



Learn to use QuantConnect and Explore 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

- [1 Key Concepts](#)
- [1.1 Getting Started](#)
- [1.2 Troubleshooting](#)
- [2 Installation](#)
- [2.1 Installing pip](#)
- [2.2 Installing Lean CLI](#)
- [3 Initialization](#)
- [3.1 Authentication](#)
- [3.2 Organization Workspaces](#)
- [3.3 Configuration](#)
- [4 Datasets](#)
- [4.1 Format and Storage](#)
- [4.2 Generating Data](#)
- [4.3 Custom Data](#)
- [4.4 QuantConnect](#)
- [4.4.1 Download By Ticker](#)
- [4.4.1.1 Key Concepts](#)
- [4.4.1.2 Costs](#)
- [4.4.2 Download in Bulk](#)
- [4.4.2.1 CFD Data](#)
- [4.4.2.2 FOREX Data](#)
- [4.4.2.3 US Equities](#)
- [4.4.2.4 US Equity Coarse Universe](#)
- [4.4.2.5 US Equity Options](#)
- [4.4.2.6 US Equity Option Universe](#)
- [4.4.2.7 US ETF Constituents](#)
- [4.4.2.8 US Futures](#)
- [4.4.2.9 US Future Universe](#)
- [4.4.2.10 International Future Universe](#)
- [4.4.2.11 US Future Options](#)
- [4.4.2.12 US Future Option Universe](#)
- [4.4.2.13 US Index Options](#)
- [4.4.2.14 US Index Option Universe](#)
- [4.5 Brokerages](#)
- [4.6 Alpha Vantage](#)
- [4.7 FactSet](#)
- [4.8 IQFeed](#)
- [4.9 Polygon](#)
- [4.10 Terminal Link](#)

- [4.11 Theta Data](#)
- [5 Projects](#)
- [5.1 Project Management](#)
- [5.2 Cloud Synchronization](#)
- [5.3 Structure](#)
- [5.4 Workflows](#)
- [5.5 Encryption](#)
- [5.6 Configuration](#)
- [5.7 Autocomplete](#)
- [5.8 Libraries](#)
- [5.8.1 Third-Party Libraries](#)
- [5.8.2 Project Libraries](#)
- [5.9 Custom Docker Images](#)
- [5.10 Version Control](#)
- [6 Research](#)
- [7 Backtesting](#)
- [7.1 Deployment](#)
- [7.2 Debugging](#)
- [8 Live Trading](#)
- [8.1 Brokerages](#)
- [8.1.1 QuantConnect Paper Trading](#)
- [8.1.2 Interactive Brokers](#)
- [8.1.3 TradeStation](#)
- [8.1.4 Tastytrade](#)
- [8.1.5 Alpaca](#)
- [8.1.6 Charles Schwab](#)
- [8.1.7 Binance](#)
- [8.1.8 Bybit](#)
- [8.1.9 Tradier](#)
- [8.1.10 Kraken](#)
- [8.1.11 Coinbase](#)
- [8.1.12 Bitfinex](#)
- [8.1.13 Bloomberg EMSX](#)
- [8.1.14 SSC Eze](#)
- [8.1.15 Trading Technologies](#)
- [8.1.16 Samco](#)
- [8.1.17 Zerodha](#)
- [8.1.18 CFD and FOREX Brokerages](#)
- [8.2 Data Providers](#)
- [8.2.1 IQFeed](#)
- [8.2.2 Polygon](#)

- [8.2.3 Theta Data](#)
- [8.3 Algorithm Control](#)
- [9 Reports](#)
- [10 Optimization](#)
- [10.1 Parameters](#)
- [10.2 Deployment](#)
- [11 Object Store](#)
- [12 API Reference](#)
- [12.1 lean backtest](#)
- [12.2 lean build](#)
- [12.3 lean cloud backtest](#)
- [12.4 lean cloud live](#)
- [12.5 lean cloud live command](#)
- [12.6 lean cloud live deploy](#)
- [12.7 lean cloud live liquidate](#)
- [12.8 lean cloud live stop](#)
- [12.9 lean cloud object-store delete](#)
- [12.10 lean cloud object-store get](#)
- [12.11 lean cloud object-store list](#)
- [12.12 lean cloud object-store ls](#)
- [12.13 lean cloud object-store properties](#)
- [12.14 lean cloud object-store set](#)
- [12.15 lean cloud optimize](#)
- [12.16 lean cloud pull](#)
- [12.17 lean cloud push](#)
- [12.18 lean cloud status](#)
- [12.19 lean config get](#)
- [12.20 lean config list](#)
- [12.21 lean config set](#)
- [12.22 lean config unset](#)
- [12.23 lean create-project](#)
- [12.24 lean data download](#)
- [12.25 lean data generate](#)
- [12.26 lean decrypt](#)
- [12.27 lean delete-project](#)
- [12.28 lean encrypt](#)
- [12.29 lean init](#)
- [12.30 lean library add](#)
- [12.31 lean library remove](#)
- [12.32 lean live](#)

- [12.33 lean live add-security](#)
- [12.34 lean live cancel-order](#)
- [12.35 lean live command](#)
- [12.36 lean live deploy](#)
- [12.37 lean live liquidate](#)
- [12.38 lean live stop](#)
- [12.39 lean live submit-order](#)
- [12.40 lean live update-order](#)
- [12.41 lean login](#)
- [12.42 lean logout](#)
- [12.43 lean logs](#)
- [12.44 lean object-store delete](#)
- [12.45 lean object-store get](#)
- [12.46 lean object-store list](#)
- [12.47 lean object-store ls](#)
- [12.48 lean object-store properties](#)
- [12.49 lean object-store set](#)
- [12.50 lean optimize](#)
- [12.51 lean private-cloud add-compute](#)
- [12.52 lean private-cloud start](#)
- [12.53 lean private-cloud stop](#)
- [12.54 lean project-create](#)
- [12.55 lean project-delete](#)
- [12.56 lean report](#)
- [12.57 lean research](#)
- [12.58 lean whoami](#)

Key Concepts

Key Concepts > Getting Started

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Prerequisites

Before you start installing the Lean CLI, check the requirements of [deploying with your brokerage](#) to ensure you have a compatible machine and review the [integration documentation for your brokerage](#) so you are aware of what functionality is available through the integration.

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.

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.

Authentication

Most of the Lean CLI commands need to communicate with the QuantConnect API. If you use any commands which interact with the cloud or [deploy a live algorithm locally](#) , you must log in using your QuantConnect account so the CLI can send authenticated API requests.

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
```

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.

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 occasionally 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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 <code>pip install --upgrade lean</code> to update to the latest version.
No such file	Follow these steps: <ol style="list-style-type: none">1. Open the advance settings in Docker Desktop.2. Disable the Allow the default Docker socket to be used setting.3. Save and restart the Docker container.4. Enable the Allow the default Docker socket to be used setting.5. Save and restart the Docker container.
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 <code>lean login</code> 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. If Docker is already running, run your command with <code>--verbose</code> for more information.

	Your venv probably has a non standard docker path or no docker access. Uninstall and reinstall docker.
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 <code>--lean-config</code> option.
We couldn't find your 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 <code>lean init</code> in an empty directory.
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 <code>./<path></code> 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: <code><</code> , <code>></code> , <code>:</code> , <code>"</code> , <code> </code> , <code>?</code> , and <code>*</code> . On top of those characters the following names are not allowed (case-insensitive): <code>CON</code> , <code>PRN</code> , <code>AUX</code> , <code>NUL</code> , <code>COM1</code> until <code>COM9</code> , and <code>LPT1</code> until <code>LPT9</code> . 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.

Could not open '/lib64/ld-linux-x86-64.so.2': No such file or directory	Your Docker installation has pulled the incorrect platform version of the LEAN Docker image. Open your terminal. Run <code>docker rmi quantconnect/lean</code> to remove the quantconnect/lean image and then run <code>docker pull quantconnect/lean --platform=linux/amd64</code> .
Could not find file '/root/ibgateway/ibgateway'.	Your Docker installation has pulled the ARM platform version of the LEAN Docker image. This version doesn't include IB Gateway, because QuantConnect doesn't support Interactive Brokers integration with ARM chips (e.g.: Apple M1, M2, and M3 chips).

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

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. If you want to install a non-Debian packaged Python application, it may be easiest to use `pipx install xyz`, which will manage a virtual environment for you.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
- 60 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 administrator privileges 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
- 60 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 administrator 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

Initialization

Authentication

Introduction

Most of the Lean CLI commands need to communicate with the QuantConnect API. If you use any commands which interact with the cloud or [deploy a live algorithm locally](#) , you must log in using your QuantConnect account so the CLI can send authenticated API requests.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 the 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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
├── storage/
└── 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.
storage /	This directory contains the Object Store data that LEAN uses to run locally.
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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 <code>lean login</code> .
config	This file contains the CLI defaults, like the default project language used when running <code>lean project-create</code> .
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
<code>show-missing-data-logs</code>	Log missing data files. This is useful for debugging.	true
<code>algorithm-manager-time-loop-maximum</code>	The maximum amount of time the algorithm can spend in a single time loop .	20
<code>maximum-warmup-history-days-look-back</code>	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
<code>maximum-chart-series</code>	The maximum number of chart series you can create in backtests.	30
<code>maximum-data-points-per-chart-series</code>	The maximum number of data points you can add to a chart series in backtests.	1,000,000

Project Configuration

For information about project configuration, see [Projects > Configuration](#).

Datasets

Datasets > Format and Storage

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

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

Data Updates

Every day, the LEAN CLI updates the [exchange market hours](#) (`data / market-hours / market-hours-database.json`) and the [symbol properties](#) database (`data / symbol-properties / symbol-properties-database.csv`). To disable the updates, open the [LEAN configuration file](#) (`lean.json`) and set the `file-database-last-update` value to a date in the future.

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

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Security Types

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

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

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:

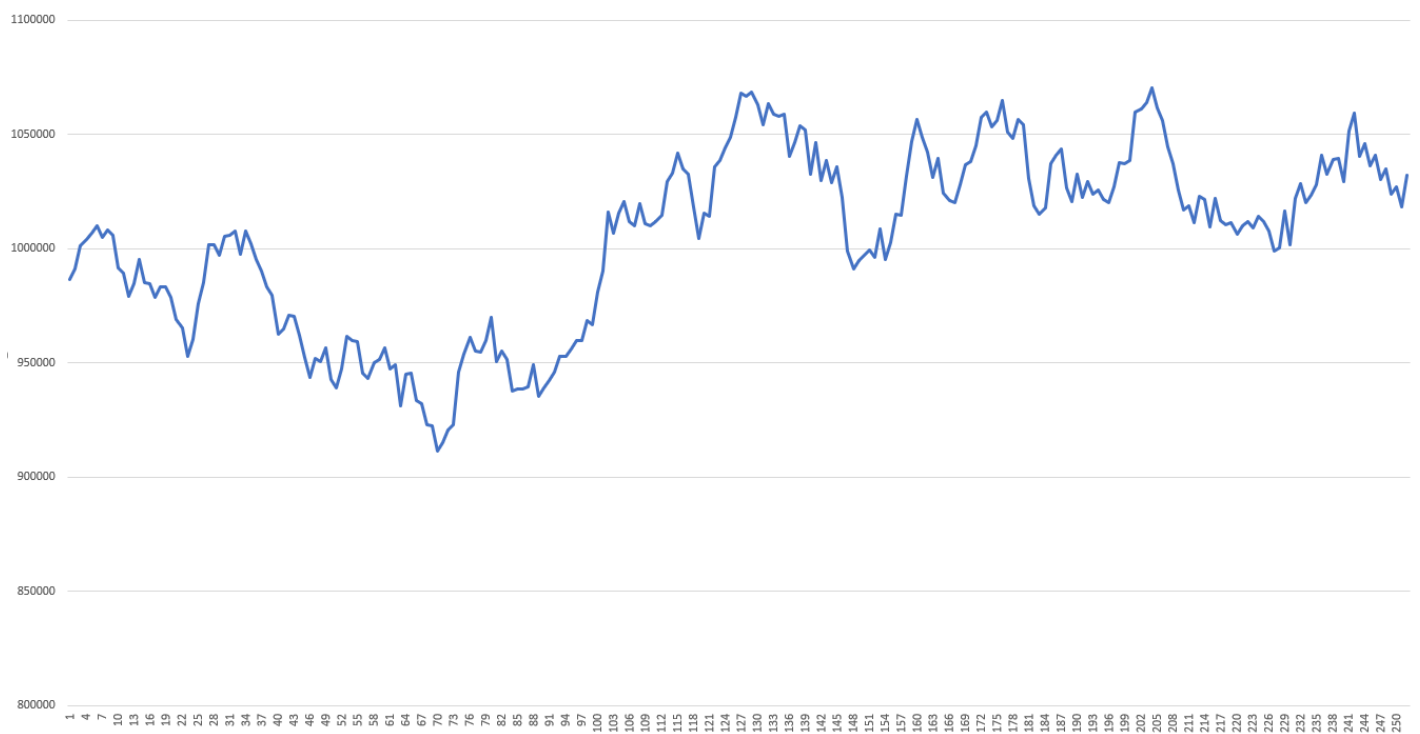
1. Open a terminal in one of your [organization workspaces](#) .
2. Run `lean data generate --start 20150101 --symbol-count 10` to generate dense minute Equity data since 01-01-2015 for 10 random symbols.

```
$ 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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 `get_source` 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:

```
import os

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

class Weather(PythonData):
    def get_source(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.REMOTE_FILE)

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

PY

4. Save the source file.

Datasets

QuantConnect

QuantConnect was founded in 2012 to serve quants everywhere with the best possible algorithmic trading technology. Seeking to disrupt a notoriously closed-source industry, QuantConnect takes a radically open-source approach to algorithmic trading. Through the QuantConnect web platform, more than 50,000 quants are served every month.

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](#)

QuantConnect

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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: BTCUSD 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) US Congress Trading
19) US ETF Constituents
20) US Equities
21) US Equity Coarse Universe
22) US Equity Options
23) US Equity Security Master
24) US Federal Reserve (FRED)
25) US Futures
26) US Futures Security Master
27) US Index Options
28) US SEC Filings
29) US Treasury Yield Curve
30) VIX Central Contango
31) VIX Daily Price
Enter an option: 20
```

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
```

- 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 .
- 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 purchase, 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 [Download Datasets During Backtests](#) .

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](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 in 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
Fundamental 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.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.universe_settings.asynchronous = True
        self.add_universe(self.universe.dollar_volume.top(100))
```

PY

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 following data:

- [US Equity Security Master](#)
- [US Equity Option Universe](#)
- Some data from the [US Equity Options](#) dataset
- 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 file format of the US Equity Option Universe data is one file per underlying Equity and each file costs 100 QCC = \$1 USD.

The US Equity Options dataset is available in 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:

```
class USEquityOptionsDataAlgorithm(QCAAlgorithm):
    def initialize(self) -> None:
        self.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.universe_settings.asynchronous = True
        underlying = self.add_equity("GOOG").symbol
        self.add_option(underlying)
```

PY

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 Option Universe	Download On Premise	1 underlying Equity over 252 trading days => 1 * 252 files = 252 files 252 files @ 100 QCC/file => 252 * 100 QCC = 25,200 QCC = \$252 USD	1 underlying Equity => 1 file/day 1 file/day @ 100 QCC/file = 100 QCC/day = \$1 USD/day
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 in 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 **CryptoUniverse** data. The file format of **CryptoUniverse** 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.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.universe_settings.asynchronous = True
        self.add_universe(CryptoUniverse.coinbase(self.universe_filter))

    def universe_filter(self, universe_day: list[CryptoUniverse]) -> list[Symbol]:
        sorted_by_dollar_volume = sorted(universe_day, key=lambda cf: cf.volume_in_usd, reverse=True)
        return [cf.symbol for cf in sorted_by_dollar_volume[:100]]
```

PY

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.

Crypto Future

Crypto Future algorithms require data from one of the price datasets and one of the margin rate datasets. The price datasets are available in several resolutions. The resolution you need depends on the Crypto 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 security per trading day per data format. Quote and trade data are separate files.	100 QCC = \$1 USD
Second	One file per security per trading day per data format. Quote and trade data are separate files.	25 QCC = \$0.25 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. Quote and trade data are separate files.	400 QCC = \$4 USD
Daily	One file per security. Quote and trade data are separate files.	100 QCC = \$1 USD

The file format of the margin rate data is one file per Crypto Future and each file costs 100 QCC = \$1 USD.

For example, the following algorithm subscribes to minute resolution data for one Crypto Future:

```
class CryptoFutureDataAlgorithm(QCAAlgorithm):
    def initialize(self) -> None:
        self.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.set_brokerage_model(BrokerageName.BINANCE_FUTURES, AccountType.MARGIN)
        self.add_crypto_future('BTCUSD')
```

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

Dataset	Package	Initial Cost	Ongoing Cost
Binance Crypto Future Price Data	Minute Download	1 security over 365 trading days with 2 data formats $\Rightarrow 1 * 365 * 2 \text{ files}$ $= 730 \text{ files}$ 730 files @ 5 QCC/file $\Rightarrow 730 * 5 \text{ QCC}$ $= 3,650 \text{ QCC}$ $= \$36.50 \text{ USD}$	2 files per day @ 5 QCC/file $\Rightarrow 2 * 5 \text{ QCC/day}$ $= 10 \text{ QCC/day}$ $= \$0.10 \text{ USD/day}$
Binance Crypto Future Margin Rate Data	Download On Premise	1 security $\Rightarrow 1 \text{ file}$ 1 file @ 100 QCC/file $\Rightarrow 1 * 100 \text{ QCC}$ $= 100 \text{ QCC}$ $= \$1 \text{ USD}$	1 security $\Rightarrow 1 \text{ file/day}$ 1 file/day @ 100 QCC/file $\Rightarrow 1 * 100 \text{ QCC/day}$ $= 100 \text{ QCC/day}$ $= \$1 \text{ USD/day}$

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 in 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(QCAAlgorithm):
    def initialize(self) -> None:
        self.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.add_forex("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 the following data:

- [US Futures Security Master](#)
- [US Future Universe](#)
- Some data from the [US Futures](#) 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

The file format of the US Future Universe dataset is one file per Future per day and each file costs 100 QCC = \$1 USD.

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

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.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        future = self.add_future(
            Futures.Indices.SP500E_MINI,
            data_normalization_mode=DataNormalizationMode.BACKWARDS_RATIO,
            data_mapping_mode=DataMappingMode.OPEN_INTEREST,
            contract_depth_offset=0
        )
        future.set_filter(0, 90)

```

PY

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 Future Universe	Download On Premise	1 ticker over 252 trading days => 1 * 252 files = 252 files 252 files @ 100 QCC/file => 25,200 QCC = \$252 USD	1 ticker => 1 file/day 1 file/day @ 100 QCC/file => 100 QCC/day = \$1 USD/day
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 the [US Index Option Universe](#) dataset and some data from the [US Index Options](#) dataset. The file format of the US Equity Option Universe data is one file per underlying Equity and each file costs 100 QCC = \$1 USD. 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:

```
class USIndexOptionsDataAlgorithm(QCAAlgorithm):
    def initialize(self):
        self.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.add_index_option("VIX")
```

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

Dataset	Package	Initial Cost	Ongoing Cost
US Index Option Universe	Download On Premise	1 underlying Index over 252 trading days => 1 * 252 files = 252 files 252 files @ 100 QCC/file => 252 * 100 QCC = 25,200 QCC = \$252 USD	1 underlying Index => 1 file/day 1 file/day @ 100 QCC/file => 100 QCC/day = \$1 USD/day
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 in 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(QCAAlgorithm):
    def initialize(self) -> None:
        self.set_start_date(2020, 1, 1)
        self.set_end_date(2021, 1, 1)
        self.add_cfd("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.

