMini,
                                dataNormalizationMode = DataNormalizationMode.BackwardsRatio,
                                dataMappingMode = DataMappingMode.OpenInterest,
                                contractDepthOffset = 0)
        future.SetFilter(0, 90)
```

The following table shows the data cost of the preceding algorithm on the Quant Researcher tier:

Dataset	Package	Initial Cost	Ongoing Cost
US Futures Security Master	Download On Premise	$600 USD	$0 USD/year
US Futures	Minute Download	1 ticker over 252 trading days with 3 data formats => 1 * 252 * 3 files = 756 files 756 files @ 5 QCC/file => 756 * 5 QCC = 3,780 QCC = $37.80 USD	1 ticker with 3 data formats => 3 files/day 3 file/day @ 5 QCC/file => 15 QCC/day = $0.15 USD/day

The preceding table assumes you download trade, quote, and open interest data. However, you can run backtests with only trade data.

Index Options

Index Options algorithms require some data from the US Index Options dataset. The US Index Options dataset is available in several resolutions. The resolution you need depends on the US Index Option subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Minute	One file per ticker per trading day per data format. Trade, quote, and open interest data are separate files.	15 QCC = $0.15 USD
Hour	One file per ticker per data format. Trade, quote, and open interest data are separate files.	900 QCC = $9 USD
Daily	One file per ticker per data format. Trade, quote, and open interest data are separate files.	300 QCC = $3 USD

For example, the following algorithm subscribes to minute resolution data for a universe of VIX Index Option contracts:

```py
class USIndexOptionsDataAlgorithm(QCAlgorithm):
    def Initialize(self):
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        self.AddIndexOption("VIX")
```

The following table shows the data cost of the preceding algorithm:

Dataset	Package	Initial Cost	Ongoing Cost
US Index Options	Minute Download	1 ticker over 252 trading days with 3 data formats => 1 * 252 * 3 files = 756 files 756 files @ 15 QCC/file => 756 * 15 QCC = 11,340 QCC = $113.40 USD	1 ticker with 3 data formats => 3 files/day 3 files/day @ 15 QCC/file => 45 QCC/day = $0.45 USD/day

The preceding table assumes you download trade, quote, and open interest data. However, you can run backtests with only trade data.

CFD

CFD algorithms require some data from the CFD dataset. The CFD dataset is available is several resolutions. The resolution you need depends on the CFD subscriptions you create in your algorithm and the resolution of data you get in history requests . The following table describes the file format and costs of each resolution:

Resolution	File Format	Cost per file
Second	One file per contract per trading day.	3 QCC = $0.03 USD
Minute	One file per contract per trading day.	3 QCC = $0.03 USD
Hour	One file per contract.	3 QCC = $0.03 USD
Daily	One file per contract.	3 QCC = $0.03 USD

For example, the following algorithm subscribes to minute resolution data for one CFD contract:

```
class CFDDataAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetEndDate(2021, 1, 1)
        self.AddCfd("XAUUSD")
```

The following table shows the data cost of the preceding algorithm:

Dataset	Package	Initial Cost	Ongoing Cost
CFD Data	Minute Download	1 contract over 314 trading days => 314 files 314 files @ 3 QCC/file => 314 * 3 QCC = 942 QCC = $9.42 USD	1 contract/day => 1 file/day 1 file/day @ 3 QCC/file => 3 QCC/day = $0.03 USD/day

Alternative Data

Algorithms that use alternative data require some data from the associated alternative dataset. To view the cost of each alternative dataset, open a dataset listing in the Dataset Market and then click the **Pricing** tab.

Downloading Data

Download in Bulk

Download any of the following datasets in bulk to get all of the data and avoid selection bias:

CFD Data

FOREX Data

US Equities

US Equity Coarse Universe

US Equity Options

US ETF Constituents

US Futures

US Index Options

See Also

Datasets

Download in Bulk

CFD Data

Introduction

Download the CFD dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain data for every ticker and trading day.

Download History

To unlock local access to the CFD dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- OANDA CFD - Daily History
- OANDA CFD - Hour History
- OANDA CFD - Minute History
- OANDA CFD - Second History

You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, follow these steps to download the data:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

Download Daily Updates

After you bulk download the CFD dataset, new daily updates are available at 3 PM Coordinated Universal Time (UTC) after each trading day. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- OANDA CFD - Daily Updates
- OANDA CFD - Hour Updates
- OANDA CFD - Minute Updates
- OANDA CFD - Second Updates

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the CFD dataset, use the CLI Command

Generator to generate your download command and then run it in a terminal in your organization workspace . Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(resolution, overwrite=False):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "CFD Data" --data-type "Bulk" --resolution "
{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update minute and second data files
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
new_data_available = False
for resolution in ["second", "minute"]:
    latest_date = sorted([f for f in os.listdir(f"cfd/oanda/{resolution}/xauusd")])[-1].split('_')[0]
    if latest_date >= END_DATE:
        print(f"{resolution} data is already up to date.")
        continue
    new_data_available = True
    download_data(resolution)

# Update daily and hourly data files
if new_data_available:
    for resolution in ["hour", "daily"]:
        download_data(resolution, True)
```

The preceding script checks the date of the most recent XAUUSD data you have for second and minute resolutions. If there is new data available for either of these resolutions, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your needs.

Size and Format

The following table shows the size and format of the CFD dataset for each resolution:

Resolution	Size	Format
Daily	500 MB	1 file per ticker
Hour	1 GB	1 file per ticker
Minute	50 GB	1 file per ticker per day
Second	200 TB	1 file per ticker per day

Price

The following table shows the price of the CFD dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	799.92	199.92
Hour	799.92	199.92
Minute	799.92	199.92
Second	799.92	199.92

Download in Bulk

FOREX Data

Introduction

Download the FOREX dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain data for every ticker and trading day.

Download History

To unlock local access to the Forex dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- OANDA Forex - Daily History
- OANDA Forex - Hour History
- OANDA Forex - Minute History
- OANDA Forex - Second History

You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, follow these steps to download the data:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

Download Daily Updates

After you bulk download the Forex dataset, new daily updates are available at 3 PM Coordinated Universal Time (UTC) after each trading day. To gain access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- OANDA Forex - Daily Updates
- OANDA Forex - Hour Updates
- OANDA Forex - Minute Updates
- OANDA Forex - Second Updates

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the Forex dataset, use the CLI Command

Generator to generate your download command and then run it in a terminal in your organization workspace . Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(resolution, overwrite=False):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "FOREX Data" --data-type "Bulk" --resolution "
{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update minute and second data files
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
new_data_available = False
for resolution in ["second", "minute"]:
    latest_date = sorted([f for f in os.listdir(f"forex/oanda/{resolution}/eurusd")])[-1].split('_')
[0]
    if latest_date >= END_DATE:
        print(f"{resolution} data is already up to date.")
        continue
    new_data_available = True
    download_data(resolution)

# Update daily and hourly data files
if new_data_available:
    for resolution in ["hour", "daily"]:
        download_data(resolution, True)
```

To update your local dataset, the preceding script checks the date of the most recent EURUSD data you have for second and minute resolutions. If there is new data available for either of these resolutions, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your needs.

Size and Format

The following table shows the size of the Forex dataset for each resolution:

Resolution	Size	Format
Daily	500 MB	1 file per ticker
Hour	1 GB	1 file per ticker
Minute	50 GB	1 file per ticker per day
Second	200 TB	1 file per ticker per day

Price

The following table shows the price of the Forex dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	799.92	199.92
Hour	799.92	199.92
Minute	799.92	199.92
Second	799.92	199.92

Download in Bulk

US Equities

Introduction

Download the US Equities dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain data for every ticker and trading day. If the resolution you download provides trade and quote data, the bulk download contains both data types. To check which data types each resolution provides, see Resolutions .

The US Equities dataset depends on the US Equity Security Master dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes.

Download History

To unlock local access to the US Equities dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Equity Daily History by AlgoSeek
- US Equity Hourly History by AlgoSeek
- US Equity Minute History by AlgoSeek
- US Equity Second History by AlgoSeek
- US Equity Tick History by AlgoSeek

If you don't already subscribe to the **US Equity Security Master by QuantConnect** data package, subscribe to it too. You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, to download the US Equities data, follow these steps:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

To download the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Download Daily Updates

After you bulk download the US Equities dataset, new daily updates are available at 7 AM Eastern Time (ET) after each trading day. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Equity Daily Updates by AlgoSeek
- US Equity Hourly Updates by AlgoSeek
- US Equity Minute Updates by AlgoSeek
- US Equity Second Updates by AlgoSeek
- US Equity Tick Updates by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Equities dataset, use the CLI Command Generator to generate your download command and then run it in a terminal in your organization workspace . To update your local copy of the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(resolution, overwrite=False):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Equities" --data-type "Bulk" --resolution "
{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update minute, second, and tick data files
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
new_data_available = False
for resolution in ["tick", "second", "minute"]:
    latest_date = sorted([f for f in os.listdir(f"equity/usa/{resolution}/spy")])[-1].split('_')[0]
    if latest_date >= END_DATE:
        print(f"{resolution} data is already up to date.")
        continue
    new_data_available = True
    download_data(resolution)

# Update daily and hourly data files
if new_data_available:
    for resolution in ["hour", "daily"]:
        download_data(resolution, True)
```

The preceding script checks the date of the most recent SPY data you have for tick, second, and minute resolutions. If there is new data available for any of these resolutions, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your

needs.

Size and Format

The following table shows the size and format of the US Equities dataset for each resolution:

Resolution	Size	Format
Daily	2 GB	1 file per ticker
Hour	4 GB	1 file per ticker
Minute	500 GB	1 file per ticker per day
Second	1.5 TB	1 file per ticker per day
Tick	1.5 TB	1 file per ticker per day

Price

The following table shows the price of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

The following table shows the price of the US Equity dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	3,480	2,640
Hour	3,480	2,640
Minute	Contact us	2,640
Second	Contact us	2,640
Tick	Contact us	2,640

Download in Bulk

US Equity Coarse Universe

Introduction

Download the US Equity Coarse Universe dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain data for every trading day.

The US Equity Coarse Universe dataset depends on the US Equity Security Master dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes.

Download History

To unlock local access to the US Equity Coarse Universe dataset, open the Pricing page of your organization and subscribe to the **US Equity Coarse Universe History by QuantConnect** data package. If you don't already subscribe to the **US Equity Security Master by QuantConnect** data package, subscribe to it too. You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, to download the US Equity Coarse Universe data, open a terminal in your organization workspace and run:

```
$ lean data download --dataset "US Equity Coarse Universe" --data-type "Bulk"
```

To download the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Download Daily Updates

After you bulk download the US Equity Coarse Universe dataset, new daily updates are available at 7 AM Eastern Time (ET) after each trading day. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to the **US Equity Coarse Universe Updates by QuantConnect** data package. You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Equity Coarse Universe dataset, open a terminal in your organization workspace and run:

```
$ lean data download --dataset "US Equity Coarse Universe" --data-type "Bulk"
```

To update your local copy of the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script downloads the latest data when it's available:

```py
import os
from datetime import datetime
from pytz import timezone

END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
latest_date = sorted([f for f in os.listdir(f"equity/usa/fundamental/coarse")])[-1]
if latest_date >= END_DATE:
    print(f"Your local data is already up to date.")
else:
    print("Updating data...")
    os.system(f'lean data download --dataset "US Equity Coarse Universe" --data-type "Bulk"')
```

The preceding script checks the date of the most recent US Equity Coarse Universe data you have. If there is new data available, it downloads the new data files.

Size and Format

The US Equity Coarse Universe dataset is 4 GB in size. We structure the data files so there is one file per day.

Price

The following table shows the price of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

All of the historical US Equity Coarse Universe data costs $1,800. An annual subscription to daily updates costs $960/year.

Download in Bulk

US Equity Options

Introduction

Download the US Equity Options dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain trade, quote, and open interest data for every ticker and trading day.

The US Equity Options dataset depends on the US Equity Security Master dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes.

Download History

To unlock local access to the US Equity Options dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Equity Options Daily History by AlgoSeek
- US Equity Options Hour History by AlgoSeek
- US Equity Options Minute History by AlgoSeek

If you don't already subscribe to the **US Equity Security Master by QuantConnect** data package, subscribe to it too. You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, to download the US Equity Options data, follow these steps:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

To download the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Download Daily Updates

After you bulk download the US Equity Options dataset, new daily updates are available at 8 PM Coordinated Universal Time (UTC) two days after each trading day. For example, the minute resolution data for Monday is available on Wednesday at 8 PM UTC. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Equity Options Daily Updates by AlgoSeek
- US Equity Options Minute Updates by AlgoSeek
- US Equity Options Hour Updates by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Equity Options dataset, use the CLI Command Generator to generate your download command and then run it in a terminal in your organization workspace . To update your local copy of the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(resolution, overwrite=False):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Equity Options" --data-type "Bulk" --option-style
"American" --resolution "{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update data files
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
latest_date = sorted([f for f in os.listdir(f"option/usa/minute/aapl")])[-1].split('_')[0]
if latest_date >= END_DATE:
    print(f"Your data is already up to date.")
else:
    download_data("minute")
    for resolution in ['hour', 'daily']:
        download_data(resolution, True)
```

The preceding script checks the date of the most recent minute resolution data you have for AAPL. If there is new minute data available, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your needs.

Size and Format

The following table shows the size and format of the US Equity Options dataset for each resolution:

Resolution	Size	Format
Daily	200 GB	1 file per ticker
Hour	500 GB	1 file per ticker
Minute	6 TB	1 file per ticker per day

Price

The following table shows the price of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

The following table shows the price of the US Equity Options dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	Contact us	1,920
Hour	Contact us	2,640
Minute	Contact us	2,880

Download in Bulk

US ETF Constituents

Introduction

Download the US ETF Constituents dataset in bulk to get the full dataset without any ETF selection bias. The bulk dataset package contains constituents data for all of the supported ETFs for every trading day.

The US ETF Constituents dataset depends on the US Equity Security Master dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes.

Download History

To unlock local access to the US ETF Constituents dataset, open the Pricing page of your organization and subscribe to the **US ETF Constituents History by QuantConnect** data package. If you don't already subscribe to the **US Equity Security Master by QuantConnect** data package, subscribe to it too. You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, to download the US ETF Consitutents data, open a terminal in your organization workspace and run:

```
$ lean data download --dataset "US ETF Constituents" --data-type "Download in Bulk"
```

To download the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Download Daily Updates

After you bulk download the US ETF Constituents dataset, new daily updates are available at 7 AM Eastern Time (ET) after each trading day. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to the **US ETF Constituents Updates by QuantConnect** data package. You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US ETF Constituents dataset, open a terminal in your organization workspace and run:

```
$ lean data download --dataset "US ETF Constituents" --data-type "Download in Bulk"
```

To update your local copy of the US Equity Security Master, run:

```
$ lean data download --dataset "US Equity Security Master"
```

Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all of the new data that's missing from your local copy:

```py
import os
from datetime import datetime
from pytz import timezone

END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
latest_date = sorted([f for f in os.listdir("equity/usa/universes/etf/spy")])[-1].split(".")[0]
if latest_date >= END_DATE:
    print("Your data is already up to date.")
else:
    print(f"Updating data...")
    os.system(f'lean data download --dataset "US ETF Constituents" --data-type "Download in Bulk"')
```

The preceding script checks the date of the most recent SPY data you have. If there is new data available for SPY, it downloads the new data files for all of the ETFs. You may need to adjust this script to fit your needs.

Size and Format

The US ETF Constituents dataset is 50 GB in size. We structure the data files so there is one file per ETF per day.

Price

The following table shows the price of an annual subscription to the US Equity Security Master for each organization tier:

Tier	Price ($/Year)
Quant Researcher	600
Team	900
Trading Firm	1,200
Institution	1,800

All of the historical US ETF Constituents data costs $3,960. An annual subscription to daily updates costs $1,200/year.

Download in Bulk

US Futures

Introduction

Download the US Futures dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain trade, quote, and open interest data for every ticker and trading day.

Download History

To unlock local access to the US Futures dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Futures Daily History by AlgoSeek
- US Futures Hour History by AlgoSeek
- US Futures Minute History by AlgoSeek
- US Futures Second History by AlgoSeek
- US Futures Tick History by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, follow these steps to download the data:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

Download Daily Updates

After you bulk download the US Futures dataset, new daily updates are available at 7 AM Eastern Time (ET) after each trading day. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Futures Daily Updates by AlgoSeek
- US Futures Hour Updates by AlgoSeek
- US Futures Minute Updates by AlgoSeek
- US Futures Second Updates by AlgoSeek
- US Futures Tick Updates by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Futures dataset, use the CLI Command Generator to generate your download command and then run it in a terminal in your organization workspace . Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions and markets:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(market, resolution, overwrite=False):
    print(f"Updating {market} {resolution} data...")
    command = f'lean data download --dataset "US Futures" --data-type "Bulk" --market "{market}" --
resolution "{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update minute, second, and tick data files
MARKETS = ['CBOT', 'CFE', 'CME', 'COMEX', 'ICE', 'INDIA', 'NYMEX']
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
new_data_available = False
for resolution in ["tick", "second", "minute"]:
    for f in os.listdir(f"future/cbot"):
        if f != resolution:
            continue
        latest_date = sorted([f for f in os.listdir(f"future/cbot/{resolution}/zc")])[-1].split('_')
[0]
        if latest_date >= END_DATE:
            print(f"{resolution} data is already up to date.")
            continue
        new_data_available = True
        for market in MARKETS:
            download_data(market, resolution)

# Update daily and hourly data files
if new_data_available:
    for resolution in ["hour", "daily"]:
        for market in MARKETS:
            download_data(market, resolution, True)
```

The preceding script checks the date of the most recent ZC data you have from the CBOT market for tick, second, and minute resolutions. If there is new data available for any of these resolutions, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions and markets, adjust this script to your needs.

Size and Format

The following table shows the size and format of the US Futures dataset for each resolution:

Resolution	Size	Format
Daily	500 MB	1 file per ticker
Hour	1 GB	1 file per ticker
Minute	24 GB	1 file per ticker per day
Second	300 GB	1 file per ticker per day
Tick	1.5 TB	1 file per ticker per day

Price

The following table shows the price of the US Futures dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	Contact us	1,920
Hour	Contact us	2,640
Minute	Contact us	2,880
Second	Contact us	2,880
Tick	Contact us	2,880

Download in Bulk

US Index Options

Introduction

Download the US Index Options dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contain trade and quote data for every ticker and trading day.

Download History

To unlock local access to the US Index Options dataset, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Index Options Daily Updates by AlgoSeek
- US Index Options Hour History by AlgoSeek
- US Index Options Minute History by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to local access, follow these steps to download the data:

1. Log in to the Algorithm Lab.
2. On the **CLI** tab of the dataset listing , use the **CLI Command Generator** to generate your download command and then copy it.

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.

3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

Download Daily Updates

After you bulk download the US Index Options dataset, new daily updates are available at 8 PM Coordinated Universal Time (UTC) two days after each trading day. For example, the minute resolution data for Monday is available on Wednesday at 8 PM UTC. To unlock local access to the data updates, open the Pricing page of your organization and subscribe to at least one of the following data packages:

- US Index Options Daily Updates by AlgoSeek
- US Index Options Minute Updates by AlgoSeek
- US Index Options Hour Updates by AlgoSeek

You need billing permissions to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Index Options dataset, use the CLI Command Generator to generate your download command and then run it in a terminal in your organization

workspace . Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of your organization workspace and run it to update your data files. The following example script updates all data resolutions:

```py
import os
from datetime import datetime
from pytz import timezone

# Define a method to download the data
def download_data(resolution, overwrite=False):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Index Options" --data-type "Bulk" --resolution "
{resolution}"'
    if overwrite:
        command += " --overwrite"
    os.system(command)

# Update data files
END_DATE = datetime.now(timezone("US/Eastern")).strftime("%Y%m%d")
latest_date = sorted([f for f in os.listdir(f"indexoption/usa/minute/spx")])[-1].split('_')[0]
if latest_date >= END_DATE:
    print(f"Your data is already up to date.")
else:
    download_data("minute")
    for resolution in ['hour', 'daily']:
        download_data(resolution, True)
```

The preceding script checks the date of the most recent minute resolution data you have for SPX. If there is new minute data available, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your needs.

Size and Format

The following table shows the size of the US Index Options dataset for each resolution:

Resolution	Size	Format
Daily	5 GB	1 file per ticker
Hour	40 GB	1 file per ticker
Minute	500 GB	1 file per ticker per day

Price

The following table shows the price of the US Index Options dataset subscriptions:

Resolution	Price of Historical Data ($)	Price of Daily Updates ($/Year)
Daily	Contact us	1,920
Hour	Contact us	2,640
Minute	Contact us	2,880

Datasets

Generating Data

Introduction

Running the LEAN engine locally with the CLI requires you to have your own local data, but real market data can be expensive and governed by difficult redistribution licenses. Instead of using actual market data, you can also opt to use realistic fake data by using LEAN's random data generator. This generator uses a Brownian motion model to generate realistic market data. It is capable of generating data for most of LEAN's supported security types and resolutions, which makes it a good solution to design and test algorithms without the need to buy real financial data.

Supported Security Types

The random data generator supports the following security types and resolutions:

Security Type	Supported Resolutions
Equity	Tick, Second, Minute, Hour, and Daily
Forex	Tick, Second, Minute, Hour, and Daily
CFD	Tick, Second, Minute, Hour, and Daily
Future	Tick, Second, Minute, Hour, and Daily
Crypto	Tick, Second, Minute, Hour, and Daily
Option	Minute

Supported Densities

The random data generator supports the following densities:

Density	Description
Dense	At least one data point per resolution step.
Sparse	At least one data point per 5 resolution steps.
VerySparse	At least one data point per 50 resolution steps.

Run the Generator

Follow these steps to generate random data:

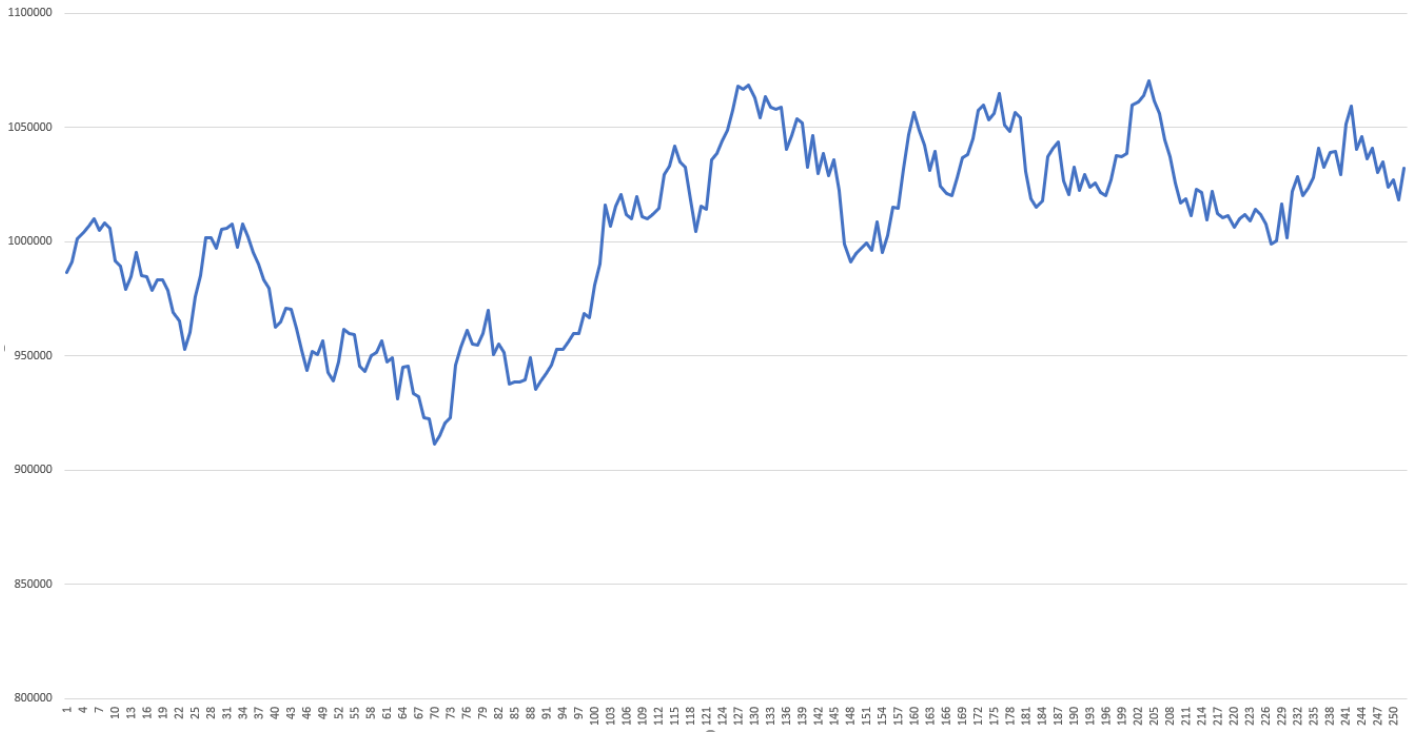
1. Open a terminal in one of your organization workspaces .
2. Run `lean data generate --start 20150101 --symbol-count 10` to generate dense minute Equity data since 01-01-2015 for 10 random symbols.

   ```
   $ lean data generate --start 20150101 --symbol-count 10
   Begin data generation of 10 randomly generated Equity assets...
   ```

   You can also specify an end date using `--end <yyyyMMdd>` , generate data for a different security type using `--security-type <type>` , for a different resolution using `--resolution <resolution>` , or with a different density using `--data-density <density>` .

   For a full list of options, run `lean data generate --help` or see Options .

The following image shows an example time series of simulated data:



Datasets

Custom Data

Introduction

Running the LEAN engine locally with the CLI requires you to have your own local data. Besides market data, LEAN also supports importing custom data into your algorithm. This page explains how to access data from local files in your algorithm when running the LEAN engine locally.

Import Local Files

When running LEAN locally using the CLI, you can already use all of the features explained in the Importing Data page of the LEAN documentation. However, sometimes you may not want to upload your file to a cloud storage service like Dropbox. You can follow these steps to convert your custom data class (the class extending `PythonData`) to retrieve data from a local file instead of a remote one:

1. Copy the data file that you want to use to the **data** directory in your organization workspace .
2. Open the source file containing your custom data class in an editor.
3. Update your `GetSource` method to load data from the local file in your **data** directory. Make sure you only use forward slashes. Backward slashes as path separators don't work. For the Weather example in the LEAN documentation, that is done like this:

```
PY
import os

from QuantConnect import Globals, SubscriptionTransportMedium
from QuantConnect.Data import SubscriptionDataSource
from QuantConnect.Python import PythonData


class Weather(PythonData):
    def GetSource(self, config: SubscriptionDataConfig, date: datetime, is_live: bool) ->
SubscriptionDataSource:
        # Old:
        # source = "https://www.dropbox.com/s/8v6z949n25hyk9o/custom_weather_data.csv?dl=1"
        # return SubscriptionDataSource(source, SubscriptionTransportMedium.RemoteFile)

        # New:
        # Replace custom_weather_data.csv with the path to your data file in the data directory
        source = os.path.join(Globals.DataFolder, "custom_weather_data.csv")
        return SubscriptionDataSource(source, SubscriptionTransportMedium.LocalFile)
```

4. Save the source file.

Projects

Projects

Project Management

Introduction

Creating new projects is an important feature of the Lean CLI. The CLI can automatically scaffold basic Python and C# projects, creating basic algorithm files, research notebooks, and the required editor configuration. Projects scaffolded by the CLI are similar to the ones created on QuantConnect, making it easy to switch between your local environment and the cloud.

Create Projects

Follow these steps to create a new Python project:

1. Open a terminal in one of your organization workspaces .
2. Run `lean project-create --language python "<projectName>"` to create a new project named **<projectName>** .

   ```
   $ lean project-create --language python "My Python Project"
   Successfully created Python project 'My Python Project'
   ```

   This command creates the **./<projectName>** directory and creates a simple **main.py** file, a Python-based research notebook, a project configuration file , and editor configuration for PyCharm and VS Code.

Follow these steps to create a new C# project:

1. Open a terminal in one of your organization workspaces .
2. Run `lean project-create --language csharp "<projectName> "` to create a new C# project named **<projectName>** .

   ```
   $ lean project-create --language csharp "My CSharp Project"
   Successfully created C# project 'My CSharp Project'
   ```

   This command creates the **./<projectName>** directory and creates a simple **Main.cs** file, a C#-based research notebook, a project configuration file , and editor configuration for Visual Studio, Rider, and VS

Code.