QuantConnect

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 Equity Option Universe](#)

[US ETF Constituents](#)

[US Futures](#)

[US Future Universe](#)

[International Future Universe](#)

[US Future Options](#)

[US Future Option Universe](#)

[US Index Options](#)

[US Index Option Universe](#)

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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:

- CFD - Daily History
- CFD - Hour History
- CFD - Minute History
- 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.
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:

- CFD - Daily Updates
- CFD - Hour Updates
- CFD - Minute Updates
- 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
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "CFD Data" --data-type "Bulk" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute", "second"]:
        dir_name = f"cfdata/oanda/{resolution}/xauusd".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"cfdata/oanda/{resolution}/xauusd.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)
```

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

For more information about the file format, see the [Data / cfd](#) directory in the LEAN repository.

Price

To view the price of the CFD dataset, see [Pricing](#) .

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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:

- Forex - Daily History
- Forex - Hour History
- Forex - Minute History
- 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.
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:

- Forex - Daily Updates
- Forex - Hour Updates
- Forex - Minute Updates
- 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
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "FOREX Data" --data-type "Bulk" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute", "second"]:
        dir_name = f"forex/oanda/{resolution}/eurusd".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"forex/oanda/{resolution}/eurusd.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[8:]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)
```

To update your local dataset, the preceding script checks the date of the most recent EURUSD data you have for all 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

For more information about the file format, see the [Data / forex](#) directory in the LEAN repository.

Price

To view the price of the Forex dataset, see [Pricing](#) .

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.
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](#) .

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:

```

import os
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Equities" --data-type "Bulk" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute", "second", "tick"]:
        dir_name = f"equity/usa/{resolution}/spy".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"equity/usa/{resolution}/spy.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)

```

The preceding script checks the date of the most recent SPY data you have for all 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.

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

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

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

For more information about the file format, see the [Data / equity](#) directory in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Equities](#) and [US Equity Security Master](#) dataset listings.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Equity Coarse Universe dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- US Equity Coarse Universe History by QuantConnect
- US Equity Security Master by QuantConnect

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"
```

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, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"equity/usa/fundamental/coarse"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US Equity Coarse Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

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.

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

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

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. For an example, see the [Data / equity / usa / fundamental / coarse / 20140407.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Equity Coarse Universe](#) and [US Equity Security Master](#) dataset listings.

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 following datasets:

- [US Equity Security Master](#) - data on splits, dividends, and symbol changes.
- [US Equity Option Universe](#) - data on the available contracts and their daily Greeks and implied volatility values.
- [US Equities](#) - trade and quote data on the underlying Equities.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 and US Equity Option Universe History by QuantConnect data packages, subscribe to them 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.
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"
```

To download the US Equity Option Universe data, run:

```
$ lean data download --dataset "US Equity Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

To download the US Equities data, see [Download History](#) .

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](#) . 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:

```

import os
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Equity Options" --data-type "Bulk" --option-style
"American" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute"]:
        dir_name = f"option/usa/{resolution}/aapl".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"option/usa/{resolution}/aapl.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)

```

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.

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

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

To update your local copy of the other dependency datasets, see the [Download Daily Updates](#) documentation for the [US Equity Option Universe](#) and [US Equities](#) datasets.

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	3 files per ticker
Hour	500 GB	3 files per ticker
Minute	6 TB	3 files per ticker per day

For more information about the file format, see the [Data / option](#) directory in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the following dataset listings:

- [US Equity Options](#)
- [US Equity Security Master](#)
- [US Equity Option Universe](#)

Download in Bulk

US Equity Option Universe

Introduction

Download the [US Equity Option Universe dataset](#) in bulk to get the full dataset without any selection bias. The bulk dataset packages contains the available Option contracts and their daily Greeks and implied volatility values for every underlying US Equity and trading day.

The US Equity Option Universe dataset depends on the [US Equity Security Master](#) dataset because the US Equity Security Master contains information on splits, dividends, and symbol changes.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Equity Option Universe dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- US Equity Option Universe History by QuantConnect
- US Equity Security Master by QuantConnect

You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the US Equity Option Universe data, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Equity Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

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 Option 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 Option 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 Option Universe dataset, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Equity Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Alternatively, instead of directly calling the `lean data download` command, you can place the following Python script in the data directory of your organization workspace and run it to update your data files.

PY

```
import os
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"option/usa/universes"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US Equity Option Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

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

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

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

Size and Format

The US Equity Option Universe dataset is 450 GB in size. We structure the data files so there is one file per underlying Equity per day. For an example, see the [Data / option / usa / universes / spy / 20240109.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Equity Option Universe](#) and [US Equity Security Master](#) dataset listings.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US ETF Constituents dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- US ETF Constituents History by QuantConnect
- US Equity Security Master by QuantConnect

You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the US ETF Constituents data, open a terminal in your [organization workspace](#) and run:

```
$ lean data download --dataset "US ETF Constituents" --data-type "Bulk" --start "20090601" --end "20500101"
```

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 "Bulk" --start "20090601" --end "20500101"
```

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, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"equity/usa/universes/etf/spy"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US ETF Constituents" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

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.

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

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

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. For an example, see the [Data / equity / usa / universes / etf / spy / 20201201.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US ETF Constituents](#) and [US Equity Security Master](#)

dataset listings.

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.

The US Futures dataset depends on the [US Futures Security Master](#) dataset because the US Futures Security Master dataset contains information to construct continuous Futures.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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

If you don't already subscribe to the US Futures 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, follow these steps to download the US Futures 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.
3. Open a terminal in your [organization workspace](#) and then run the command from the CLI Command Generator

To download the US Futures Security Master, run:

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

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:

```

import os
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Futures" --data-type "Bulk" --market "CBOT" --
resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute", "second", "tick"]:
        dir_name = f"future/cbot/{resolution}/zc".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"future/cbot/{resolution}/zc.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)

```

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.

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

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

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

For more information about the file format, see the [Data / future](#) directory in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Futures](#) and [US Futures Security Master](#) dataset listings.

Download in Bulk

US Future Universe

Introduction

Download the [US Future Universe dataset](#) in bulk to get the full dataset without any selection bias. The bulk dataset packages contains the available Futures contracts, their daily prices, and open interest for every trading day.

The Future Universe dataset depends on the [US Futures Security Master](#) dataset because the US Futures Security Master dataset contains information on symbol changes of the contracts.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Future Universe dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- US Future Universe History by QuantConnect
- US Futures Security Master by QuantConnect

You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the US Future Universe data, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Future Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

To download the US Futures Security Master, run:

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

Download Daily Updates

After you bulk download the US Future 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 Future 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 Future Universe dataset, open a terminal in your [organization workspace](#) and then run:


```
$ lean data download --dataset "US Future Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Alternatively, instead of directly calling the `lean data download` command, you can place the following Python script in the data directory of your organization workspace and run it to update your data files.

PY

```
import os
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"future/cme/universes"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US Future Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

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

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

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

Size and Format

The US Future Universe dataset is about 2 GB in size. We structure the data files so there is one file per Future per day. For an example, see the [Data / future / cme / universes / es / 20200211.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Future Universe](#) and [US Futures Security Master](#) dataset listings.

Download in Bulk

International Future Universe

Introduction

Download the [International Future Universe dataset](#) in bulk to get the full dataset without any selection bias. The bulk dataset packages contains the available Futures contracts, their daily prices, and open interest for every trading day.

The International Future Universe dataset depends on the [US Futures Security Master](#) dataset because the US Futures Security Master dataset contains information on symbol changes of the contracts.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the International Future Universe dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- International Future Universe History by QuantConnect
- US Futures Security Master by QuantConnect

You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the International Future Universe 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.
3. Open a terminal in your [organization workspace](#) and then run the command from the CLI Command Generator

To download the US Futures Security Master, run:

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

Download Daily Updates

After you bulk download the International Future 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 International Future 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 International Future Universe 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 the following Python script in the data directory of your organization workspace and run it to update your data files.

PY

```
import os
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"future/hkfe/universes"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "International Future Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

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

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

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

Size and Format

The International Future Universe dataset is about 1 GB in size. We structure the data files so there is one file per Future per day. For an example, see the [Data / future / hkfe / universes / hsi / 20131028.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [International Future Universe](#) and [US Futures Security Master](#) dataset listings.

Download in Bulk

US Future Options

Introduction

Download the [US Future 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 Future Options dataset depends on the following datasets:

- [US Future Option Universe](#) - data on the available US Future Options contracts and their current Open Interest.
- [US Futures Security Master](#) - data to construct continuous Futures.
- [US Future Universe](#) - data on the available US Future contracts, their daily trading volume, and Open Interest.
- [US Futures](#) - data on the underlying Futures contracts, including price, volume, Open Interest, and expiry.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Future Options dataset, [contact us](#) . 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.
3. Open a terminal in your [organization workspace](#) and then run the command from the CLI Command Generator .

To download the US Futures Security Master, run:

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

To download the data of the other dependency datasets, see the Download History documentation for the [US Future Option Universe](#) , [US Future Universe](#) , and [US Futures](#) datasets.

Download Daily Updates

After you bulk download the US Future Options dataset, new daily updates are available at 7 AM Eastern Time (ET) after each trading day. To unlock local access to the data updates, [contact us](#) . You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Future 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 and markets:

PY

```
import os
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Future Options" --data-type "Bulk" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute"]:
        dir_name = f"futureoption/cme/{resolution}/es".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"futureoption/cme/{resolution}/es.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)
```

The preceding script checks the date of the most recent ES data you have from the CME market for minute resolution. If there is new data available, 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.

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

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

To update your local copy of the other dependency datasets, see the Download Daily Updates documentation for the [US Future Option Universe](#) , [US Future Universe](#) , and [US Futures](#) datasets.

Size and Format

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

Resolution	Size	Format
Daily	1.5 GB	1 file per ticker
Hour	11 GB	1 file per ticker
Minute	400 GB	1 file per ticker per day

For an example, see the [Data / futureoption / cme / daily / es / 20200619 / es_2020_trade_american.zip](#) file in the LEAN repository.

Price

To view the price of the US Future Options dataset, [contact us](#) .

Download in Bulk

US Future Option Universe

Introduction

Download the [US Future Option Universe](#) dataset in bulk to get the full dataset without any selection bias. The bulk dataset packages contains the available Future Option contracts, their daily prices, and volume for every trading day.

The US Future Option Universe dataset depends on the following datasets:

- [US Futures Security Master](#) - information on symbol changes of the contracts.
- [US Future Universe](#) - the universe of underlying Futures contracts.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Future Option Universe dataset, open the [Pricing](#) page of your organization and subscribe to the following data packages:

- US Future Option Universe History by QuantConnect
- US Futures Security Master by QuantConnect
- US Future Universe History by QuantConnect

You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the US Future Option Universe data, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Future Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

To download the US Futures Security Master, run:

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

To download the US Future Universe data, run:


```
$ lean data download --dataset "US Future Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Download Daily Updates

After you bulk download the US Future Option 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 Future Option Universe Updates by QuantConnect and US Future Universe Updates by QuantConnect data packages. You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to dataset updates, to update your local copy of the US Future Option Universe dataset, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Future Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Alternatively, instead of directly calling the `lean data download` command, you can place the following Python script in the data directory of your organization workspace and run it to update your data files.

PY

```
import os
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"futureoption/cme/universes"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US Future Option Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)
```

The preceding script checks the date of the most recent US Future Option Universe data you have. If there is new

data available, it downloads the new data files.

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

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

To update your local copy of the US Future Universe data, run:

```
$ lean data download --dataset "US Future Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Size and Format

The US Future Option Universe dataset is about 6 GB in size. We structure the data files so there is one file per underlying Future contract per day. For an example, see the [Data / futureoption / cme / universes / es / 20200619 / 20200108.csv](#) file in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the following dataset listings:

- [US Future Option Universe](#)
- [US Futures Security Master](#)
- [US Future Universe](#)

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.

The US Index Options dataset depends on the [US Index Option Universe](#) dataset because the US Index Options Universe dataset contains information on the available contracts, including their daily Greeks and implied volatility values.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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

If you don't already subscribe to the US Index Option Universe by QuantConnect data package, subscribe to it too. 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.
3. Open a terminal in your [organization workspace](#) and then run the command from the CLI Command Generator

To download the US Index Option Universe data, run:

```
$ lean data download --dataset "US Index Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

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:

```

import os
import pandas as pd
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

# Define a method to download the data
def __download_data(resolution, start=None, end=None):
    print(f"Updating {resolution} data...")
    command = f'lean data download --dataset "US Index Options" --data-type "Bulk" --resolution "{resolution}"'
    if start:
        end = end if end else start
        command += f" --start {start} --end {end}"
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7,30):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:30 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

def __download_high_frequency_data(latest_on_cloud):
    for resolution in ["minute"]:
        dir_name = f"indexoption/usa/{resolution}/spx".lower()
        if not os.path.exists(dir_name):
            __download_data(resolution, '19980101')
            continue
        latest_on_disk = sorted(os.listdir(dir_name))[-1].split('_')[0]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution, latest_on_disk, latest_on_cloud)

def __download_low_frequency_data(latest_on_cloud):
    for resolution in ["daily", "hour"]:
        file_name = f"indexoption/usa/{resolution}/spx.zip".lower()
        if not os.path.exists(file_name):
            __download_data(resolution)
            continue
        latest_on_disk = str(pd.read_csv(file_name, header=None)[0].iloc[-1])[0:8]
        if latest_on_disk >= latest_on_cloud:
            print(f"{resolution} data is already up to date.")
            continue
        __download_data(resolution)

if __name__ == "__main__":
    latest_on_cloud = __get_end_date()
    __download_low_frequency_data(latest_on_cloud)
    __download_high_frequency_data(latest_on_cloud)

```

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.

To update your local copy of the US Index Option Universe data, see [Download Daily Updates](#) .

Size and Format

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

Resolution	Size	Format
Daily	5 GB	3 files per contract
Hour	40 GB	3 files per contract
Minute	500 GB	3 files per contract per day

For examples, see the [Data / indexoption / usa / daily](#) directory in the LEAN repository.

Price

To view the prices of the datasets, see the Pricing tab of the [US Index Options](#) and [US Index Option Universe](#) dataset listings.

Download in Bulk

US Index Option Universe

Introduction

Download the [US Index Option Universe dataset](#) in bulk to get the full dataset without any selection bias. The bulk dataset packages contains the available Option contracts and their daily Greeks and implied volatility values for every underlying US Index and trading day.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Download History

To unlock local access to the US Index Option Universe dataset, open the [Pricing](#) page of your organization and subscribe to the US Index Option Universe History by QuantConnect data package. You need [billing permissions](#) to change the organization's subscriptions.

After you subscribe to local access, to download the US Index Option Universe data, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Index Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Download Daily Updates

After you bulk download the US Index Option 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 Index Option 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 Index Option Universe dataset, to update your local copy of the US Equity Option Universe dataset, open a terminal in your [organization workspace](#) and then run:

```
$ lean data download --dataset "US Index Option Universe" --data-type "bulk" --start "20250403" --end "20250403"
```

Alternatively, instead of directly calling the `lean data download` command, you can place the following Python script in the data directory of your organization workspace and run it to update your data files.

```

import os
from datetime import datetime, time, timedelta
from pytz import timezone
from os.path import abspath, dirname
os.chdir(dirname(abspath(__file__)))

OVERWRITE = False

def __get_start_date() -> str:
    dir_name = f"indexoption/usa/universes"
    files = [] if not os.path.exists(dir_name) else sorted(os.listdir(dir_name))
    return files[-1].split(".")[0] if files else '19980101'

def __get_end_date() -> str:
    now = datetime.now(timezone("US/Eastern"))
    if now.time() > time(7, 0):
        return (now - timedelta(1)).strftime("%Y%m%d")
    print('New data is available at 07:00 AM EST')
    return (now - timedelta(2)).strftime("%Y%m%d")

if __name__ == "__main__":
    start, end = __get_start_date(), __get_end_date()
    if start >= end:
        exit("Your data is already up to date.")

    command = f'lean data download --dataset "US Index Option Universe" --data-type "Bulk" --start {start} --end {end}'
    if OVERWRITE:
        command += " --overwrite"
    print(command)
    os.system(command)

```

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

Size and Format

The US Index Option Universe dataset is 9 GB in size. We structure the data files so there is one file per underlying Index per day. For an example, see the [Data / indexoption / usa / universes / spx / 20210112.csv](#) file in the LEAN repository.

Price

To view the price of the US Index Option Universe dataset, see [Pricing](#) .

Datasets

Brokerages

Introduction

Instead of using the data from QuantConnect, you can use data from your brokerage.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

To view the asset classes that are available for your brokerage, follow these steps:

1. Open the [Brokerages](#) page.
2. Click the name of your brokerage.
3. In the table of contents, click Asset Classes .

The brokerage data providers serve raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data in local deployments, [download the US Futures Security Master](#) .

Universe Selection

In local deployments, [universe selection](#) is available with the brokerage data provider if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. For live trading, you'll need to periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)

In cloud deployments, QuantConnect Cloud provides the universe selection datasets.

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the brokerage data provider for research, backtesting, and live trading.

Download

To download brokerage data, open a terminal in your [organization workspace](#) and then run

`lean data download --data-provider-historical <brokerageName> --data-type <data-type> --resolution <resolution> --security-type <security-type> --ticker <tickers> --market <market> --start <YYYYMMDD> --end <YYYYMMDD> <requiredBrokerageOptions>.`

```
$ lean data download --data-provider-historical "Interactive Brokers" --data-type Trade --resolution Daily --security-type Equity --ticker NVDA,AMD --market USA --start 20240303 --end 20240404 --ib-user-name userName --ib-account account --ib-password password

$ lean data download --data-provider-historical Binance --data-type Trade --resolution Daily --security-type Crypto --ticker BTCUSD,ETHUSD --start 20240303 --end 20240404 --binance-exchange-name Binance --binance-api-key apiKey --binance-api-secret apiSecret
```

To download Coinbase data, include the `--market Coinbase` option.

```
$ lean data download --data-provider-historical "Coinbase Advanced Trade" --data-type Trade --resolution hour --security-type Crypto --ticker BTCUSD --market Coinbase --start 20230101 --end 20240404 --coinbase-api-name apiName --coinbase-api-private-key apiPrivateKey
```

If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access brokerage data from the local Research Environment, open a terminal in your [organization workspace](#) and then run

`lean research <projectName> --data-provider-historical <brokerageName> <requiredBrokerageOptions>`

.

```
$ lean research "My Project" --data-provider-historical "Interactive Brokers" --ib-user-name userName --ib-account account --ib-password password
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with data from your brokerage, open a terminal in your [organization workspace](#) and then run

`lean backtest <projectName> --data-provider-historical <brokerageName> <requiredBrokerageOptions>`

.

```
$ lean backtest "My Project" --data-provider-historical "Interactive Brokers" --ib-user-name userName --ib-account account --ib-password password
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with your brokerage data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical <brokerageName> <requiredBrokerageOptions>
```

.

```
$ lean optimize "My Project" --data-provider-historical "Interactive Brokers" --ib-user-name  
userName --ib-account account --ib-password password
```

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select your brokerage and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses your brokerage as the data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-live <dataBrokerageName> <requiredDataBrokerageOptions> --brokerage <brokerageName> <requiredBrokerageOptions>
```

.

```
$ lean live deploy "My Project" --data-provider-live "Interactive Brokers" --ib-user-name userName --ib-  
account account --ib-password password --brokerage "Paper Trading"
```

Depending on the brokerage you select, you may need to provide some [required brokerage options](#) . To use a different provider for historical data, include the `--data-provider-historical` option. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

Pricing

To view the prices of your brokerage data, visit their website.

Supported Brokerages

The following table shows the brokerages that provide live (`--data-provider-live`) and historical (`--data-provider-historical`) data for research, backtesting, and optimization:

Brokerage Name	Required Options
Alpaca	<code>--alpaca-environment</code>
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>
	<code>--binance-api-secret</code> or <code>--binanceus-api-secret</code>
	<code>--binance-use-testnet</code>
Bitfinex	<code>--bitfinex-api-key</code>
	<code>--bitfinex-api-secret</code>
Bybit	<code>--bybit-api-key</code>
	<code>--bybit-api-secret</code>
	<code>--bybit-vip-level</code>
CharlesSchwab	<code>--charles-schwab-account-number</code>
Coinbase Advanced Trade	<code>--coinbase-api-name</code>
	<code>--coinbase-api-private-key</code>
Interactive Brokers	<code>--ib-user-name</code>
	<code>--ib-account</code>
	<code>--ib-password</code>
Kraken	<code>--kraken-api-key</code>
	<code>--kraken-api-secret</code>
	<code>--kraken-verification-tier</code>
Oanda	<code>--oanda-account-id</code>
	<code>--oanda-access-token</code>
	<code>--oanda-environment</code>
TradeStation	<code>--trade-station-environment</code>
	<code>--trade-station-account-id</code>

For brokerages that exclusively provide a live data feed, see the [lean live deploy](#) reference.

If you have an ARM M1, M2, or M3 chip, [Interactive Brokers](#) is not supported, see the [Troubleshooting](#) .

Datasets

Alpha Vantage

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use Alpha Vantage for historical data if you're deploying a local project. This page explains our integration with their API and its functionality. To use Alpha Vantage, you need to [get a free](#) or [premium API key](#) .

To view the implementation of the Alpha Vantage integration, see the [Lean.DataSource.AlphaVantage repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

The Alpha Vantage data provider serves asset price data directly from [Alpha Vantage's Time Series Stock Data APIs](#) . Our integration supports [US Equity](#) securities. It only provides the security price data, so you need to [download the US Equity Security Master](#) .

Universe Selection

[Universe selection](#) is available with the Alpha Vantage data provider if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. For live trading, you'll need to periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US ETF Constituents](#)
- [US Equity Coarse Universe](#)

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the Alpha Vantage data provider for research, backtesting, and live trading.

Download

To download Alpha Vantage data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical AlphaVantage --data-type Trade --resolution <resolution> --security-type EQUITY --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --alpha-vantage-api-key <apiKey> --alpha-vantage-price-plan <pricePlan>
```

.

```
$ lean data download --data-provider-historical AlphaVantage --data-type Trade --resolution Daily --security-type Equity --ticker NVDA,AMD --start 20240303 --end 20240404 --alpha-vantage-api-key apiKey --alpha-vantage-price-plan Free
```

The `--alpha-vantage-price-plan` option must be one of the following values: `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` . If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access Alpha Vantage data from the local Research Environment, open a terminal in your [organization workspace](#) and then run

```
lean research <projectName> --data-provider-historical AlphaVantage --alpha-vantage-api-key <apiKey> --alpha-vantage-price-plan <pricePlan>
```

.

```
$ lean research "My Project" --data-provider-historical AlphaVantage --alpha-vantage-api-key apiKey --alpha-vantage-price-plan Free
```

The `--alpha-vantage-price-plan` option must be one of the following values: `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` . If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with Alpha Vantage data, open a terminal in your [organization workspace](#) and then run

```
lean backtest <projectName> --data-provider-historical AlphaVantage --alpha-vantage-api-key <apiKey> --alpha-vantage-price-plan <pricePlan>
```

.

```
$ lean backtest "My Project" --data-provider-historical AlphaVantage --alpha-vantage-api-key apiKey --alpha-vantage-price-plan Free
```

The `--alpha-vantage-price-plan` option must be one of the following values: `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` . If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with Alpha Vantage data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical AlphaVantage --alpha-vantage-api-key <apiKey> --alpha-vantage-price-plan <pricePlan>
```

.

```
$ lean optimize "My Project" --data-provider-historical AlphaVantage --alpha-vantage-api-key apiKey --alpha-vantage-price-plan Free
```

The `--alpha-vantage-price-plan` option must be one of the following values: `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` .

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select Alpha Vantage and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses Alpha Vantage as the historical data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-historical AlphaVantage --alpha-vantage-api-key <apiKey> --alpha-vantage-price-plan <pricePlan> --brokerage <brokerageName> <requiredBrokerageOptions>
```

. Alpha Vantage doesn't support streaming live data. The paper trading brokerage also doesn't provide live data, so if you select it, you need to include

```
--data-provider-live <liveDataProviderName> <requiredLiveDataProviderOptions> .
```

```
$ lean live deploy "My Project" --data-provider-historical AlphaVantage --alpha-vantage-api-key apiKey --alpha-vantage-price-plan Free --brokerage "Paper Trading" --data-provider-live "Custom data only"
```

The `--alpha-vantage-price-plan` option must be one of the following values: `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` . Depending on the brokerage you select, you may need to provide some [additional required options](#) . If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

Pricing

To view the prices of the Alpha Vantage API packages, see the [Premium API Key](#) page on the Alpha Vantage website.

Datasets

FactSet

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use data from [FactSet](#) if you have the authentication JSON file. This page explains our integration with their API and its functionality.

To view the implementation of the FactSet integration, see the [Lean.DataSource.FactSet repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Authetication

To access FactSet data, ask [FactSet Support](#) for your authentication JSON file and then save it to your local machine. When you run any of the following commands, you'll provide the path to its location using the `--factset-auth-config-file` option. This file has the following [format](#) :

```
{
  "clientId": "abcde123",
  "clientAuthType": "Confidential",
  "jwk": {
    "kty": "RSA",
    "use": "sig",
    "alg": "RS256",
    "kid": "34n2j4n23j42m34kn234",
    "d": "42r34t345k3m45-34t3k45k-345v",
    "n": "42r34t345k3m45-34t3k45k-345v",
    "e": "42r34t345k3m45-34t3k45k-345v",
    "p": "42r34t345k3m45-34t3k45k-345v",
    "q": "42r34t345k3m45-34t3k45k-345v",
    "dp": "42r34t345k3m45-34t3k45k-345v",
    "dq": "42r34t345k3m45-34t3k45k-345v",
    "qi": "42r34t345k3m45-34t3k45k-345v"
  },
  "wellKnownUri": "https://auth.factset.com/.well-known/openid-configuration"
}
```

Supported Datasets

The FactSet data provider serves asset price data directly from [FactSet's Options API](#) . Our integration supports [US Index Options](#) securities.

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the FactSet data provider for research and backtesting.

Download

To download FactSet data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical FactSet --data-type <data-type> --resolution Daily  
--security-type IndexOption --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --factset-auth  
-config-file=<auth.json>
```

.

```
$ lean data download --data-provider-historical FactSet --data-type Trade --resolution Daily --security-  
type IndexOption --ticker SPX --start 20240422 --end 20240423 --factset-auth-config-file=auth.json
```

If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access FactSet data from the local Research Environment, open a terminal in your [organization workspace](#) and then run

```
lean research <projectName> --data-provider-historical FactSet --factset-auth-config-file <auth.j  
son>
```

.

```
$ lean research "My Project" --data-provider-historical FactSet --factset-auth-config-file auth.json
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with FactSet data, open a terminal in your [organization workspace](#) and then run

```
lean backtest <projectName> --data-provider-historical FactSet --factset-auth-config-file <auth.j  
son>
```

.

```
$ lean backtest "My Project" --data-provider-historical FactSet --factset-auth-config-file auth.json
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with FactSet data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical FactSet --factset-auth-config-file <auth.json>
```

.

```
$ lean optimize "My Project" --data-provider-historical FactSet --factset-auth-config-file auth.json
```

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select Polygon and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses FactSet as the historical data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-historical FactSet --factset-auth-config-file <auth.json> --brokerage <brokerageName> <requiredBrokerageOptions>
```

. FactSet doesn't support streaming live data. The paper trading brokerage also doesn't provide live data, so if you select it, you need to include

```
--data-provider-live <liveDataProviderName> <requiredLiveDataProviderOptions> .
```

```
$ lean live deploy "My Project" --data-provider-historical FactSet --factset-auth-config-file auth.json --brokerage "Paper Trading" --data-provider-live "Custom data only"
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

Pricing

To learn more about FactSet data solutions and pricing, see the [Data Solutions](#) page on the FactSet website.

Datasets

IQFeed

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use IQFeed if you're deploying a local project on Windows. This page explains our integration with their API and its functionality. To use IQFeed, you need to [create an account](#) and [install IQFeed Client](#) .

To view the implementation of the IQFeed integration, see the [Lean.DataSource.IQFeed repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

Our IQFeed integration supports securities from the following asset classes:

- [US Equity](#)
- [US Equity Options](#)
- [Forex](#) (listed on FXCM)
- [US Futures](#)

The IQFeed data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data in local deployments, [download the US Futures Security Master](#) .

Universe Selection

In local deployments, [universe selection](#) is available with the IQFeed data provider if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. For live trading, you'll need to periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US ETF Constituents](#)
- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)

In cloud deployments, QuantConnect Cloud provides the universe selection datasets.

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the IQFeed data provider for research, backtesting, and live trading.

Download

To download IQFeed data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical IQFeed --data-type <data-type> --resolution <resolution> --security-type <security-type> --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --iqfeed-username <username> --iqfeed-password <password>
```

.

```
$ lean data download --data-provider-historical IQFeed --data-type Trade --resolution Daily --security-type Equity --ticker NVDA,AMD --start 20240303 --end 20240404 --iqfeed-username username --iqfeed-password password
```

If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access IQFeed data from the local Research Environment, open a terminal in your [organization workspace](#) and then run

```
lean research <projectName> --data-provider-historical IQFeed --iqfeed-username <username> --iqfeed-password <password>
```

.

```
$ lean research "My Project" --data-provider-historical IQFeed --iqfeed-username username --iqfeed-password password
```

The `lean research` command also accepts additional [options](#) for IQFeed. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with IQFeed data, open a terminal in your [organization workspace](#) and then run

```
lean backtest <projectName> --data-provider-historical IQFeed --iqfeed-username <username> --iqfeed-password <password>
```

.

```
$ lean backtest "My Project" --data-provider-historical IQFeed --iqfeed-username username --iqfeed-password password
```

The `lean backtest` command also accepts additional [options](#) for IQFeed. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with IQFeed data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical IQFeed --iqfeed-username <username> --  
iqfeed-password <password>
```

```
$ lean optimize "My Project" --data-provider-historical IQFeed --iqfeed-username username --iqfeed-  
password password
```

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select IQFeed and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses IQFeed as the data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-live IQFeed --iqfeed-username <username> --iqfeed-  
password <password> --brokerage <brokerageName> <requiredBrokerageOptions>
```

```
$ lean live deploy "My Project" --data-provider-live IQFeed --iqfeed-username username --iqfeed-password  
password --brokerage "Paper Trading"
```

The `lean live deploy` command also accepts additional [options](#) for IQFeed. Depending on the brokerage you select, you may need to provide some [required brokerage options](#) . To use a different provider for historical data, include the `--data-provider-historical` option. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

To deploy the live algorithm through the interactive mode of the CLI, see [IQFeed](#) .

Pricing

To view the prices of the IQFeed services, see the [IQFeed Core Service Fees](#) page on the DTN IQFeed Help Site.

Datasets

Polygon

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use data from [Polygon](#) if you have an API key. To get an API key, see the [API Keys](#) page on the Polygon website

To view the implementation of the Polygon integration, see the [Lean.DataSource.Polygon repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

Our Polygon integration supports securities from the following asset classes:

- [US Equity](#)
- [US Equity Options](#)
- [US Indices](#)
- [US Index Options](#)

The Polygon data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) .

Universe Selection

In local deployments, [universe selection](#) is available with the Polygon data provider if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. For live trading, you'll need to periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)

In cloud deployments, QuantConnect Cloud provides the universe selection datasets.

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the Polygon data provider for research, backtesting, and live trading.

Download

To download Polygon data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical Polygon --data-type <data-type> --resolution <resolution> --security-type <security-type> --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --polygon-api-key <apiKey>
```

```
$ lean data download --data-provider-historical Polygon --data-type Trade --resolution Hour --security-type Equity --ticker SOFI,TSLA,NIO --start 20200101 --end 20240404 --polygon-api-key apiKey
```

If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access Polygon data from the local Research Environment, open a terminal in your [organization workspace](#) and then run `lean research <projectName> --data-provider-historical Polygon --polygon-api-key <apiKey> .`

```
$ lean research "My Project" --data-provider-historical Polygon --polygon-api-key apiKey
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with Polygon data, open a terminal in your [organization workspace](#) and then run `lean backtest <projectName> --data-provider-historical Polygon --polygon-api-key <apiKey> .`

```
$ lean backtest "My Project" --data-provider-historical Polygon --polygon-api-key apiKey
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with Polygon data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run `lean optimize <projectName> --data-provider-historical Polygon --polygon-api-key <apiKey> .`

```
$ lean optimize "My Project" --data-provider-historical Polygon --polygon-api-key apiKey
```

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select Polygon and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses Polygon as the data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-live Polygon --polygon-api-key <apiKey> --brokerage <brokerageName> <requiredBrokerageOptions>
```

.