The project name must only contain `-` , `_` , letters, numbers, and spaces. The project name can't start with a space or be any of the following reserved names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, or LPT9.

You can provide a project name containing forward slashes to create a project in a sub-directory. In case any of the given sub-directories does not exist yet, the CLI creates them for you.

Set the Default Language

It is also possible to set the default language to use when running `lean project-create` :

- Run `lean config set default-language python` to set the default language to Python, after which you no longer need to provide the `--language python` option to create Python projects.

```
$ lean config set default-language python
$ lean project-create "My Python Project"
Successfully created Python project 'My Python Project'
```

- Run `lean config set default-language csharp` to set the default language to C#, after which you no longer need to provide the `--language csharp` option to create C# projects.

```
$ lean config set default-language csharp
$ lean project-create "My CSharp Project"
Successfully created C# project 'My CSharp Project'
```

Rename Projects

Follow these steps to rename a project that you have on your local machine and in QuantConnect Cloud:

1. Open the organization workspaces on your local machine where you store the project.
2. Rename the project file.

   The project name must only contain `-` , `_` , letters, numbers, and spaces. The project name can't start with a space or be any of the following reserved names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, or LPT9.

3. Log in to the CLI if you haven't done so already.
4. Open a terminal in the same organization workspace.
5. Run `lean cloud push --project "<projectName>"` .

```
$ lean cloud push --project "My Renamed Project"
[1/1] Pushing "My Renamed Project"
Renaming project in the cloud from 'My Project' to 'My Renamed Project'
Successfully updated name and files and libraries for 'My Project'
```

Alternatively, you can rename the project in QuantConnect Cloud and then pull the project to your local machine.

Delete Projects

Follow these steps to delete a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that stores the project.
3. Run `lean project-delete "<projectName>"` .

```
$ lean project-delete "My Project"
Successfully deleted project 'My Project'
```

This command deletes the project on your local machine and in QuantConnect Cloud.

If you are a collaborator on the project, this command doesn't delete the project for the other collaborators, but it removes you as a collaborator.

Projects

Cloud Synchronization

Introduction

Cloud synchronization allows you to synchronize your projects in QuantConnect Cloud with your local drive using the Lean CLI. Cloud synchronization makes it possible to use your local development environment when writing your algorithms while using QuantConnect's infrastructure and data library when executing them.

Pulling Cloud Projects

Follow these steps to pull all the cloud projects that you store in an organization to your local drive:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace from which you want to pull projects.
3. Run `lean cloud pull` to pull all your cloud projects to the current directory, creating directories where necessary.

```
$ lean cloud pull
[1/3] Pulling 'Creative Red Mule'
Successfully pulled 'Creative Red Mule/main.py'
[2/3] Pulling 'Determined Yellow-Green Duck'
Successfully pulled 'Determined Yellow-Green Duck/main.py'
Successfully pulled 'Determined Yellow-Green Duck/research.ipynb'
[3/3] Pulling 'Halloween Strategy'
Successfully pulled 'Halloween Strategy/benchmark.py'
Successfully pulled 'Halloween Strategy/main.py'
Successfully pulled 'Halloween Strategy/research.ipynb'
```

4. Update your projects to include the required imports to run the projects locally and to make autocomplete work.

Follow these steps to pull a single cloud project to your local drive:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that stores the project.
3. Run `lean cloud pull --project "<projectName>"` to pull the project named "<projectName>" to **./<projectName>**.

```
$ lean cloud pull --project "My Project"
[1/1] Pulling 'My Project'
Successfully pulled 'My Project/main.py'
Successfully pulled 'My Project/research.ipynb'
```

4. Update your project to include the required imports to run the project locally and to make autocomplete work.

If you have a local copy of the project when you pull the project from the cloud, the configuration values of the cloud project overwrite the configuration values of your local copy .

If one of your team members creates a project library , adds it to a project, and then adds you as a collaborator to the project, you can pull the project but not the library. To pull the library as well, your team member must add you as a collaborator on the library project.

Pushing Local Projects

Follow these steps to push all the local projects in an organization workspace to the cloud:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace.
3. Run `lean cloud push` to push all your local projects in the organization to the cloud, creating new cloud projects where necessary.

```
$ lean cloud push
[1/3] Pushing 'Alpha'
Successfully created cloud project 'Alpha'
Successfully created cloud file 'Alpha/benchmark.py'
Successfully updated cloud file 'Alpha/main.py'
Successfully updated cloud file 'Alpha/research.ipynb'
[2/3] Pushing 'Buy and Hold BNBUSDT'
Successfully created cloud project 'Buy and Hold BNBUSDT'
Successfully updated cloud file 'Buy and Hold BNBUSDT/main.py'
Successfully updated cloud file 'Buy and Hold BNBUSDT/research.ipynb'
[3/3] Pushing 'Creative Red Mule'
Successfully updated cloud file 'Creative Red Mule/main.py'
```

Follow these steps to push a single local project to the cloud:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project you want to push.
3. Run `lean cloud push --project "<projectName>"` to push the project stored in **. / <projectName>** to the cloud.

```
$ lean cloud push --project "My Project"
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
```

If you create a project on your local machine and push it to the cloud, the `lean cloud push` command creates the cloud version of the project in the organization that's linked to your current organization workspace.

If you have a cloud copy of the project when you push the project from your local machine, the configuration values of your local project overwrite the configuration values of your cloud copy.

Detecting Environment

Sometimes it might be useful to run certain logic only when your algorithm is running locally, or when it is running in the cloud. You can use the following code snippet to check where your algorithm is running (replace `Computer` with your computer's hostname):

```py
import platform

if platform.node() == "Computer":
    # Running locally
    pass
else:
    # Running in the cloud
    pass
```

Projects

Structure

Introduction

When you run the `lean project-create` or `lean cloud pull` commands, the CLI creates the basic files and folders most editors need to open your source code, provide autocomplete, and enable local debugging. This page documents exactly which files are created when you create a new local project with `lean project-create` or `lean cloud pull` .

Project Structure

New projects have the following structure:

```
.
├── .idea/
│   ├── misc.xml
│   ├── modules.xml
│   ├── <projectName>.iml
│   └── workspace.xml
├── .vscode/
│   ├── launch.json
│   └── settings.json
├── config.json
├── main.py (only generated by lean project-create)
└── research.ipynb (only generated by lean project-create)
```

These files contain the following content:

File	Content
.idea / misc.xml , **.idea / modules.xml** , **.idea / <projectName>.iml**	These files contain PyCharm configuration so PyCharm can provide accurate autocomplete.
.idea / workspace.xml	This file contains debug configuration to make debugging with PyCharm easier.
.vscode / launch.json	This file contains debug configuration to make debugging with VS Code easier.
.vscode / settings.json	This file contains VS Code configuration so that VS Code's Python and Pylance extensions can provide accurate autocomplete.
config.json	This file contains the project configuration of the created project.
main.py	This file contains a basic Python algorithm to help you get started.
research.ipynb	This file contains a Python-based research notebook that can be opened in a research environment .

Projects

Workflows

Introduction

The Lean CLI supports multiple workflows, ranging from running everything locally to just using your local development environment but keeping all execution in the cloud. This page contains several examples of common workflows, but you're free to mix local and cloud features in any way you'd like.

Cloud-focused Workflow

A cloud-focused workflow (local development, cloud execution) with the CLI might look like this:

1. Open a terminal in one of your organization workspaces .
2. Run `lean cloud pull` to pull remotely changed files.
3. Start programming locally and run backtests in the cloud with `lean cloud backtest "<projectName>" --open --push` whenever there is something to backtest. The `--open` flag means that the backtest results are opened in the browser when done, while the `--push` flag means that local changes are pushed to the cloud before running the backtest.
4. Whenever you want to create a new project, run `lean project-create "<projectName>" --push` and `lean cloud push --project "<projectName>"` to create a new project containing some basic code and to push it to the cloud.
5. When you're finished for the moment, run `lean cloud push` to push all locally changed files to the cloud.

The advantage of this workflow is that you can use all the tools in your local development environment to write your code (i.e. autocomplete, auto-formatting, etc.) while using QuantConnect's computing infrastructure and data when running your code. This advantage means you don't need to have a powerful computer and you don't need to have your own data, since that's all provided by us. The downside to this workflow is that you need a constant internet connection because without an internet connection the CLI can't communicate with the cloud.

Locally-focused Workflow

A locally-focused workflow (local development, local execution) with the CLI might look like this:

1. Open a terminal in one of your organization workspaces .
2. Run `lean project-create "<projectName>"` to create a new project with some basic code to get you started.
3. Work on your strategy in **./ <projectName>** .
4. Run `lean research "<projectName>"` to start a Jupyter Lab session to perform research.
5. Run `lean backtest "<projectName>"` to run a backtest whenever there's something to test. This command runs your strategy in a Docker container containing the same packages as the ones used on QuantConnect.com, but with your own data.

6. Whenever you want to debug a strategy in your local development environment, see Debugging .

With this workflow, you are not limited to the computing power that's available in QuantConnect's infrastructure, because everything runs locally. On the other hand, this also means you must have all your required data available locally. To download some of QuantConnect's data for local usage, see Downloading Data .

Mixed Workflow

A mixed workflow (local development, local debugging, and cloud execution) with the CLI might look like this:

1. Open a terminal in one of your organization workspaces .
2. Run `lean cloud pull` to pull remotely changed files.
3. Start programming on your strategies.
4. Whenever you want to debug a strategy in your local development environment, see Debugging .
5. Whenever you want to backtest a strategy, run `lean cloud backtest "<projectName>" --open --push` to push local changes to the cloud, run a backtest in the cloud, and open the results in the browser once finished.
6. When you're finished for the moment, run `lean cloud push` to push all locally changed files in the organization workspace to the cloud.

The advantage of this workflow is that you can use your local development environment and its debugging tools when writing your algorithms while using QuantConnect's infrastructure and data in backtesting and live trading. Although this does require you to have a local copy of the data that your strategy needs, it doesn't require you to maintain your own infrastructure and data feeds in live trading. To download some of QuantConnect's data for local usage, see Downloading Data .

Projects

Configuration

Introduction

Project-specific configuration is stored in the project directory in the **config.json** file. This file is automatically generated when you run `lean project-create` or `lean cloud pull` . Just like the global and Lean configuration files, the project configuration is stored as JSON but without support for comments.

Properties

The following properties are stored in the **config.json** file:

Property	Description
`description`	This property contains the project's description, which is displayed in backtest reports. It must always be a string.
`parameters`	This property is a key/value object containing the project's parameters. Both the keys and the values must be strings. To load parameter values into your algorithm, see Get Parameters . The parameter values are sent to your algorithm when you deploy the algorithm, so it's not possible to change the parameter values while the algorithm runs.
`cloud-id`	This property is set automatically after the project has been pulled from or pushed to the cloud. It contains the id of the project's counterpart in the cloud and must not be modified or removed manually.
`local-id`	This property is set automatically when the CLI needs to uniquely identify the current project. It contains a unique id that is specific to the project and must not be modified or removed manually.
`libraries`	This property is set automatically when you add a project library to the project. It contains a list of dictionaries that hold the name and path of each library.
`organization-id`	This property is set automatically after the project has been pulled from or pushed to the cloud. It contains the Id of the organization that the project is saved to in the cloud.
`algorithm-language`	This property contains the language of the project. It is automatically set when the project is created and must not be modified or removed manually.
`docker`	This property is a key/value object containing the docker instance's "environment" and "ports" command line run arguments. For example, to expose host port 999 to internal port 6006, write `"docker": { "ports": { "999": "6006"} }` .
`lean-engine`	This property is the version number of the LEAN Engine version that the project uses.
`python-venv`	This property is an integer that represents the Python environment that the project uses.

Projects

Autocomplete

Introduction

Local autocomplete is an important feature of the Lean CLI, making you more productive in your local development environment. The CLI automatically generates the necessary editor configuration when creating new projects to make local autocomplete almost seamless for most popular editors. However, not everything can be configured automatically. This page explains how to make local autocomplete work for Python and C# with all editors supported by the CLI.

Python and PyCharm

Follow these steps to set up local autocomplete for Python in PyCharm:

1. Open a project directory, generated by the CLI, with PyCharm and wait for the project to load.
2. Wait for PyCharm to index all packages and autocomplete starts working.
3. Update your project to include the required imports .

Python and VS Code

Follow these steps to set up local autocomplete for Python in VS Code:

1. Install the Python and Pylance extensions in VS Code.
2. Open a project directory, generated by the CLI, with VS Code and Python autocomplete works automatically.
3. Update your project to include the required imports .

C# and Visual Studio

Follow these steps to set up local autocomplete for C# in Visual Studio:

1. Make sure the .NET 6.0 Runtime is installed in the Visual Studio Installer by clicking **Modify** in the installed Visual Studio version and opening the **Individual components** tab on the window that shows up.
2. Click **Open a project or solution** on Visual Studio's home screen and open the **.csproj** file in one of the project directories generated by the CLI. Visual Studio automatically downloads the required dependencies and indexes them to provide autocomplete.
3. Update your project to include the required imports .

C# and Rider

Follow these steps to set up local autocomplete for C# in Rider:

1. Make sure the .NET 6.0 Runtime is installed.
2. Open the **.csproj** file in a project directory, generated by the CLI, with Rider and wait for the project to load. Rider automatically downloads the required dependencies and indexes them to provide autocomplete.

3. Update your project to include the required imports .

C# and VS Code

Follow these steps to set up local autocomplete for C# in VS Code:

1. Make sure the .NET 6.0 Runtime is installed.
2. Install the C# extension in VS Code.
3. Open a project directory, generated by the CLI, with VS Code and wait for the project to load.
4. Click **Restore** if a pop-up shows in the bottom-right telling you there are dependencies to restore.
5. Update your project to include the required imports .

Imports

Some imports are automatically added to your files when you run them in the cloud. This does not happen locally, so in your local environment, you need to manually import all the classes that you use. You can copy the following code snippet to the top of every file to have the same imports as the ones used in the cloud:

```
from AlgorithmImports import *
```

Staying Up-to-date

Follow these steps to update Python autocomplete to be aware of the latest changes to LEAN:

1. Open a terminal.
2. Run `pip install --upgrade quantconnect-stubs` to update the Python autocomplete.

```
$ pip install --upgrade quantconnect-stubs
Collecting quantconnect-stubs
Installing collected packages: quantconnect-stubs
Successfully installed quantconnect-stubs-11657
```

Follow these steps to update C# autocomplete to be aware of the latest changes to LEAN:

1. Open a terminal in one of your organization workspaces .
2. Run `dotnet add "<projectName>" package QuantConnect.Lean` to update the C# autocomplete for the project in **./ <projectName>** .

```
$ dotnet add "My Project" package QuantConnect.Lean
info : Adding PackageReference for package 'QuantConnect.Lean' into project '/home/john/My
Project/My Project.csproj'.
info : PackageReference for package 'QuantConnect.Lean' version '2.5.11800' updated in file
'/home/john/My Project/My Project.csproj'.
```

Additionally, you can also update C# autocomplete by updating the version of the QuantConnect.Lean package reference in the C# project file (the file ending with **.csproj**). This can be done manually or through your editor's built-in NuGet tooling if your editor has such a feature.

Projects

Libraries

Libraries

Third-Party Libraries

Introduction

The Lean CLI supports using dozens of open source packages in your algorithms. These packages are reviewed by our security team, and when approved, can be used in backtesting, live trading, and research. To use these packages in your algorithm, you will need to add the relevant `import` statement at the top of your code file.

By default, only the libraries in the official LEAN Docker images can be referenced in your algorithms. However, the CLI also supports using custom libraries. This makes it possible to use a library that is not available in the official LEAN Docker images or to use a newer version of an existing library.

Supported Libraries for AMD64 Systems

The CLI supports many of the most popular C# and Python open-source libraries. On AMD64-based systems, the CLI supports the same C# and Python libraries as are supported on QuantConnect. If you're unsure about the architecture of your system, it's most likely AMD64. The following libraries are available on AMD64-based systems:

```
# Name                      Version
absl-py                     1.4.0
adagio                      0.2.4
aesara                      2.8.12
aiodns                      3.0.0
aiohttp                     3.8.4
aiohttp-cors                0.7.0
aiosignal                   1.3.1
alabaster                   0.7.13
ale-py                      0.7.4
alembic                     1.10.3
alpaca-trade-api            0.26
alphalens-reloaded          0.4.3
altair                      4.2.2
ansi2html                   1.8.0
antlr4-python3-runtime      4.11.1
anyio                       3.6.2
appdirs                     1.4.4
arch                        5.3.1
argon2-cffi                 21.3.0
argon2-cffi-bindings        21.2.0
arviz                       0.15.1
astropy                     5.2.1
asttokens                   2.2.1
astunparse                  1.6.3
async-timeout               4.0.2
asyncio-nats-client         0.11.5
```

```
attrs                          23.1.0
autograd                       1.5
autokeras                      1.1.0
autoray                        0.6.3
AutoROM                        0.4.2
AutoROM.accept-rom-license     0.6.1
ax-platform                    0.3.0
Babel                          2.12.1
backcall                       0.2.0
bayesian-optimization          1.4.2
beautifulsoup4                 4.11.2
bleach                         6.0.0
blessed                        1.20.0
blis                           0.7.9
blosc2                         2.0.0
bokeh                          2.4.3
boltons                        23.0.0
botorch                        0.8.2
Bottleneck                     1.3.7
brotlipy                       0.7.0
cachetools                     5.3.0
captum                         0.6.0
catalogue                      2.0.8
catboost                       1.1.1
category-encoders              2.6.0
ccxt                           3.0.72
certifi                        2022.12.7
cffi                           1.15.1
chardet                        5.1.0
charset-normalizer             2.0.4
check-shapes                   1.0.0
click                          7.1.2
clikit                         0.6.2
cloudpickle                    2.2.1
cmaes                          0.9.1
cmdstanpy                      1.1.0
colorama                       0.4.6
colorcet                       3.0.1
colorful                       0.5.5
colorlog                       6.7.0
colorlover                     0.3.0
colour                         0.1.5
comm                           0.1.3
commonmark                     0.9.1
conda                          23.3.1
conda-content-trust            0.1.3
conda-package-handling         2.0.2
conda_package_streaming        0.7.0
confection                     0.0.4
cons                           0.4.5
contourpy                      1.0.7
convertdate                    2.4.0
copulae                        0.7.7
copulas                        0.8.0
cramjam                        2.6.2
crashtest                      0.3.1
creme                          0.6.1
cryptography                   38.0.4
cufflinks                      0.17.3
cvxopt                         1.3.0
cvxpy                          1.3.0
cycler                         0.11.0
cymem                          2.0.7
Cython                         0.29.33
darts                          0.23.1
dash                           2.9.3
dash-core-components           2.0.0
dash-cytoscape                 0.3.0
dash-html-components           2.0.0
dash-table                     5.0.0
dask                           2023.4.0
deap                           1.3.3
debugpy                        1.5.1
decorator                      5.1.1
defusedxml                     0.7.1
Deprecated                     1.2.13
dgl                            1.0.1
```

```
dill                          0.3.6
dimod                         0.12.3
diskcache                     5.6.1
distlib                       0.3.6
distributed                   2021.4.1
dm-tree                       0.1.8
docutils                      0.19
DoubleML                      0.5.2
dtreeviz                      2.2.1
dtw-python                    1.3.0
dwave-cloud-client            0.10.3
dwave-drivers                 0.4.4
dwave-greedy                  0.3.0
dwave-hybrid                  0.6.9
dwave-inspector               0.3.0
dwave-inspectorapp            0.3.0
dwave-neal                    0.6.0
dwave-networkx                0.8.12
dwave-ocean-sdk               6.1.1
dwave-preprocessing           0.5.3
dwave-samplers                1.0.0
dwave-system                  1.18.0
dwave-tabu                    0.5.0
dwavebinarycsp                0.2.0
dx                            0.1.22
ecos                          2.0.12
EMD-signal                    1.4.0
empyrical                     0.5.5
empyrical-reloaded            0.5.9
en-core-web-md                3.5.0
en-core-web-sm                3.5.0
entrypoints                   0.4
ephem                         4.1.4
et-xmlfile                    1.1.0
etuples                       0.3.8
exceptiongroup                1.1.1
exchange-calendars            4.2.6
executing                     1.2.0
ezyrb                         1.3.0.post2304
fastai                        2.7.11
fastai2                       0.0.30
fastcore                      1.5.29
fastdownload                  0.0.7
fasteners                     0.18
fastjsonschema                2.16.3
fastparquet                   2023.2.0
fastprogress                  1.0.3
fasttext                      0.9.2
featuretools                  0.23.1
filelock                      3.12.0
findiff                       0.9.2
finrl                         0.3.1
FixedEffectModel              0.0.5
Flask                         2.0.3
flatbuffers                   23.3.3
fonttools                     4.39.3
fracdiff                      0.9.0
frozendict                    2.3.7
frozenlist                    1.3.3
fs                            2.4.16
fsspec                        2023.4.0
fst-pso                       1.8.1
fugue                         0.8.3
fugue-sql-antlr               0.1.6
future                        0.18.3
fuzzy-c-means                 1.6.3
FuzzyTM                       2.0.5
gast                          0.4.0
gensim                        4.3.0
gevent                        22.10.2
gluonts                       0.12.3
google-api-core               2.11.0
google-auth                   2.17.3
google-auth-oauthlib          0.4.6
google-pasta                  0.2.0
googleapis-common-protos      1.59.0
gpflow                        2.7.0
```

```
gplearn                       0.4.2
gpustat                       1.1
GPUtil                        1.4.0
gpytorch                      1.9.1
graphviz                      0.20.1
greenlet                      2.0.2
grpcio                        1.54.0
gym                           0.21.0
Gymnasium                     0.26.3
gymnasium-notices             0.0.1
h2o                           3.40.0.1
h5netcdf                      1.1.0
h5py                          3.8.0
hijri-converter               2.2.4
hmmlearn                      0.2.8
holidays                      0.23
holoviews                     1.15.4
homebase                      1.0.1
hopcroftkarp                  1.2.5
html5lib                      1.1
httpstan                      4.9.1
hurst                         0.0.5
hvplot                        0.8.2
hyperopt                      0.2.7
idna                          3.4
iisignature                   0.24
ijson                         3.2.0.post0
imageio                       2.27.0
imagesize                     1.4.1
imbalanced-learn              0.10.1
importlib-metadata            4.13.0
importlib-resources           5.12.0
iniconfig                     2.0.0
injector                      0.20.1
interpret                     0.3.0
interpret-core                0.3.2
ipykernel                     6.22.0
ipython                       8.12.0
ipython-genutils              0.2.0
ipywidgets                    8.0.4
itsdangerous                  2.1.2
jax                           0.4.4
jaxlib                        0.4.4
jedi                          0.18.2
Jinja2                        3.1.2
joblib                        1.2.0
jqdatasdk                     1.8.11
json5                         0.9.11
jsonpatch                     1.32
jsonpointer                   2.0
jsonschema                    4.17.3
jupyter                       1.0.0
jupyter-bokeh                 3.0.5
jupyter-console               6.6.3
jupyter-dash                  0.4.2
jupyter-resource-usage        0.7.2
jupyter-server                1.24.0
jupyter_client                8.2.0
jupyter_core                  5.3.0
jupyterlab                    3.4.4
jupyterlab-pygments           0.2.2
jupyterlab-widgets            3.0.7
jupyterlab_server             2.22.1
keract                        4.5.1
keras                         2.11.0
keras-nlp                     0.4.1
keras-rl                      0.4.2
keras-tuner                   1.3.5
kiwisolver                    1.4.4
kmapper                       2.0.1
korean-lunar-calendar         0.3.1
kt-legacy                     1.0.5
langcodes                     3.3.0
lark                          1.1.5
lazy_loader                   0.2
lazypredict                   0.2.12
libclang                      16.0.0
```

```
lightgbm                      3.3.5
lime                          0.2.0.1
line-profiler                 4.0.2
linear-operator               0.3.0
littleutils                   0.2.2
livelossplot                  0.5.5
llvmlite                      0.39.1
locket                        1.0.0
logical-unification           0.4.5
LunarCalendar                 0.0.9
lxml                          4.9.2
lz4                           4.3.2
Mako                          1.2.4
Markdown                      3.4.3
MarkupSafe                    2.1.2
marshmallow                   3.19.0
matplotlib                    3.7.0
matplotlib-inline             0.1.6
miniful                       0.0.6
miniKanren                    1.0.3
minorminer                    0.2.9
mistune                       2.0.5
mljar-supervised              0.11.5
mlxtend                       0.21.0
mmh3                          2.5.1
modin                         0.18.1
mpi4py                        3.1.4
mplfinance                    0.12.9b7
mpmath                        1.2.1
msgpack                       1.0.4
mthree                        2.4.0
multidict                     6.0.4
multipledispatch              0.6.0
multiprocess                  0.70.14
multitasking                  0.0.11
murmurhash                    1.0.9
nbclassic                     0.5.5
nbclient                      0.7.3
nbconvert                     7.3.1
nbeats-keras                  1.8.0
nbeats-pytorch                1.8.0
nbformat                      5.8.0
nest-asyncio                  1.5.6
networkx                      2.8.8
neural-tangents               0.6.2
neuralprophet                 0.5.0
nfoursid                      1.0.1
nltk                          3.8.1
nose                          1.3.7
notebook                      6.5.4
notebook_shim                 0.2.2
ntlm-auth                     1.5.0
numba                         0.56.4
numerapi                      2.13.2
numexpr                       2.8.4
numpy                         1.23.5
numpydoc                      1.5.0
nvidia-cublas-cu11            11.10.3.66
nvidia-cuda-nvrtc-cu11        11.7.99
nvidia-cuda-runtime-cu11      11.7.99
nvidia-cudnn-cu11             8.5.0.96
nvidia-ml-py                  11.525.112
oauthlib                      3.2.2
openai                        0.26.5
opencensus                    0.11.2
opencensus-context            0.1.3
opencv-python                 4.7.0.72
openpyxl                      3.1.1
opt-einsum                    3.3.0
optuna                        3.1.0
orjson                        3.8.10
ortools                       9.4.1874
osqp                          0.6.2.post9
outdated                      0.2.2
packaging                     23.1
pandas                        1.5.3
pandas-datareader             0.10.0
```

```
pandas-flavor                 0.5.0
pandas-market-calendars       4.1.4
pandas-ta                     0.3.14b0
pandocfilters                 1.5.0
panel                         0.14.3
param                         1.13.0
parso                         0.8.3
partd                         1.4.0
pastel                        0.2.1
pathos                        0.3.0
pathy                         0.10.1
patsy                         0.5.3
pbr                           5.11.1
penaltymodel                  1.0.2
PennyLane                     0.29.0
PennyLane-Lightning           0.29.0
PennyLane-qiskit              0.29.0
persim                        0.3.1
pexpect                       4.8.0
pickleshare                   0.7.5
Pillow                        9.5.0
pingouin                      0.5.3
pip                           22.3.1
pkgutil_resolve_name          1.3.10
platformdirs                  3.2.0
plotly                        5.13.1
plotly-resampler              0.8.3.2
plucky                        0.4.3
pluggy                        1.0.0
ply                           3.11
pmdarima                      2.0.2
polars                        0.16.9
pox                           0.3.2
ppft                          1.7.6.6
pprofile                      2.1.0
ppscore                       1.2.0
preshed                       3.0.8
prometheus-client             0.16.0
prompt-toolkit                3.0.38
property-cached               1.6.4
prophet                       1.1.2
protobuf                      3.19.6
psutil                        5.9.5
psycopg2-binary               2.9.6
ptvsd                         4.3.2
ptyprocess                    0.7.0
PuLP                          2.7.0
pure-eval                     0.2.2
py-cpuinfo                    9.0.0
py-heat                       0.0.6
py-heat-magic                 0.0.2
py-lets-be-rational           1.0.1
py-spy                        0.3.14
py-vollib                     1.0.1
py4j                          0.10.9.7
pyaml                         21.10.1
pyarrow                       11.0.0
pyasn1                        0.4.8
pyasn1-modules                0.2.8
pybind11                      2.10.4
pycares                       4.3.0
pycosat                       0.6.4
pycparser                     2.21
pyct                          0.5.0
pydantic                      1.10.7
pyDeprecate                   0.3.2
pydevd-pycharm                201.8538.36
pydmd                         0.4.0.post2301
pyerfa                        2.0.0.3
pyfolio                       0.9.2
pyfolio-reloaded              0.9.5
pyFUME                        0.2.25
Pygments                      2.15.1
pykalman                      0.9.5
pylev                         1.4.0
pyluach                       2.2.0
pymannkendall                 1.4.3
```

```
pymc                          5.0.2
pymdptoolbox                  4.0b3
PyMeeus                       0.5.12
PyMySQL                       1.0.3
pynndescent                   0.5.9
pyod                          1.0.9
Pyomo                         6.5.0
pyOpenSSL                     22.0.0
pyparsing                     3.0.9
pyportfolioopt                1.5.4
pyqubo                        1.3.1
pyrb                          1.0.1
pyro-api                      0.1.2
pyro-ppl                      1.8.4
pyrsistent                    0.19.3
pysimdjson                    5.0.2
PySocks                       1.7.1
pystan                        3.6.0
pytensor                      2.9.1
pytest                        7.3.1
python-dateutil               2.8.2
python-statemachine           1.0.3
pytorch-ignite                0.4.11
pytorch-lightning             1.7.4
pytz                          2023.3
pyvinecopulib                 0.6.2
pyviz-comms                   2.2.1
PyWavelets                    1.4.1
PyYAML                        6.0
pyzmq                         25.0.2
qdldl                         0.1.7
qiskit                        0.41.1
qiskit-aer                    0.11.2
qiskit-ibmq-provider          0.20.1
qiskit-terra                  0.23.2
qpd                           0.4.1
qtconsole                     5.4.2
QtPy                          2.3.1
quadprog                      0.1.11
quantecon                     0.6.0
QuantLib                      1.29
QuantStats                    0.0.59
rauth                         0.7.3
ray                           2.3.0
rectangle-packer              2.0.1
regex                         2023.3.23
requests                      2.28.1
requests-ntlm                 1.1.0
requests-oauthlib             1.3.1
retrying                      1.3.4
retworkx                      0.12.1
rich                          12.4.4
ripser                        0.6.4
Riskfolio-Lib                 4.0.3
riskparityportfolio           0.4
river                         0.14.0
rsa                           4.9
ruamel.yaml                   0.17.21
ruamel.yaml.clib              0.2.6
ruptures                      1.1.7
rustworkx                     0.12.1
SALib                         1.4.7
scikeras                      0.10.0
scikit-image                  0.19.3
scikit-learn                  1.2.1
scikit-learn-extra            0.2.0
scikit-multiflow              0.5.3
scikit-optimize               0.9.0
scikit-plot                   0.3.7
scikit-tda                    1.0.0
scipy                         1.10.1
scs                           3.2.3
sdeint                        0.3.0
seaborn                       0.12.2
semantic-version              2.10.0
Send2Trash                    1.8.0
setuptools                    64.0.2
```

```
setuptools-scm                7.1.0
shap                          0.41.0
simpful                       2.10.0
simplejson                    3.19.1
simpy                         4.0.1
six                           1.16.0
sklearn-json                  0.1.0
skope-rules                   1.0.1
sktime                        0.16.1
slicer                        0.0.7
smart-open                    6.3.0
sniffio                       1.3.0
snowballstemmer               2.2.0
sortedcontainers              2.4.0
soupsieve                     2.4.1
spacy                         3.5.0
spacy-legacy                  3.0.12
spacy-loggers                 1.0.4
Sphinx                        6.1.3
sphinxcontrib-applehelp       1.0.4
sphinxcontrib-devhelp         1.0.2
sphinxcontrib-htmlhelp        2.0.1
sphinxcontrib-jsmath          1.0.1
sphinxcontrib-qthelp          1.0.3
sphinxcontrib-serializinghtml 1.1.5
SQLAlchemy                    1.4.47
sqlglot                       11.5.5
srsly                         2.4.6
ssm                           0.0.1
stable-baselines3             1.7.0
stack-data                    0.6.2
statsforecast                 1.5.0
statsmodels                   0.13.5
stellargraph                  1.2.1
stevedore                     5.0.0
stochastic                    0.7.0
stockstats                    0.5.2
stumpy                        1.11.1
symengine                     0.10.0
sympy                         1.11.1
ta                            0.10.2
TA-Lib                        0.4.18
tables                        3.8.0
tabulate                      0.8.10
tadasets                      0.0.4
tbats                         1.1.3
tblib                         1.7.0
tenacity                      8.2.2
tensorboard                   2.11.2
tensorboard-data-server       0.6.1
tensorboard-plugin-wit        1.8.1
tensorboardX                  2.6
tensorflow                    2.11.0
tensorflow-addons             0.19.0
tensorflow-decision-forests   1.2.0
tensorflow-estimator          2.11.0
tensorflow-hub                0.13.0
tensorflow-io-gcs-filesystem  0.32.0
tensorflow-probability        0.19.0
tensorflow-text               2.11.0
tensorly                      0.8.0
tensortrade                   1.0.3
termcolor                     2.2.0
terminado                     0.17.1
tf2jax                        0.3.3
thinc                         8.1.9
threadpoolctl                 3.1.0
thriftpy2                     0.4.16
thundergbm                    0.3.17
tick                          0.7.0.1
tifffile                      2023.4.12
tinycss2                      1.2.1
toml                          0.10.2
tomli                         2.0.1
toolz                         0.12.0
torch                         1.13.1
torch-cluster                 1.6.0
```

```
torch-geometric               2.2.0
torch-scatter                 2.1.0
torch-sparse                  0.6.16
torch-spline-conv             1.2.1
torchmetrics                  0.9.3
torchvision                   0.14.1
tornado                       6.3
tqdm                          4.64.1
trace-updater                 0.0.9.1
trading-calendars             2.1.1
traitlets                     5.9.0
treeinterpreter               0.2.3
triad                         0.8.6
tsfresh                       0.20.0
tslearn                       0.5.3.2
tweepy                        4.10.0
typeguard                     3.0.2
typer                         0.3.2
typing_extensions             4.5.0
umap-learn                    0.5.3
urllib3                       1.26.14
virtualenv                    20.21.0
wasabi                        1.1.1
wcwidth                       0.2.6
webargs                       8.2.0
webencodings                  0.5.1
websocket-client              1.5.1
websockets                    10.4
Werkzeug                      2.2.3
wheel                         0.37.1
widgetsnbextension            4.0.7
wordcloud                     1.8.2.2
wrapt                         1.14.1
wrds                          3.1.6
wurlitzer                     3.0.3
xarray                        2023.1.0
xarray-einstats               0.5.1
xgboost                       1.7.4
xlrd                          2.0.1
XlsxWriter                    3.1.0
yarl                          1.8.2
yellowbrick                   1.5
yfinance                      0.2.18
zict                          3.0.0
zipp                          3.15.0
zope.event                    4.6
zope.interface                6.0
zstandard                     0.18.0
```