```
$ lean live deploy "My Project" --data-provider-live Polygon --polygon-api-key apiKey --brokerage "Paper Trading"
```

Depending on the brokerage you select, you may need to provide some [required brokerage options](#) . To use a different provider for historical data, include the `--data-provider-historical` option. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

To deploy the algorithm in QuantConnect Cloud while still using Polygon as the data provider, run

```
lean cloud live deploy <projectName> --data-provider-live Polygon --polygon-api-key <apiKey> --brokerage <brokerageName> <requiredBrokerageOptions> --node <nodeName> --auto-restart <enableAutoRestarts> --notify-order-events <enableOrderEventNotifications> --notify-insights <enableInsightNotifications> <requiredNotificationOptions>
```

.

```
$ lean cloud live deploy "My Project" --data-provider-live Polygon --polygon-api-key apiKey --brokerage "Paper Trading" --node "My Node" --auto-restart yes --notify-order-events no --notify-insights no
```

To deploy the live algorithm through the interactive mode of the CLI, see [Polygon](#) .

Pricing

To view the prices of the Polygon API packages, see the [Simple Pricing](#) page on the Polygon website.

Datasets

Terminal Link

Introduction

Instead of using the data from QuantConnect, you can use data from Bloomberg EMSX through Terminal Link if you're in an organization on the [Institution tier](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

Terminal Link supports securities from the following asset classes:

- [US Equity](#)
- [US Equity Options](#)
- [US Futures](#)
- [US Index Options](#)

Terminal Link serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data in local deployments, [download the US Futures Security Master](#) .

Universe Selection

In local deployments, [universe selection](#) is available with Terminal Link if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. For live trading, you'll need to periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)

In cloud deployments, QuantConnect Cloud provides the universe selection dataset.

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the Terminal Link data provider for research, backtesting, and live trading.

Download

To download Bloomberg DAPI data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical "Terminal Link" --data-type <data-type> --resolution <resolution> --security-type <security-type> --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --terminal-link-connection-type DAPI --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey>
```

.

```
$ lean data download --data-provider-historical "Terminal Link" --data-type Trade --resolution Daily --security-type Equity --ticker NVDA,AMD --start 20240303 --end 20240404 --terminal-link-connection-type DAPI --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey
```

If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access Bloomberg DAPI data from the local Research Environment through Terminal Link, open a terminal in your [organization workspace](#) and then run

```
lean research <projectName> --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey>
```

.

```
$ lean research "My Project" --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with Terminal Link that uses data from the Bloomberg DAPI, open a terminal in your [organization workspace](#) and then run

```
lean backtest <projectName> --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey>
```

.

```
$ lean backtest "My Project" --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey
```

If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with data from Bloomberg DAPI through Terminal Link:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey>
```

.

```
$ lean optimize "My Project" --data-provider-historical "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey
```

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select Terminal Link and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses Terminal Link as the data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-live "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-account <emsxAccount> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey> --brokerage <brokerageName> <requiredBrokerageOptions>
```

.

```
$ lean live deploy "My Project" --data-provider-live "Terminal Link" --terminal-link-connection-type DAPI --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-account emsxAccount --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey --brokerage "Paper Trading"
```

Depending on the brokerage you select, you may need to provide some [required brokerage options](#) . To use a different provider for historical data, include the `--data-provider-historical` option. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

To deploy the algorithm in QuantConnect Cloud while still using Terminal Link as the data provider, run

```
lean cloud live deploy <projectName> --data-provider-live "Terminal Link" --terminal-link-environment <environment> --terminal-link-server-host <serverHost> --terminal-link-server-port <serverPort> --terminal-link-emsx-account <emsxAccount> --terminal-link-emsx-broker <emsxBroker> --terminal-link-openfigi-api-key <openfigiApiKey> --brokerage <brokerageName> <requiredBrokerageOptions> --node <nodeName> --auto-restart <enableAutoRestarts> --notify-order-events <enableOrderEventNotifications> --notify-insights <enableInsightNotifications> <requiredNotificationOptions>
```

.

```
$ lean cloud live deploy "My Project" --data-provider-live "Terminal Link" --terminal-link-environment Production --terminal-link-server-host serverHost --terminal-link-server-port serverPort --terminal-link-emsx-account emsxAccount --terminal-link-emsx-broker emsxBroker --terminal-link-openfigi-api-key openfigiApiKey --brokerage "Paper Trading" --node "My Node" --auto-restart yes --notify-order-events no --notify-insights no
```

To deploy the live algorithm through the interactive mode of the CLI, see [Deploy Local Algorithms](#) or [Deploy Cloud Algorithms](#) .

Pricing

To view the prices of Bloomberg's Execution Management Solutions, see [Execution Management System](#) on the Bloomberg website.

Datasets

Theta Data

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use Theta Data if you're deploying a local project. This page explains our integration with their API and its functionality. To use Theta Data, you need to [install and launch the Theta Terminal](#) .

To view the implementation of the Theta Data integration, see the [Lean.DataSource.ThetaData repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Supported Datasets

Our Theta Data integration supports securities from the following asset classes:

- [Equity](#)
- [Equity Options](#)
- [Index](#)
- [Index Options](#)

The Theta Data data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) .

Alternative Data

If you have licensed alternative data with QuantConnect, it works as expected with the Theta Data data provider for research, backtesting, and live trading.

Download

To download Theta Data data, open a terminal in your [organization workspace](#) and then run

```
lean data download --data-provider-historical ThetaData --data-type <data-type> --resolution <resolution> --security-type <security-type> --ticker <tickers> --start <YYYYMMDD> --end <YYYYMMDD> --thetadatasubscription-plan <plan>
```

```
$ lean data download --data-provider-historical ThetaData --data-type Trade --resolution Daily --security-type Option --ticker NVDA,AMD --start 20240303 --end 20240404 --thetadatasubscription-plan Standard
```

The `--thetadatasubscription-plan` option must be `Free` , `Value` , `Standard` , or `Pro` . If you provide your credentials, your [Lean configuration file](#) saves them.

Research

To access Theta Data data from the local Research Environment, open a terminal in your [organization workspace](#) and then run

```
lean research <projectName> --data-provider-historical ThetaData --thetadatasubscription-plan <plan>
```

.

```
$ lean research "My Project" --data-provider-historical ThetaData --thetadatasubscription-plan Standard
```

The `--thetadatasubscription-plan` option must be `Free` , `Value` , `Standard` , or `Pro` . If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean research <projectName>` to open the Research Environment with the same options.

Backtesting

To run a local backtest with data from Theta Data, open a terminal in your [organization workspace](#) and then run

```
lean backtest <projectName> --data-provider-historical ThetaData --thetadatasubscription-plan <plan>
```

.

```
$ lean backtest "My Project" --data-provider-historical ThetaData --thetadatasubscription-plan Standard
```

The `--thetadatasubscription-plan` option must be `Free` , `Value` , `Standard` , or `Pro` . If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean backtest <projectName>` to run another backtest with the same options.

Optimization

Follow these steps to run a local optimization job with data from Theta Data:

1. Add some [parameters](#) to your project.
2. Open a terminal in your [organization workspace](#) .
3. Run

```
lean optimize <projectName> --data-provider-historical ThetaData --thetadatasubscription-plan <plan>
```

.

```
$ lean optimize "My Project" --data-provider-historical ThetaData --thetadatasubscription-plan Standard
```

The `--thetadatasubscription-plan` option must be `Free` , `Value` , `Standard` , or `Pro` .

4. Follow [the steps in the interactive wizard](#) to configure your optimization job settings.

The `lean optimize` command also accepts additional [options](#) so that you can select Theta Data and run the command in non-interactive mode. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run `lean optimize <projectName>` to run another optimization job with the same options.

Live Trading

To deploy a local live algorithm that uses Theta Data as the data provider, open a terminal in your [organization workspace](#) and then run

```
lean live deploy <projectName> --data-provider-live ThetaData --thetadatasubscription-plan <plan> --brokerage <brokerageName> <requiredBrokerageOptions>
```

.

```
$ lean live deploy "My Project" --data-provider-live ThetaData --thetadatasubscription-plan Standard --brokerage "Paper Trading"
```

The `--thetadatasubscription-plan` option must be `Free`, `Value`, `Standard`, or `Pro`. Depending on the brokerage you select, you may need to provide some [required brokerage options](#). To use a different provider for historical data, include the `--data-provider-historical` option. If you provide any of the preceding options, your [Lean configuration file](#) saves them so that you only need to run

`lean live deploy <projectName> --brokerage <brokerageName>` to deploy another live algorithm with the same options.

To deploy the live algorithm through the interactive mode of the CLI, see [ThetaData](#).

The CLI doesn't currently support deploying cloud algorithms with the Theta Data data provider.

Pricing

To view the prices of the Theta Data API packages, see the [Pricing](#) page on the Theta Data website.

Projects

Projects > Project Management

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

Get Project Id

To get the project Id, open the `<organizationWorkspace> / <projectName> / config.json` file and look for the value of the `local-id` or `cloud-id` key. An example project Id is 13946911.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

The CLI only pushes the [supported file types](#) in your projects.

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):

```
import platform

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

PY

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`.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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

Encryption

Introduction

You can save encrypted versions of your project files on your local machine and in QuantConnect Cloud instead of the raw, human readable, file content. To use the encryption system, you provide your own encryption key. The encryption key file must be a txt file with at least 32 characters. It's content can be arbitrary.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Encrypt Projects

To encrypt a local project, run `lean encrypt "<projectName>" --key "<keyPath>"`.

```
$ lean encrypt "My Project" --key "C:\Users\john\qc-encryption-key.txt"
Local files encrypted successfully with key C:\Users\john\qc-encryption-key.txt
```

To [push](#) an encrypted version of a local project to QuantConnect Cloud, [add your encryption key to your browser](#) and then run `lean cloud push --project "<projectName>" --encrypt --key "<keyPath>"`. If your local project is unencrypted when you run this command, the project sent to QuantConnect Cloud is encrypted but the project on your local machine remains unencrypted. If your local project is unencrypted and has a [library](#) when you run this command, the project sent to QuantConnect Cloud is encrypted but the library sent to QuantConnect Cloud remains unencrypted.

```
$ lean cloud push --project "My Project" --encrypt --key "C:\Users\john\qc-encryption-key.txt"
[1/1] Pushing 'My Project'
Successfully created cloud project 'My Project' in organization 'd6d62db48592c72e67b534553413b691'
Successfully updated name, files, and libraries for 'My Project'
```

To [pull](#) an encrypted version of a project in QuantConnect Cloud to your local machine, run

`lean cloud pull --project "<projectName>" --encrypt --key "<keyPath>"`. If your cloud project isn't encrypted when you run this command, the project you pull to your local machine is encrypted but the project in the cloud remains unencrypted.

```
$ lean cloud pull --project "My Project" --encrypt --key "C:\Users\john\qc-encryption-key.txt"
[1/1] Pulling 'My Project'
Successfully pulled 'My Project/main.py'
Successfully pulled 'My Project/research.ipynb'
```

Decrypt Projects

To decrypt a local project, run `lean decrypt "<projectName>" --key "<keyPath>"` .

```
$ lean decrypt "My Project" --key "C:\Users\john\qc\encryption-key.txt"
Successfully decrypted project C:\Users\john\qc\workspaces\AI Capital\My Project
```

To **pull** a decrypted version of a project in QuantConnect Cloud to your local machine, run

`lean cloud pull --project "<projectName>" --decrypt --key "<keyPath>"` . If your cloud project is encrypted when you run this command, the project you pull to your local machine is decrypted but the project in the cloud remains encrypted.

```
$ lean cloud pull --project "My Project" --decrypt --key "C:\Users\john\qc\encryption-key.txt"
[1/1] Pulling 'My Project'
Successfully pulled 'My Project/main.py'
Successfully pulled 'My Project/research.ipynb'
```

To **push** a decrypted version of a local project to QuantConnect Cloud, run

`lean cloud push --project "<projectName>" --decrypt --key "<keyPath>"` . If your local project is encrypted when you run this command, the project you push to the cloud is decrypted but the project on your local machine remains encrypted.

```
$ lean cloud push --project "My Project" --decrypt --key "C:\Users\john\qc\encryption-key.txt"
[1/1] Pushing 'My Project'
Successfully created cloud project 'My Project' in organization 'd6d62db48592c72e67b534553413b691'
Successfully updated name, files, and libraries for 'My Project'
```

Collaboration Support

Encryption isn't available for projects that have **collaborators** .

Libraries

Encrypted projects can use **libraries** encrypted with the same project key or unencrypted libraries. However, you cannot use a library encrypted with a different project encryption key.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Properties

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

Property	Description
<code>description</code>	This property contains the project's description, which is displayed in backtest reports. It must always be a string.
<code>parameters</code>	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.
<code>cloud-id</code>	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.
<code>local-id</code>	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.
<code>libraries</code>	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.
<code>organization-id</code>	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.
encrypted	This property is automatically set when you encrypt or decrypt a project. It's a boolean that represents if the local project is currently encrypted.
encryption-key-path	This property is automatically set when the local project is currently encrypted. It's a string that represents the path to the file that contains the encryption key.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 *
```

PY

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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:

		PY
<code>_libgcc_mutex</code>	0.1	
<code>_openmp_mutex</code>	5.1	
<code>absl-py</code>	2.3.0	
<code>accelerate</code>	1.7.0	
<code>adagio</code>	0.2.6	
<code>aesara</code>	2.9.4	
<code>aiohappyeyeballs</code>	2.6.1	
<code>aiohttp</code>	3.12.12	
<code>aiosignal</code>	1.3.2	
<code>aiosqlite</code>	0.21.0	
<code>alembic</code>	1.16.1	
<code>alibi-detect</code>	0.12.0	
<code>alphalens-reloaded</code>	0.4.6	
<code>altair</code>	5.5.0	
<code>ampl-asl</code>	1.0.0	
<code>anaconda-anon-usage</code>	0.4.4	
<code>annotated-types</code>	0.7.0	
<code>anyio</code>	4.9.0	
<code>aplr</code>	10.9.0	
<code>appdirs</code>	1.4.4	
<code>apricot-select</code>	0.6.1	

arch	7.2.0
archspect	0.2.3
argon2-cffi	25.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
arviz	0.21.0
astropy	7.1.0
astropy-iers-data	0.2025.6.9.14.9.37
asttokens	3.0.0
astunparse	1.6.3
async-lru	2.0.5
attr	2.5.1
attrs	25.3.0
authlib	1.6.0
autograd	1.8.0
autograd-gamma	0.5.0
autokeras	2.0.0
autoray	0.7.1
ax-platform	1.0.0
babel	2.17.0
banks	2.1.2
bayesian-optimization	2.0.4
beartype	0.21.0
beautifulsoup4	4.13.4
bleach	6.2.0
blinker	1.9.0
blis	1.2.1
blosc2	3.3.4
bokeh	3.6.3
boltons	23.0.0
botorch	0.14.0
bottle	0.13.3
bottleneck	1.5.0
brotli-python	1.0.9
bzip2	1.0.8
c-ares	1.34.5
ca-certificates	2025.4.26
cachetools	5.5.2
captum	0.8.0
catalogue	2.0.10
catboost	1.2.8
category-encoders	2.8.1
causal-conv1d	1.5.0.post8
cbor	1.0.0
certifi	2025.4.26
cesium	0.12.1
cffi	1.17.1
chardet	5.2.0
charset-normalizer	3.3.2
check-shapes	1.1.1
chronos-forecasting	1.5.2
clarabel	0.11.1
click	8.2.1
clikit	0.6.2
cloudpathlib	0.21.1
cloudpickle	3.1.1
cmdstanpy	1.2.5
coin-or-cbc	2.10.12
coin-or-cgl	0.60.9
coin-or-clp	1.17.10
coin-or-osi	0.108.11
coin-or-utls	2.11.12
coincbc	2.10.12
colorama	0.4.6
colorcet	3.1.0
coloredlogs	15.0.1
colorlog	6.9.0
colorlover	0.3.0
colour	0.1.5
comm	0.2.2
conda	24.9.2
conda-content-trust	0.2.0
conda-libmamba-solver	24.9.0

conda-conda-server	24.7.0
conda-package-handling	2.3.0
conda-package-streaming	0.10.0
confection	0.1.5
cons	0.4.6
contourpy	1.3.2
control	0.10.1
copulae	0.7.9
copulas	0.12.2
coreforecast	0.0.16
cramjam	2.10.0
crashtest	0.3.1
creme	0.6.1
cryptography	43.0.0
cucim-cu12	25.4.0
cuda-bindings	12.9.0
cuda-compiler	12.3.2
cuda-cuobjdump	12.9.82
cuda-cuxxfilt	12.9.82
cuda-nvcc	12.4.131
cuda-nvdisasm	12.9.88
cuda-nvprune	12.9.82
cuda-python	12.9.0
cuda-version	12.9
cudf-cu12	25.4.0
cufflinks	0.17.3
cugraph-cu12	25.4.1
cuml-cu12	25.4.0
cuproj-cu12	25.4.0
cupy-cuda12x	13.4.1
curl-cffi	0.11.3
cuspatial-cu12	25.4.0
cuvs-cu12	25.4.0
cuxfilter-cu12	25.4.0
cvxopt	1.3.2
cvxportfolio	1.5.0
cvxpy	1.6.6
cycler	0.12.1
cymem	2.0.11
cython	3.1.2
darts	0.35.0
dash	2.9.3
dash-core-components	2.0.0
dash-cytoscape	1.0.2
dash-html-components	2.0.0
dash-table	5.0.0
dask	2025.2.0
dask-cuda	25.4.0
dask-cudf-cu12	25.4.0
databricks-sdk	0.56.0
dataclasses-json	0.6.7
datasets	2.21.0
datashader	0.18.1
deap	1.4.3
debugpy	1.6.7.post1
decorator	5.2.1
deepmerge	2.0
defusedxml	0.7.1
deprecated	1.2.18
deslib	0.3.7
dgl	2.1.0
diastatic-malt	2.15.2
dill	0.3.8
dimod	0.12.20
dirtyjson	1.0.8
diskcache	5.6.3
distributed	2025.2.0
distributed-ucxx-cu12	0.43.0
distro	1.9.0
dm-tree	0.1.9
docker	7.1.0
docutils	0.21.2
doubleml	0.10.0

dropstackframe	0.1.1
dtreeviz	2.2.2
dtw-python	1.5.3
dwave-cloud-client	0.13.4
dwave-gate	0.3.3
dwave-hybrid	0.6.14
dwave-inspector	0.5.3
dwave-inspectorapp	0.3.3
dwave-networkx	0.8.17
dwave-ocean-sdk	8.3.0
dwave-optimization	0.6.0
dwave-preprocessing	0.6.8
dwave-samplers	1.5.0
dwave-system	1.30.0
dwavebinarycsp	0.3.1
ecos	2.0.14
einops	0.8.1
einx	0.3.0
emd-signal	1.6.4
empyrical-reloaded	0.5.12
en-core-web-md	3.8.0
en-core-web-sm	3.8.0
et-xmlfile	2.0.0
etuples	0.3.9
exchange-calendars	4.10.1
executing	2.2.0
faiss-cpu	1.11.0
farama-notifications	0.0.4
fastai	2.8.2
fastai2	0.0.30
fastapi	0.115.12
fastcore	1.8.2
fastdownload	0.0.7
fasteners	0.19
fastjsonschema	2.21.1
fastparquet	2024.11.0
fastprogress	1.0.3
fastrlock	0.8.3
fasttext	0.9.3
fasttransform	0.0.2
feature-engine	1.8.3
featuretools	1.31.0
filelock	3.18.0
filetype	1.2.0
filterpy	1.4.5
findiff	0.12.1
fixedeffectmodel	0.0.5
flagembedding	1.3.5
flask	3.1.1
flatbuffers	25.2.10
fmt	9.1.0
fonttools	4.58.2
formulaic	1.1.1
fqdn	1.5.1
frozendict	2.4.2
frozenset	1.7.0
fs	2.4.16
fsspec	2024.6.1
fugue	0.9.1
future	1.0.0
fuzzy-c-means	1.7.2
gast	0.6.0
gatspy	0.3
gensim	4.3.3
geopandas	1.1.0
gevent	25.5.1
gitdb	4.0.12
gitpython	3.1.44
gluonts	0.16.1
google-auth	2.40.3
google-genai	1.19.0
google-pasta	0.2.0
gpflow	2.10.0

gplearn	0.4.2
gpytorch	1.14
graphene	3.4.3
graphql-core	3.2.6
graphql-relay	3.2.0
greenlet	3.2.3
griffe	1.7.3
grpcio	1.73.0
unicorn	23.0.0
gym	0.26.2
gym-notices	0.0.8
gymnasium	1.0.0
h11	0.16.0
h2o	3.46.0.7
h5netcdf	1.6.1
h5py	3.14.0
hf-xet	1.1.3
hmmlearn	0.3.3
holidays	0.74
holoviews	1.20.2
homebase	1.0.1
hopcroftkarp	1.2.5
httpcore	1.0.9
httpstan	4.13.0
httpx	0.28.1
httpx-sse	0.4.0
huggingface-hub	0.33.0
humanfriendly	10.0
hurst	0.0.5
hvplot	0.11.3
hyperopt	0.2.7
ibm-cloud-sdk-core	3.24.2
ibm-platform-services	0.66.2
idna	3.7
iisignature	0.24
ijson	3.4.0
imageio	2.37.0
imbalanced-learn	0.13.0
immutabledict	4.2.1
importlib-metadata	8.7.0
importlib-resources	6.5.2
iniconfig	2.1.0
injector	0.22.0
inscriptis	2.6.0
interface-meta	1.3.0
interpret	0.6.11
interpret-core	0.6.11
ipopt	3.14.17
ipykernel	6.29.5
ipynb	0.9.7
ipython	9.3.0
ipython-pygments-lexers	1.1.1
ipywidgets	8.1.7
ir-datasets	0.5.10
isoduration	20.11.0
itsdangerous	2.2.0
jax	0.5.3
jaxlib	0.5.3
jaxtyping	0.3.2
jedi	0.19.2
jinja2	3.1.6
jiter	0.10.0
joblib	1.4.2
json5	0.12.0
jsonpatch	1.33
jsonpath-ng	1.7.0
jsonpointer	2.1
jsonschema	4.24.0
jsonschema-specifications	2025.4.1
jupyter	1.1.1
jupyter-ai	2.31.5
jupyter-ai-magics	2.31.5
jupyter-bokeh	4.0.5

jupyter-boken	4.0.0
jupyter-client	8.6.3
jupyter-console	6.6.3
jupyter-core	5.8.1
jupyter-events	0.12.0
jupyter-lsp	2.2.5
jupyter-resource-usage	1.1.1
jupyter-server	2.16.0
jupyter-server-proxy	4.4.0
jupyter-server-terminals	0.5.3
jupyterlab	4.4.3
jupyterlab-pygments	0.3.0
jupyterlab-server	2.27.3
jupyterlab-widgets	3.0.15
kagglehub	0.3.12
kaleido	0.2.1
kdepy	1.1.12
keras	3.10.0
keras-hub	0.21.1
keras-nlp	0.21.1
keras-rl	0.4.2
keras-tcn	3.5.6
keras-tuner	1.4.7
keyutils	1.6.1
kiwisolver	1.4.8
kmapper	2.1.0
korean-lunar-calendar	0.3.1
krb5	1.21.3
kt-legacy	1.0.5
langchain	0.3.25
langchain-community	0.3.25
langchain-core	0.3.65
langchain-text-splitters	0.3.8
langcodes	3.5.0
langsmith	0.3.45
language-data	1.3.0
lark	1.2.2
lazy-loader	0.4
lazypredict	0.2.16
ld_impl_linux-64	2.40
libarchive	3.7.7
libblas	3.9.0
libcap	2.75
libcbblas	3.9.0
libclang	18.1.1
libcudf-cu12	25.4.0
libcugraph-cu12	25.4.1
libcuml-cu12	25.4.0
libcurl	8.14.1
libcuspatal-cu12	25.4.0
libcups-cu12	25.4.0
libedit	3.1.20230828
libev	4.33
libevent	2.1.12
libexpat	2.7.0
libfabric	2.1.0
libfabric1	2.1.0
libffi	3.4.6
libgcc	15.1.0
libgcc-ng	15.1.0
libgcrypt-lib	1.11.1
libgfortran	15.1.0
libgfortran-ng	15.1.0
libgfortran5	15.1.0
libgomp	15.1.0
libgpg-error	1.55
libhwloc	2.11.2
libiconv	1.18
libkvikio-cu12	25.4.0
liblapack	3.9.0
liblapacke	3.9.0
liblzma	5.8.1
liblzma-devel	5.8.1

libmamba	1.5.8
libmambapy	1.5.8
libnghttp2	1.64.0
libnl	3.11.0
libnsl	2.0.1
libopenblas	0.3.29
libpmix	5.0.8
libraft-cu12	25.4.0
librmm-cu12	25.4.0
libscotch	7.0.4
libsolv	0.7.33
libspral	2025.03.06
libsqlite	3.50.1
libssh2	1.11.1
libstdcxx	15.1.0
libstdcxx-ng	15.1.0
libsystemd0	257.6
libucx-cu12	1.18.1
libucxx-cu12	0.43.0
libudev1	257.6
libuuid	2.38.1
libxcrypt	4.4.36
libxml2	2.13.8
libzlib	1.3.1
lifelines	0.30.0
lightgbm	4.6.0
lightning	2.5.1.post0
lightning-utilities	0.14.3
lightweight-charts	2.1
lime	0.2.0.1
line-profiler	4.2.0
linear-operator	0.6
linkify-it-py	2.0.3
livelossplot	0.5.6
llama-cloud	0.1.26
llama-cloud-services	0.6.31
llama-index	0.12.41
llama-index-agent-openai	0.4.11
llama-index-cli	0.4.3
llama-index-core	0.12.42
llama-index-embeddings-openai	0.3.1
llama-index-indices-managed-llama-cloud	0.7.7
llama-index-llms-openai	0.4.5
llama-index-multi-modal-llms-openai	0.5.1
llama-index-program-openai	0.3.2
llama-index-question-gen-openai	0.3.1
llama-index-readers-file	0.4.9
llama-index-readers-llama-parse	0.4.0
llama-parse	0.6.31
llvmlite	0.42.0
locket	1.0.0
logical-unification	0.4.6
loguru	0.7.3
lxml	5.4.0
lz4	4.4.4
lz4-c	1.10.0
lzo	2.10
mako	1.3.10
mamba-ssm	2.2.4
mapie	1.0.1
marisa-trie	1.2.1
markdown	3.8
markdown-it-py	3.0.0
markupsafe	3.0.2
marshmallow	3.26.1
matplotlib	3.8.4
matplotlib-inline	0.1.7
mdit-py-plugins	0.4.2
mdurl	0.1.2
menuinst	2.1.2
metis	5.1.0
mgarch	0.3.0
minikanren	1.0.3

minorminer	0.2.18
mistune	3.1.3
ml-dtypes	0.4.1
mlflow	3.1.0
mlflow-skinny	3.1.0
mlforecast	1.0.2
mljar-scikit-plot	0.3.12
mljar-supervised	1.1.17
mlxtend	0.23.4
mmh3	2.5.1
modin	0.33.1
mpi	1.0
mplfinance	0.12.10b0
mpmath	1.3.0
msgpack	1.1.0
multidict	6.4.4
multipledispatch	1.0.0
multiprocess	0.70.16
multitasking	0.0.11
mumps-include	5.7.3
mumps-seq	5.7.3
murmurhash	1.0.13
mypy-extensions	1.1.0
namex	0.1.0
narwhals	1.42.1
nbclient	0.10.2
nbconvert	7.16.6
nbformat	5.10.4
ncurses	6.5
ndindex	1.10.0
nest-asyncio	1.6.0
networkx	3.5
neural-tangents	0.6.5
neuralforecast	3.0.1
neuralprophet	0.9.0
nfoursid	1.0.1
ngboost	0.5.6
ninja	1.11.1.4
nixtla	0.6.6
nlTK	3.9.1
nolds	0.6.2
nose	1.3.7
notebook	7.4.3
notebook-shim	0.2.4
numba	0.59.1
numba-cuda	0.4.0
numerapi	2.20.6
numexpr	2.11.0
numpy	1.26.4
nvidia-cublas-cu12	12.4.5.8
nvidia-cuda-cupti-cu12	12.4.127
nvidia-cuda-nvrtc-cu12	12.4.127
nvidia-cuda-runtime-cu12	12.4.127
nvidia-cudnn-cu12	9.3.0.75
nvidia-cufft-cu12	11.2.1.3
nvidia-curand-cu12	10.3.5.147
nvidia-cusolver-cu12	11.6.1.9
nvidia-cusparse-cu12	12.3.1.170
nvidia-ml-py	12.575.51
nvidia-nccl-cu12	2.21.5
nvidia-nvcomp-cu12	4.2.0.11
nvidia-nvjitlink-cu12	12.4.127
nvidia-nvtx-cu12	12.4.127
nvtx	0.2.12
nx-cugraph-cu12	25.4.0
oauthlib	3.2.2
onnx	1.18.0
onnx-mltools	1.14.0
onnxruntime	1.22.0
openai	1.86.0
opencv-contrib-python-headless	4.11.0.86
opencv-python	4.11.0.86
openmpi	5.0.8

openmpi	3.0.0
openpyxl	3.1.5
openssl	3.5.0
opentelemetry-api	1.34.1
opentelemetry-sdk	1.34.1
opentelemetry-semantic-conventions	0.55b1
opt-einsum	3.4.0
optree	0.16.0
optuna	4.3.0
optuna-integration	4.3.0
orjson	3.10.18
ormsgpack	1.7.0
ortools	9.11.4210
osqp	0.6.7.post3
overrides	7.7.0
packaging	24.1
pandas	2.2.3
pandas-flavor	0.7.0
pandas-market-calendars	5.1.0
pandas-ta	0.3.14b0
pandocfilters	1.5.1
panel	1.7.1
param	2.2.1
parso	0.8.4
partd	1.4.2
pastel	0.2.1
pathos	0.3.2
patsy	1.0.1
pbr	6.1.1
peewee	3.17.3
peft	0.15.2
penaltymodel	1.2.0
pennylane	0.41.1
pennylane-lightning	0.41.1
pennylane-qiskit	0.36.0
persim	0.3.8
pexpect	4.9.0
pgmpy	1.0.0
pillow	10.4.0
pingouin	0.5.5
platformdirs	4.3.8
plotly	5.24.1
plucky	0.4.3
pluggy	1.6.0
plum-dispatch	2.5.7
ply	3.11
pmdarima	2.0.4
polars	1.30.0
pomegranate	1.1.2
pot	0.9.5
pox	0.3.6
ppft	1.7.7
pprofile	2.2.0
preshed	3.0.10
prometheus-client	0.22.1
prompt-toolkit	3.0.51
propcache	0.3.2
prophet	1.1.7
protobuf	5.26.1
proxy-tools	0.1.0
psutil	5.9.8
ptyprocess	0.7.0
pulp	3.2.1
pure-eval	0.2.3
py-cpuinfo	9.0.0
py-heat	0.0.6
py-heat-magic	0.0.2
py-lets-be-rational	1.0.1
py-volib	1.0.1
py4j	0.10.9.9
pyaml	25.5.0
pyarrow	19.0.1
pyasn1	0.6.1

pyasn1-modules	0.4.2
pybind11	2.13.6
pybind11-abi	4
pycosat	0.6.6
pycparser	2.21
pyct	0.5.0
pydantic	2.11.5
pydantic-core	2.33.2
pydantic-settings	2.9.1
pydmd	2025.6.1
pyerfa	2.0.1.5
pyfolio-reloaded	0.9.9
pygments	2.19.1
pyjwt	2.10.1
pykalman	0.10.1
pylev	1.4.0
pylibcudf-cu12	25.4.0
pylibcugraph-cu12	25.4.1
pylibraft-cu12	25.4.0
pyluach	2.2.0
pymannkendall	1.4.3
pymc	5.23.0
pymdptoolbox	4.0b3
pynndescent	0.5.13
pynvjitlink-cu12	0.6.0
pynvml	12.0.0
pyod	2.0.5
pyogrio	0.11.0
pyomo	6.9.2
pyarsing	3.2.3
pypdf	5.6.0
pyportfolioopt	1.5.6
pyproj	3.7.1
pyqt6	6.9.1
pyqt6-qt6	6.9.1
pyqt6-sip	13.10.2
pyrb	1.0.1
pyre-extensions	0.0.32
pyro-api	0.1.2
pyro-ppl	1.9.1
pysimdjson	6.0.2
pysocks	1.7.1
pyspnego	0.11.2
pystan	3.10.0
pytensor	2.31.3
pytest	8.4.0
pytest-runner	6.0.1
python	3.11.13
python-dateutil	2.9.0.post0
python-dotenv	1.1.0
python-graphviz	0.20.3
python-json-logger	3.3.0
python-statemachine	2.5.0
python_abi	3.11
pytorch-forecasting	1.3.0
pytorch-ignite	0.5.2
pytorch-lightning	2.5.1.post0
pytorch-tabnet	4.1.0
pytz	2025.2
pyvinecopulib	0.6.5
pyviz-comms	3.0.4
pywavelets	1.8.0
pywebview	5.4
pyyaml	6.0.2
pyzmq	26.4.0
qdlDL	0.1.7.post5
qiskit	2.0.2
qiskit-aer	0.17.1
qiskit-ibm-provider	0.11.0
qiskit-ibm-runtime	0.40.1
quadprog	0.1.13
quantecon	0.8.1
quantlib	1.38

quantstats	0.0.64
raft-dask-cu12	25.4.0
rapids-dask-dependency	25.4.0
rapids-logger	0.1.1
rauth	0.7.3
ray	2.46.0
rbeast	0.1.23
rdma-core	57.0
readline	8.2
referencing	0.36.2
regex	2024.11.6
reproc	14.2.4
reproc-cpp	14.2.4
requests	2.32.4
requests-ntlm	1.3.0
requests-oauthlib	2.0.0
requests-toolbelt	1.0.0
rfc3339-validator	0.1.4
rfc3986-validator	0.1.1
rich	14.0.0
ripser	0.6.12
riskfolio-lib	7.0.1
riskparityportfolio	0.6.0
river	0.21.0
rmm-cu12	25.4.0
rpds-py	0.25.1
rsa	4.9.1
ruamel.yaml	0.18.6
ruamel.yaml.clib	0.2.8
ruptures	1.1.9
rustworkx	0.16.0
safetensors	0.5.3
salib	1.5.1
scikeras	0.13.0
scikit-base	0.12.3
scikit-image	0.22.0
scikit-learn	1.6.1
scikit-learn-extra	0.3.0
scikit-optimize	0.10.2
scikit-tda	1.1.1
scipy	1.11.4
scipy-openblas32	0.3.29.265.1
scs	3.2.7.post2
sdeint	0.3.0
seaborn	0.13.2
send2trash	1.8.3
sentence-transformers	4.1.0
sentencepiece	0.2.0
setuptools	73.0.1
setuptools-scm	8.3.1
shap	0.47.2
shapely	2.0.7
shimmy	2.0.0
simpervisor	1.0.0
simplejson	3.20.1
simpy	4.1.1
six	1.17.0
skfolio	0.7.0
skl2onnx	1.19.1
sklearn-compat	0.1.3
sklearn-json	0.1.0
sktime	0.37.0
slicer	0.0.8
smart-open	7.1.0
smmmap	5.0.2
sniffio	1.3.1
sortedcontainers	2.4.0
soupsieve	2.7
spacy	3.8.7
spacy-legacy	3.0.12
spacy-loggers	1.0.5
sqlalchemy	2.0.41
sqlparse	0.5.3

supabase	0.0.0
srsly	2.5.1
ssm	0.0.1
stable-baselines3	2.6.0
stack-data	0.6.3
stanio	0.5.1
starlette	0.46.2
statsforecast	2.0.1
statsmodels	0.14.4
stevedore	5.4.1
stochastic	0.6.0
stockstats	0.6.5
stopit	1.1.2
stripptf	0.0.26
stumpy	1.13.0
sweetviz	2.3.1
symengine	0.13.0
sympy	1.13.1
ta	0.11.0
ta-lib	0.6.4
tables	3.10.2
tabulate	0.9.0
tadatasets	0.2.1
tblib	3.1.0
tenacity	9.1.2
tensorboard	2.18.0
tensorboard-data-server	0.7.2
tensorboardx	2.6.4
tensordict	0.6.2
tensorflow	2.18.0
tensorflow-decision-forests	1.11.0
tensorflow-io-gcs-filesystem	0.37.1
tensorflow-probability	0.25.0
tensorflow-text	2.18.1
tensorly	0.9.0
tensorrt	10.11.0.33
tensorrt-cu12	10.11.0.33
tensorrt-cu12-bindings	10.11.0.33
tensorrt-cu12-libs	10.11.0.33
tensortrade	1.0.3
termcolor	3.1.0
terminado	0.18.1
tf-keras	2.18.0
tf2jax	0.3.7
tf2onnx	1.16.1
thinc	8.3.4
threadpoolctl	3.6.0
thundergbm	0.3.17
tifffile	2025.6.11
tigramite	5.2.7.0
tiktoken	0.9.0
tinycss2	1.4.0
tinygrad	0.10.3
tk	8.6.13
tokenizers	0.21.1
toml	0.10.2
tomlkit	0.13.3
toolz	0.12.1
torch	2.5.1
torch-cluster	1.6.3
torch-geometric	2.6.1
torch-scatter	2.1.2
torch-sparse	0.6.18
torch-spline-conv	1.2.2
torchdata	0.11.0
torchmetrics	1.7.2
torchrl	0.6.0
torchvision	0.20.1
tornado	6.5.1
tpot	0.12.2
tqdm	4.66.5
traitlets	5.14.3
transformers	4.52.4

trec-car-tools	2.6
treelite	4.4.1
triad	0.9.8
triton	3.1.0
truststore	0.8.0
tsfel	0.1.9
tsfresh	0.20.2
tslearn	0.6.3
tweepy	4.15.0
typer	0.9.4
typer-config	1.4.2
types-python-dateutil	2.9.0.20250516
typing-extensions	4.14.0
typing-inspect	0.9.0
typing-inspection	0.4.1
tzdata	2025.2
uc-micro-py	1.0.3
ucc	1.4.4
ucx	1.18.1
ucx-py-cu12	0.43.0
ucxx-cu12	0.43.0
umap-learn	0.5.7
unlzw3	0.2.3
update-checker	0.18.0
uri-template	1.3.0
urllib3	2.4.0
utilsforecast	0.2.12
uvicorn	0.34.3
wadler-lindig	0.1.6
warc3-wet	0.2.5
warc3-wet-clueweb09	0.2.5
wasabi	1.1.3
wcwidth	0.2.13
weasel	0.4.1
webargs	8.7.0
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
websockets	15.0.1
werkzeug	3.1.3
wheel	0.44.0
widgetsnextextension	4.0.14
woodwork	0.31.0
wordcloud	1.9.4
wrapt	1.17.2
wurlitzer	3.1.1
x-transformers	2.3.12
xarray	2025.6.0
xarray-einstats	0.9.0
xgboost	3.0.2
xlrd	2.0.1
xlsxwriter	3.2.3
xxhash	3.5.0
xyzservices	2025.4.0
xz	5.8.1
xz-gpl-tools	5.8.1
xz-tools	5.8.1
yaml-cpp	0.8.0
yaml	1.20.1
ydf	0.9.0
yellowbrick	1.5
yfinance	0.2.63
zict	3.0.0
zipp	3.23.0
zlib	1.3.1
zlib-state	0.1.9
zope-event	5.0
zope-interface	7.2
zstandard	0.23.0
zstd	1.5.7

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
	botocore	1.20.72	pypi_0	pypi
	brctlipy	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	py36hcf9a06_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
immutable	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
libcbblas	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
pyarsing	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
pysistent	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-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	py36hfb0944_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

Request New Libraries

To request a new library, [contact us](#) . We will add the library to the queue for review and deployment. Since the libraries run on our servers, we need to ensure they are secure and won't cause harm. The process of adding new libraries takes 2-4 weeks to complete. View the list of libraries currently under review on the [Issues list of the Lean GitHub repository](#) .

Add 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 3.11 (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.

Additionally, you can also add 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.

Remove Libraries

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 remove custom Python libraries by modifying the project's requirements.txt file.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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. To create a library directory, set the name to Library / directoryName / libraryName (for example, Library / Tools / Calculators).

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
```