Supported Libraries for ARM64 Systems

On ARM64-based systems, the list of available libraries is a bit shorter because ARM64 is not as well supported as AMD64. The following libraries are available on ARM64-based systems:

PY

```
# Name                    Version                   Build  Channel
_openmp_mutex             4.5                       1_gnu    conda-forge
absl-py                   0.12.0                  pypi_0    pypi
alembic                   1.6.2                   pypi_0    pypi
appdirs                   1.4.4                   pypi_0    pypi
argon2-cffi               20.1.0           py36h269c3a8_2    conda-forge
async_generator           1.10                      py_0    conda-forge
attrs                     21.1.0             pyhd8ed1ab_0    conda-forge
auto-ks                   0.1                     pypi_0    pypi
autograd                  1.3                     pypi_0    pypi
backcall                  0.2.0              pyh9f0ad1d_0    conda-forge
backports                 1.0                       py_2    conda-forge
backports.functools_lru_cache 1.6.4              pyhd8ed1ab_0    conda-forge
bayesian-optimization     1.2.0                   pypi_0    pypi
beautifulsoup4            4.9.0                   pypi_0    pypi
bleach                    3.3.0              pyh44b312d_0    conda-forge
boto3                     1.17.72                 pypi_0    pypi
```

```
boto3                     1.17.72                  pypi_0    pypi
botocore                  1.20.72                  pypi_0    pypi
brotlipy                  0.7.0           py36h269c3a8_1001    conda-forge
ca-certificates           2020.12.5            h4fd8a4c_0    conda-forge
certifi                   2020.12.5         py36h704843e_1    conda-forge
cffi                      1.14.4            py36hcfd9a06_0    conda-forge
chardet                   4.0.0             py36h704843e_1    conda-forge
click                     8.0.0                    pypi_0    pypi
cliff                     3.7.0                    pypi_0    pypi
cloudpickle               1.3.0                    pypi_0    pypi
cmaes                     0.8.2                    pypi_0    pypi
cmd2                      1.5.0                    pypi_0    pypi
cmdstanpy                 0.4.0                    pypi_0    pypi
colorama                  0.4.4                    pypi_0    pypi
colorlog                  5.0.1                    pypi_0    pypi
colorlover                0.3.0                    pypi_0    pypi
conda                     4.10.1            py36h704843e_0    conda-forge
conda-package-handling    1.7.3             py36h269c3a8_0    conda-forge
contextvars               2.4                      pypi_0    pypi
copulalib                 1.1.0                    pypi_0    pypi
copulas                   0.3.3                    pypi_0    pypi
cryptography              3.4.7             py36h70ab5b5_0    conda-forge
cufflinks                 0.17.3                   pypi_0    pypi
cycler                    0.10.0                     py_2    conda-forge
cython                    0.29.17           py36h831f99a_0    conda-forge
dask                      2021.3.0                 pypi_0    pypi
dataclasses               0.8                      pypi_0    pypi
datashape                 0.5.2                    pypi_0    pypi
deap                      1.3.1                    pypi_0    pypi
decorator                 4.4.2                    pypi_0    pypi
defusedxml                0.7.1               pyhd8ed1ab_0    conda-forge
dill                      0.3.1.1                  pypi_0    pypi
distributed               2.20.0                   pypi_0    pypi
docutils                  0.14                     pypi_0    pypi
dtw-python                1.0.5                    pypi_0    pypi
dx                        0.1.2                    pypi_0    pypi
entrypoints               0.3              pyhd8ed1ab_1003    conda-forge
fasttext                  0.9.2                    pypi_0    pypi
feature-selector          1.0.0                    pypi_0    pypi
featuretools              0.14.0                   pypi_0    pypi
findiff                   0.8.5                    pypi_0    pypi
freetype                  2.10.4               hdf53a3c_1    conda-forge
frozendict                2.0.2                    pypi_0    pypi
future                    0.18.2                   pypi_0    pypi
gluonts                   0.4.3                    pypi_0    pypi
gplearn                   0.4.1                    pypi_0    pypi
greenlet                  1.1.0                    pypi_0    pypi
gym                       0.17.2                   pypi_0    pypi
h2o                       3.30.0.3                 pypi_0    pypi
heapdict                  1.0.1                    pypi_0    pypi
hmmlearn                  0.2.3                    pypi_0    pypi
holidays                  0.9.12                   pypi_0    pypi
hyperopt                  0.2.5                    pypi_0    pypi
icu                       64.2                 h4c5d2ac_1    conda-forge
idna                      2.10                pyh9f0ad1d_0    conda-forge
immutables                0.15                     pypi_0    pypi
importlib-metadata        4.0.1             py36h704843e_0    conda-forge
iniconfig                 1.1.1                    pypi_0    pypi
ipykernel                 5.5.4             py36h103942d_0    conda-forge
ipython                   7.16.1            py36h0e46ebc_2    conda-forge
ipython_genutils          0.2.0                      py_1    conda-forge
ipywidgets                7.5.1                    pypi_0    pypi
jax                       0.1.68                   pypi_0    pypi
jedi                      0.17.2            py36h704843e_1    conda-forge
jinja2                    2.11.3              pyh44b312d_0    conda-forge
jmespath                  0.10.0                   pypi_0    pypi
joblib                    1.0.1                    pypi_0    pypi
json5                     0.9.5               pyh9f0ad1d_0    conda-forge
jsonschema                3.2.0               pyhd8ed1ab_3    conda-forge
jupyter_client            6.1.12              pyhd8ed1ab_0    conda-forge
jupyter_core              4.7.1             py36h704843e_0    conda-forge
jupyterlab                2.1.2                      py_0    conda-forge
jupyterlab_pygments       0.1.2               pyh9f0ad1d_0    conda-forge
jupyterlab_server         1.2.0                      py_0    conda-forge
kiwisolver                1.3.1             py36h72e8208_1    conda-forge
ld_impl_linux-aarch64     2.35.1               h02ad14f_2    conda-forge
libblas                   3.9.0                8_openblas    conda-forge
libcblas                  3.9.0                8_openblas    conda-forge
```

```
libcblas                  3.9.0              8_openblas    conda-forge
libffi                    3.2.1           h4c5d2ac_1007    conda-forge
libgcc-ng                 9.3.0              he1ea209_19    conda-forge
libgfortran-ng            7.5.0              h6d8ffed_18    conda-forge
libgfortran4              7.5.0              h6d8ffed_18    conda-forge
libgomp                   9.3.0              he1ea209_19    conda-forge
liblapack                 3.9.0               8_openblas    conda-forge
libopenblas               0.3.12          pthreads_hb3c22a3_1    conda-forge
libpng                    1.6.37              hbd635b3_2    conda-forge
libsodium                 1.0.18              hb9de7d4_1    conda-forge
libstdcxx-ng              9.3.0              h1ed1776_19    conda-forge
lightgbm                  2.3.0                    pypi_0    pypi
mako                      1.1.4                    pypi_0    pypi
markupsafe                1.1.1            py36h269c3a8_3    conda-forge
matplotlib                3.2.1                         0    conda-forge
matplotlib-base           3.2.1            py36h0f30586_0    conda-forge
mistune                   0.8.4            py36h269c3a8_1003    conda-forge
mpi                       1.0                     openmpi    conda-forge
mplfinance                0.12.4a0                 pypi_0    pypi
msgpack                   1.0.0                    pypi_0    pypi
multipledispatch          0.6.0                    pypi_0    pypi
mxnet                     1.6.0                    pypi_0    pypi
nbclient                  0.5.3              pyhd8ed1ab_0    conda-forge
nbconvert                 6.0.7            py36h704843e_3    conda-forge
nbformat                  5.1.3              pyhd8ed1ab_0    conda-forge
ncurses                   6.2                  h7fd3ca4_4    conda-forge
nest-asyncio              1.5.1              pyhd8ed1ab_0    conda-forge
networkx                  2.5.1                    pypi_0    pypi
neural-tangents           0.2.1                    pypi_0    pypi
nltk                      3.4.5                    pypi_0    pypi
notebook                  6.3.0            py36h704843e_0    conda-forge
numpy                     1.18.1           py36h3849536_1    conda-forge
nvidia-ml-py3             7.352.0                  pypi_0    pypi
oauthlib                  3.1.0                    pypi_0    pypi
odo                       0+unknown                pypi_0    pypi
openmpi                   4.0.3                hd49bf07_1    conda-forge
openssl                   1.1.1k               hf897c2e_0    conda-forge
opt-einsum                3.3.0                    pypi_0    pypi
optuna                    2.3.0                    pypi_0    pypi
packaging                 20.9               pyh44b312d_0    conda-forge
pandas                    0.25.3           py36h59fbc97_0    conda-forge
pandas-market-calendars   1.7                      pypi_0    pypi
pandocfilters             1.4.2                      py_1    conda-forge
parso                     0.7.1              pyh9f0ad1d_0    conda-forge
pbr                       5.6.0                    pypi_0    pypi
pennylane                 0.9.0                    pypi_0    pypi
pexpect                   4.8.0              pyh9f0ad1d_2    conda-forge
pickleshare               0.7.5                 py_1003    conda-forge
pip                       21.1.1             pyhd8ed1ab_0    conda-forge
plotly                    4.7.1                    pypi_0    pypi
pluggy                    0.13.1                   pypi_0    pypi
ppscore                   0.0.2                    pypi_0    pypi
prettytable               2.1.0                    pypi_0    pypi
prometheus_client         0.10.1             pyhd8ed1ab_0    conda-forge
prompt-toolkit            3.0.18             pyha770c72_0    conda-forge
psutil                    5.8.0                    pypi_0    pypi
ptvsd                     4.3.2                    pypi_0    pypi
ptyprocess                0.7.0              pyhd3deb0d_0    conda-forge
pulp                      1.6.8                    pypi_0    pypi
py                        1.10.0                   pypi_0    pypi
pyaml                     20.4.0                   pypi_0    pypi
pybind11                  2.6.2                    pypi_0    pypi
pycosat                   0.6.3            py36h269c3a8_1006    conda-forge
pycparser                 2.20               pyh9f0ad1d_2    conda-forge
pydantic                  1.8.2                    pypi_0    pypi
pydevd-pycharm            201.8538.36              pypi_0    pypi
pyglet                    1.5.0                    pypi_0    pypi
pygments                  2.9.0              pyhd8ed1ab_0    conda-forge
pykalman                  0.9.5                    pypi_0    pypi
pyopenssl                 20.0.1             pyhd8ed1ab_0    conda-forge
pyparsing                 2.4.7              pyh9f0ad1d_0    conda-forge
pyperclip                 1.8.2                    pypi_0    pypi
pyro-api                  0.1.2                    pypi_0    pypi
pyro-ppl                  1.3.1                    pypi_0    pypi
pyrsistent                0.17.3           py36h269c3a8_2    conda-forge
pysocks                   1.7.1            py36h704843e_3    conda-forge
pytest                    6.2.4                    pypi_0    pypi
python                    3.6.7            h357f687_1008_cpython    conda-forge
```

```
python                    3.6.7           h357f687_1008_cpython    conda-forge
python-dateutil           2.8.0                    pypi_0    pypi
python-editor             1.0.4                    pypi_0    pypi
python-graphviz           0.8.4                    pypi_0    pypi
python_abi                3.6                   1_cp36m    conda-forge
pytz                      2021.1                pyhd8ed1ab_0    conda-forge
pywavelets                1.1.1                    pypi_0    pypi
pyyaml                    5.4.1                    pypi_0    pypi
pyzmq                     22.0.3               py36hfbd0944_1    conda-forge
rauth                     0.7.3                    pypi_0    pypi
readline                  8.1                   h1a49cc3_0    conda-forge
requests                  2.25.1                pyhd3deb0d_0    conda-forge
requests-oauthlib         1.3.0                    pypi_0    pypi
retrying                  1.3.3                    pypi_0    pypi
rpy2                      3.3.6                    pypi_0    pypi
ruamel_yaml               0.15.80              py36h269c3a8_1004    conda-forge
ruptures                  1.1.3                    pypi_0    pypi
scikit-learn              0.24.2                   pypi_0    pypi
scikit-learn-extra        0.2.0                    pypi_0    pypi
scikit-multiflow          0.4.1                    pypi_0    pypi
scikit-optimize           0.7.4                    pypi_0    pypi
scipy                     1.4.1                py36h3a855aa_3    conda-forge
sdeint                    0.2.1                    pypi_0    pypi
seaborn                   0.11.0                   pypi_0    pypi
semantic-version          2.6.0                    pypi_0    pypi
send2trash                1.5.0                      py_0    conda-forge
setuptools                49.6.0               py36h704843e_3    conda-forge
setuptools-git            1.2                      pypi_0    pypi
simpy                     4.0.1                    pypi_0    pypi
six                       1.16.0                pyh6c4a22f_0    conda-forge
sklearn-contrib-py-earth  0.1.0                    pypi_0    pypi
sklearn-json              0.1.0                    pypi_0    pypi
sortedcontainers          2.3.0                    pypi_0    pypi
soupsieve                 2.2.1                    pypi_0    pypi
sqlalchemy                1.4.15                   pypi_0    pypi
sqlite                    3.35.5                h43e6a2a_0    conda-forge
statistics                1.0.3.5                  pypi_0    pypi
stevedore                 3.3.0                    pypi_0    pypi
ta                        0.5.25                   pypi_0    pypi
tabulate                  0.8.9                    pypi_0    pypi
tblib                     1.7.0                    pypi_0    pypi
terminado                 0.9.4                py36h704843e_0    conda-forge
testpath                  0.4.4                      py_0    conda-forge
theano                    1.0.4                    pypi_0    pypi
threadpoolctl             2.1.0                    pypi_0    pypi
tigramite                 4.1.0                    pypi_0    pypi
tk                        8.6.10                ha99a2a3_1    conda-forge
toml                      0.10.2                   pypi_0    pypi
toolz                     0.11.1                   pypi_0    pypi
torch                     1.8.1                    pypi_0    pypi
tornado                   6.1                  py36h269c3a8_1    conda-forge
tqdm                      4.60.0                pyhd8ed1ab_0    conda-forge
trading-calendars         2.1.1                    pypi_0    pypi
traitlets                 4.3.3                py36h9f0ad1d_1    conda-forge
tweepy                    3.8.0                    pypi_0    pypi
typing_extensions         3.7.4.3                    py_0    conda-forge
tzdata                    2021a                 he74cb21_0    conda-forge
tzlocal                   2.1                      pypi_0    pypi
ujson                     1.35                     pypi_0    pypi
urllib3                   1.26.4                pyhd8ed1ab_0    conda-forge
wcwidth                   0.2.5                 pyh9f0ad1d_2    conda-forge
webencodings              0.5.1                      py_1    conda-forge
wheel                     0.36.2                pyhd3deb0d_0    conda-forge
widgetsnbextension        3.5.1                    pypi_0    pypi
wrapt                     1.12.1               py36h269c3a8_3    conda-forge
xarray                    0.15.1                   pypi_0    pypi
xz                        5.2.5                 h6dd45c4_1    conda-forge
yaml                      0.2.5                 h516909a_0    conda-forge
zeromq                    4.3.4                 h01db608_0    conda-forge
zict                      2.0.0                    pypi_0    pypi
zipp                      3.4.1                 pyhd8ed1ab_0    conda-forge
zlib                      1.2.11               h516909a_1009    conda-forge
```

Custom C# Libraries

Follow these steps to add custom libraries to your C# project:

1. Find the name of the package that you want to add on NuGet .
2. Open a terminal in the organization workspace that stores the project.
3. Run `lean library add "<projectName>" <packageName>` to add the `<packageName>` NuGet package to the project in **. / <projectName>** .

   ```
   $ lean library add "My Project" Microsoft.ML
   Retrieving latest available version from NuGet
   Adding Microsoft.ML 1.5.5 to 'My Project/My Project.csproj'
   Restoring packages in 'My Project' to provide local autocomplete
   ```

   This command installs the latest version of the `Microsoft.ML` package. If you want to use a different version you can use the `--version <value>` option. Additionally, you can pass the `--no-local` flag to skip restoring the packages locally.

Follow these steps to remove custom libraries from your C# project:

1. Open a terminal in the organization workspace that stores the project.
2. Run `lean library remove "<projectName>" <packageName>` to remove the `<packageName>` NuGet package from the project in **. / <projectName>** .

   ```
   $ lean library remove "My Project" Microsoft.ML
   Removing Microsoft.ML from 'My Project/My Project.csproj'
   Restoring packages in 'My Project'
   ```

   You can pass the `--no-local` flag to skip restoring the packages locally.

Additionally, you can also add or remove custom C# libraries by modifying the C# project file (the file ending with **.csproj**). This can be done manually or through your editor's built-in NuGet tooling if your editor has such a feature.

Custom Python Libraries

Follow these steps to add custom libraries to your Python project:

1. Find the name of the package that you want to add on PyPI .
2. Open a terminal in the organization workspace that stores the project.
3. Run `lean library add "<projectName>" <packageName>` to add the `<packageName>` PyPI package to the project in **. / <projectName>** .

```
$ lean library add "My Project" altair
Retrieving latest compatible version from PyPI
Adding altair 4.1.0 to 'My Project/requirements.txt'
Installing altair 4.1.0 in local Python environment to provide local autocomplete
```

This command installs the latest version of the `altair` package that is compatible with Python 3.8 (which is what the official LEAN Docker images use). If you want to use a different version you can use the `--version <value>` option. Additionally, you can pass the `--no-local` flag to skip installing the package in your local Python environment.

4. If you are using VS Code, restart your editor for autocomplete to start working on the new library.

Follow these steps to remove custom libraries from your Python project:

1. Open a terminal in the organization workspace that stores the project.
2. Run `lean library remove " <projectName> " <packageName>` to remove the `<packageName>` PyPI package from the project in **./ <projectName>** .

```
$ lean library remove "My Project" altair
Removing altair from 'My Project/requirements.txt'
```

Additionally, you can also add or remove custom Python libraries by modifying the project's **requirements.txt** file. If you choose to do this, make sure that the library versions that you add to this file are compatible with Python 3.6, because that's what the official LEAN Docker images use.

Libraries

Project Libraries

Introduction

Project libraries are QuantConnect projects you can merge into your project to avoid duplicating code files. If you have tools that you use across several projects, create a library.

Create Libraries

To create a library, open a terminal in one of your organization workspaces and then create a project in the **Library** directory.

```
$ lean project-create "Library/MyLibrary"
Successfully created Python project 'Library/MyLibrary'
```

The library name can only contain letters (a-z), numbers (0-9), and underscores (_). The library name can't contain spaces or start with a number.

The `lean project-create` command creates a new project based on your default programming language . To create a C# library, add the `--language csharp` option.

```
$ lean project-create "Library/MyLibrary" --language csharp
Successfully created C# project 'Library/MyLibrary'
Restoring packages in 'Library\MyLibrary' to provide local autocomplete
Restored successfully
Successfully created C# project 'Library/MyLibrary'
```

Add Libraries

Follow these steps to add a library to your project:

1. Open a terminal in your organization workspace that contains the library.
2. Run `lean library add "<projectName>" "Library/<libraryName>"` .

   ```
   $ lean library add "My Project" "Library/MyLibrary"
   Adding Lean CLI library D:\qc\lean-cli\Library\MyLibrary to project D:\qc\lean-cli\My Project
   ```

3. In your project files, import the library class at the top of the page.

   ```
   from MyLibrary.main import MyLibrary
   ```
   PY

In general, use `from <libraryName>.<fileNameWithoutExtension> import <memberName>` .

4. In your project files, instantiate the library class and call its methods.

```
x = MyLibrary()
value = x.Add(1, 2)
```

PY

Rename Libraries

Follow these steps to rename a library:

1. Open the organization workspace that contains the library.
2. In the **Library** directory, rename the library project.

   The library name can only contain letters (a-z), numbers (0-9), and underscores (_). The library name can't contain spaces or start with a number.

3. If you have a copy of the library in QuantConnect Cloud, open a terminal in your organization workspace and push the library project.

```
$ lean cloud push --project "Library/MySpecialLibrary"
[1/1] Pushing 'Library\MySpecialLibrary'
Renaming project in cloud from 'Library/MyLibrary' to 'Library/MySpecialLibrary'
Successfully updated name, files, and libraries for 'Library/MyLibrary'
```

Remove Libraries

Follow these steps to remove a library from a project, open a terminal in your organization workspace that stores the project and then run `lean library remove "<projectName>" "Library/<libraryName>"` .

```
$ lean library remove "My Project" "Library/MyLibrary"
```

Delete Libraries

To delete a library, open a terminal in your organization workspace that contains the library and then run `lean project-delete "Library/<libraryName>"` .

```
$ lean project-delete "Library/MyLibrary"
Successfully deleted project 'Library/MyLibrary'
```

Projects

Custom Docker Images

Introduction

By default, the CLI uses the official LEAN Docker images when running the LEAN engine or the research environment. However, the CLI also supports custom Docker images, making it possible to use your own version of LEAN. To make this feature easier to use, the CLI is also capable of building Docker images of your own version of LEAN using a single command.

Using Custom Images

Follow these steps to make the CLI use custom Docker images when running the LEAN engine or the research environment:

1. Open a terminal.
2. Run `lean config set engine-image <value>` , where `<value>` is the full name of your Docker image containing the LEAN engine (example: `quantconnect/lean:latest`).

```
$ lean config set engine-image quantconnect/lean:latest
Successfully updated the value of 'engine-image' to 'quantconnect/lean:latest'
```

3. Run `lean config set research-image <value>` , where `<value>` is the full name of your Docker image containing the research environment (example: `quantconnect/research:latest`).

```
$ lean config set research-image quantconnect/research:latest
Successfully updated the value of 'research-image' to 'quantconnect/research:latest'
```

Follow these steps to revert the CLI to using the default Docker images when running the LEAN engine or the research environment:

1. Open a terminal.
2. Run `lean config unset engine-image` to configure the CLI to use the default engine image instead of a custom one.

```
$ lean config unset engine-image
Successfully unset 'engine-image'
```

3. Run `lean config unset research-image` to configure the CLI to use the default research image instead of a custom one.

```
$ lean config unset research-image
Successfully unset 'research-image'
```

Building Custom Images

Follow these steps to build custom LEAN Docker images using the CLI:

1. Create a new directory that will hold the LEAN repository.
2. Clone the **QuantConnect / Lean** GitHub repository using `git` or download and extract the latest master branch archive . Save this repository to a directory called **Lean** in the directory created in step 1.
3. Make your changes to LEAN.
4. Open a terminal in the directory created in step 1.
5. Run `lean build` to build the foundation image, compile LEAN, build the engine image, and build the research image.

```
$ lean build
Building 'lean-cli/foundation:latest' from
'/home/johndoe/QuantConnect/Lean/DockerfileLeanFoundation'
Compiling the C# code in '/home/johndoe/QuantConnect/Lean'
Building 'lean-cli/engine:latest' from '/home/johndoe/QuantConnect/Lean/Dockerfile' using 'lean-
cli/foundation:latest' as base image
Building 'lean-cli/research:latest' from '/home/johndoe/QuantConnect/Lean/DockerfileJupyter'
using 'lean-cli/engine:latest' as base image
Setting default engine image to 'lean-cli/engine:latest'
Setting default research image to 'lean-cli/research:latest'
```

   After running this command the CLI uses your newly built images instead of the official ones.

By default the `lean build` command tags all custom images with `latest` . You can specify a different tag using the `--tag <value>` option.

If you haven't changed the foundation Dockerfile, the CLI automatically skips building the custom foundation image and uses the official `quantconnect/lean:foundation` image instead.

Research

Introduction

Starting local Jupyter Lab environments is a powerful feature of the Lean CLI. Jupyter Lab environments allow you to work on research notebooks locally. These environments contain the same features as QuantConnect's research environment but run locally with your own data.

Running Local Research Environment

You can run the Research Environment with your current configuration or an old configuration from QuantConnect.

Current Configuration

The default Research Environment configuration is the latest master branch of LEAN. If you set a different research image , the image you set is your current configuration. Follow these steps to start a local research environment with your current configuration:

1. Open a terminal in one of your organization workspaces .
2. Run `lean research "<projectName>"` to start a local research environment for the project in **. / <projectName>** on port `8888` .

   ```
   $ lean research "My Project"
   Starting JupyterLab, access in your browser at localhost:8888
   ```

   You can run the environment on a different port by providing the `--port <port>` option.
3. In the browser window that opens, open a research notebook.

If your configuration is set to the master branch of LEAN, the CLI automatically checks if your image is behind master every seven days. If your image falls behind master, the CLI automatically updates your image to the lastest version. To force an update before the automatic check, add the `--update` option. To avoid updates, add the `--no-update` option.

Old Configurations from QuantConnect

Follow these steps to start a local Research Environment with an old research configuration from QuantConnect:

1. View the available versions on the quantconnect/research Docker Hub tags page .
2. Copy the name of the tag you want to run.
3. Open a terminal in one of your organization workspaces .
4. Run `lean research "<projectName>" --image quantconnect/research:<tagFromStep2>` to start a local Research Environment for the project in **. / <projectName>** .

```
$ lean research "My Project" --image quantconnect/research:11154
Pulling quantconnect/research:11154...
20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
```

Opening Research Notebooks in PyCharm

Follow these steps to open a research notebook in PyCharm:

1. Start a local research environment for the project containing the notebook.
2. Open the project containing the notebook in PyCharm.
3. Open PyCharm's settings and go to **Build, Execution, Deployment > Jupyter > Jupyter Servers** .
4. Tick the radio box in front of **Configured Server** and enter **http://localhost:8888/?token=** as the Jupyter Server URL.
5. Click **Apply** in the bottom-right to save the changes and **OK** to exit settings window.
6. Open the notebook file you want to work in PyCharm's file tree.

Opening Research Notebooks in VS Code

Follow these steps to open a research notebook in VS Code:

1. Start a local research environment for the project containing the notebook.
2. Open the project containing the notebook in VS Code.
3. Open the notebook file you want to work in VS Code's file tree.
4. Open the Kernel Picker button on the top right-hand side of the notebook (or run the **Notebook: Select Notebook Kernel** command from the Command Palette by pressing **Ctrl+Shift+P**).
5. Pick **Select Another Kernel** in the list of choices that pops up.
6. Select **Existing Jupyter Server...** in the list of choices that pops up.
7. Enter **http://localhost:8888/** when asked for the URI of the running Jupyter server.
8. Select **Foundation-Py-Default** in the list of choices that pops up. This choice gives you access to a kernel for Python notebooks. For C# notebooks, select **Foundation-C#-Default** .

Retrieving Local Backtests

Sometimes it might be useful to retrieve the results of a previously ran local backtest in the research environment. By default, backtests are saved in the **<projectName> / backtests / <timestamp>** directory, which is also available in the research environment. You can use the following code snippet to read the contents of a backtest's result file into a local variable and to print its statistics (make sure to replace the path to the backtest with your own):

```py
import json

backtest_path = "backtests/2021-03-03_01-57-38/main.json"

with open(backtest_path) as file:
    data = json.load(file)

for key, value in data["Statistics"].items():
    print(f"{key}: {value}")
```

Retrieving Cloud Backtests

If you are logged in using `lean login` you can also retrieve cloud backtest results in your local research environment. If you know the name of the project and the backtest you can use the following code snippet to retrieve the backtest's results and to print its statistics:

```py
project_name = "Python Template"
backtest_name = "Adaptable Tan Frog"

project = next(p for p in api.ListProjects().Projects if p.Name == project_name)
partial_backtest = next(b for b in api.ListBacktests(project.ProjectId).Backtests if b.Name ==
backtest_name)
backtest = api.ReadBacktest(project.ProjectId, partial_backtest.BacktestId)

for key in backtest.Statistics.Keys:
    print(f"{key}: {backtest.Statistics[key]}")
```

Backtesting

Backtesting

Deployment

Introduction

Backtesting is a way to test your algorithm on historic data. The CLI makes backtesting easier by providing simple commands to backtest your algorithms locally with your own data, or in the cloud with QuantConnect's data.

Run Local Backtests

By default, local backtests run in the LEAN engine in the quantconnect/lean Docker image. This Docker image contains all the libraries available on QuantConnect , meaning your algorithm also has access to those libraries. If the specified project is a C# project, it is first compiled using the same Docker image. See Third-Party Libraries to learn how to use custom libraries and see Custom Docker Images to learn how to build and use custom Docker images.

Because algorithms run in a Docker container, `localhost` does not point to your computer's `localhost` . Substitute `localhost` with `host.docker.internal` if your algorithm needs to connect to other services running on your computer. In other words, instead of connecting to `http://localhost:<port>/` , connect to `http://host.docker.internal:<port>/` .

You can run local backtests with the regular version of the LEAN engine or a custom version.

Regular LEAN Engine

Follow these steps to run a local backtest with the latest version of LEAN engine:

1. Set up your local data for all the data required by your project.
2. Open a terminal in the organization workspace that contains the project you want to backtest.
3. Run `lean backtest "<projectName>"` to run a local backtest for the project in **. / <projectName>** .

```
$ lean backtest "My Project"
20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
Successfully ran 'My Project' in the 'backtesting' environment and stored the output in 'My
Project/backtests/2021-03-22_18-51-28'
```

4. View the result in the **<projectName> / backtests / <timestamp>** directory. Results are stored in JSON files and can be analyzed in a local research environment . You can save results to a different directory by providing the `--output <path>` option in step 3.

```
$ lean backtest "My Project" --output "My Project/custom-output"
20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
Successfully ran 'My Project' in the 'backtesting' environment and stored the output in 'My
Project/custom-output'
```

Custom LEAN Engine

Follow these steps to run a local backtest with a custom version of the LEAN engine:

1. Set up your local data for all the data required by your project.
2. View the available versions on the quantconnect/lean Docker Hub tags page .
3. Copy the name of the tag that you want to run.
4. Run `lean backtest "<projectName> --image quantconnect/lean:<tagFromStep2>"` to run a local backtest for the project in **. / <projectName>** .

```
$ lean backtest "My Project" --image quantconnect/lean:11154
Pulling quantconnect/lean:11154...
20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
```

US Equity Options Algorithms

Follow these steps to run a local US Equity Options backtest:

1. Download the US Equity Security Master dataset.

```
$ lean data download --dataset "US Equity Security Master"
```

2. Download minute resolution trade data from the US Equity dataset.

```
$ lean data download --dataset "US Equities" --data-type "Trade" --ticker "SPY" --resolution
"Minute" --start "20210101" --end "20210720"
```

3. Download minute resolution trade and quote data from the US Equity Options dataset.

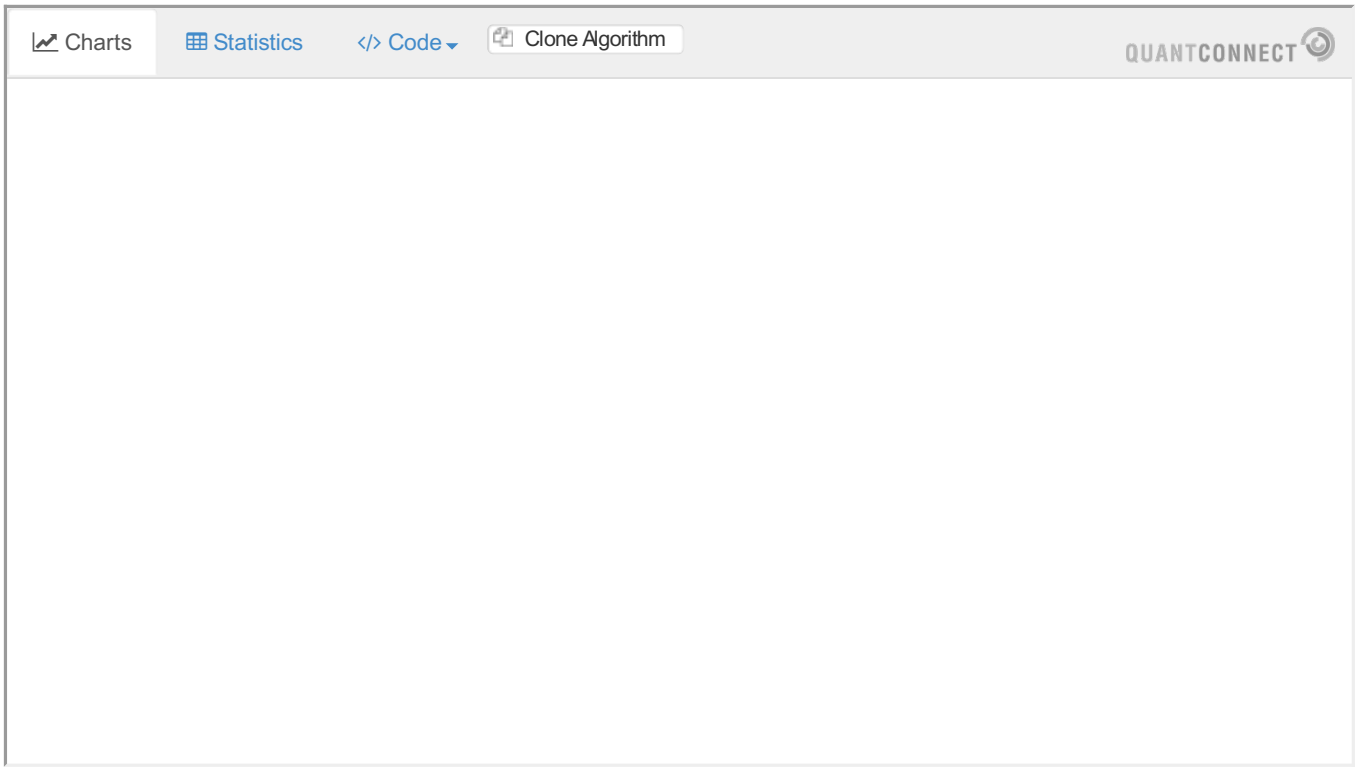
```
$ lean data download --dataset "US Equity Options" --data-type "Trade" --option-style "American"
--ticker "SPY" --resolution "Minute" --start "20210101" --end "20210720"

$ lean data download --dataset "US Equity Options" --data-type "Quote" --option-style "American"
--ticker "SPY" --resolution "Minute" --start "20210101" --end "20210720"
```

4. Create a new local project .

```
$ lean project-create --language python "<projectName>"
```

You can use the following example algorithm:



If you have the latest version of LEAN and you get different overall statistics when you run the algorithm on your local machine versus in the cloud, delete your local data files and re-download them . Some of your local files may be outdated and the preceding download commands didn't update them.

The following table shows a breakdown of the data costs for this example algorithm:

Dataset	Initial Cost (USD)	Update Cost (USD)
US Equity Security Master	$1,200/year	$1,200/year
US Equity	$7.05	$0.05/day
US Equity Options	$41.70	$0.30/day

Run Cloud Backtests

Follow these steps to run a cloud backtest:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project you want to backtest.
3. Run `lean cloud backtest "<projectName>" --push --open` to push **./ <projectName>** to the cloud, run a cloud backtest for the project, and open the results in the browser.

   ```
   $ lean cloud backtest "My Project" --push --open
   [1/1] Pushing 'My Project'
   Successfully updated cloud file 'My Project/main.py'
   Started compiling project 'My Project'
   Successfully compiled project 'My Project'
   Started backtest named 'Muscular Black Chinchilla' for project 'My Project'
   ```

4. Inspect the result in the browser, which opens automatically after the backtest finishes.

Download Datasets During Backtests

An alternative to manually downloading all required data before running the backtest is to use the `ApiDataProvider` in LEAN. This data provider automatically downloads the required data files when your backtest requests them. After it downloads a data file, it stores it in your local data directory so that in future runs, it'll not have to download it again.

Follow these steps to use the `ApiDataProvider` to automatically download data when needed:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project you want to backtest.
3. Run `lean backtest "<projectName>" --download-data` to run a local backtest for the project in **./ <projectName>** and update the Lean configuration to use the `ApiDataProvider` .

   ```
   $ lean backtest "My Project" --download-data
   20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
   DEBUG (64bit)
   20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
   Successfully ran 'My Project' in the 'backtesting' environment and stored the output in 'My
   Project/backtests/2021-03-22_18-51-28'
   ```

   Setting the `--download-data` flag updates your Lean configuration. This means that you only need to use the flag once, all future backtests will automatically use the `ApiDataProvider` .

Follow these steps to revert the Lean configuration changes so that it uses only local data again:

1. Open a terminal in your organization workspace .
2. Run `lean backtest "<projectName>" --data-provider Local` to run a local backtest for the project in **./ <projectName>** and update the Lean configuration to only use local data.

```
$ lean backtest "My Project" --data-provider Local
20210322 17:27:46.658 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 17:27:46.664 TRACE:: Engine.Main(): Started 5:27 PM
Successfully ran 'My Project' in the 'backtesting' environment and stored the output in 'My
Project/backtests/2021-03-22_18-51-28'
```

The `--data-provider` option updates your Lean configuration. This means that you only need to use the option once, all future backtests will automatically use the newly configured data provider.

By default the `ApiDataProvider` does not have a spending limit and will keep downloading files until your QuantConnect organization runs out of QuantConnect Credit (QCC). You can use the `--data-purchase-limit <value>` option to set the QCC spending limit for the backtest.

All the options documented above are also available on the `lean research` command.

Backtesting

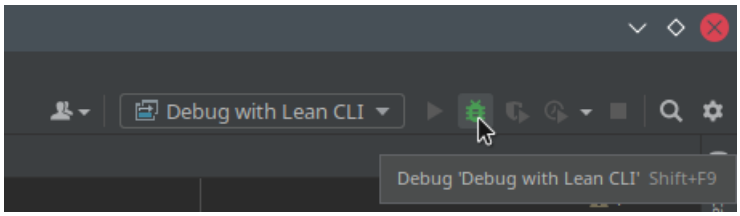
Debugging

Introduction

Debugging is an important part of writing any algorithm. The CLI makes it easy to use the builtin debugger of the most popular editors to debug LEAN algorithms. This page explains how to start local debugging for Python and C# with all editors supported by the CLI.

Python and PyCharm

Local debugging for Python in PyCharm requires PyCharm's remote debugging functionality, which is only available in PyCharm Professional. After making sure you are running the Professional edition, follow these steps to start local debugging for Python in PyCharm:

1. Follow the How to set up local autocomplete for Python in PyCharm tutorial.
2. Open the directory containing the **main.py** file in a new PyCharm window. It is important that you open the project directory itself, not your organization workspace .
3. Start debugging using the **Debug with Lean CLI** run configuration (this configuration is created when you create a new project with the CLI). 

4. Run the `lean backtest` command with the `--debug pycharm` option.

```
$ lean backtest "My Project" --debug pycharm
20210322 18:58:23.355 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 18:58:23.360 TRACE:: Engine.Main(): Started 6:58 PM
```

5. Terminate the debugger in PyCharm once LEAN has exited.

After finishing Python debugging with PyCharm, you will see a message saying "Connection to Python debugger failed". You can safely ignore this message.

Python and VS Code

Follow these steps to start local debugging for Python in VS Code:

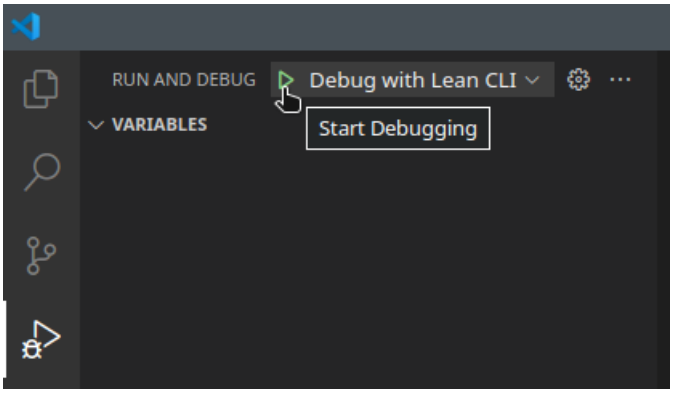
1. Follow the How to set up local autocomplete for Python in VS Code tutorial.
2. Open the directory containing the **main.py** file in a new VS Code window. It is important that you open the

project directory itself, not your organization workspace .

3. Run the `lean backtest` command with the `--debug ptvsd` option and wait until the CLI tells you to attach to the debugger.

```
$ lean backtest "My Project" --debug ptvsd
20210322 18:59:37.352 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode:
DEBUG (64bit)
20210322 18:59:37.359 TRACE:: Engine.Main(): Started 6:59 PM
20210322 18:59:39.055 TRACE:: DebuggerHelper.Initialize(): waiting for PTVSD debugger to attach
at localhost:5678...
```

4. In VS Code, open the **Run** tab and run the configuration called **Debug with Lean CLI** (this configuration is created when you create a new project with the CLI).



C# and Visual Studio

Follow these steps to start local debugging for C# in Visual Studio:

1. Follow the How to set up local autocomplete for C# in Visual Studio tutorial.
2. Open the project containing the **Main.cs** file in a new Visual Studio window. It is important that you open the project directory itself, not your organization workspace .
3. Run the `lean backtest` command with the `--debug vsdbg` option and wait until the CLI tells you to attach to the debugger.

```
$ lean backtest "My Project" --debug vsdbg
20210423 13:50:54.634 TRACE:: DebuggerHelper.Initialize(): waiting for debugger to attach...
```

4. In Visual Studio, open the process selector using **Debug > Attach to Process...** .
5. Select **Docker (Linux Container)** as the connection type.
6. Select **lean_cli_vsdbg** as connection target.
7. Double-click on the process named **dotnet** .
8. Tick the checkbox in front of **Managed (.NET Core for Unix)** and click **OK** to start debugging.

After finishing C# debugging with Visual Studio you will see a message saying "The debug adapter exited unexpectedly.". You can safely ignore this message.

C# and Rider

Follow these steps to start local debugging for C# in Rider:

1. Follow the How to set up local autocomplete for C# in Rider tutorial.
2. Open the project containing the **Main.cs** file in a new Rider window. It is important that you open the project directory itself, not your organization workspace .
3. Run the `lean backtest` command with the `--debug rider` option and wait until the CLI tells you to attach to the debugger.

```
$ lean backtest "My Project" --debug rider
20210423 13:50:54.634 TRACE:: DebuggerHelper.Initialize(): waiting for debugger to attach...
```

4. In Rider, select **Run > Attach To Remote Process...** .
5. In the pop-up that opens, select the target named **root@localhost:2222** .
6. Wait for Rider to connect and select the process named **dotnet QuantConnect.Lean.Launcher.dll** when a selector pops up to start debugging. You may have to select **Remote debugger tools are not loaded to the remote host. Click to load** first.

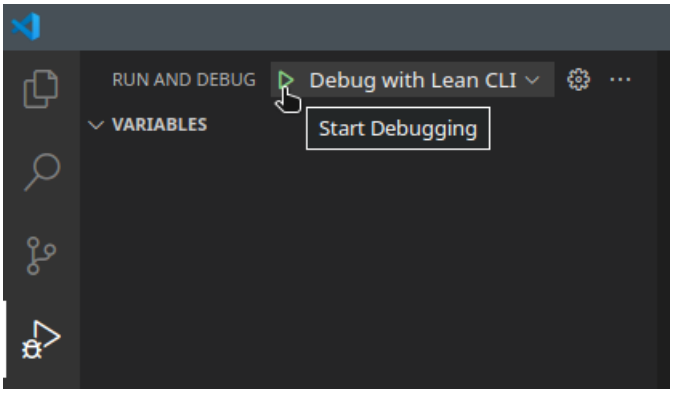
C# and VS Code

Follow these steps to start local debugging for C# in VS Code:

1. Follow the How to set up local autocomplete for C# in VS Code tutorial.
2. Open the directory containing the **Main.cs** file in a new VS Code window. It is important that you open the project directory itself, not your organization workspace .
3. Run the `lean backtest` command with the `--debug vsdbg` option and wait until the CLI tells you to attach to the debugger.

```
$ lean backtest "My Project" --debug vsdbg
20210423 13:50:54.634 TRACE:: DebuggerHelper.Initialize(): waiting for debugger to attach...
```

4. In VS Code, open the **Run** tab and run the configuration called **Debug with Lean CLI** (this configuration is created when you create a new project with the CLI).



After finishing C# debugging with VS Code you will see a message saying "The pipe program 'docker' exited

unexpectedly with code 137.". You can safely ignore this message.

Live Trading

Live Trading

Brokerages

Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

QuantConnect Paper Trading

Equities, Forex, CFD, Crypto, Futures, & Future Options

Binance

Crypto

Bitfinex

Crypto

Coinbase

Crypto

Interactive Brokers

Equities, Options, Forex, Futures, & Future Options

Kraken

Crypto

Oanda

Forex & CFD

Prime Brokerages

Equities, Forex, Crypto, Futures, & Options

Samco

India Equities

TD Ameritrade

Equities

Tradier

Equities & Options

Trading Technologies

Futures

Zerodha

India Equities

See Also

IQ Feed
Polygon

Brokerages

QuantConnect Paper Trading

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the QuantConnect Paper Trading brokerage.

Deploy Local Algorithms

Follow these steps to start local live trading with the QuantConnect Paper Trading brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

If your algorithm only uses custom data, you can select the "Custom data only" data feed option. This data feed doesn't require any brokerage credentials, but only works if your algorithm doesn't subscribe to non-custom data. Your algorithm crashes if it attempts to subscribe to non-custom data with this data feed in place, including the benchmark security. To avoid data issues with the benchmark security, either set the benchmark to the subscribed custom data or a constant.

```
self.SetBenchmark(lambda x: 0)
```
PY

4. Set your initial cash balance.

```
$ lean live "My Project"
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

5. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step

2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the QuantConnect Paper Trading brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the QuantConnect Paper Trading brokerage.
5. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

6. Configure your notification settings. You can configure any combination of email, webhook, SMS, and

Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

7. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

8. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: QuantConnect Paper TradingProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None



Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

9. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: QuantConnect Paper TradingLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Binance

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Binance or Binance US brokerage.

To view the implementation of the Binance brokerage integration, see the Lean.Brokerages.Binance repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Binance or Binance US brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter the exchange to use.

```
$ lean live "My Project"
Binance Exchange (Binance, BinanceUS): BinanceUS
```

4. Enter the environment to use.

```
$ lean live "My Project"
Use the testnet? (live, paper): live
```

5. Enter your API key and API secret.

```
$ lean live "My Project"
API key: 6d3ef5ca2d2fa52e4ee55624b0471261
API secret: ********************************
```

To create a new API key, see the API Management page on Binance or Binance US .

6. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

7. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Binance brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Binance brokerage.
5. Enter the exchange to use.

```
$ lean cloud live "My Project" --push --open
Binance Exchange (Binance, BinanceUS):
```

6. Enter the environment to use.

```
$ lean cloud live "My Project" --push --open
Environment (live, paper):
```

7. Enter your Binance API key and secret.

```
$ lean cloud live "My Project" --push --open
API key: wL1waeOC7VD447skCFeiat9pP3r1uKXfYomGg43uyCOgzl8xsI9SZsX97AXP4zWv
API secret: ************************************************************
```

   To create a new API key, see the API Management page on Binance or Binance US .

8. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

9. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

12. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: BinanceProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

14. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: BinanceLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Bitfinex

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Bitfinex brokerage.

To view the implementation of the Bitfinex brokerage integration, see the Lean.Brokerages.Bitfinex repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Bitfinex brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your API key Id and secret.

```
$ lean live "My Project"
API key: bbbMsqbxjytVM9cGvnLpKguz9rZf2T5qACxaVx7E8Mm
API secret: ****************************************
```

To create new API credentials, see the API Management page on the Bitfinex website.

4. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

5. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Bitfinex brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Bitfinex brokerage.
5. Enter your API key Id and secret.

```
$ lean cloud live "My Project" --push --open
API key: bbbMsqbxjytVM9cGvnLpKguz9rZf2T5qACxaVx7E8Mm
Secret key: ****************************************
```

To create new API credentials, see the API Management page on the Bitfinex website.

6. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

7. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

8. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

9. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

10. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

11. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: BitfinexProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

12. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: BitfinexLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Coinbase

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Coinbase brokerage.

To view the implementation of the Coinbase brokerage integration, see the Lean.Brokerages.CoinbasePro repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Coinbase brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter the environment to use.

```
$ lean live "My Project"
Use sandbox? (live, paper): live
```

4. Enter your API key, API secret, and passphrase.

```
$ lean live "My Project"
API key: 6d3ef5ca2d2fa52e4ee55624b0471261
API secret:
********************************************************************************
Passphrase: ****************
```

To create new API credentials, see the API settings page on the Coinbase website.

5. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

6. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Coinbase brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **./ <projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Coinbase brokerage.
5. Enter whether the sandbox should be used.

```
$ lean cloud live "My Project" --push --open
Use sandbox? (live, paper): live
```

6. Enter your API key, API secret, and passphrase.

```
$ lean cloud live "My Project" --push --open
API key: 6d3ef5ca2d2fa52e4ee55624b0471261
API secret:
********************************************************************************
Passphrase: ****************
```

To create new API credentials, see the API settings page on the Coinbase website.

7. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

8. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

9. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

10. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

11. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

12. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: CoinbaseProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

13. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: CoinbaseLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Interactive Brokers

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Interactive Brokers (IB) brokerage.

To view the implementation of the IB brokerage integration, see the Lean.Brokerages.InteractiveBrokers repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Interactive Brokers brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Set up IB Key Security via IBKR Mobile. For instructions, see IB Key Security via IBKR Mobile on the IB website.
4. Go back to the terminal and enter your Interactive Brokers username, account id, and password.

```
$ lean live "My Project"
Username: trader777
Account id: DU1234567
Account password: ****************
```

5. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

   If you select IQFeed, see IQFeed for set up instructions.

   If you select Polygon Data Feed, see Polygon for set up instructions.

6. Enter whether you want to enable delayed market data.

```
$ lean live "My Project"
Enable delayed market data? [yes/no]:
```

   This property configures the behavior when your algorithm attempts to subscribe to market data for which you don't have a market data subscription on Interactive Brokers. When enabled, your algorithm continues running using delayed market data. When disabled, live trading will stop and LEAN will shut down.

7. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an

environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Interactive Brokers brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Interactive Brokers brokerage.
5. Set up IB Key Security via IBKR Mobile. For instructions, see IB Key Security via IBKR Mobile on the IB website.
6. Go back to the terminal and enter your Interactive Brokers username, account id, and password.

```
$ lean cloud live "My Project" --push --open
Username: trader777
Account id: DU1234567
Account password: ****************
```

7. Enter whether you want to use the price data feed from Interactive Brokers instead of the one from QuantConnect. Enabling this feature requires you to have active Interactive Brokers market data subscriptions for all data required by your algorithm.

```
$ lean cloud live "My Project" --push --open
Do you want to use the Interactive Brokers price data feed instead of the QuantConnect price
data feed? (yes/no): y
```

8. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

9. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

12. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: Interactive BrokersProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

14. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: Interactive BrokersLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Kraken

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Kraken brokerage.

To view the implementation of the Kraken brokerage integration, see the Lean.Brokerages.Kraken repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Kraken brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your API key and API secret.