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)
```

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. To create a library directory, set the name to Library / directoryName / libraryName (for example, Library / Tools / Calculators).

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

Projects

Version Control

Introduction

Version control is the practice of tracking and managing changes to code files. By using version control, you can save an extra back up of your project files in the cloud, keep a history of all code changes, and easily revert changes to your projects.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Create Workspace Repositories

Follow these steps to set up a new version control repository for one of your [organization workspaces](#) :

1. In your version control system, [create a new repository](#) for the organization workspace.
2. Open a terminal in your organization workspace and then [clone](#) the new repository to a temporary directory.

```
$ git clone https://github.com/<userName>/<repoName>.git temp
```

3. Move the .git directory from the temporary directory to the workspace directory.

```
$ mv temp/.git <workspaceDirectory>/.git
```

4. Delete the temporary directory.

```
$ rm -r temp
```

Push Changes to Git

Follow these steps to push the changes of your organization workspace to your version control system:

1. [Pull all your cloud projects](#) , creating directories where necessary.

```
$ lean cloud pull
```

2. Add the project directories and the Library .

```
$ git add Library/  
$ git add <projectDirectory1>/  
$ git add <projectDirectory2>/
```

3. Commit the changes.

```
$ git commit -am "Latest Updates"
```

4. Push the changes to the repository.

```
$ git push
```

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 on your local machine.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Start the 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. To start a local research environment with your current configuration, open a terminal in one of your [organization workspaces](#) and then 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
```

JupyterLab automatically opens in your default browser. If you intend to use PyCharm or VS Code instead of JupyterLab, include the `--no-open` option to not open JupyterLab. To run the environment on a different port by providing the `--port <port>` option.

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 latest 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
```

Data Providers


When you run the Research Environment, the default data provider is your local machine. To use data from QuantConnect Cloud, a brokerage, or a third-party data provider, include the `--data-provider-historical` option. To view what brokerages and third-party data providers are available, see [the reference page for this command](#).

Use JupyterLab

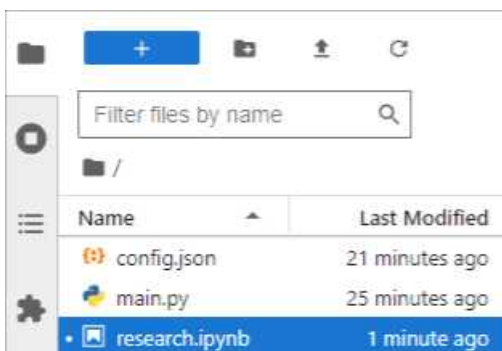
The following sections explain how to operate the Research Environment with [JupyterLab](#), the default IDE.

Open Notebooks

Follow these steps to open a research notebook in JupyterLab:

1. [Start a local research environment](#) for the project that contains the notebook.
2. In the left navigation menu, click the  File Browser icon.
3. In the File Browser panel, double-click the notebook file you want to open.

The default notebook is `research.ipynb`.



For more information about opening files in JupyterLab, see [Opening Files](#) in the JupyterLab documentation.

Run Notebook Cells

Notebooks are a collection of cells where you can write code snippets or Markdown. To execute a cell, press `Shift+Enter`.

```
[1] 3+7
10

[2] print("Hello world!")
Hello world!
```


The following describes some helpful keyboard shortcuts to speed up your research:

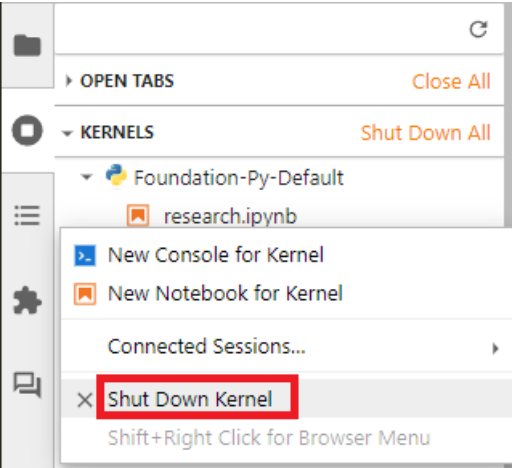
Keyboard Shortcut	Description
Shift+Enter	Run the selected cell.
a	Insert a cell above the selected cell.
b	Insert a cell below the selected cell.
x	Cut the selected cell.
v	Paste the copied or cut cell.
z	Undo cell actions.

For more information about keyboard shortcuts in JupyterLab, see [Keyboard Shortcuts](#) in the JupyterLab documentation.

Stop Nodes

Follow these steps to stop a research node in JupyterLab:

1. In the left navigation menu, click the  Running Terminals and Kernels icon.
2. In the Kernals section of the Running Terminals and Kernels panel, right-click the name of the notebook file and then click Shut Down Kernel from the drop-down menu.



For more information about the Running Terminals and Kernels panel in JupyterLab, see [Managing Kernels and](#)

[Terminals](#) in the JupyterLab documentation.

Add Notebooks


Follow these steps to add notebook files to a project in JupyterLab:

1. [Start a local research environment](#) for the project.
2. In the top navigation menu, click File > New > Notebook .
3. In the Select Kernel window, click Select .

For more information about the creating files in JupyterLab, see [Creating Files and Activities](#) in the JupyterLab documentation.


Rename Notebooks

Follow these steps to rename notebook files in JupyterLab:

1. [Start a local research environment](#) for the project that contains the notebook.
2. In the left navigation menu, click the  File Browser icon.
3. In the File Browser panel, right-click the notebook file you want to rename and then click Rename from the drop-down menu.
4. Enter the new name.
5. Press Enter .

Delete Notebooks

Follow these steps to delete notebook files in JupyterLab:

1. [Start a local research environment](#) for the project that contains the notebook.
2. In the left navigation menu, click the  File Browser icon.
3. In the File Browser panel, right-click the notebook file you want to delete and then click Delete from the drop-down menu.
4. In the Delete window, click Delete .


Use PyCharm

The following sections explain how to operate the Research Environment with [PyCharm](#) .

Open Notebooks

Follow these steps to open a research notebook in PyCharm:

1. [Start a local research environment](#) for the project that contains the notebook.
2. Open the same project in PyCharm.
3. In the top navigation menu of PyCharm, click File > Settings .
4. In the Settings window, click Language & Frameworks > Jupyter > Jupyter Servers .
5. Click Configured Server .
6. Enter `http://localhost:8888/?token=` into the field.
7. Click Apply to save the changes.

8. Click OK to exit the window.
9. In the left navigation menu, click the  Project icon.
10. In the Project panel, double-click the notebook file you want to open.

The default notebook is research.ipynb .

Run Notebook Cells

Notebooks are a collection of cells where you can write code snippets or Markdown. To execute a cell, press **Shift+Enter** .

```
[1] 3+7
... 10
Python
```

```
[2] print("Hello world!")
... Hello world!
Python
```


The following describes some helpful keyboard shortcuts to speed up your research:

Keyboard Shortcut	Description
Shift+Enter	Run the selected cell.
a	Insert a cell above the selected cell.
b	Insert a cell below the selected cell.
x	Cut the selected cell.
v	Paste the copied or cut cell.
z	Undo cell actions.

For more information about keyboard shortcuts in PyCharm, see [Keyboard Shortcuts](#) in the PyCharm documentation.

Stop Nodes

Follow these steps to stop a research node in PyCharm:

1. In the left navigation menu, click the  Project icon.
2. In the Project panel, right-click the name of the notebook file and then click Shutdown Kernel from the drop-down menu.


For more information about the Jupyter notebook servers in PyCharm, see [Manage Jupyter notebook servers](#) in the PyCharm documentation.

Add Notebooks

To add notebook files to a project in PyCharm, see [Create a notebook file](#) in the PyCharm documentation.


Rename Notebooks

Follow these steps to rename notebook files in PyCharm:

1. [Start a local research environment](#) for the project in PyCharm.
2. In the left navigation menu, click the  Project icon.
3. In the Project panel, right-click the notebook file you want to rename and then click Refactor > Rename from the drop-down menu.
4. Enter the new name.
5. Press Enter .

Delete Notebooks

Follow these steps to delete notebook files in PyCharm:

1. [Start a local research environment](#) for the project in PyCharm.
2. In the left navigation menu, click the  Project icon.
3. In the Project panel, right-click the notebook file you want to rename and then click Delete from the drop-down menu.
4. In the Delete window, click OK .

Use VS Code

The following sections explain how to operate the Research Environment with VS Code.

Open Notebooks

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

1. [Start a local research environment](#) for the project that contains the notebook.
2. Open the same project VS Code.
3. In the Explore panel, click the notebook file you want to open.

The default notebook is `research.ipynb` .

4. In the top-right corner of the notebook, click Select Kernel .
5. In the Select Another Kernel window, click Existing Jupyter Server... .
6. Enter `http://localhost:8888/` into the field.
7. Press Enter .
8. (Optional) Enter a display name for the server and then press Enter .
9. Click Foundation-Py-Default .

Run Notebook Cells

Notebooks are a collection of cells where you can write code snippets or Markdown. To execute a cell, press

Shift+Enter .

```
[1] 3+7 Python
... 10

[2] print("Hello world!") Python
... Hello world!
```



The following describes some helpful keyboard shortcuts to speed up your research:

Keyboard Shortcut	Description
Shift+Enter	Run the selected cell.
a	Insert a cell above the selected cell.
b	Insert a cell below the selected cell.
x	Cut the selected cell.
v	Paste the copied or cut cell.
z	Undo cell actions.

For more information about keyboard shortcuts in VS Code, see [Key Bindings for Visual Studio Code](#) in the VS Code documentation.



Add Notebooks

Follow these steps to add notebook files to a project in VS Code:

1. [Open the project](#) .
2. In the right navigation menu, click the  Explorer icon.
3. In the Explorer panel, expand the QC (Workspace) section.
4. Click the  New File icon.
5. Enter fileName .ipynb .
6. Press Enter .

Rename Notebooks


Follow these steps to rename notebook files in VS Code:

1. [Start a local research environment](#) for the project that contains the notebook.
2. In the right navigation menu, click the  Explorer icon.
3. In the Explorer panel, expand the QC (Workspace) section.
4. Click the  New File icon.

5. Enter fileName .ipynb .
6. Press Enter .

Delete Notebooks

Follow these steps to delete notebook files in VS Code:

1. [Start a local research environment](#) for the project that contains the notebook.
2. In the right navigation menu, click the  Explorer icon.
3. In the Explorer panel, right-click the notebook you want to rename and then click Rename .
4. Enter the new name and then press Enter .

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):

```
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}")
```

PY

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:

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

project = next(p for p in api.list_projects().projects if p.name == project_name)
partial_backtest = next(b for b in api.list_backtests(project.project_id).backtests if b.name == backtest_name)
backtest = api.read_backtest(project.project_id, partial_backtest.backtest_id)

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

PY

Backtesting

Backtesting > Deployment

Backtesting

Deployment

Introduction

Backtesting is a way to test your algorithm on historical data. The CLI makes backtesting easier by providing simple commands to backtest your algorithms locally or in QuantConnect Cloud.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
```

Data Providers

When you run a local backtest, the default data provider is your local machine. To use data from [QuantConnect Cloud](#) , a brokerage, or a third-party data provider, include the `--data-provider-historical` option. To view what brokerages and third-party data providers are available, see [the reference page for this command](#) .

Required Datasets

To run backtests with Equity or Equity Option data, download the [download the US Equity Security Master](#) .

To run backtests with Futures data, [download the US Futures Security Master](#) .

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"
```

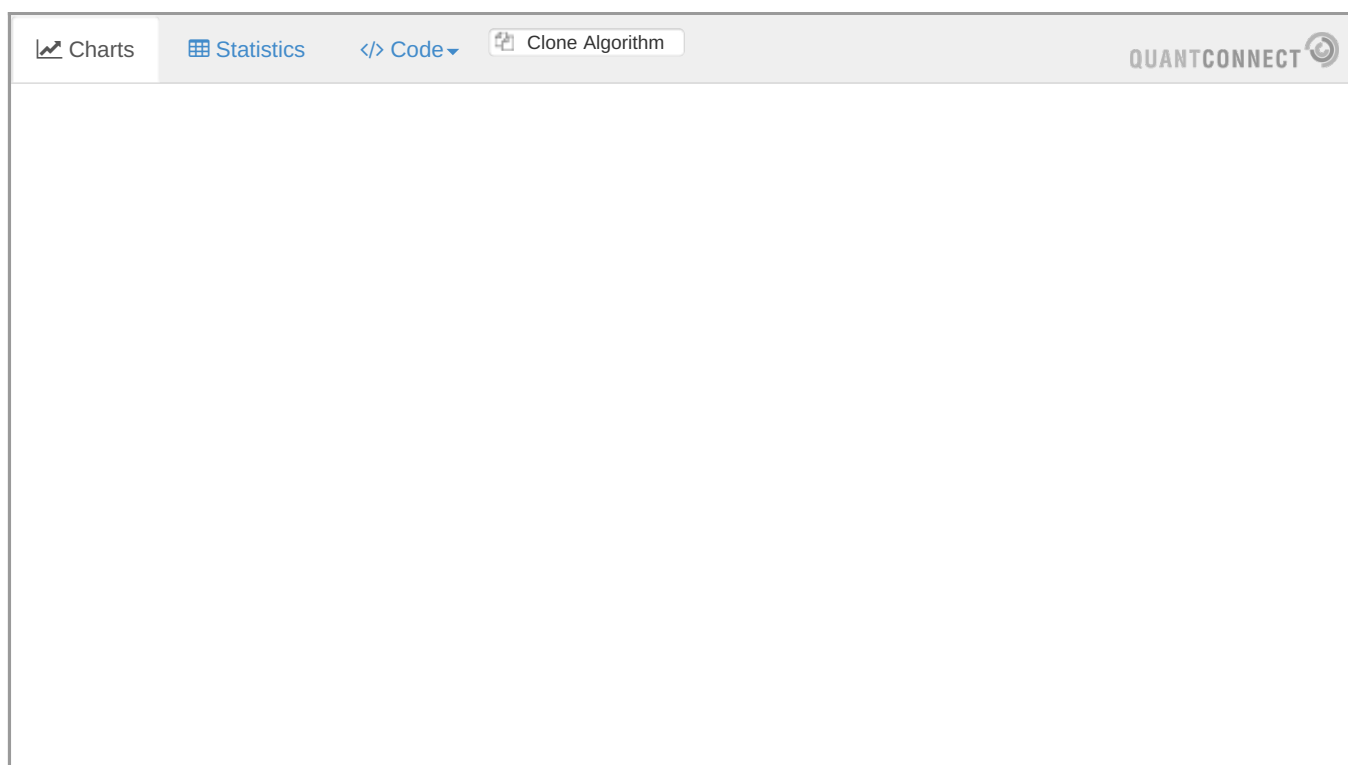
3. Download minute resolution trade and quote data from the [US Equity Options](#) dataset.

```
$ 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

When you run a backtest in QuantConnect Cloud, it uses the data from the [Dataset Market](#) . 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 the data you need before you run a 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 won't have to download it again. If the files contain data for multiple days (for example, daily Equity price data files), the `ApiDataProvider` re-downloads the files if your local files are at least 7 days old. To adjust this setting, update the `downloader-data-update-period` value in your `Lean configuration` file.

Follow these steps to use the `ApiDataProvider` to automatically download the data you need:

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-historical 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-historical 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-historical` 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.

Get Backtest Id

To get the Id of a local backtest, check the name of the `<organizationWorkspace> / <projectName> / <deploymentTimestamp> / <backtestId>.json` file. An example local backtest Id is 1710698424.

To get the Id of a cloud backtest, check the output of the `lean cloud backtest` command in the terminal. If you no longer have the output, [get the backtest Id from the Algorithm Lab](#) . An example cloud backtest Id is 8b16cec0c44f75188d82f9eadb310e17.

Backtesting

Debugging

Introduction

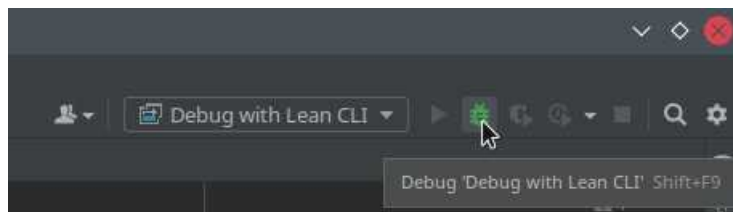
Debugging is an important part of writing any algorithm. The CLI makes it easy to use the builtin debugger of the most popular editors to debug LEAN algorithms. This page explains how to start local debugging for Python and C# with all editors supported by the CLI.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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. Open a terminal in your organization workspace and run `lean backtest "<projectName>" --debug pycharm`.

```
$ 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.

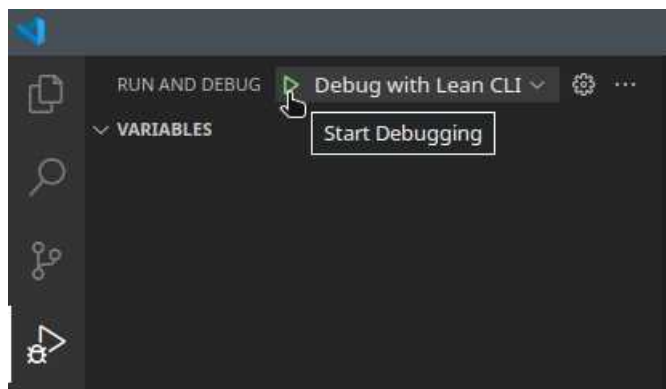
Python and VS Code

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. Open a terminal in your organization workspace, run `lean backtest "<projectName>" --debug debugpy` , and then wait until the CLI tells you to attach to the debugger.

```
$ lean backtest "My Project" --debug debugpy
20240507 20:35:54.970 TRACE:: Engine.Main(): LEAN ALGORITHMIC TRADING ENGINE v2.5.0.0 Mode: DEBUG (64bit)
20240507 20:35:55.272 TRACE:: Engine.Main(): Started 8:35 PM
20240507 20:36:01.104 TRACE:: DebuggerHelper.Initialize(): debugpy waiting for attach at port 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. Open a terminal in your organization workspace, run `lean backtest "<projectName>" --debug vsdbg` , 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. Open a terminal in your organization workspace, run `lean backtest "<projectName>" --debug rider` , 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.

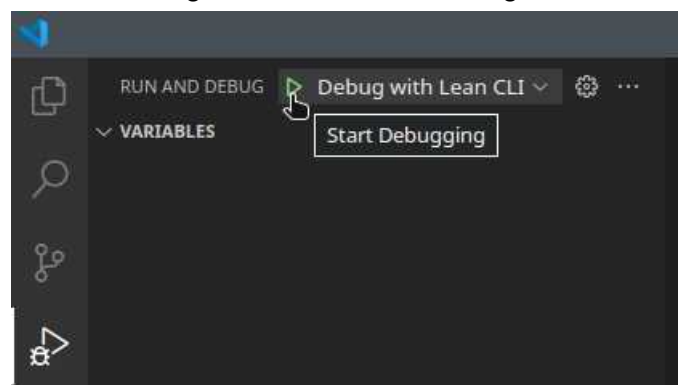
C# and VS Code

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. Open a terminal in your organization workspace, run `lean backtest "<projectName>" --debug vsdbg` , 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

Live Trading

Brokerages

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

[QuantConnect Paper Trading](#)

[US Equities, FOREX, CFD, Crypto, Futures, & Future Options](#)

[Interactive Brokers](#)

[US Equities, Equity Options, FOREX, Futures, Future Options, Index, Index Options, & CFD](#)

[TradeStation](#)

[US Equities, Equity Options, Futures, Index, & Index Options](#)

[Tastytrade](#)

[US Equities, Equity Options, Futures, Future Options, Index, & Index Options](#)

[Alpaca](#)

[US Equities, Equity Options, & Crypto](#)

[Charles Schwab](#)

[US Equities, Equity Options, Index, & Index Options](#)

[Binance](#)

[Crypto & Crypto Futures](#)

[Bybit](#)

[Crypto & Crypto Futures](#)

[Tradier](#)

[US Equities & Equity Options](#)

[Kraken](#)

[Crypto](#)

[Coinbase](#)

[Crypto](#)

[Bitfinex](#)

[Crypto](#)

[Bloomberg EMSX](#)

[US Equities, Equity Options, Futures, & Index Options](#)

[SSC Eze](#)

[US Equities, Equity Options, Futures, Future Options, Index, & Index Options](#)

[Trading Technologies](#)

[Futures](#)

[Samco](#)

[India Equities](#)

[Zerodha](#)

[India Equities](#)

[CFD and FOREX Brokerages](#)

[CFD & FOREX](#)

See Also

[IQ Feed](#)

[Polygon](#)

[Theta Data](#)

Brokerages

QuantConnect Paper Trading

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the QuantConnect Paper Trading brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 1 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Set your initial cash balance.

```
$ lean live deploy "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. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

If your algorithm only uses custom data, you can select the "Custom data only" data provider 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 provider 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.set_benchmark(lambda x: 0)
```

PY

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 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 deploy "<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 deploy "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'
```

4. Enter 1 to select the QuantConnect Paper Trading brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

5. 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 deploy "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
```

6. 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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

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 deploy "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. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: QuantConnect Paper Trading
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: QuantConnectBrokerage
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
```

10. 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 Trading
Launched: 2021-06-09 15:10:12 UTC
```

Brokerages

Interactive Brokers

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Interactive Brokers (IB) brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the IB brokerage integration, see the [Lean.Brokerages.InteractiveBrokers repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the [Interactive Brokers](#) brokerage and the [Interactive Brokers](#) data provider:

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 deploy "<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 deploy "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'
```

4. Enter 1 to select the [Interactive Brokers](#) brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

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 a weekly restart time that's convenient for you.

```
$ lean cloud live "My Project" --push --open
Weekly restart UTC time (hh:mm:ss) [21:00:00]:
```

You'll receive a notification on your IB Key device every Sunday to re-authenticate the connection between IB and your live algorithm. Enter a time on Sunday to receive the notification. If you don't re-authenticate before the timeout period, your algorithm quits executing. Ensure your IB Key device has sufficient battery for the time you expect to receive the notification. If you don't receive a notification, see [I am not receiving IBKR Mobile notifications](#) on the IB website.

8. Enter whether you want to use the [price data from Interactive Brokers](#) instead of the data 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
```

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

11. Set your initial cash balance.

```
$ lean cloud live deploy "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 deploy "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: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. 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 deploy "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
```

14. Enter 1 to select the [Interactive Brokers](#) data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 1
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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


```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Interactive Brokers
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Interactive Brokers
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': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', '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
```

16. 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 Brokers
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

If you have an ARM M1, M2, or M3 chip, you can't deploy a local live algorithm with the IB brokerage, see the [Troubleshooting](#).

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 1.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Set up IB Key Security via IBKR Mobile. For instructions, see [IB Key Security via IBKR Mobile](#) on the IB website.
5. 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: *****
```

6. Enter a weekly restart time that's convenient for you.

```
$ lean live "My Project"
Weekly restart UTC time (hh:mm:ss) [21:00:00]:
```

You'll receive a notification on your IB Key device every Sunday to re-authenticate the connection between IB and your live algorithm. Enter a time on Sunday to receive the notification. If you don't re-authenticate before the timeout period, your algorithm quits executing. Ensure your IB Key device has sufficient battery for the time you expect to receive the notification. If you don't receive a notification, see [I am not receiving IBKR Mobile notifications](#) on the IB website.

7. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

8. 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. Delayed market data is on a time lag that is usually 10-20 minutes behind real-time quotes, see [Delayed Market Data Timing](#) for more information. When disabled, live trading will stop and LEAN will shut down.

9. 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` .

Data Provider

For information about the IB data provider, see [Interactive Brokers](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)
- [US Futures Universe](#)
- [US Futures Options Universe](#)

The IB data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data in local deployments, [download the US Futures Security Master](#) .

Troubleshooting

The following table describes errors you may see when deploying to IB:

Error Message(s)	Possible Cause and Fix
Login failed.	The credentials you provided are incorrect. Typically, the password contains leading and/or trailing white spaces. Copy the password to a text editor to ensure the password is correct. If you can't log in to Trader Workstation (TWS) with your credentials, contact IB. If you can log in to TWS but can't log in to the deployment wizard, contact us and provide the algorithm Id and deployment Id.
Login to the IB Gateway failed because a user account-tasks is required.	Download IB Gateway , run it, and follow the instructions provided.
An existing session was detected and will not be automatically disconnected. Historical Market Data Service error message: Trading TWS session is connected from a different IP address.	IB still recognizes your previous live deployment as being partially connected. It can take a minute to fully disconnect. For more information, see Security and Stability > Connections .

<p>The two factor authentication request timed out.</p> <p>A security dialog was detected for Code Card Authentication.</p> <p>Unknown message window detected: Challenge: 123 456</p>	<p>You haven't replied to the two factor authentication requests. The code card authentication ("Challenge") is triggered when you don't reply to the IB mobile 2FA requests. Ensure your IB Key device has sufficient battery for the time you expect to receive the notification. If you don't receive a notification, see I am not receiving IBKR Mobile notifications on the IB website.</p>
<p>API support is not available for accounts that support free trading.</p>	<p>Upgrade your plan from IBKR Lite to IBKR Pro.</p>
<p>No security definition has been found for the request.</p>	<p>Your algorithm added an invalid or unsupported security. For example, a delisted stock, an expired contract, inexistent contract (invalid expiration date or strike price), or a warrant (unsupported). If the security should be valid and supported , open a support ticket and attach the live deployment Id. The algorithm will continue running, but it won't trade the security. If you don't want to deploy to an account with an invalid or unsupported security, set <code>self.settings.ignore_unknown_assets</code> is <code>False</code> .</p>
<p>Requested market data is not subscribed.</p> <p>Historical Market Data Service error message: No market data permissions for ...</p>	<p>Your algorithm uses the Interactive Brokers Data Provider , but you don't have a subscription to it. Subscribe to the data bundle you need , contact IB, or re-deploy the algorithm with a different data provider. Try the QuantConnect or the hybrid QuantConnect + Interactive Brokers data providers on QuantConnect Cloud or try a third-party provider.</p>
<p>Timeout waiting for brokerage response for brokerage order id 37 lean id 31</p>	<p>IB didn't respond to an order request. Stop and re-deploy the algorithm. On the next deployment, LEAN retrieves this order or the positions it opened or closed.</p>
<p>Could not find file '/root/ibgateway/ibgateway'.</p>	<p>Your Docker installation has pulled the ARM platform version of the LEAN Docker image. This version doesn't include IB Gateway, because QuantConnect doesn't support Interactive Brokers integration with ARM chips (e.g.: Apple M1, M2, and M3 chips).</p>

To view the description of less common errors, see [Error Codes](#) in the TWS API Documentation. If you need further support, [open a new support ticker](#) and add the live deployment with the error.

Brokerages

TradeStation

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the [TradeStation](#) brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the TradeStation brokerage integration, see the [Lean.Brokerages.TradeStation repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the [TradeStation](#) brokerage and the [TradeStation](#) data provider:

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 deploy "<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 deploy "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'
```

4. Enter 1 to select the [TradeStation](#) brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

5. In the browser window that automatically opens, log in to your TradeStation account.

```
$ lean cloud live "My Project" --push --open
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tradestation
Will sleep 5 seconds and retry fetching authorization...
```

6. Enter the TradeStation account ID.

```
$ lean cloud live "My Project" --push --open
The TradeStation account Id (11810357, 210NKH33, SIM2829935F, SIM2829934M): SIM2829935F
```

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

9. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

11. 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 deploy "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
```

12. Enter 1 to select the [TradeStation](#) data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 1
```

13. In the browser window that automatically opens, log in to your TradeStation account.

```
$ lean cloud live "My Project" --push --open

Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tradestation
Will sleep 5 seconds and retry fetching authorization...
```

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

```

$ lean cloud live deploy "My Project" --push --open
Brokerage: TradeStation
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: TradeStation
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': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', '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: TradeStation
Launched: 2021-06-09 15:10:12 UTC

```

Deploy Local Algorithms

Follow these steps to start local live trading with the [TradeStation](#) 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 1.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. LEAN CLI opens your browser for authorization. Login to TradeStation.

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tradestation
Will sleep 5 seconds and retry fetching authorization...
```

5. Enter the TradeStation account ID.

```
$ lean live "My Project"
The TradeStation account Id (11810357, 210NKH33, SIM2829935F, SIM2829934M): SIM2829935F
```

6. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

7. LEAN CLI opens your browser for authorization. Login to TradeStation if necessary.

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tradestation
Will sleep 5 seconds and retry fetching authorization...
```

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` .

Data Provider

For information about the [TradeStation](#) data provider, see [TradeStation](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)

The TradeStation data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data, [download the US Futures Security Master](#) .

Brokerages

Tastytrade

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the [tastytrade](#) brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the tastytrade brokerage integration, see the [Lean.Brokerages.tastytrade repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the [tastytrade](#) brokerage and the [tastytrade](#) data provider:

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 deploy "<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 deploy "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'
```

4. Enter 1 to select the [tastytrade](#) brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

5. In the browser window that automatically opens, log in to your tastytrade account.

```
$ lean cloud live "My Project" --push --open
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tastytrade
Will sleep 5 seconds and retry fetching authorization...
```

6. Enter the tastytrade account ID.

```
$ lean cloud live "My Project" --push --open
The tastytrade account Id (11810357, 210NKH33, SIM2829935F, SIM2829934M): SIM2829935F
```

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

9. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

11. 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 deploy "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
```

12. Enter 1 to select the [tastytrade](#) data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 1
```

13. In the browser window that automatically opens, log in to your tastytrade account.