```
$ lean live "My Project"
API key:
API secret:
```

To get your API credentials, see the API Management Settings page on the Kraken website.

4. Enter your verification tier.

```
$ lean live "My Project"
Select the Verification Tier (Starter, Intermediate, Pro):
```

For more information about verification tiers, see Verification levels explained on the Kraken website.

5. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

6. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Kraken brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **./ <projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Kraken brokerage.
5. Enter your API key and API secret.

```
$ lean cloud live "My Project" --push --open
API key:
Secret key:
```

   To get your API credentials, see the API Management Settings page on the Kraken website.
6. Enter your verification tier.

```
$ lean cloud live "My Project" --push --open
Select the Verification Tier (Starter, Intermediate, Pro):
```

   For more information about verification tiers, see Verification levels explained on the Kraken website.
7. Select the live node that you want to use. If you only have one idle live trading node, it is selected

automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

8. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

9. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

10. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

11. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

12. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: KrakenProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

13. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: KrakenLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Oanda

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Oanda brokerage.

To view the implementation of the Oanda brokerage integration, see the Lean.Brokerages.OANDA repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Oanda brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter the environment to use. Enter `Trade` for fxTrade or `Practice` for fxTrade Practice.

```
$ lean live "My Project"
Environment? (Practice, Trade): Trade
```

4. Enter your OANDA account ID.

```
$ lean live "My Project"
Account id: 001-011-5838423-001
```

To get your account ID, see your Account Statement page on the OANDA website.

5. Enter your OANDA API token.

```
$ lean live "My Project"
API token: ****************
```

To create a token, see the Manage API Access page on the OANDA website.

6. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

7. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step

2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Oanda brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Oanda brokerage.
5. Enter the environment to use. Enter `Trade` for fxTrade or `Practice` for fxTrade Practice.

```
$ lean cloud live "My Project" --push --open
Environment? (Practice, Trade):
```

6. Enter your OANDA account ID.

```
$ lean cloud live "My Project" --push --open
Account id: 001-011-5838423-001
```

To get your account ID, see your Account Statement page on the OANDA website.

7. Enter your OANDA API token.

```
$ lean cloud live "My Project" --push --open
API token: ****************
```

To create a token, see the Manage API Access page on the OANDA website.

8. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

9. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

12. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: OandaProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

14. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: OandaLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Prime Brokerages

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Terminal Link brokerage.

Asset Classes

Terminal Link supports trading the following asset classes:

- Equities
- Equity Options
- Futures

You may not be able to trade all assets with Terminal Link. For example, if you live in the EU, you can't trade US ETFs. Check with your local regulators to know which assets you are allowed to trade. You may need to adjust settings in your brokerage account to live trade some assets.

Data Feeds

Terminal Link lets you source a live data feed from the Bloomberg™ Terminal for live trading. You also can use the LEAN paper trading functionality to test your strategy on Bloomberg™'s live data feed.

Orders

Terminal Link enables you to create and manage Bloomberg™ orders.

Order Types

The following table describes the available order types for each asset class that Terminal Link supports:

Order Type	Equity	Equity Options	Futures
MarketOrder	✓	✓	✓
LimitOrder	✓	✓	✓
StopMarketOrder	✓	✓	✓
StopLimitOrder	✓	✓	✓

Time In Force

Terminal Link supports the following TimeInForce instructions:

- `Day`
- `GoodTilCanceled`
- `GoodTilDate`

Get Open Orders

Terminal Link lets you access open orders .

Monitor Fills

Terminal Link allows you to monitor orders as they fill through order events .

Updates

Terminal Link doesn't support order updates .

Cancellations

Terminal Link enables you to cancel open orders .

Fees

Orders filled with Terminal Link are subject to the fees of the Bloomberg Execution Management System.

Historical Data

When LEAN taps into Bloomberg™ via Terminal Link, it can run backtests and research notebooks with rich historical data sourced from the Bloomberg™ Terminal. LEAN provides accurate slippage, spread, and transaction fee models for realistic backtesting. All models are customizable to adapt to your strategy requirements. Historical data is cached locally in an efficient format for quick backtesting in the LEAN engine. If you request intraday historical data, you can request data from within the last 6 months. Historical open interest and custom data isn't available.

Compliance

Bloomberg™ is not affiliated with QuantConnect, nor does it endorse Terminal Link. All users of the integration must hold a Bloomberg™ License to be defined as an “Entitled User.” All products must be used in accordance with

established licensing terms set by Bloomberg™. All data accessed via the Bloomberg™ Desktop API must remain on the host computer. The Bloomberg™ Terminal and the LEAN instance must be on the same computer. The Bloomberg™ Server API cannot be used for black-box trading. Any Bloomberg™ Server API Data usage will require soliciting permission via the Bloomberg™ Permission System. QuantConnect requires installing the LEAN GUI along with any Terminal Link subscription.

The following rules apply:

- All users of the integration must hold a Bloomberg License to be defined as an "Entitled User".
- All data accessed via the Bloomberg Desktop API must remain on the host computer. The Bloomberg Terminal and the LEAN instance must be on the same computer.
- The Bloomberg Server API cannot be used for black-box trading. Any Bloomberg Server API Data usage will require soliciting permission via the Bloomberg Permission System.

The following table shows the activities each of the Bloomberg technologies support:

Technology	Research	Backtesting	Paper Trading	Live Trading
Desktop API	✓	✓	✓	✓
B.PIPE	✓	✓	✓	✓
Server API	✓	✓	✓	-

CLI Commands

Execute the CLI commands in the following sections to interact with Terminal Link. If you need further assistance, see the CLI Reference .

Run Local Backtests

Launch local backtests with data from the Bloomberg Terminal desktop API. Lean automatically fetches the data required for your backtest.

```
$ lean backtest "<projectName>" --data-provider "Terminal Link"
```

Launch Research Notebooks

Start Jupyter Research Notebooks, tapping into the entire QuantConnect API with the data sourced from a Bloomberg Terminal.

```
$ lean research "<projectName>" --data-provider "Terminal Link"
```

Deploy Live Algorithms

Launch live trading algorithms to trade with any of the 1300+ routing destinations in the Bloomberg EMSX network.

```
$ lean live "<projectName>" --brokerage "Terminal Link" --data-feed "Terminal Link"
```

Deploy Local Algorithms

Follow these steps to start local live trading with the Terminal Link brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter the environment to use.

```
$ lean live "My Project"
Environment (Production, Beta): Production
```

4. Enter the host and port of the Bloomberg server.

```
$ lean live "My Project"
Server host: 127.0.0.1
Server port: 8194
```

5. Enter your EMSX configuration

```
$ lean live "My Project"
EMSX broker: someValue
EMSX account:
```

6. Enter your Open FIGI API key.

```
$ lean live "My Project"
Open FIGI API key:
```

7. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

   If you select IQFeed, see IQFeed for set up instructions.

   If you select Polygon Data Feed, see Polygon for set up instructions.

8. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

The CLI doesn't currently support deploying cloud algorithms with Terminal Link.

Brokerages

Samco

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Samco brokerage.

To view the implementation of the Samco brokerage integration, see the Lean.Brokerages.Samco repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Samco brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your Samco credentials.

```
$ lean live "My Project"
Client ID:
Client Password:
```

4. Enter your year of birth.

```
$ lean live "My Project"
Year of Birth:
```

5. Enter the product type.

```
$ lean live "My Project"
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

6. Enter the trading segment.

```
$ lean live "My Project"
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

7. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

8. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Samco brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Samco brokerage.
5. Enter your Samco credentials.

```
$ lean cloud live "My Project" --push --open
Client ID:
Client Password:
```

6. Enter your year of birth.

```
$ lean cloud live "My Project" --push --open
Year of Birth:
```

7. Enter the product type.

```
$ lean cloud live "My Project" --push --open
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

8. Enter the trading segment.

```
$ lean cloud live "My Project" --push --open
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

9. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

10. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

11. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

12. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

13. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

14. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: SamcoProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

15. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: SamcoLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

TD Ameritrade

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the TD Ameritrade brokerage.

To view the implementation of the TD Ameritrade brokerage integration, see the Lean.Brokerages.TDAmeritrade repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the TD Ameritrade brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your TD Ameritrade credentials.

```
$ lean live "My Project"
API key:
OAuth Access Token:
Account number:
```

To get your account credentials, see Account Types .

4. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

5. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the TD Ameritrade brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.

3. Run `lean cloud live "<projectName>" --push --open` to push **./<projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the TD Ameritrade brokerage.
5. Enter your TD Ameritrade credentials.

```
$ lean cloud live "My Project"
API key:
OAuth Access Token:
Account number:
```

To get your account credentials, see Account Types .

6. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

7. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send

for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

8. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

9. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

10. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

11. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: TD AmeritradeProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

12. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: TD AmeritradeLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Tradier

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Tradier brokerage.

To view the implementation of the Tradier brokerage integration, see the Lean.Brokerages.Tradier repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Tradier brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your Tradier account Id and access token.

```
$ lean live "My Project"
Account id: VA000001
Access token: ****************
```

To get these credentials, see your Settings/API Access page on the Tradier website.

4. Enter whether the developer sandbox should be used.

```
$ lean live "My Project"
Use the developer sandbox? (live, paper): live
```

5. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

6. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Tradier brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **./ <projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Tradier brokerage.
5. Enter your Tradier account Id and access token.

```
$ lean cloud live "My Project" --push --open
Account id: VA000001
Access token: ****************
```

   To get these credentials, see your Settings/API Access page on the Tradier website.
6. Enter whether the developer sandbox should be used.

```
$ lean cloud live "My Project" --push --open
Use the developer sandbox? (live, paper):
```

7. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

8. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

9. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

10. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

11. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

12. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: TradierProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

13. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: TradierLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Trading Technologies

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Trading Technologies brokerage.

To view the implementation of the Trading Technologies integration, see the Lean.Brokerages.TradingTechnologies repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Trading Technologies brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your Trading Technologies credentials.

```
$ lean live "My Project"
User name: john
Session password: ****************
Account name: jane
```

4. Enter the REST configuration.

```
$ lean live "My Project"
REST app key: my-rest-app-key
REST app secret: ******************
REST environment: my-environment
```

5. Enter the market data configuration.

```
$ lean live "My Project"
Market data sender comp id:
Market data target comp id:
Market data host:
Market data port:
```

6. Enter the order routing configuration.

```
$ lean live "My Project"
Order routing sender comp id:
Order routing target comp id:
Order routing host:
Order routing port:
```

7. Enter whether FIX messages must be logged.

```
$ lean live "My Project"
Log FIX messages (yes/no): yes
```

8. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

9. Set your initial cash balance.

```
$ lean live "My Project"
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

10. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Trading Technologies brokerage:

1. Log in to the CLI if you haven't done so already.

2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **./<projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Trading Technologies brokerage.
5. Enter your Trading Technologies credentials.

```
$ lean cloud live "My Project" --push --open
User name: john
Session password: ****************
Account name: jane
```

6. Enter the REST configuration.

```
$ lean cloud live "My Project" --push --open
REST app key: my-rest-app-key
REST app secret: ******************
REST environment: my-environment
```

7. Enter the order routing configuration.

```
$ lean cloud live "My Project" --push --open
Order routing sender comp id:
```

Our TT integration routes orders via the TT FIX 4.4 Connection. Contact your TT representative to set the exchange where you would like your orders sent. Your account details are not saved on QuantConnect.

8. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

9. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

12. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: Trading TechnologiesProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None


 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

13. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.

2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: Trading TechnologiesLaunched: 2021-06-09 15:10:12 UTC
```

Brokerages

Zerodha

Introduction

The Lean CLI supports local live trading with all brokerages supported by LEAN, which makes the transfer from backtesting to live trading as seamless as possible. The Lean CLI also supports starting live trading for a cloud project on any of the brokerages supported in the cloud. We recommend live trading your projects in our cloud because we provide a battle-tested, colocated infrastructure racked in Equinix, maintained by our engineers to ensure the best possible stability and uptime. This page contains instructions on how to start live trading with the Zerodha brokerage.

To view the implementation of the Zerodha brokerage integration, see the Lean.Brokerages.Zerodha repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the Zerodha brokerage:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option:
```

3. Enter your Kite Connect API key and access token.

```
$ lean live "My Project"
API key: hp9erb9ct0lqaxpm
Access token: ********************
```

4. Enter the product type.

```
$ lean live "My Project"
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

5. Enter the trading segment.

```
$ lean live "My Project"
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

6. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

If you select IQFeed, see IQFeed for set up instructions.

If you select Polygon Data Feed, see Polygon for set up instructions.

7. Enter whether you have a history API subscription.

```
$ lean live "My Project"
Do you have a history API subscription? (true, false): true
```

8. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Zerodha brokerage:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

```
$ lean cloud live "My Project" --push --open
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Trading Technologies
11) Kraken
12) TD Ameritrade
Enter an option:
```

4. Enter the number of the Zerodha brokerage.
5. Enter your Kite Connect API key and access token.

```
$ lean cloud live "My Project" --push --open
API key: hp9erb9ct0lqaxpm
Access token: ********************
```

6. Enter the product type.

```
$ lean cloud live "My Project" --push --open
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

7. Enter the trading segment.

```
$ lean cloud live "My Project" --push --open
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

8. Enter whether you have a history API subscription.

```
$ lean live "My Project"
Do you have a history API subscription? (true, false): true
```

9. Select the live node that you want to use. If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

10. Configure your notification settings. You can configure any combination of email, webhook, SMS, and Telegram notifications for order events and emitted insights. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

```
$ lean cloud live "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

11. Enable or disable automatic algorithm restarting. This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

```
$ lean cloud live "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

12. Set your initial cash balance.

```
$ lean cloud live "My Project" --push --open
Previous cash balance: [{'currency': 'USD', 'amount': 100000.0}]
Do you want to set a different initial cash balance? [y/N]: y
Setting initial cash balance...
Currency: USD
Amount: 95800
Cash balance: [{'currency': 'USD', 'amount': 95800.0}]
Do you want to add more currency? [y/N]: n
```

13. Set your initial portfolio holdings.

```
$ lean cloud live "My Project" --push --open
Do you want to set the initial portfolio holdings? [y/N]: y
Do you want to use the last portfolio holdings? [] [y/N]: n
Setting custom initial portfolio holdings...
Symbol: GOOG
Symbol ID: GOOCV VP83T1ZUHROL
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

14. Verify the configured settings and confirm them to start the live deployment in the cloud.

```
$ lean cloud live "My Project" --push --open
Brokerage: ZerodhaProject id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Data provider: QuantConnect
LEAN version: 11157
Order event notifications: Yes
Insight notifications: Yes
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None

 Initial live cash balance: [{'currency': 'USD', 'amount': 95800.0}]
 Initial live portfolio holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHROL',
'quantity': 10, 'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

15. Inspect the result in the browser, which opens automatically after the deployment starts.

Follow these steps to see the live status of a project:

1. Log in to the CLI if you haven't done so already.
2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud status "<projectName>"` to show the status of the cloud project named "<projectName>".

```
$ lean cloud status "My Project"
Project id: 1234567
Project name: My Project
Project url: https://www.quantconnect.com/project/1234567
Live status: Running
Live id: L-1234567a8901d234e5e678ddd9b0123c
Live url: https://www.quantconnect.com/project/1234567/live
Brokerage: ZerodhaLaunched: 2021-06-09 15:10:12 UTC
```

Live Trading

Data Feeds

Data feeds supply data to run your live algorithm using LEAN. You can use multiple data feeds in live trading algorithms.

IQFeed

US Equities, US Options, Forex, & Futures

Polygon

Equities, Forex, & Cryptos

See Also

Brokerages

Data Feeds

IQFeed

Introduction

Instead of using the data feed from your brokerage, you can also use IQFeed if you're on Windows. Using IQFeed requires you to have an IQFeed developer account and you need to have the IQFeed client installed locally. This tutorial demonstrates how to set up the IQ Feed data feed with the QuantConnect Paper Trading brokerage.

Deploy Local Algorithms

Follow these steps to start local live trading with the IQ Feed data feed:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter a brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option: 1
```

3. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

4. Enter the path to the IQConnect binary.

   The default path is **C: / Program Files (x86) / DTN / IQFeed / iqconnect.exe** if you used the default settings when installing the IQFeed client.

```
$ lean live "My Project"
IQConnect binary location [C:/Program Files (x86)/DTN/IQFeed/iqconnect.exe]:
```

5. Enter your IQFeed username and password.

```
$ lean live "My Project"
Username: 123456
Password: **********
```

6. If you have an IQFeed developer account, enter the product Id and version of your account.

```
$ lean live "My Project"
Product id: <yourID>
Product version: 1.0
```

7. If you don't have an IQFeed developer account, open **iqlink.exe** , log in to IQLink with your username and password, and then enter random numbers for the product id and version.

```
$ lean live "My Project"
Product id: 123
Product version: 1.0
```

8. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Supported Assets

Our IQFeed integration supports securities from the following asset classes:

- US Equity
- Forex (listed on FXCM)
- US Option
- Future

Data Feeds

Polygon

Introduction

Instead of using the data feed from your brokerage, you can also use Polygon . This tutorial demonstrates how to set up the Polygon data feed with the QuantConnect Paper Trading brokerage.

Deploy Local Algorithms

Follow these steps to start local live trading with the Polygon data feed:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter a brokerage number.

```
$ lean live "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Pro
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Atreyu
12) Trading Technologies
13) Kraken
14) TD Ameritrade
Enter an option: 1
```

3. Enter the number of the data feed to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Pro
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Kraken
11) TD Ameritrade
12) IQFeed
13) Polygon Data Feed
14) Custom data only
To enter multiple options, separate them with comma.:
```

4. Enter your Polygon API key.

```
$ lean live "My Project"
Configure credentials for Polygon Data Feed

Your Polygon data feed API Key:
```

   To get your API key, see the API Keys page on the Polygon website.

5. View the result in the **<projectName> / live / <timestamp>** directory. Results are stored in real-time in JSON format. You can save results to a different directory by providing the `--output <path>` option in step 2.

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

Supported Assets

Our Polygon integration supports securities from the following asset classes:

- Equity
- Forex
- Crypto (listed on Coinbase or Bitfinex)

Live Trading

Algorithm Control

Introduction

The algorithm control features let you adjust your algorithm while it executes live so that you can perform actions that are not written in the project files. The control features let you intervene in the execution of your algorithm and make adjustments. The control features that are available to you depend on if you deploy the algorithm on your local machine or on the QuantConnect cloud servers.

Control Local Algorithms

While your local algorithms run, you can add security subscriptions, submit orders, adjust orders, and stop their execution.

Add Security Subscriptions

You can manually create security subscriptions for your algorithm instead of calling the `Add securityType` methods in your code files. If you add security subscriptions to your algorithm, you can place manual trades without having to edit and redeploy the algorithm. To add security subscriptions, open a terminal in the organization workspace that contains the project and then run `lean live add-security "My Project"` .

```
$ lean live add-security "My Project" --ticker "SPY" --market "usa" --security-type "equity"
```

For more information about the command options, see Options .

You can't manually remove security subscriptions.

Submit Orders

In local deployments, you can manually place orders instead of calling the automated methods in your project files. You can use any order type that is supported by the brokerage that you used when deploying the algorithm. To view the supported order types of your brokerage, see the **Orders** section of your brokerage model . Some example situations where it may be helpful to place manual orders instead of stopping and redeploying the algorithm include the following:

- Your brokerage account had holdings in it before you deployed your algorithm
- Your algorithm had bugs in it that caused it to purchase the wrong security
- You want to add a hedge to your portfolio without adjusting the algorithm code
- You want to rebalance your portfolio before the rebalance date

To submit orders, open a terminal in the organization workspace that contains the project and then run

`lean live submit-order "My Project"` .

```
$ lean live submit-order "My Project" --ticker "SPY" --market "usa" --security-type "equity" --order-
type "market" --quantity 10
```

For more information about the command options, see Options .

Update Orders

To update an existing order, open a terminal in the organization workspace that contains the project and then run `lean live update-order "My Project"` .

```
$ lean live update-order "My Project" --order-id 1 --quantity 5
```

For more information about the command options, see Options .

Cancel Orders

To cancel an existing order, open a terminal in the organization workspace that contains the project and then run `lean live cancel-order "My Project"` .

```
$ lean live cancel-order "My Project" --order-id 1
```

For more information about the command options, see Options .

Liquidate Positions

To liquidate a specific asset in your algorithm, open a terminal in the organization workspace that contains the project and then run `lean live liquidate "My Project"` .

```
$ lean live liquidate "My Project" --ticker "SPY" --market "usa" --security-type "equity"
```

When you run the command, if the market is open for the asset, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market on open orders.

For more information about the command options, see Options .

Stop Algorithms

The `lean live stop` command immediately stops your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Update the settings you entered into the deployment command

- Place manual orders through your brokerage account

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean live stop "My Project"` .

```
$ lean live stop "My Project"
```

For more information about the command options, see Options .

Control Cloud Algorithms

While your cloud algorithms run, you can liquidate their positions and stop their exeuction.

Liquidate Positions

The `lean cloud live liquidate` command acts as a "kill switch" to sell all of your portfolio holdings. If your algorithm has a bug in it that caused it to purchase a lot of securities that you didn't want, this command let's you easily liquidate your portfolio instead of placing many manual trades. When you run the command, if the market is open for an asset you hold, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market on open orders. After the algorithm submits the liquidation orders, it stops executing.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean cloud live liquidate "My Project"` .

```
$ lean cloud live liquidate "My Project"
```

For more information about the command options, see Options .

Stop Algorithms

The `lean live stop` command immediately stops your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Update the settings you entered into the deployment command
- Place manual orders through your brokerage account

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean cloud live stop "My Project"` .

```
$ lean cloud live stop "My Project"
```

For more information about the command options, see Options .

Reports

Introduction

The `lean report` command in the Lean CLI is a wrapper around the LEAN Report Creator. The LEAN Report Creator is a program included with LEAN which allows you to quickly generate polished, professional-grade reports of your backtests and live trading results. We hope that you can use these reports to share your strategy performance with prospective investors.

Generate Reports

Follow these steps to generate a report of a trading algorithm:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean report` to generate a report of the most recent backtest.

   ```
   $ lean report
   20210322 20:03:48.718 TRACE:: QuantConnect.Report.Main(): Parsing source files...backtest-data-source-file.json,
   20210322 20:03:51.602 TRACE:: QuantConnect.Report.Main(): Instantiating report...
   Successfully generated report to './report.html'
   ```

   By default, the generated report is saved to **./report.html** , although you can change this by providing a custom path with the `--report-destination <path>` option. To generate a report of a backtest that is not the most recent one, you can use the `--backtest-results <path>` option to specify the path to the backtest results JSON file to generate a report for it.
3. Open the generated report in the browser and inspect its results.

You can also configure the following optional details:

Detail	Description
Strategy name	This name is displayed in the top-right corner of each page and can be configured using `--strategy-name <value>` . This value defaults to the name of the project directory.
Strategy version	This version is displayed next to the strategy name and can be configured using `--strategy-version <value>` .
Strategy description	This description is displayed underneath the "Strategy Description" header on the first page and can be configured using `--strategy-description` . This value defaults to the description stored in the project's configuration .
Live results	These results are displayed over the backtest results and can be configured using `--live-results <path>` . The provided path must point to a JSON file containing live results. For example, `--live-results "My Project/live/2022-03-17_10-53-12/L-3578882079.json"` .

Key Statistics

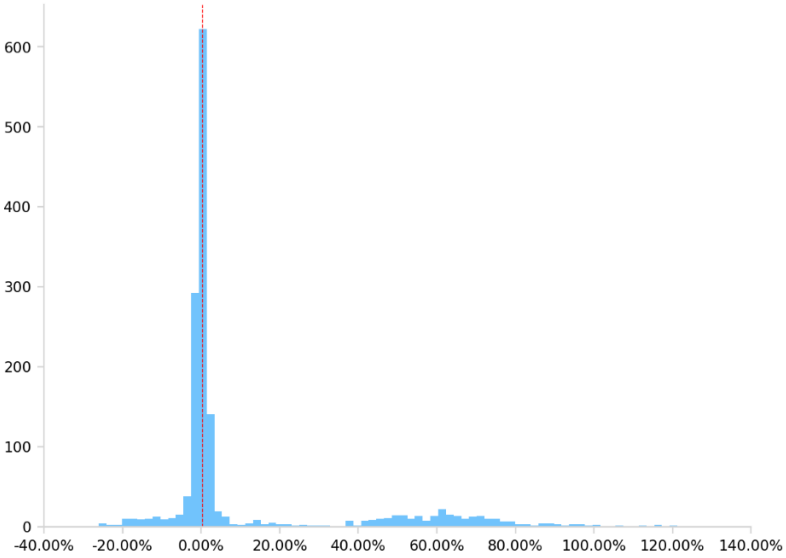
The top of the backtest report displays statistics to summarize your algorithm's performance. The following table describes the key statistics in the report:

Statistic	Description
Runtime Days	The number of days in the backtest or live trading period.
Turnover	The percentage of the algorithm's portfolio that was replaced in a given year.
CAGR	The annual percentage return that would be required to grow a portfolio from its starting value to its ending value.
Markets	The asset classes that the algorithm trades.
Trades per day	The total number of trades during the backtest divided by the number of days in the backtest. Trades per day is an approximation of the algorithm's trading frequency.
Drawdown	The largest peak to trough decline in an algorithm's equity curve.
Probabilistic SR	The probability that the estimated Sharpe ratio of an algorithm is greater than a benchmark (1).
Sharpe Ratio	A measure of the risk-adjusted return, developed by William Sharpe.
Information Ratio	The amount of excess return from the risk-free rate per unit of systematic risk.
Strategy Capacity	The maximum amount of money an algorithm can trade before its performance degrades from market impact.

Returns

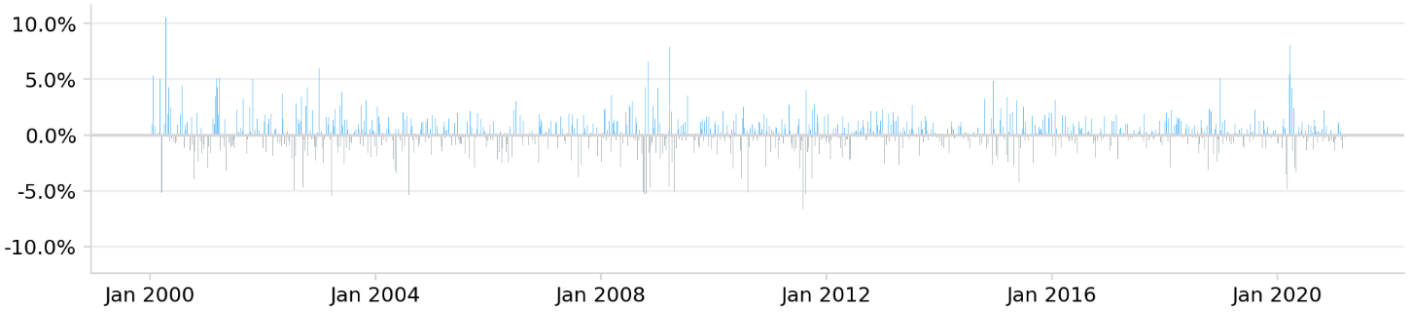
The backtest report displays charts to show the algorithm's returns per trade, per day, per month, per year, and the cumulative returns over the backtest.

Returns per Trade



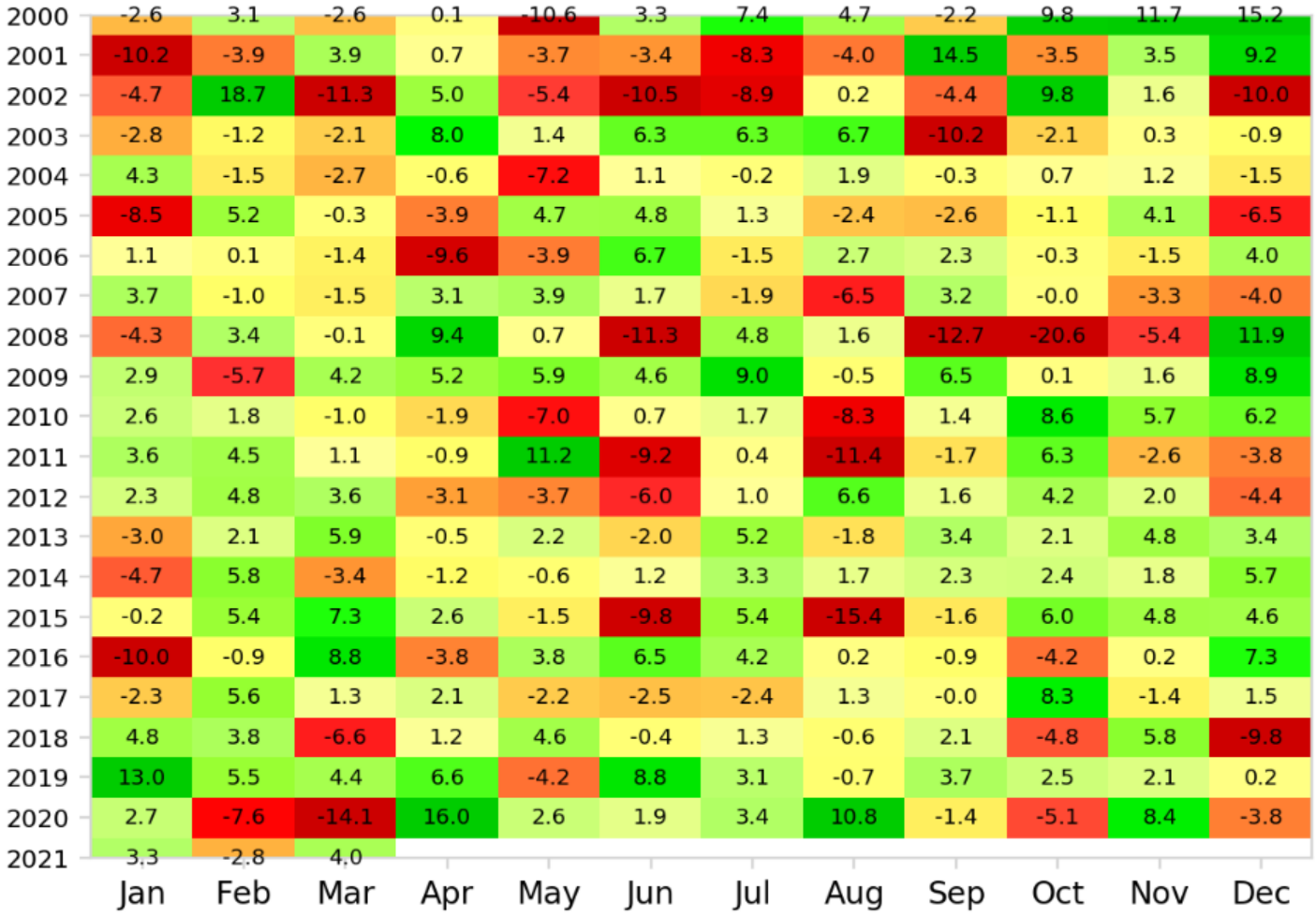
This chart displays a histogram that shows the distribution of returns per trade over the backtesting period.

Daily Returns



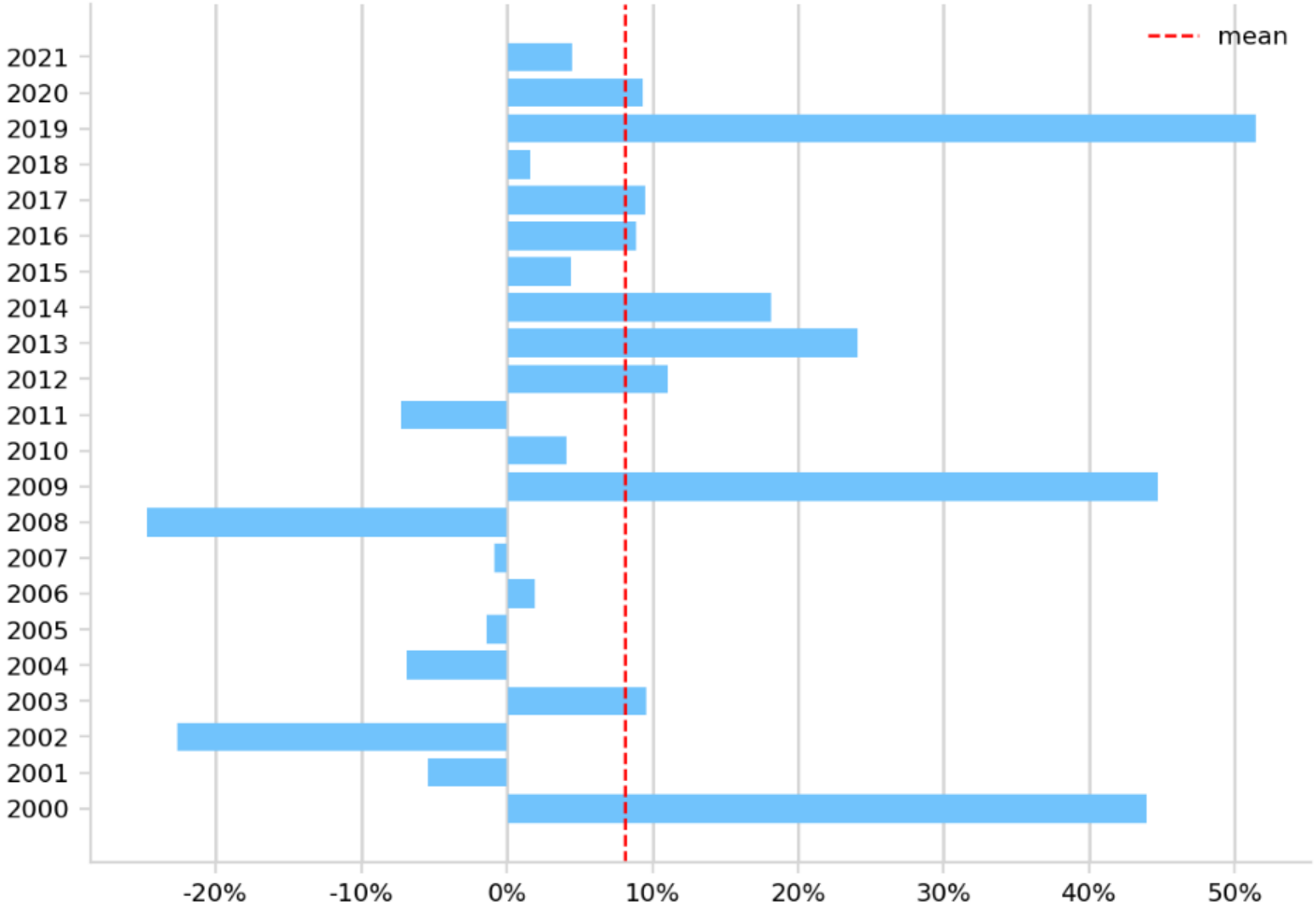
This chart displays the returns of each day. Blue bars represent profitable days and gray bars represent unprofitable days.

Monthly Returns



This chart displays the return of each month. We convert the original equity curve series into a monthly series and calculate the returns of each month. Green cells represent months with a positive return and red cells represent months with a negative return. Months that have a greater magnitude of returns are represented with darker cells. Yellow cells represent months with a relatively small gain or loss. White rectangles represent months that are not included in the backtest period. The values in the cells are percentages.

Annual Returns

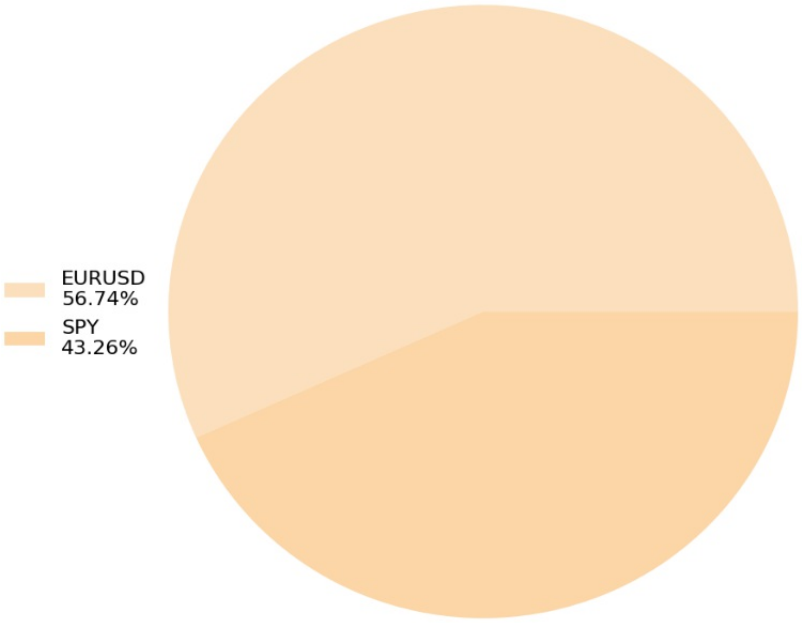


This chart displays the return of each year. We calculate the total return within each year and represent each year with a blue bar. The red dotted line represents the average of the annual returns.

Cumulative Returns

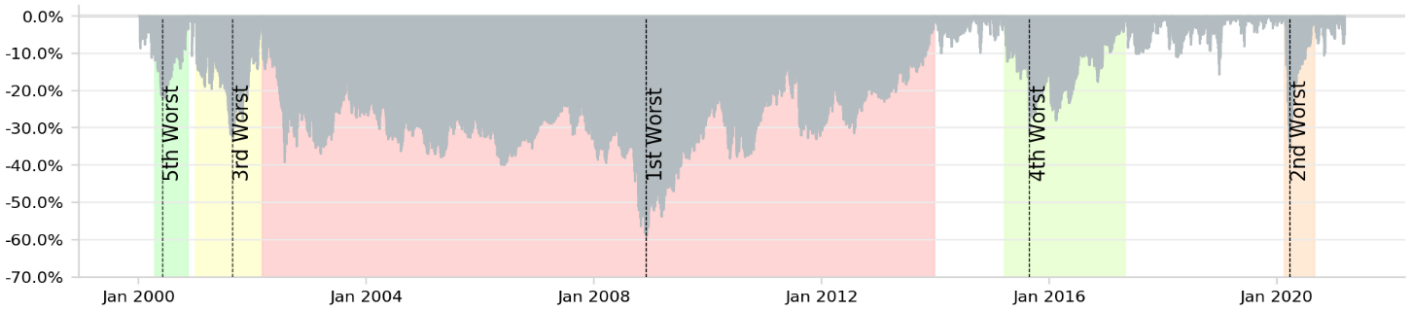
This chart displays the cumulative returns of your algorithm. The blue line represents your algorithm and the gray line represents the benchmark.

Asset Allocation



This chart displays a time-weighted average of the absolute holdings value for each asset that entered your portfolio during the backtest. When an asset has a percentage that is too small to be shown in the pie chart, it is incorporated into an "Others" category.

Drawdown

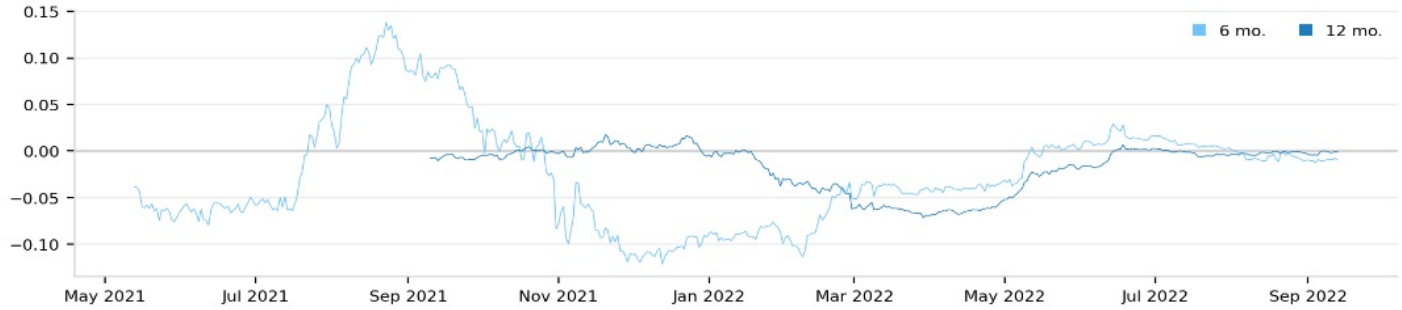


This chart displays the peak-to-trough drawdown of your portfolio's equity throughout the backtest period. The drawdown of each day is defined as the percentage loss since the maximum equity value before the current day. The drawdowns are calculated based on daily data. The top 5 drawdown periods are marked in the chart with different colors.

Rolling Statistics

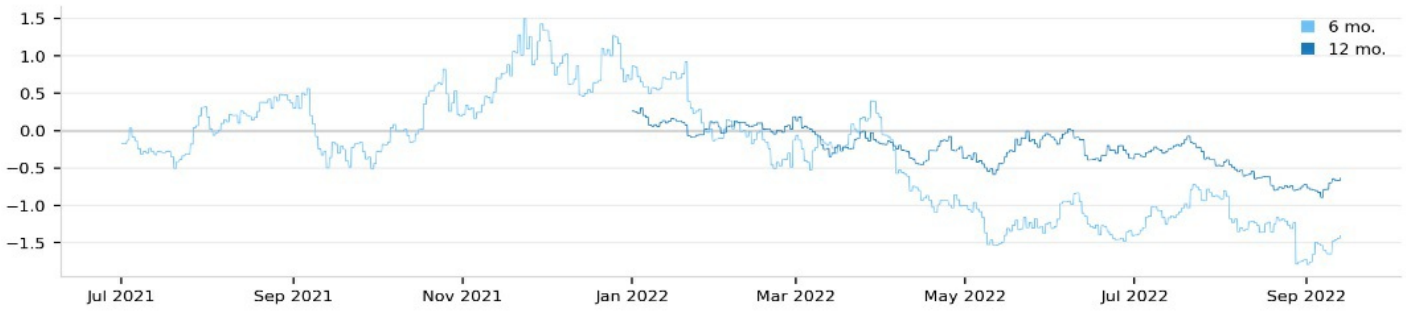
The backtest report displays time series for your portfolio's rolling beta and Sharpe ratio .

Rolling Portfolio Beta



This chart displays the rolling portfolio beta over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Rolling Sharpe Ratio

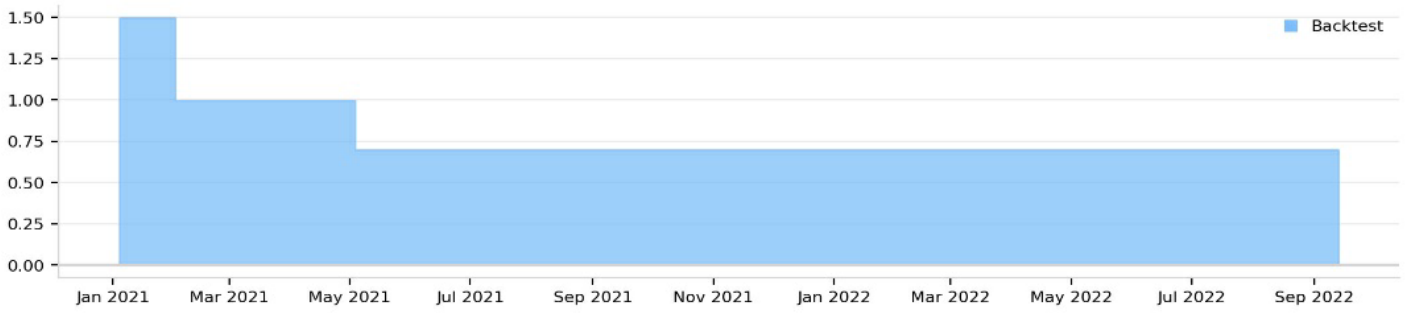


This chart displays the rolling portfolio Sharpe ratio over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Exposure

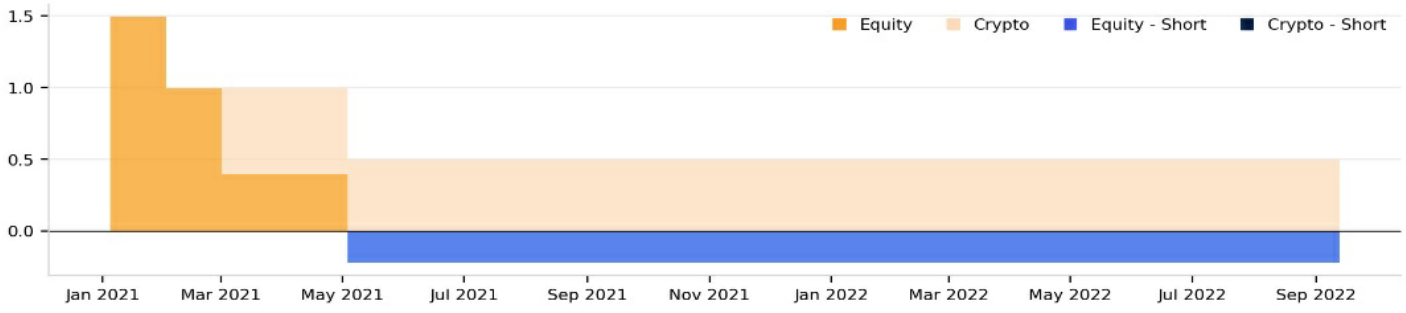
The backtest report displays time series for your portfolio's overall leverage and your portfolio's long-short exposure by asset class.

Leverage



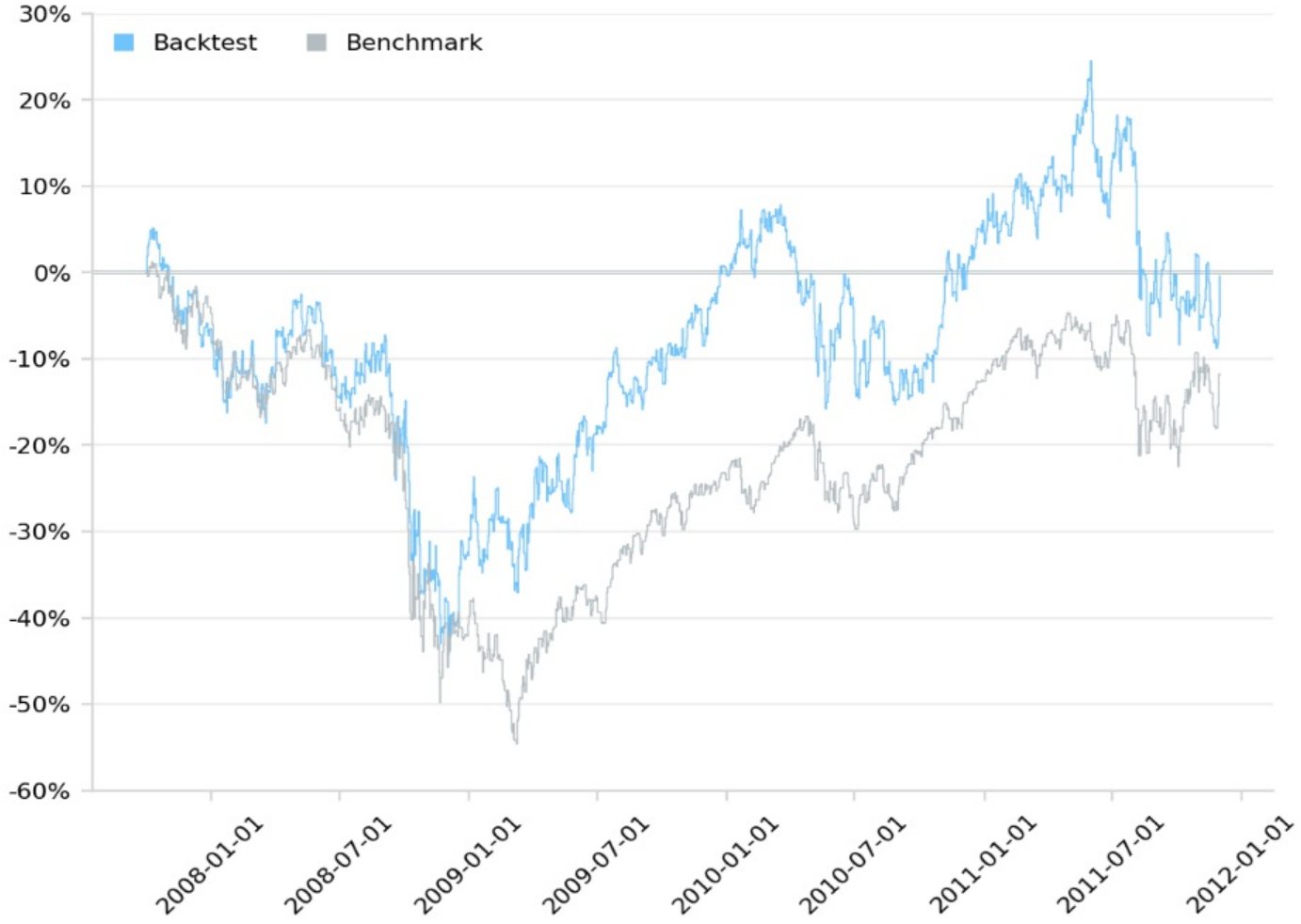
This chart displays your algorithm's utilization of leverage over time.

Long-Short Exposure By Asset Class



This chart displays your algorithm's long-short exposure by asset class over time.

Crisis Events



This set of charts displays the cumulative returns of your algorithm and the benchmark during various historical periods. The blue line represents the cumulative returns of your algorithm and the grey line represents the cumulative return of the benchmark. The report only contains the crisis event that occurred during your algorithm's backtest period. The following table shows the crisis events that may be included in your backtest report:

Crisis Name	Start Date	End Date
DotCom Bubble 2000	2/26/2000	9/10/2000
September 11, 2001	9/5/2001	10/10/2001
U.S. Housing Bubble 2003	1/1/2003	2/20/2003
Global Financial Crisis 2007	10/1/2007	12/1/2011
Flash Crash 2010	5/1/2010	5/22/2010
Fukushima Meltdown 2011	3/1/2011	4/22/2011
U.S. Credit Downgrade 2011	8/5/2011	9/1/2011
ECB IR Event 2012	9/5/2012	10/12/2012
European Debt Crisis 2014	10/1/2014	10/29/2014
Market Sell-Off 2015	8/10/2015	10/10/2015
Recovery 2010-2012	1/1/2010	10/1/2012
New Normal 2014-2019	1/1/2014	1/1/2019
COVID-19 Pandemic 2020	2/10/2020	9/20/2020

Optimization

Optimization

Parameters

Introduction

Project parameters are parameters that are defined in your project's configuration file . These parameters are a replacement for constants in your algorithm and can be optimized using one of LEAN's optimization strategies either locally or in the cloud.

Configure Project Parameters

Follow these steps to make your algorithm use project parameters instead of constant values:

1. Open your project in your preferred editor.
2. Open the project's **config.json** file.
3. Add the required parameters in the `parameters` property. All keys and values of this object must be strings. Example:

```
{
    "parameters": {
        "ema-fast": "10",
        "ema-medium": "30",
        "ema-slow": "50"
    }
}
```

4. Open your algorithm in the editor.
5. Call `QCAlgorithm.GetParameter(name)` in your algorithm to retrieve the string value of a parameter and use that instead of constant values.

PY

```
class ParameterizedAlgorithm(QCAlgorithm):
    def Initialize(self) -> None:
        self.SetStartDate(2020, 1, 1)
        self.SetCash(100000)
        self.AddEquity("SPY")

        fast_period = self.GetParameter("ema-fast", 10)
        medium_period = self.GetParameter("ema-medium", 30)
        slow_period = self.GetParameter("ema-slow", 50)

        self._fast = self.EMA("SPY", fast_period)
        self._medium = self.EMA("SPY", medium_period)
        self._slow = self.EMA("SPY", slow_period)
```

Optimization

Deployment

Introduction

The Lean CLI supports optimizing a project's parameters on your local machine or in the cloud using LEAN's powerful optimization strategies. Optimization is helpful when you want to find the best combination of parameters to minimize or maximize a certain statistic, like the algorithm's Sharpe ratio or drawdown . If your run optimizations in the cloud, you don't need your own powerful machine or data library.

Run Local Optimizations

Follow these steps to run a local optimization:

1. Set up your local data for all the data required by your project.
2. Convert your project to use project parameters instead of constants for all values that must be optimized.
3. Open a terminal in the organization workspace that contains the project.
4. Run `lean optimize "<projectName>"` to start optimizing the project in **. / <projectName>** . This command starts an interactive wizard which lets you configure the optimizer.

```
$ lean optimize "My Project"
Select the optimization strategy to use:
1) Grid Search
2) Euler Search
Enter an option:
```

5. Enter the number of the optimization strategy to use. You can either choose for Grid Search, which runs through all possible combinations of parameters, or for Euler Search, which performs an Euler-like which gradually works towards smaller optimizations.

```
$ lean optimize "My Project"
Select the optimization strategy to use:
1) Grid Search
2) Euler Search
Enter an option: 1
```

6. Enter the number of the optimization target to use. The target specifies what statistic you want to optimize and whether you want to minimize or maximize it.

```
$ lean optimize "My Project"
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
3) Compounding Annual Return (min)
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option: 2
```

7. For each parameter, enter whether you want to optimize it and what its values can be.

```
$ lean optimize "My Project"
Should the 'ema-fast' parameter be optimized? [Y/n]: y
Minimum value for 'ema-fast': 5
Maximum value for 'ema-fast': 10
Step size for 'ema-fast' [1.0]: 1
Should the 'ema-medium' parameter be optimized? [Y/n]: y
Minimum value for 'ema-medium': 25
Maximum value for 'ema-medium': 30
Step size for 'ema-medium' [1.0]: 1
Should the 'ema-slow' parameter be optimized? [Y/n]: y
Minimum value for 'ema-slow': 45
Maximum value for 'ema-slow': 50
Step size for 'ema-slow' [1.0]: 1
```

8. Enter the constraints of the optimization. An example optimization is "Drawdown <= 0.25", which discards all parameter combinations resulting in a drawdown higher than 25%.

```
$ lean optimize "My Project"
Current constraints: None
Do you want to add a constraint? [y/N]: y
Select a constraint target:
1) Sharpe Ratio
2) Compounding Annual Return
3) Probabilistic Sharpe Ratio
4) Drawdown
Enter an option: 4
Select a constraint operator (<value> will be asked after this):
1) Less than <value>
2) Less than or equal to <value>
3) Greater than <value>
4) Greater than or equal to <value>
5) Equal to <value>
6) Not equal to <value>
Enter an option: 2
Set the <value> for the selected operator: 0.25
Current constraints: TotalPerformance.PortfolioStatistics.Drawdown <= 0.25
Do you want to add a constraint? [y/N]: n
```

After configuring the constraints the optimizer starts running.

9. View the results in the terminal after the optimizer finished. The logs contains the optimal parameter combination.

```
$ lean optimize "My Project"
20220223 18:26:20.000 TRACE:: Program.Main(): Exiting Lean...
20220223 18:26:20.079 TRACE:: LeanOptimizer.TriggerOnEndEvent(OID 2313bbba-9c71-4b9e-8b91-
c70cc117b0c7): Optimization has ended. Result for Target: ['TotalPerformance'].
['PortfolioStatistics'].['SharpeRatio'] at: 3.6205: was reached using ParameterSet: (ema-
slow:47,ema-medium:26,ema-fast:5) backtestId '21c30000-dc5a-4dec-b75a-da5b1796ccba'.
Constraints: (['TotalPerformance'].['PortfolioStatistics'].['Drawdown'] 'LessOrEqual' 0.25)
Optimal parameters: ema-slow: 47, ema-medium: 26, ema-fast: 5
Successfully optimized 'My Project' and stored the output in 'My Project/optimizations/2021-03-
24_00-22-15'
```

10. View the individual backtest results in the **<project> / optimizations / <timestamp>** directory. Results are stored in JSON files and can be analyzed in a local research environment . You can save results to a different directory by providing the `--output <path>` option in step 4.

```
$ lean optimize "My Project" --output "My Project/custom-output"
20220223 18:28:20.000 TRACE:: Program.Main(): Exiting Lean...
20220223 18:28:20.079 TRACE:: LeanOptimizer.TriggerOnEndEvent(OID 1ac5e638-aae0-4aa9-80d4-
02c51bb7b84d): Optimization has ended. Result for Target: ['TotalPerformance'].
['PortfolioStatistics'].['SharpeRatio'] at: 3.6205: was reached using ParameterSet: (ema-
slow:47,ema-medium:26,ema-fast:5) backtestId 'e2aa3abf-bb60-4e91-a281-59c882ada62f'.
Constraints: (['TotalPerformance'].['PortfolioStatistics'].['Drawdown'] 'LessOrEqual' 0.25)
Optimal parameters: ema-slow: 47, ema-medium: 26, ema-fast: 5
Successfully optimized 'My Project' and stored the output in 'My Project/custom-output'
```

By default, local optimizations run in the LEAN engine in the quantconnect/lean Docker image. This Docker image contains all the libraries available on QuantConnect , meaning your algorithm also has access to those libraries. If the specified project is a C# project it is first compiled using the same Docker image. See Project Libraries to learn how to use project libraries, and Custom Docker Images to learn how to build and use custom Docker images.

Run Cloud Optimizations

Follow these steps to run a cloud optimization:

1. Log in to the CLI if you haven't done so already.
2. Convert your project to use project parameters instead of constants for all values that must be optimized.
3. Open a terminal in the organization workspace that contains the project.
4. Run `lean cloud optimize "<projectName>" --push` to push **. / <projectName>** to the cloud and start optimizing the project in the cloud.

```
$ lean cloud optimize "My Project" --push
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
3) Compounding Annual Return (min)
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option:
```

5. Enter the number of the optimization target to use. The target specifies what statistic you want to optimize and whether you want to minimize or maximize it.

```
$ lean cloud optimize "My Project" --push
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
3) Compounding Annual Return (min)
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option: 2
```

6. For each parameter, enter whether you want to optimize it and what its values can be.

```
$ lean cloud optimize "My Project" --push
Should the 'ema-fast' parameter be optimized? [Y/n]: y
Minimum value for 'ema-fast': 1
Maximum value for 'ema-fast': 10
Step size for 'ema-fast' [1.0]: 1
Should the 'ema-slow' parameter be optimized? [Y/n]: y
Minimum value for 'ema-slow': 21
Maximum value for 'ema-slow': 30
Step size for 'ema-slow' [1.0]: 1
```

7. Enter the constraints of the optimization. An example optimization is "Drawdown <= 0.25", which discards all parameter combinations resulting in a drawdown higher than 25%.

```
$ lean cloud optimize "My Project" --push
Current constraints: None
Do you want to add a constraint? [y/N]: y
Select a constraint target:
1) Sharpe Ratio
2) Compounding Annual Return
3) Probabilistic Sharpe Ratio
4) Drawdown
Enter an option: 4
Select a constraint operator (<value> will be asked after this):
1) Less than <value>
2) Less than or equal to <value>
3) Greater than <value>
4) Greater than or equal to <value>
5) Equal to <value>
6) Not equal to <value>
Enter an option: 2
Set the <value> for the selected operator: 0.25
Current constraints: TotalPerformance.PortfolioStatistics.Drawdown <= 0.25
Do you want to add a constraint? [y/N]: n
```

8. Enter the number of the optimization node type to use.

```
$ lean cloud optimize "My Project" --push
Select the optimization node type:
1) O2-8 (2 cores, 8 GB RAM) @ $0.15 per hour
2) O4-12 (4 cores, 12 GB RAM) @ $0.30 per hour
3) O8-16 (8 cores, 16 GB RAM) @ $0.60 per hour
Enter an option: 2
```

9. Enter the number of nodes that should run in parallel.

```
$ lean cloud optimize "My Project" --push
How many nodes should run in parallel (1-12) [6]: 10
```

10. Confirm the given input to start the optimizer.

```
$ lean cloud optimize "My Project" --push
Estimated number of backtests: 100
Estimated batch time: 8 minutes
Estimated batch cost: $0.38
Organization balance: 173,368 QCC ($1,733.68)
Do you want to start the optimization on the selected node type? [Y/n]: y
```

11. Inspect the optimal parameter combination and the full statistics of the backtest that ran with this combination at the bottom of the logs when the optimizer has finished.

API Reference

API Reference

lean backtest

Introduction

Backtest a project locally using Docker.

```
$ lean backtest <project> [options]
```

Description

Runs a local backtest in a Docker container using the quantconnect/lean Docker image. The logs of the backtest are shown in real-time and the full results are stored in the **<project> / backtest / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to backtest. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to be the algorithm to run.

If the `--debug` option is given, this command configures the Docker container in such a way to allow debugging using your editor's debugger. The exact ways to get local debugging to work depends on your editor and language, see Debugging for more information on how to set this up.

You can use the `--data-provider` option to change where the data is retrieved. This option updates the Lean configuration file , so you don't need to use this option multiple times for the same data provider if you are not switching between them. If you use the Terminal Link data provider, you must also provide the following options:

- `--terminal-link-environment`
- `--terminal-link-server-host`
- `--terminal-link-server-port`
- `--terminal-link-openfigi-api-key`

You can use the `--download-data` flag as an alias for `--data-provider QuantConnect` and the `--data-purchase-limit` option to set the maximum amount of QuantConnect Credit (QCC) to spend during the backtest when using QuantConnect as data provider. The `--data-purchase-limit` option is not persistent.

The Docker image that's used contains the same libraries as the ones available on QuantConnect . If the selected

project is a C# project, it is compiled before starting the backtest.

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>`. The image is pulled before running the backtest if it doesn't exist locally yet or if you pass the `--update` flag.

Arguments

The `lean backtest` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to backtest.

Options

The `lean backtest` command supports the following options:

Option	Description
`--output <path>`	Directory to store results in (defaults to **<project> / backtests / <timestamp>**).
`--detach` , `-d`	Run the backtest in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--debug`	Enable a certain debugging method, see Debugging for more information.
`--data-provider <value>`	Update the Lean configuration file to retrieve data from the given provider, which must be `Local` , `QuantConnect` , or `Terminal Link` .
`--download-data`	Update the Lean configuration file to download data from the QuantConnect API, alias for `--data-provider QuantConnect` .
`--data-purchase-limit <value>`	The maximum amount of QCC to spend on downloading data during the backtest when using QuantConnect as data provider.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.

Option	Description
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--python-venv <value>`	The path of the python virtual environment to use.
`--update`	Pull the LEAN engine image before running the backtest.
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--release`	Compile C# projects in release configuration instead of debug.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean backtest` command and exit.

API Reference

lean build

Introduction

Build Docker images of your own version of LEAN.

```
$ lean build [root] [options]
```

Description

Builds local Docker images of a local version of LEAN and sets them up for local usage with the CLI. After running this command, all commands that run the LEAN engine or the research environment use your custom images. This command performs the following actions:

1. The `lean-cli/foundation:latest` image is built from **Lean / DockerfileLeanFoundation** (if you're using an AMD64-based system) or **Lean / DockerfileLeanFoundationARM** (if you're using an ARM64-based system).
2. LEAN is compiled in a Docker container using the `lean-cli/foundation:latest` image.
3. The `lean-cli/engine:latest` image is built from **Lean / Dockerfile** using `lean-cli/foundation:latest` as the base image.
4. The `lean-cli/research:latest` image is built from **Lean / DockerfileJupyter** using `lean-cli/engine:latest` as the base image.
5. The default engine image is set to `lean-cli/engine:latest` .
6. The default research image is set to `lean-cli/research:latest` .

When the foundation Dockerfile is the same as the one used for the official foundation image, step 1 is skipped and `quantconnect/lean:foundation` is used instead of `lean-cli/foundation:latest` .

Arguments

The `lean build` command expects the following arguments:

Argument	Description
`<lean>`	The path to the directory containing the LEAN repository. Defaults to the current working directory.

Options

The `lean build` command supports the following options:

Option	Description
`--tag <value>`	The tag to apply to custom images (defaults to `latest`).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean build` command and exit.

API Reference

lean cloud backtest

Introduction

Backtest a project in the cloud.

```
$ lean cloud backtest <project> [options]
```

Description

Runs a backtest for a cloud project. While running the backtest, a progress bar shows to keep you up-to-date on the status of the backtest. After running the backtest, the resulting statistics and a link to the full results on QuantConnect are logged. You can use the `--open` option to automatically open the full results in the browser after the backtest has finished.

If you have a local copy of the cloud project, you can use the `--push` option to push local modifications to the cloud before running the backtest.

Arguments

The `lean cloud backtest` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project for which to run a backtest.

Options

The `lean cloud backtest` command supports the following options:

Option	Description
`--name <value>`	The name of the backtest (a random one is generated if not specified).
`--push`	Push local modifications to the cloud before running the backtest.
`--open`	Automatically open the results in the browser when the backtest is finished.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud backtest` command and exit.

API Reference

lean cloud live

Introduction

Start live trading for a project in the cloud.

```
$ lean cloud live <project> [options]
```

Description

Starts live trading for a cloud project. Before starting live trading, the CLI shows an interactive wizard letting you configure the brokerage, the live node, and the notifications. After starting live trading, the CLI displays a URL to the live results. You can use the `--open` flag to automatically open this URL in the browser once the deployment starts.

If you specify the `--brokerage` option, the interactive wizard is skipped and the command runs in non-interactive mode. In this mode, the command doesn't prompt for input or confirmation and reads all configuration from the provided command-line options. In non-interactive mode, all options specific to the selected brokerage become required, as well as `--node` , `--auto-restart` , `--notify-order-events` , and `--notify-insights` .

The following options are required for each brokerage in non-interactive mode:

`--brokerage`	**Required Options**
`"Paper Trading"`	N/A
`"Binance"`	`--binance-exchange-name`
	`--binance-api-key` or `--binanceus-api-key`
	`--binance-api-secret` or `--binanceus-api-secret`
	`--binance-use-testnet`
`"Bitfinex"`	`--bitfinex-api-key`
	`--bitfinex-api-secret`
	`--gdax-api-key`
	`--gdax-api-secret`

"Coinbase Pro" --brokerage	**Required Options**
	--gdax-passphrase
	--gdax-use-sandbox
"Interactive Brokers"	--ib-user-name
	--ib-account
	--ib-password
	--ib-data-feed
"Kraken"	--kraken-api-key
	--kraken-api-secret
	--kraken-verification-tier
"OANDA"	--oanda-account-id
	--oanda-access-token
	--oanda-environment
"Samco"	--samco-client-id
	--samco-client-password
	--samco-year-of-birth
	--samco-product-type
	--samco-trading-segment
"TDAmeritrade"	--tdameritrade-api-key
	--tdameritrade-access-token
	--tdameritrade-account-number
"Tradier"	--tradier-account-id
	--tradier-access-token
	--tradier-environment
	--tt-user-name

`--brokerage`	**Required Options**
`"Trading Technologies"`	`--tt-session-password`
	`--tt-account-name`
	`--tt-rest-app-key`
	`--tt-rest-app-secret`
	`--tt-rest-environment`
	`--tt-order-routing-sender-comp-id`
`"Zerodha"`	`--zerodha-api-key`
	`--zerodha-access-token`
	`--zerodha-product-type`
	`--zerodha-trading-segment`
	`--zerodha-history-subscription`

If you omit some of the required brokerage options when running in non-interactive mode, the CLI uses the option values in your LEAN configuration file .

Example non-interactive usage:

```
$ lean cloud live "My Project" \
    --brokerage "Paper Trading" \
    --node "My Node" \
    --auto-restart yes
    --notify-order-events no \
    --notify-insights no \
    --push \
    --open
```

If you have a local copy of the cloud project, you can use the `--push` option to push local modifications to the cloud before starting live trading.

Arguments

The `lean cloud live` command expects the following arguments:

Argument	**Description**
`<project>`	The name or Id of the project to start live trading.

Options

The `lean cloud live` command supports the following options:

Option	Description
`--brokerage <value>`	The brokerage to use when running in non-interactive mode.
`--binance-exchange-name <value>`	`Binance` or `BinanceUS`
`--binance-api-key <value>`	Your Binance API key, which you can generate on the API Management page on the Binance website.
`--binanceus-api-key <value>`	Your Binance US API key, which you can generate on the API Management page on the Binance US website.
`--binance-api-secret <value>`	Your Binance API secret.
`--binanceus-api-secret <value>`	Your Binance US API secret.
`--binance-use-testnet <value>`	`live` to use the production environment or `paper` to use the testnet.
`--bitfinex-api-key <value>`	Your Bitfinex API key, which you can generate on the API Management page on the Bitfinex website.
`--bitfinex-api-secret <value>`	Your Bitfinex API secret.
`--gdax-api-key <value>`	Your Coinbase API key, which you can generate on the API settings page on the Coinbase website.
`--gdax-api-secret <value>`	Your Coinbase API secret.
`--gdax-passphrase <value>`	Your Coinbase API passphrase.
`--gdax-use-sandbox <value>`	`live` to use the live trading environment or `paper` to use the sandbox.
`--ib-user-name <value>`	Your Interactive Brokers username (example: trader777).
`--ib-account <value>`	Your Interactive Brokers account Id (example: DU1234567).
`--ib-password <value>`	Your Interactive Brokers password.
`--ib-data-feed <boolean>`	Whether the Interactive Brokers price data feed must be used instead of the one from QuantConnect (`yes` or `no`).
`--kraken-api-key <value>`	Your Kraken API key, which you can find on the API Management Settings page on the Kraken website.
`--kraken-api-secret <value>`	Your Kraken API secret.

Option	Description
`--kraken-verification-tier <value>`	Your Kraken verification tier (`Starter` , `Intermediate` , or `Pro`). For more information about verification tiers, see Verification levels explained on the Kraken website.
`--oanda-account-id <value>`	Your OANDA account id, which you can find on your Account Statement page on the OANDA website.
`--oanda-access-token <value>`	Your OANDA API token, which you can generate on the Manage API Access page on the OANDA website.
`--oanda-environment <value>`	`Practice` to trade on fxTrade Practice or `Trade` to trade on fxTrade.
`--samco-client-id <value>`	Your Samco account Client ID.
`--samco-client-password <value>`	Your Samco account password.
`--samco-year-of-birth <value>`	Your year of birth (YYYY) registered with Samco.
`--samco-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--samco-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--tdameritrade-api-key <value>`	Your TDAmeritrade API key.
`--tdameritrade-access-token <value>`	Your TDAmeritrade OAuth Access Token.
`--tdameritrade-account-number <value>`	Your TDAmeritrade account number.
`--tradier-account-id <value>`	Your Tradier account id, which you can find on your Settings > API Access page on the Tradier website.
`--tradier-access-token <value>`	Your Tradier access token.
`--tradier-environment <value>`	`live` to use the live environment or `paper` to use the developer sandbox.
`--tt-user-name <value>`	Your Trading Technologies username.
`--tt-session-password <value>`	Your Trading Technologies session password.
`--tt-account-name <value>`	Your Trading Technologies account name.
`--tt-rest-app-key <value>`	Your Trading Technologies REST app key.
`--tt-rest-app-secret <value>`	Your Trading Technologies REST app secret.

Option	Description
`--tt-rest-environment <value>`	The REST environment in which to run.
`--tt-order-routing-sender-comp-id <value>`	The order routing sender comp Id to use.
`--zerodha-api-key <value>`	Your Kite Connect API key.
`--zerodha-access-token <value>`	Your Zerodha access token.
`--zerodha-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--zerodha-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--zerodha-history-subscription <boolean>`	Whether you have a history API subscription for Zerodha.
`--node <value>`	The name or Id of the live node to run on.
`--auto-restart <boolean>`	Whether automatic algorithm restarting must be enabled.
`--notify-order-events <boolean>`	Whether notifications must be sent for order events.
`--notify-insights <boolean>`	Whether notifications must be sent for emitted insights.
`--notify-emails <email> <subject>`	A comma-separated list of "email:subject" pairs configuring email-notifications.
`--notify-webhooks <url> <headers>`	A comma-separated list of "url:HEADER_1=VALUE_1:HEADER_2=VALUE_2:etc" pairs configuring webhook-notifications.
`--notify-sms <value>`	A comma-separated list of phone numbers configuring SMS notifications.
`--notify-telegram <value>`	A comma-separated list of "user/group Id:token(optional)" pairs configuring telegram notifications.
`--live-cash-balance <value>`	A comma-separated list of "currency:amount" pairs that define the initial cash balance.
`--live-holdings <value>`	A comma-separated list of "symbol:symbolId:quantity:averagePrice" pairs that define the initial portfolio holdings. For example, `"GOOG:GOOCV VP83T1ZUHROL:10:50.0"` .

Option	Description
`--push`	Push local modifications to the cloud before starting live trading.
`--open`	Automatically open the live results in the browser once the deployment starts.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live` command and exit.

API Reference

lean cloud live liquidate

Introduction

Stop live trading and liquidate existing positions for a certain project.

```
$ lean cloud live liquidate <project> [options]
```

Description

Arguments

The `lean cloud live liquidate` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to liquidate.

Options

The `lean cloud live liquidate` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live liquidate` command and exit.

API Reference

lean cloud live stop

Introduction

Stop live trading a certain project without liquidating existing positions.

```
$ lean cloud live stop <project> [options]
```

Description

Arguments

The `lean cloud live stop` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to stop live trading.

Options

The `lean cloud live stop` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live stop` command and exit.

API Reference

lean cloud optimize

Introduction

Optimize a project in the cloud.

```
$ lean cloud optimize <project> [options]
```

Description

Runs an optimization for a cloud project. While running the optimization, a progress bar shows to keep you up-to-date on the status of the optimization. After running the optimization, the optimal parameters and the statistics of the backtest with the optimal parameters are logged.

By default, an interactive wizard is shown, letting you configure the target, the parameters, the constraints, the node type, and the number of parallel nodes. When `--target` is given the command runs in non-interactive mode and does not prompt for input or confirmation.

When `--target` is given, the optimizer configuration is read from the command-line options. This means the `--target` , `--target-direction` , `--parameter` , `--node` , and `--parallel-nodes` options become required. Additionally, you can also use `--constraint` to specify optimization constraints.

In non-interactive mode, the parameters can be configured using the `--parameter` option. This option takes the following values: the name of the parameter, its minimum value, its maximum value, and its step size. You can provide this option multiple times to configure multiple parameters.

In non-interactive mode, the constraints can be configured using the `--constraint` option. This option takes a "statistic operator value" string as value, where the statistic must be a path to a property in a backtest's output file, like "TotalPerformance.PortfolioStatistics.SharpeRatio". This statistic can also be shortened to "SharpeRatio" or "Sharpe Ratio", in which case, the command automatically converts it to the longer version. The value must be a number and the operator must be `<` , `>` , `<=` , `>=` , `==` , or `==` . You can provide this option multiple times to configure multiple constraints.

Example non-interactive usage:

```
$ lean cloud optimize "My Project" \
    --target "Sharpe Ratio" \
    --target-direction "max" \
    --parameter my-first-parameter 1 10 0.5 \
    --parameter my-second-parameter 20 30 5 \
    --constraint "Drawdown < 0.5" \
    --constraint "Sharpe Ratio >= 1" \
    --node O4-12 \
    --parallel-nodes 12 \
    --push
```

If you have a local copy of the cloud project, you can use the `--push` option to push local modifications to the cloud before starting the optimization.

Arguments

The `lean cloud optimize` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to optimize.

Options

The `lean cloud optimize` command supports the following options:

Option	Description
`--target <value>`	The path to the property in the backtest's output file to target in non-interactive mode, like "TotalPerformance.PortfolioStatistics.SharpeRatio". May also be a shortened version, like "SharpeRatio" or "Sharpe Ratio".
`--target-direction <value>`	`min` if the target must be minimized, `max` if it must be maximized.
`--parameter <name> <min> <max> <step>`	The 'parameter min max step' pairs configuring the parameters to optimize. May be used multiple times.
`--constraint <value>`	The 'statistic operator value' pairs configuring the constraints of the optimization. May be used multiple times.
`--node <value>`	The node to optimize on, must `O2-8` , `O4-12` , or `O8-16` .
`--parallel-nodes <value>`	The number of nodes to run in parallel.
`--name <value>`	The name of the optimization (a random one is generated if not specified).
`--push`	Push local modifications to the cloud before starting the optimization.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud optimize` command and exit.

API Reference

lean cloud pull

Introduction

Pull projects from QuantConnect to the local drive.

```
$ lean cloud pull [options]
```

Description

Pulls projects from QuantConnect to your local directory while preserving the directory structure of your projects on QuantConnect. The project's files, description, and parameters are pulled from the cloud. By default, all cloud projects are pulled from the organization that's linked to your current organization workspace . If you provide a `--project` option, you only pull a single project from the cloud.

Before pulling a cloud project, the CLI checks if the local directory with the same path already exists. If it does and the local directory is not linked to the cloud project (because of an earlier `lean cloud pull` or `lean cloud push`), the CLI skips pulling the cloud project and logs a descriptive warning message.

If you have a local copy of a project when you pull it from the cloud, local files that don't exist in the cloud are not deleted, but the configuration values of your cloud project overwrite the configuration values of the local version . If you have renamed the project in the cloud, when you pull the project from the cloud, the local project is renamed to match the name of the cloud project.

If one of your team members creates a project library , adds it to a project, and then adds you as a collaborator to the project, you can pull the project but not the library. To pull the library as well, your team member must add you as a collaborator on the library project.

Options

The `lean cloud pull` command supports the following options:

Option	Description
`--project <value>`	The name or Id of the cloud project to pull (all your cloud projects in the organization if not specified).
`--pull-bootcamp`	Pull Boot Camp projects (disabled by default).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud pull` command and exit.

API Reference

lean cloud push

Introduction

Push local projects to QuantConnect.

```
$ lean cloud push [options]
```

Description

Pushes local projects to QuantConnect while preserving the directory structure of your local projects. The project's files, description, and parameters are pushed to the cloud. By default, all local projects in your current organization workspace directory are pushed. If you provide a `--project` option, you only push a single project to the cloud.

Before pushing a local project, the CLI checks if the cloud project with the same path already exists. If it does and the cloud project is not linked to the local project (because of an earlier `lean cloud pull` or `lean cloud push`), the CLI adds a 1 to the end of the name of your local project and then pushes it to the cloud. If the cloud project doesn't exist yet, the CLI creates it for you and pushes the contents of the local project to the newly created cloud project.

If you have a cloud copy of a project when you push it from your local machine, files in the cloud which don't exist locally are deleted and the configuration values of your local project overwrite the configuration values of the cloud version. If you have renamed the project on your local machine or in the cloud before you push, the cloud project is renamed to match the name of the local project.

Options

The `lean cloud push` command supports the following options:

Option	Description
`--project <path>`	Path to the local project to push (all local projects in your current organization workspace if not specified).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud push` command and exit.

API Reference

lean cloud status

Introduction

Show the live trading status of a project in the cloud.

```
$ lean cloud status <project> [options]
```

Description

Shows the live status of a cloud project. Displays the project id, project name, project URL, and live status. If the project has been deployed before, it also shows the deployment id, results URL, brokerage, and when it launched. If the project has been stopped, it also shows when it was stopped and the error that caused it, if there is one.

Arguments

The `lean cloud status` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the cloud project for which to show the status.

Options

The `lean cloud status` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud status` command and exit.

API Reference

lean config get

Introduction

Get the current value of a configurable option.

```
$ lean config get <key> [options]
```

Description

Prints the value of the requested option or aborts if the value is not set.

This command doesn't print the values of credentials for security reasons. Open the **credentials** file in your global configuration directory to see your stored credentials.

Arguments

The `lean config get` command expects the following arguments:

Argument	Description
`<key>`	The key of the value to retrieve. Run `lean config list` to get a list of all available keys.

Options

The `lean config get` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config get` command and exit.

API Reference

lean config list

Introduction

List the configurable options and their current values.

```
$ lean config list [options]
```

Description

Displays a table containing all configurable options and their current values. Credentials are masked with asterisks for security reasons.

Options

The `lean config list` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config list` command and exit.

API Reference

lean config set

Introduction

Set a configurable option.

```
$ lean config set <key> <value> [options]
```

Description

Updates the value of a configurable option in your global configuration directory. The command aborts with a descriptive error message if the given value is invalid for the given key.

The following keys can be set:

Key	Description
`user-id`	The user Id used when making authenticated requests to the QuantConnect API.
`api-token`	The API token used when making authenticated requests to the QuantConnect API.
`default-language`	The default language used when creating new projects (must be either `python` or `csharp`).
`engine-image`	The Docker image used when running the LEAN engine (`quantconnect/lean:latest` if not set).
`research-image`	The Docker image used when running the research environment (`quantconnect/research:latest` if not set).

Arguments

The `lean config set` command expects the following arguments:

Argument	Description
`<key>`	The key of the option to update.
`<value>`	The new value for the option with the given key.

Options

The `lean config set` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config set` command and exit.

API Reference

lean config unset

Introduction

Unset a configurable option.

```
$ lean config unset <key> [options]
```

Description

Unsets the value of a configurable option in your global configuration directory. The command aborts with a descriptive error message if the given value is invalid for the given key.

Arguments

The `lean config unset` command expects the following arguments:

Argument	Description
`<key>`	The key of the option to unset.

Options

The `lean config unset` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config unset` command and exit.

API Reference

lean data download

Introduction

Purchase and download data from QuantConnect Datasets.

```
$ lean data download [options]
```

Description

Lets you purchase and download data from QuantConnect Datasets. This command performs the following actions:

1. An interactive wizard is shown allowing you to configure exactly which data you want to download.
2. You are asked to accept the CLI API Access and Data Agreement in your web browser.
3. You are asked to confirm the purchase one last time.
4. The CLI downloads the requested files while charging the organization that's linked to your current organization workspace.

The available datasets and their pricing can be found in QuantConnect Datasets . Data is priced on a per-file per-download basis, meaning you pay a certain number of QuantConnect Credits (QCC) for each file you download. The required QCC is deducted from your organization's QCC balance when a file is downloaded. If you force-quit the command before it downloaded all requested files, you are not charged for the files you didn't download.

When `--dataset` is given, the command runs in non-interactive mode and several steps in the preceding list are skipped. Instead, the command reads the downloading configuration from the given command-line options. In this mode, the `--dataset` option is required, as well as all options specific to the selected dataset.

Example non-interactive usage:

```
$ lean data download \
    --dataset "Bitfinex Crypto Price Data" \
    --data-type "Trade" \
    --ticker "BTCUSDT" \
    --resolution "Daily" \
    --start "20201208" \
    --end "20221208"
```

In case the local data already exists, a warning is logged and you are given the choice of whether you want to enable overwriting existing data or not. Use the `--overwrite` flag to override this behavior and enable overwriting existing data in all such cases.

Options

The `lean data download` command supports the following options:

Option	Description
`--dataset <value>`	The name of the dataset to download data from in non-interactive mode.
`--overwrite`	Overwrite existing local data.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean data download` command and exit.

API Reference

lean data generate

Introduction

Generate realistic market data.

```
$ lean data generate [options]
```

Description

Runs the random data generator in the LEAN ToolBox to generate realistic market data using a Brownian motion model. Requires `--start <yyyyMMdd>` and `--symbol-count <amount>` to be set. The rest of the options have default values.

If `--end <yyyyMMdd>` isn't set, data is generated from the start date until the current date. If the `--end` option is set, data is generated between the given `--start` and `--end` values (inclusive).

By default, dense data is generated, which means the generated data contains at least one data point per resolution step. You can use `--data-density Sparse` to change this to at least one data point per 5 resolution steps, or `--data-density VerySparse` to change it to at least one data point per 50 resolution steps.

If the security type is set to `Equity` , this command will automatically generate map files, factor files, and coarse universe data. To not generate coarse universe data, set the `--include-coarse` option to `false` .

The following combinations of security types and resolutions are supported:

Security Type	Supported Resolutions				
	Tick	Second	Minute	Hour	Daily
Equity	✓	✓	✓	✓	✓
Forex	✓	✓	✓	✓	✓
CFD	✓	✓	✓	✓	✓
Future	✓	✓	✓	✓	✓
Crypto	✓	✓	✓	✓	✓
Option			✓		

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>`. The image is pulled before running the random data generator if it doesn't exist locally yet or if you pass the `--update` flag.

Options

The `lean data generate` command supports the following options:

Option	Description
`--start <yyyyMMdd>`	The inclusive start date for the data to generate in `yyyyMMdd` format. Must be at least `19980101` .
`--end <yyyyMMdd>`	The inclusive end date for the data to generate in `yyyyMMdd` format (defaults to today).
`--symbol-count <value>`	The number of symbols for which to generate data. This value is ignored if you provide the `--tickers` option.
`--tickers <value>`	Comma-separated list of tickers to use for generating data.
`--security-type <value>`	The security type for which to generate data (defaults to `Equity`). Must be `Equity` , `Forex` , `Cfd` , `Future` , `Crypto` or `Option` .
`--resolution <value>`	The resolution of the generated data (defaults to `Minute`). Must be `Tick` , `Second` , `Minute` , `Hour` or `Daily` . See the description for the supported combinations of security types and resolutions.
`--data-density <value>`	The density of the generated data (defaults to `Dense`). Must be `Dense` (at least one data point per resolution step), `Sparse` (at least one data point per 5 resolution steps), or `VerySparse` (at least one data point per 50 resolution steps).
`--include-coarse <true/false>`	Whether coarse universe data must be generated for Equity data (defaults to `true`).
`--market <market>`	The market for which to generate data. This option defaults to LEAN's default market for the requested security type.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the generator.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean data generate` command and exit.

API Reference

lean init

Introduction

Scaffold a Lean configuration file and data directory.

```
$ lean init [options]
```

Description

Fills the current directory with all the files needed to get going. It'll create a Lean configuration file and a data directory containing some sample data. To view a full list of the created files, see Directory Structure .

Options

The `lean init` command supports the following options:

Option	Description
`--organization &ltvalue>`	The name or Id of the organization to link with the organization workspace.
`--language &ltvalue>` , `-l &ltvalue>`	The default language of new projects (`python` or `csharp`).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean init` command and exit.

API Reference

lean library add

Introduction

Add a custom library to a project.

```
$ lean library add <project> <name> [options]
```

Description

Adds a third-party or project library to a project so you can use it in local backtesting, local live trading, local optimizations, and the local research environment. Additionally, this command updates your local environment so you get autocomplete on the libraries.

C# libraries are added to your C# project file (the file ending in **.csproj**). If `dotnet` is on your `PATH` and `--no-local` is not given, the CLI also restores all dependencies using `dotnet restore` to make local autocomplete work.

Third-party Python libraries are added to your project's **requirements.txt** file. If `pip` is on your `PATH` and `--no-local` is not given, the CLI also installs the Python package in your local Python environment to make local autocomplete work.

If `--version` is not given, the package is pinned to the latest compatible version. For C# projects, this is the latest available version. For Python projects, this is the latest version compatible with Python 3.8 (which is what the Docker images use).

If `--version` is given and the project is a Python project, the CLI will additionally check whether the given version is compatible with Python 3.6. If this is not the case, the command aborts because libraries incompatible with Python 3.8 cannot be installed in the official Docker images.

If you add a project library to the project, the name and path of the library is added to the project configuration file .

Arguments

The `lean library add` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory.
`<name>`	For third-party C# libraries, the name of the NuGet package to add. For third-party Python libraries, the name of the PyPI package to add. For project libraries, the path to the library in the **Library** directory of your organization workspace .

Options

The `lean library add` command supports the following options:

Option	Description
`--version <value>`	The version of the package to add to the project (defaults to the latest compatible version).
`--no-local`	Skip making changes to the local environment.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean library add` command and exit.

API Reference

lean library remove

Introduction

Remove a custom library from a project.

```
$ lean library remove <project> <name> [options]
```

Description

Removes a library from a project. This command can remove libraries that are added using `lean library add` , as well as libraries that were manually added to the C# project file or a Python project's **requirements.txt** file.

C# libraries are removed from your C# project file (the file ending in **.csproj**). If `dotnet` is on your `PATH` and `--no-local` is not given, the CLI also restores all dependencies using `dotnet restore` .

Third-party Python libraries are removed from your project's **requirements.txt** file.

Project libraries are removed from your project configuration file .

Arguments

The `lean library remove` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory.
`<name>`	For third-party C# libraries, the name of the NuGet package to remove. For third-party Python libraries, the name of the PyPI package to remove. For project libraries, the path to the library in the **Library** directory of your organization workspace .

Options

The `lean library remove` command supports the following options:

Option	Description
`--no-local`	Skip making changes to the local environment.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean library remove` command and exit.

API Reference

lean live

Introduction

Start live trading a project locally using Docker.

```
$ lean live <project> [options]
```

Description

Starts local live trading in a Docker container using the quantconnect/lean Docker image. The logs of the algorithm are shown in real-time and the full results are stored in the **<project> / live / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to backtest. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to contain the algorithm to run.

By default, an interactive wizard is shown letting you configure the brokerage and data feed to use. When `--environment` , `--brokerage` , or `--data-feed` is given, the command runs in non-interactive mode and does not prompt for input.

When the `--environment` option is given, the environment with the given name is used. The given environment must be one of the live environments stored in your Lean configuration file . This means the environment must have the `live-mode` property set to `true` .

When `--brokerage` or `--data-feed` is given, the live configuration is read from the command-line options. In case a required option has not been provided, the command falls back to the property with the same name in your Lean configuration file. The command aborts if this property also hasn't been set. The required options depend on the selected brokerage or data feed.

The following options are required for each brokerage in non-interactive mode:

`--brokerage`	**Required Options**
`"Paper Trading"`	N/A
	`--binance-exchange-name`
	`--binance-api-key` or `--binanceus-api-key`

"Binance" --brokerage	Required Options
	--binance-api-secret or --binanceus-api-secret
	--binance-use-testnet
"Bitfinex"	--bitfinex-api-key
	--bitfinex-api-secret
"Coinbase Pro"	--gdax-api-key
	--gdax-api-secret
	--gdax-passphrase
	--gdax-use-sandbox
"Interactive Brokers"	--ib-user-name
	--ib-account
	--ib-password
"Kraken"	--kraken-api-key
	--kraken-api-secret
	--kraken-verification-tier
"OANDA"	--oanda-account-id
	--oanda-access-token
	--oanda-environment
"Samco"	--samco-client-id
	--samco-client-password
	--samco-year-of-birth
	--samco-product-type
	--samco-trading-segment
"TDAmeritrade"	--tdameritrade-api-key
	--tdameritrade-access-token
	--tdameritrade-account-number

--brokerage	Required Options
"Terminal Link"	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
"Tradier"	--tradier-account-id
	--tradier-access-token
	--tradier-environment
"Trading Technologies"	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key
	--tt-rest-app-secret
	--tt-rest-environment
	--tt-market-data-sender-comp-id
	--tt-market-data-target-comp-id
	--tt-market-data-host
	--tt-market-data-port
	--tt-order-routing-sender-comp-id
	--tt-order-routing-target-comp-id
	--tt-order-routing-host
	--tt-order-routing-port
	--tt-log-fix-messages
	--zerodha-api-key
	--zerodha-access-token

"Zerodha" --brokerage	Required Options
	--zerodha-product-type
	--zerodha-trading-segment

The --data-feed option is required. The following table shows the available data feeds and their required options in non-interactive mode:

--data-feed	Required Options
"Binance"	--binance-exchange-name
	--binance-api-key or --binanceus-api-key
	--binance-api-secret or --binanceus-api-secret
"Bitfinex"	All options required by --brokerage "Bitfinex" .
"Coinbase Pro"	--gdax-api-key
	--gdax-api-secret
	--gdax-passphrase
"Custom data only"	N/A
"Interactive Brokers"	All options required by --brokerage "Interactive Brokers" .
	--ib-enable-delayed-streaming-data
"IQFeed"	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-product-name
	--iqfeed-version
"Kraken"	All options required by --brokerage "Kraken" .
"OANDA"	--oanda-account-id
	--oanda-access-token
"Polygon Data Feed"	--polygon-api-key
"Samco"	All options required by --brokerage "Samco" .

`--data-feed`	**Required Options**
`"TDAmeritrade"`	All options required by `--brokerage "TDAmeritrade"` .
`"Terminal Link"`	All options required by `--brokerage "Terminal Link"` .
`"Tradier"`	`--tradier-account-id`
	`--tradier-access-token`
`"Zerodha"`	All options required by `--brokerage "Zerodha"` .
	`--zerodha-history-subscription`

If you omit some of the required brokerage or data feed options when running in non-interactive mode, the CLI uses the option values in your LEAN configuration file .

Example non-interactive usage:

```
$ lean live "My Project" \
    --brokerage "Paper Trading" \
    --data-feed "Interactive Brokers" \
    --ib-user-name "trader777" \
    --ib-account "DU1234567" \
    --ib-password "hunter2" \
    --ib-enable-delayed-streaming-data yes
```

The Docker image that is used contains the same libraries as the ones available on QuantConnect . If the selected project is a C# project, it is compiled before starting live trading.

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>` . The image is pulled before starting the local live trading if it doesn't exist locally yet or if you pass the `--update` flag.

Arguments

The `lean live` command expects the following arguments:

Argument	**Description**
`<project>`	The path to the project directory or algorithm file to start local live trading.

Options

The `lean live` command supports the following options:

Option	**Description**

Option	Description
`--environment <value>`	The name of the environment in the Lean configuration file to use.
`--output <path>`	Directory to store results in (defaults to **<project> / live / <timestamp>**).
`--detach` , `-d`	Run the live deployment in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--brokerage <value>`	The brokerage to use when running in non-interactive mode.
`--data-feed <value>`	The data feed to use when running in non-interactive mode.
`--data-provider <value>`	Update the Lean configuration file to retrieve data from the given provider (`QuantConnect` or `Local`).
`--binance-exchange-name <value>`	`Binance` or `BinanceUS`
`--binance-api-key <value>`	Your Binance API key, which you can generate on the API Management page on the Binance website.
`--binanceus-api-key <value>`	Your Binance US API key, which you can generate on the API Management page on the Binance US website.
`--binance-api-secret <value>`	Your Binance API secret.
`--binanceus-api-secret <value>`	Your Binance US API secret.
`--binance-use-testnet <value>`	`live` to use the production environment or `paper` to use the testnet.
`--bitfinex-api-key <value>`	Your Bitfinex API key, which you can generate on the API Management page on the Bitfinex website.
`--bitfinex-api-secret <value>`	Your Bitfinex API secret.
`--gdax-api-key <value>`	Your Coinbase API key, which you can generate on the API settings page on the Coinbase website.
`--gdax-api-secret <value>`	Your Coinbase API secret.
`--gdax-passphrase <value>`	Your Coinbase API passphrase.
`--gdax-use-sandbox <value>`	`live` to use the live trading environment or `paper` to use the sandbox.

Option	Description
`--ib-user-name <value>`	Your Interactive Brokers username (example: trader777).
`--ib-account <value>`	Your Interactive Brokers account Id (example: DU1234567).
`--ib-password <value>`	Your Interactive Brokers password.
`--ib-enable-delayed-streaming-data <boolean>`	Whether delayed data may be used when your algorithm subscribes to a security for which you don't have a market data subscription.
`--iqfeed-iqconnect <path>`	The path to your IQConnect binary.
`--iqfeed-username <value>`	Your IQFeed username.
`--iqfeed-password <value>`	Your IQFeed password.
`--iqfeed-product-name <value>`	The product name of your IQFeed developer account.
`--iqfeed-version <value>`	The product version of your IQFeed developer account.
`--kraken-api-key <value>`	Your Kraken API key, which you can find on the API Management Settings page on the Kraken website.
`--kraken-api-secret <value>`	Your Kraken API secret.
`--kraken-verification-tier <value>`	Your Kraken verification tier (`Starter` , `Intermediate` , or `Pro`). For more information about verification tiers, see Verification levels explained on the Kraken website.
`--oanda-account-id <value>`	Your OANDA account id, which you can find on your Account Statement page on the OANDA website.
`--oanda-access-token <value>`	Your OANDA API token, which you can generate on the Manage API Access page on the OANDA website.
`--oanda-environment <value>`	`Practice` to trade on fxTrade Practice or `Trade` to trade on fxTrade.
`--polygon-api-key <value>`	Your Polygon data feed API Key.
`--samco-client-id <value>`	Your Samco account Client ID.
`--samco-client-password <value>`	Your Samco account password.
`--samco-year-of-birth <value>`	Your year of birth (YYYY) registered with Samco.

Option	Description
`--samco-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--samco-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--tdameritrade-api-key <value>`	Your TDAmeritrade API key.
`--tdameritrade-access-token <value>`	Your TDAmeritrade OAuth Access Token.
`--tdameritrade-account-number <value>`	Your TDAmeritrade account number.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-emsx-broker <value>`	The EMSX broker to use.
`--terminal-link-emsx-account <value>`	The EMSX account to use (optional).
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--tradier-account-id <value>`	Your Tradier account id, which you can find on your Settings > API Access page on the Tradier website.
`--tradier-access-token <value>`	Your Tradier access token.
`--tradier-environment <value>`	`live` to use the live environment or `paper` to use the developer sandbox.
`--tt-user-name <value>`	Your Trading Technologies username.
`--tt-session-password <value>`	Your Trading Technologies session password.
`--tt-account-name <value>`	Your Trading Technologies account name.
`--tt-rest-app-key <value>`	Your Trading Technologies REST app key.
`--tt-rest-app-secret <value>`	Your Trading Technologies REST app secret.
`--tt-rest-environment <value>`	The REST environment in which to run.
`--tt-market-data-sender-comp-id <value>`	The market data sender comp Id to use.
`--tt-market-data-target-comp-id <value>`	The market data target comp Id to use.

Option	Description
`--tt-market-data-host <value>`	The host of the market data server.
`--tt-market-data-port <value>`	The port of the market data server.
`--tt-order-routing-sender-comp-id <value>`	The order routing sender comp Id to use.
`--tt-order-routing-target-comp-id <value>`	The order routing target comp Id to use.
`--tt-order-routing-host <value>`	The host of the order routing server.
`--tt-order-routing-port <value>`	The port of the order routing server.
`--tt-log-fix-messages <boolean>`	Whether FIX messages should be logged.
`--zerodha-api-key <value>`	Your Kite Connect API key.
`--zerodha-access-token <value>`	Your Zerodha access token.
`--zerodha-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--zerodha-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--zerodha-history-subscription <boolean>`	Whether you have a history API subscription for Zerodha.
`--live-cash-balance <value>`	A comma-separated list of currency:amount pairs that define the initial cash balance.
`--live-holdings <value>`	A comma-separated list of "symbol:symbolId:quantity:averagePrice" pairs that define the initial portfolio holdings. For example, `"GOOG:GOOCV VP83T1ZUHROL:10:50.0"` .
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--python-venv <value>`	The path of the python virtual environment to use.
`--update`	Pull the LEAN engine image before starting live trading.
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).

Option	Description
`--release`	Compile C# projects in release configuration instead of debug.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live` command and exit.

API Reference

lean live add-security

Introduction

Add a security subscription to a local live algorithm.

```
$ lean live add-security <project> [options]
```

Description

Arguments

The `lean live add-security` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to which you want to add the security.

Options

The `lean live add-security` command supports the following options:

Option	Description
`--ticker <value>`	The ticker of the symbol to add.
`--market <value>`	The market of the symbol to add.
`--security-type <value>`	The security type of the symbol to add.
`--resolution <value>`	The resolution of the symbol to add. Defaults to `"Minute` .
`--fill-data-forward <boolean>`	The fill forward behavior. `true` to fill forward or `false` to not fill forward. Defaults to `true` .
`--leverage <value>`	The leverage for the security. Defaults to 2 for Equity, 50 for Forex, and 1 for everything else.
`--extended-market-hours <boolean>`	The extended market hours flag. `true` to allow pre/post market data or `false` for only in market data. Defaults to `false`
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live add-security` command and exit.

API Reference

lean live cancel-order

Introduction

Cancel an order with a specific Id in a local live trading project.

```
$ lean live cancel-order <project> [options]
```

Description

Arguments

The `lean live cancel-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file that contains the order you want to cancel.

Options

The `lean live cancel-order` command supports the following options:

Option	Description
`--order-id <value>`	The Id of the order to cancel.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live cancel-order` command and exit.

API Reference

lean live liquidate

Introduction

Liquidate a specific symbol from the latest local deployment of a project.

```
$ lean live liquidate <project> [options]
```

Description

Arguments

The `lean live liquidate` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to liquidate.

Options

The `lean live update-order` command supports the following options:

Option	Description
`--ticker <value>`	The ticker of the symbol to liquidate.
`--market <value>`	The market of the symbol to liquidate.
`--security-type <value>`	The security type of the symbol to liquidate.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live liquidate` command and exit.

API Reference

lean live stop

Introduction

Stop a local live trading algorithm.

```
$ lean live stop <project> [options]
```

Description

Arguments

The `lean live stop` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to stop live trading.

Options

The `lean live stop` command supports the following options:

Option	Description
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live stop` command and exit.

API Reference

lean live submit-order

Introduction

Submit an order in a local live trading project.

```
$ lean live submit-order <project> [options]
```

Description

Arguments

The `lean live submit-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file in which you want to submit the order.

Options

The `lean live submit-order` command supports the following options:

Option	Description
`--ticker <value>`	The ticker for the order.
`--market <value>`	The market for the order.
`--security-type <value>`	The security type for the order.
`--order-type <value>`	The type of the order.
`--quantity <value>`	The number of units to be ordered (directional).
`--limit-price <value>`	The limit price of the order.
`--stop-price <value>`	The stop price of the order.
`--tag <value>`	The order tag.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live submit-order` command and exit.

API Reference

lean live update-order

Introduction

Update an order with a specific Id in a local live trading project.

```
$ lean live update-order <project> [options]
```

Description

Arguments

The `lean live update-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file that contains the order you want to update.

Options

The `lean live update-order` command supports the following options:

Option	Description
`--order-id <value>`	The Id of the order to update.
`--quantity <value>`	The number of units to be updated (directional).
`--limit-price <value>`	The limit price of the order to be updated.
`--stop-price <value>`	The stop price of the order to be updated.
`--tag <value>`	The new order tag.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live update-order` command and exit.

API Reference

lean login

Introduction

Log in with a QuantConnect account.

```
$ lean login [options]
```

Description

Lets you log in with your QuantConnect API credentials and stores the given values in the **credentials** file in your global configuration directory.

If `--user-id` or `--api-token` is not provided, an interactive prompt is shown and the missing values are read from stdin.

You can request your user Id and API token on your Account page.

Options

The `lean login` command supports the following options:

Option	Description
`-u, --user-id <value>`	The QuantConnect user Id to log in with.
`-t, --api-token <value>`	The QuantConnect API token to log in with.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean login` command and exit.

API Reference

lean logout

Introduction

Log out and remove stored credentials.

```
$ lean logout [options]
```

Description

Removes the credentials stored in the **credentials** file in your global configuration directory.

Options

The `lean logout` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean logout` command and exit.

API Reference

lean logs

Introduction

Display the most recent backtest/live/optimization logs.

```
$ lean logs [options]
```

Description

Displays the most recent backtest/live/optimization logs. By default, the most recent backtest logs are shown unless `--live` or `--optimization` is given. You can pass in a project with `--project <directory>` to display the most recent logs from a specific project.

Options

The `lean logs` command supports the following options:

Option	Description
`--backtest`	Display the most recent backtest logs (default).
`--live`	Display the most recent live logs.
`--optimization`	Display the most recent optimization logs.
`--project <directory>`	The project of which to show the most recent logs of.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean logs` command and exit.

API Reference

lean optimize

Introduction

Optimize a project's parameters locally using Docker.

```
$ lean optimize <project> [options]
```

Description

Runs a local optimization in a Docker container using the quantconnect/lean Docker image. The logs of the optimizer are shown in real-time and the full results of the optimizer and all executed backtests are stored in the **<project> / optimizations / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to optimize. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to contain the algorithm to optimize.

By default, an interactive wizard is shown letting you configure the optimizer. When `--optimizer-config` or `--strategy` is given, the command runs in non-interactive mode and doesn't prompt for input.

When the `--optimizer-config <config file>` option is given, the specified config file is used. This option must point to a file containing a full optimizer config (the `algorithm-type-name` , `algorithm-language` and `algorithm-location` properties may be omitted). See the **Optimizer.Launcher / config.example.json** file in the LEAN repository for an example optimizer configuration file, which also contains documentation on all the required properties.

When `--strategy` is given, the optimizer configuration is read from the command-line options. This means the `--strategy` , `--target` , `--target-direction` , and `--parameter` options become required. Additionally, you can also use `--constraint` to specify optimization constraints.

In non-interactive mode, the parameters can be configured using the `--parameter` option. This option takes the following values: the name of the parameter, its minimum value, its maximum value, and its step size. You can provide this option multiple times to configure multiple parameters.

In non-interactive mode, the constraints can be configured using the `--constraint` option. This option takes a "statistic operator value" string as value, where the statistic must be a path to a property in a backtest's output file, like "TotalPerformance.PortfolioStatistics.SharpeRatio". This statistic can also be shortened to "SharpeRatio" or "Sharpe Ratio", in which case the command automatically converts it to the longer version. The value must be a

number and the operator must be `<` , `>` , `<=` , `>=` , `==` , or `==` . You can provide this option multiple times to configure multiple constraints.

Example non-interactive usage:

```
$ lean optimize "My Project" \
    --strategy "Grid Search" \
    --target "Sharpe Ratio" \
    --target-direction "max" \
    --parameter my-first-parameter 1 10 0.5 \
    --parameter my-second-parameter 20 30 5 \
    --constraint "Drawdown < 0.5" \
    --constraint "Sharpe Ratio >= 1"
```

To estimate the cost of running an optimization job without actually running it, add the `--estimate` option to the command. You need to backtest the project at least once in order to estimate the cost of optimizing it.

To set the maximum number of concurrent backtests to run, use the `--max-concurrent-backtests` option.

The Docker image that's used contains the same libraries as the ones available on QuantConnect . If the selected project is a C# project, it is compiled before starting the optimization.

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>` . The image is pulled before running the optimizer if it doesn't exist locally yet or if you pass the `--update` flag.

Arguments

The `lean optimize` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to optimize.

Options

The `lean optimize` command supports the following options:

Option	Description
`--output <path>`	Directory to store results in (defaults to **<project> / optimizations / <timestamp>**).
`--detach` , `-d`	Run the optimization in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.

Option	Description
`--optimizer-config <path>`	The optimizer configuration file that should be used (the `algorithm-type-name` , `algorithm-language` and `algorithm-location` properties may be omitted). See the **Optimizer.Launcher / config.example.json** file in the LEAN repository for an example optimizer config file.
`--strategy <value>`	The optimization strategy to use in non-interactive mode. Must be `Grid Search` or `Euler Search` .
`--target <value>`	The path to the property in the backtest's output file to target, like "TotalPerformance.PortfolioStatistics.SharpeRatio". May also be a shortened version, like "SharpeRatio" or "Sharpe Ratio".
`--target-direction <value>`	`min` if the target must be minimized, `max` if it must be maximized.
`--parameter <name> <min> <max> <step>`	The 'parameter min max step' pairs configuring the parameters to optimize. May be used multiple times.
`--constraint <value>`	The 'statistic operator value' pairs configuring the constraints of the optimization. May be used multiple times.
`--estimate`	Estimate optimization runtime without running it.
`--max-concurrent-backtests <value>`	Maximum number of concurrent backtests to run (x >= 1).
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the optimizer.
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--release`	Compile C# projects in release configuration instead of debug.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean optimize` command and exit.

API Reference

lean project-create

Introduction

Create a new project containing starter code.

```
$ lean project-create <name> [options]
```

Description

Creates a new project with some basic starter code. If the language is set to `python` , this generates a **main.py** file, a Python-based research notebook, a project configuration file, and editor configuration files for PyCharm and VS Code.

If the language is set to `csharp` this generates a **Main.cs** file, a C#-based research notebook, a project configuration file, and editor configuration files for Visual Studio, Rider, and VS Code.

A full list of the created files can be found on the Projects > Structure page.

If no `--language` is given, the default language saved in the global configuration is used. You can update the default language to Python by running `lean config set default-language python` or to C# by running `lean config set default-language csharp` .

If the given project name contains slashes, the name is parsed as a path and the project is created in a subdirectory. Any subdirectories that don't exist yet are created automatically.

Arguments

The `lean project-create` command expects the following arguments:

Argument	Description
`<name>`	The name of the project to create. This name may contain slashes to create a project in a subdirectory. The project name must only contain `-` , `_` , letters, numbers, and spaces. The project name can't start with a space or be any of the following reserved names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, or LPT9. If the project is a Python library, the library name can only contain letters (a-z), numbers (0-9), and underscores (_). Python library names can't contain spaces or start with a number.

Options

The `lean project-create` command supports the following options:

Option	Description
`--language <value>`	The language of the project to create, must be either `python` or `csharp` .
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean project-create` command and exit.

API Reference

lean project-delete

Introduction

Delete a project on your local machine and in the cloud.

```
$ lean project-delete <project> [options]
```

Description

Deletes a project from your local machine and in the cloud. If you are a collaborator on the project, this command doesn't delete the project for the other collaborators, but it removes you as a collaborator.

Options

The `lean project-delete` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean project-delete` command and exit.

API Reference

lean report

Introduction

Generate a report of a backtest or live trading algorithm.

```
$ lean report [options]
```

Description

Runs the LEAN Report Creator in a Docker container using the quantconnect/lean Docker image. By default, this command generates a report of the most recent backtest. This behavior can be overridden by using `--backtest-results <path>` and providing the path to the backtest results JSON file. If `--live-results <path>` is also given, the generated report will contain both the backtest and the live results.

The `--strategy-name` , `--strategy-version` , and `--strategy-description` options can be used to set the name, version, and description that are shown in the report. The name and version are shown in the top-right corner of each page, while the description is shown on the top of the first page. These fields are blank by default.

When the given backtest results are stored in a project directory or one of its subdirectories, the default name is the name of the project directory and the default description is the description in the project's **config.json** file.

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>` . The image is pulled before running the report creator if it doesn't exist locally yet or if you pass the `--update` flag.

Options

The `lean report` command supports the following options:

Option	Description
`--backtest-results <path>`	The path to the JSON file containing the backtest results. Defaults to the most recent backtest results file in the current working directory or one of its subdirectories.
`--live-results <path>`	The path to the JSON file containing the live trading results.
`--report-destination <path>`	The path where the generated report is stored as HTML (defaults to **./report.html**).
`--detach`, `-d`	Run the report creator in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--strategy-name <value>`	The name of the strategy. It will appear at the top-right corner of each page.
`--strategy-version <value>`	The version of the strategy. It will appear next to the project name.
`--strategy-description <value>`	The description of the strategy. It will appear under the **Strategy Description** section on the first page of the report.
`--pdf`	Create a PDF version along with the HTML version of the report.
`--overwrite`	Overwrite the given `--report-destination` if it already contains a file.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the report creator.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean report` command and exit.

API Reference

lean research

Introduction

Run a Jupyter Lab environment locally using Docker.

```
$ lean research <project> [options]
```

Description

Runs a local Jupyter Lab environment in a Docker container using the quantconnect/research Docker image. The project directory is mounted in the Docker container and the Jupyter Lab instance is exposed on a local port. After the Jupyter Lab instance has started, the browser automatically opens.

By default, Jupyter Lab is exposed on port 8888. To use a custom port, you can use the `--port` option, which is required to run two Jupyter Lab instances side-by-side.

You can use the `--data-provider` option to change where data is retrieved. This option updates the Lean configuration file , so you don't need to use this option multiple times for the same data provider if you are not switching between them. If you use the Terminal Link data provider, you must also provide the following options:

- `--terminal-link-environment`
- `--terminal-link-server-host`
- `--terminal-link-server-port`
- `--terminal-link-openfigi-api-key`

You can use the `--download-data` flag as an alias for `--data-provider QuantConnect` and the `--data-purchase-limit` option to set the maximum amount of QuantConnect Credit (QCC) to spend during the research session when using QuantConnect as data provider. The `--data-purchase-limit` option is not persistent.

If you have previously logged in using `lean login` , the CLI automatically makes your credentials available in the Jupyter Lab instance. If this happens, the `api` variable is automatically assigned an instance of Api in your research notebooks, which you can use to make authenticated requests to the QuantConnect API.

The default Research Environment configuration is the latest master branch of LEAN. If you set a different research image , the image you set is your current configuration. To start the Research Environment with a different configuration than your current configuration, use the `--image <value>` option. If the image doesn't exist on your local machine or you pass the `--update` flag, the image is pulled before starting the Research Environment. To avoid updating the image, pass the `--no-update` flag.

Arguments

The `lean research` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory for which to run a research environment.

Options

The `lean research` command supports the following options:

Option	Description
`--port <value>`	The port to run Jupyter Lab on (defaults to 8888).
`--data-provider <value>`	Update the Lean configuration file to retrieve data from the given provider, which must be `Local` , `QuantConnect` , or `Terminal Link` .
`--download-data`	Update the Lean configuration file to download data from the QuantConnect API, alias for `--data-provider QuantConnect` .
`--data-purchase-limit <value>`	The maximum amount of QCC to spend on downloading data during the research session when using QuantConnect as data provider.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--detach` , `-d`	Run Jupyter Lab in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--no-open`	Don't open the Jupyter Lab environment in the browser after starting it.
`--image <value>`	The LEAN research image to use (defaults to `quantconnect/research:latest`).
`--update`	Pull the LEAN research image before starting the research environment.
`--no-update`	Use the current LEAN research image.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean research` command and exit.

API Reference

lean whoami

Introduction

Display who is logged in.

```
$ lean whoami [options]
```

Description

Displays the name and the email address of the user who is currently logged in or "You are not logged in" if no-one is logged in.

Options

The `lean whoami` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean whoami` command and exit.