```
$ lean cloud live "My Project" --push --open

Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tastytrade
Will sleep 5 seconds and retry fetching authorization...
```

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

```

$ lean cloud live deploy "My Project" --push --open
Brokerage: tastytrade
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: tastytrade
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': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', '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: tastytrade
Launched: 2021-06-09 15:10:12 UTC

```

Deploy Local Algorithms

Follow these steps to start local live trading with the [tastytrade](#) 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 1.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. LEAN CLI opens your browser for authorization. Login to tastytrade.

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tastytrade
Will sleep 5 seconds and retry fetching authorization...
```

5. Enter the tastytrade account ID.

```
$ lean live "My Project"
The tastytrade account Id (11810357, 210NKH33, SIM2829935F, SIM2829934M): SIM2829935F
```

6. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

7. LEAN CLI opens your browser for authorization. Login to tastytrade if necessary.

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=tastytrade
Will sleep 5 seconds and retry fetching authorization...
```

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` .

Data Provider

For information about the [tastytrade](#) data provider, see [tastytrade](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)
- [US Futures Universe](#)
- [US Futures Options Universe](#)

The tastytrade data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data, [download the US Futures Security Master](#) .

Brokerages

Alpaca

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Alpaca brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the [Alpaca](#) brokerage integration, see the [Lean.Brokerages.Alpaca repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Alpaca brokerage and the Alpaca data provider:

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 deploy "<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 deploy "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'
```

4. Enter 16 to select the Alpaca brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 16
```

5. In the browser window that automatically opens, click Allow .

```
$ lean cloud live "My Project" --push --open
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=alpaca
Will sleep 5 seconds and retry fetching authorization...
```

6. Enter the environment to use.

```
$ lean cloud live "My Project" --push --open
Live or Paper environment? (live, paper): live
```

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

9. Set your initial cash balance.

```
$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

11. 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 deploy "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
```

12. Enter 19 to select the Alpaca data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 19
```

13. In the browser window that automatically opens, click Allow .

```
$ lean cloud live "My Project" --push --open

Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=alpaca
Will sleep 5 seconds and retry fetching authorization...
```

14. Enter **your API key and API secret** .

```
$ lean live "My Project"
Alpaca Api Key: PKEFXE36AR50EG5K5KNQ
Alpaca Api Secret: *****
```

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

```

$ lean cloud live deploy "My Project" --push --open
Brokerage: Alpaca
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Alpaca
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': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', '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

```

16. 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: Alpaca
Launched: 2021-06-09 15:10:12 UTC

```

Deploy Local Algorithms

Follow these steps to start local live trading with the Alpaca 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 16 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 16
```

4. In the browser window that automatically opens, click Allow .

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=alpaca
Will sleep 5 seconds and retry fetching authorization...
```

5. Enter the environment to use.

```
$ lean live "My Project"
Live or Paper environment? (live, paper): live
```

6. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

7. In the browser window that automatically opens, click Allow .

```
$ lean live "My Project"
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=alpaca
Will sleep 5 seconds and retry fetching authorization...
```

8. Enter [your API key and API secret](#) .

```
$ lean live "My Project"
Alpaca Api Key: PKEFXE36AR50EG5K5KNQ
Alpaca Api Secret: *****
```

9. 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` .

Data Provider

For information about the Alpaca data provider, see [Alpaca](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)

The Alpaca data provider serves raw Equity data. To get adjusted data in local deployments, [download the US Equity Security Master](#) .

Brokerages

Charles Schwab

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Charles Schwab brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

Charles Schwab only supports authenticating one account at a time per user. If you have an algorithm running with Charles Schwab and then deploy a second one, the first algorithm stops running.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Charles Schwab brokerage and the Charles Schwab data provider:

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 deploy "<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 deploy "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'
```

4. Enter 13 to select the Charles Schwab brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 13
```

5. In the browser window that automatically opens, click Allow .

```
$ lean cloud live "My Project" --push --open

Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=charlesschwab&projectId=
<projectId>

Will sleep 5 seconds and retry fetching authorization...
```

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

8. Set your initial cash balance.

```
$ lean cloud live deploy "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
```

9. Set your initial portfolio holdings.

```
$ lean cloud live deploy "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: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

10. 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 deploy "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
```

11. Enter 13 to select the Charles Schwab data provider.


```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 13
```

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Charles Schwab
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Charles Schwab
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': 'G00CV VP83T1ZUHR0L', '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: Charles Schwab
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading with the Charles Schwab 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 13 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 13
```

4. If your project is local-only project, LEAN CLI will prompt you to enter the ID of any cloud project of your organization to proceed with authentication, see [Get Project ID](#) for more information.

```
$ lean live "My Project"
Please enter any cloud project ID to proceed with Auth0 authentication:
```

5. In the browser window that automatically opens. On the Charles Schwab website, log in, and select your Schwab accounts to link. Click Allow to grant QuantConnect access to your account information and authorization.

```
$ lean live "My Project"
Please enter any cloud project ID to proceed with Auth0 authentication: <projectId>
Please open the following URL in your browser to authorize the LEAN CLI.
https://www.quantconnect.com/api/v2/live/auth0/authorize?brokerage=charlesschwab&projectId=
<projectId>
Will sleep 5 seconds and retry fetching authorization...
```

6. Enter the Charles Schwab account number.

```
$ lean live "My Project"
The CharlesSchwab account number (12345678, 23456789): 12345678
```

7. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Data Provider

For information about the Charles Schwab data provider, see [Charles Schwab](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)

The Charles Schwab data provider serves raw Equity data. To get adjusted data in local deployments, [download the US Equity Security Master](#) .

Brokerages

Binance

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Binance or Binance US brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Binance brokerage integration, see the [Lean.Brokerages.Binance repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

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

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 deploy "<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 deploy "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'
```

4. Enter 7 to select the Binance brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 7
```

5. Enter the exchange to use.

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

6. Enter your Binance API key and secret.

```
$ lean cloud live "My Project" --push --open
API key: wL1wae0C7VD447skCFeiat9pP3r1uKXfYomGg43uyC0gzl8xsI9SZsX97AXP4zWv
API secret: *****
```

To create new credentials, see [How to Create API Keys on Binance](#) .

7. Enter the environment to use.

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

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

10. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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: G00G
Symbol ID: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

12. 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 deploy "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
```

13. Enter 7 to select the Binance data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 7
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Binance
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Binance
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 VP83T1ZUHR0L', '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: Binance
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading with the Binance or Binance US 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 1 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Enter the exchange to use.

```
$ lean live "My Project"
Binance Exchange (Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures): BinanceUS
```

5. Enter your API key and API secret.

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

To create new credentials, see [How to Create API Keys on Binance](#) .

6. Enter the environment to use.

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

7. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Brokerages

Bybit

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Bybit brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Bybit brokerage integration, see the [Lean.Brokerages.Bybit repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Bybit brokerage and the Bybit data provider:

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 deploy "<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 deploy "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'
```

4. Enter 14 to select the Bybit brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 14
```

5. Enter your API key and secret.

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

To generate your API credentials, see [Account Types](#) . Your account details are not saved on QuantConnect.

6. Enter your VIP level.

```
$ lean live "My Project"
VIP Level (VIP0, VIP1, VIP2, VIP3, VIP4, VIP5, SupremeVIP, Pro1, Pro2, Pro3, Pro4, Pro5): VIP0
```

For more information about VIP levels, see [FAQ — Bybit VIP Program](#) on the Bybit website.

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

9. Set your initial cash balance.

```

$ lean cloud live deploy "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. 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 deploy "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

```

11. Enter 17 to select the Bybit data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 17
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Bybit
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Bybit
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}]
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: Bybit
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading with the Bybit 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 14 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 14
```

4. Enter your API key and secret.

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


To generate your API credentials, see [Account Types](#) . Your account details are not saved on QuantConnect.

5. Enter your VIP level.

```
$ lean live "My Project"
VIP Level (VIP0, VIP1, VIP2, VIP3, VIP4, VIP5, SupremeVIP, Pro1, Pro2, Pro3, Pro4, Pro5): VIP0
```

For more information about VIP levels, see [FAQ — Bybit VIP Program](#) on the Bybit website.

6. Enter the environment to use.

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

The paper environment is [Bybit Demo Trading](#) .

7. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Brokerages

Tradier

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Tradier brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Tradier brokerage integration, see the [Lean.Brokerages.Tradier repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Tradier brokerage and the Tradier data provider:

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 deploy "<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 deploy "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'
```

4. Enter 3 to select the Tradier brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 3
```

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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

9. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

11. 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 deploy "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
```

12. Enter 3 to select the Tradier data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 3
```

In the demo environment, Tradier doesn't offer streaming market data due to exchange restrictions related to delayed data.

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Tradier
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Tradier
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 VP83T1ZUHR0L', '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: Tradier
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 3 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 3
```

4. 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.

5. Enter whether the developer sandbox should be used.

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

6. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)

- [Polygon](#)
- [Theta Data](#)

In the demo environment, Tradier doesn't offer streaming market data due to exchange restrictions related to delayed data.

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` .

Data Provider

For information about the Tradier data provider, see [Tradier](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)

The Tradier data provider serves raw Equity data. To get adjusted data in local deployments, [download the US Equity Security Master](#) .

Brokerages

Kraken

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Kraken brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Kraken brokerage integration, see the [Lean.Brokerages.Kraken repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Kraken brokerage and the Kraken data provider:

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 deploy "<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 deploy "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'
```

4. Enter 12 to select the Kraken brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 12
```

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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

9. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

11. 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 deploy "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
```

12. Enter 12 to select the Kraken data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 12
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Kraken
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Kraken
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 VP83T1ZUHR0L', '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: Kraken
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 12 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 12
```

4. 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.

5. 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.

6. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze

To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Brokerages

Coinbase

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Coinbase brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Coinbase brokerage integration, see the [Lean.Brokerages.Coinbase repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Coinbase Advanced Trade brokerage and the Coinbase Advanced Trade data provider:

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 deploy "<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 deploy "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'
```

4. Enter 6 to select the Coinbase Advanced Trade brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 6
```

5. Enter your API name and private key.

```
$ lean live "My Project"
API name: organizations/2c7dhs-a3a3-4acf-aa0c-f68584f34c37/apiKeys/41090ffa-asd2-8080-815f-
afaf63747e35
API private key:
*****
```

To create new API credentials, see [Account Types](#) .

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

8. Set your initial cash balance.

```

$ lean cloud live deploy "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

```

9. Set your initial portfolio holdings.

```

$ lean cloud live deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

10. 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 deploy "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
```

11. Enter 6 to select the Coinbase Advanced Trade data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 6
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Coinbase Advanced Trade
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Coinbase Advanced Trade
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 VP83T1ZUHR0L', '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: Coinbase Advanced Trade
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading with the Coinbase Advanced Trade 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 6 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 6
```

4. Enter your API name and private key.

```
$ lean live "My Project"
API name: organizations/2c7dhs-a3a3-4acf-aa0c-f68584f34c37/apiKeys/41090ffa-asd2-8080-815f-
afaf63747e35
API private key:
*****
```

To create new API credentials, see [Account Types](#) .

5. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Brokerages

Bitfinex

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Bitfinex brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Bitfinex brokerage integration, see the [Lean.Brokerages.Bitfinex repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

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

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 deploy "<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 deploy "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'
```

4. Enter 5 to select the Bitfinex brokerage.


```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 5
```

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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

8. Set your initial cash balance.

```
$ lean cloud live deploy "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
```

9. Set your initial portfolio holdings.

```
$ lean cloud live deploy "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: G00G
Symbol ID: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

10. 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 deploy "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
```

11. Enter 5 to select the Bitfinex data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 5
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Bitfinex
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Bitfinex
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': 'G00CV VP83T1ZUHR0L', '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: Bitfinex
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 5 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 5
```

4. Enter your API key Id and secret.

```
$ lean live "My Project"
API key: bbbMsbxjytVM9cGvnLpKguz9rZf2T5qACxaVx7E8Mm
API secret: *****
```

To create new API credentials, see the [API Management page](#) on the Bitfinex website.

5. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Brokerages

Bloomberg EMSX

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Terminal Link brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

The LEAN CLI integrates with the Bloomberg Desktop API (DAPI) to get data for research, backtests, optimization, and live trading or through your existing Bloomberg data subscriptions. In addition, you can route live trading strategies and UAT paper trading through Bloomberg order routing via the EMSX network.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Asset Classes

Terminal Link supports trading the following asset classes:

- [Equities](#)
- [Equity Options](#)
- [Futures](#)
- [Index Options](#)

Data Feeds

Terminal Link with the LEAN CLI lets you source historical data from Bloomberg and save it onto your desktop computer for backtesting as within your license permissions from Bloomberg. The Bloomberg API includes up to six months of history in low resolutions like second and minute resolutions.

Orders

Terminal Link enables you to create and manage Bloomberg™ orders. You can also use the LEAN CLI with the Terminal Link integration to test paper trading with LEAN. In this case, LEAN models order fills using the live tick data feed from Bloomberg™.

Order Types

The following table describes the available order types for each asset class that Terminal Link supports:

Order Type	Equity	Equity Options	Futures	Index Options
MarketOrder	✓	✓	✓	✓
LimitOrder	✓	✓	✓	✓
StopMarketOrder	✓	✓	✓	✓
StopLimitOrder	✓	✓	✓	✓

Time In Force

Terminal Link supports the following [TimeInForce](#) instructions:

- [DAY](#)
- [GOOD_TIL_CANCELED](#)
- [good_til_date](#)

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](#) .

Handling Splits

If you're using raw [data normalization](#) and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a [stock split](#) occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price
- Trigger price

Fees

Orders filled with Terminal Link are subject to the fees of the Bloomberg™ Execution Management System and your prime brokerage destination. To view how we model their fees, see [Fees](#) .

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. QuantConnect requires a Trading Firm or Institutional license to use the Terminal Link integration.

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 following table shows the activities each of the Bloomberg technologies support:

Technology	Research	Backtesting	Paper Trading	Live Trading
Desktop API	✓	✓	✓	✓
B.PIPE	✓	✓	✓	✓

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-historical "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-historical "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-provider-live "Terminal Link"
```

Deploy Cloud Algorithms

You need to [set up the Bloomberg SAPI](#) before you can deploy cloud algorithms with Terminal Link.

Follow these steps to start live trading a project in the cloud with the Terminal Link brokerage and the Terminal Link data provider:

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 deploy "<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 deploy "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'
```

4. Enter 10 to select the Terminal Link brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 10
```

5. Enter your unique user identifier (UUID).

```
$ lean cloud live "My Project" --push --open
Configure credentials for Terminal Link
Using 'SAPI' Connection Type
Server Auth ID:
```

The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.

6. Enter the environment to use.

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

7. Enter the SAPI host and port.

```
$ lean cloud live "My Project" --push --open
Server host:
Server port:
```

The default port is 8194.

8. Enter the EMSX broker to use.

```
$ lean cloud live "My Project" --push --open
EMSX broker:
```

9. Enter the account to which LEAN should route orders.

```
$ lean cloud live "My Project" --push --open
EMSX account []:
```

10. Enter your OpenFIGI API key.

```
$ lean cloud live "My Project" --push --open
OpenFIGI API key:
```

11. 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 deploy "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

```

12. 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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

13. Set your initial cash balance.

```

$ lean cloud live deploy "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

```

14. Set your initial portfolio holdings.

```

$ lean cloud live deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

15. 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 deploy "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
```

16. Enter 10 to select the Terminal Link data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 10
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Terminal Link
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Terminal Link
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 VP83T1ZUHR0L', '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
```

18. 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: Terminal Link
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

You need to [install the Bloomberg Terminal and the BBComm component](#) from the Bloomberg website before you can deploy local algorithms with Terminal Link.

Follow these steps to start local live trading with the Terminal Link 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 10 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 10
```

4. Enter the environment to use.

```
$ lean live "My Project"
Environment (Production, Beta): Production
```

5. Enter the host and port of the Bloomberg server.

```
$ lean live "My Project"
Server host: 127.0.0.1
Server port: 8194
```

6. Enter your EMSX configuration

```
$ lean live "My Project"
EMSX broker: someValue
EMSX account:
```

7. Enter your Open FIGI API key.

```
$ lean live "My Project"
```

Open FIGI API key:

8. Enter 9 to select the Terminal Link live data provider.

```
$ lean live deploy "My Project"
```

Select a live data provider:

- 1) Interactive Brokers
- 2) Tradier
- 3) Oanda
- 4) Bitfinex
- 5) Coinbase Advanced Trade
- 6) Binance
- 7) Zerodha
- 8) Samco
- 9) Terminal Link
- 10) Trading Technologies
- 11) Kraken
- 12) Charles Schwab
- 13) IQFeed
- 14) Polygon
- 15) CoinApi
- 16) ThetaData
- 17) Custom data only
- 18) Bybit
- 19) TradeStation
- 20) Alpaca
- 21) Tastytrade
- 22) Eze

To enter multiple options, separate them with comma: 9

9. 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` .

Brokerages

SSC Eze

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the [SS&C Eze](#) brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the [SS&C Eze](#) brokerage and the [SS&C Eze](#) data provider:

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 deploy "<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 deploy "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'
```

4. Enter 1 to select the [SS&C Eze](#) brokerage.


```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

5. Enter the trading domain, region, username, password, and trading account in BBCE (BANK;BRANCH;CUSTOMER;DEPOSIT) format.

```
$ lean cloud live "My Project" --push --open
Provided by brokerage: trading domain: LIGHTSPEEDDELAY
Provided by brokerage: locale/market region: Live and Delayed
Username:
Account password:
Enter your trading account (BBCE format): LATEST;TEST;01;ACQUIREDMEDIA
```

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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: 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 deploy "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. Enter 1 to select the [SS&C Eze](#) data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 1
```

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: SS&C Eze
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: SS&C Eze
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
```

11. 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: SS&C Eze
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading with the [SS&C Eze](#) 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 1 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Enter the trading domain, region, username, password, and trading account in BBCD (BANK;BRANCH;CUSTOMER;DEPOSIT) format.

```
$ lean cloud live "My Project" --push --open
Provided by brokerage: trading domain: LIGHTSPEEDDELAY
Provided by brokerage: locale/market region: Live and Delayed
Username:
Account password:
Enter your trading account (BBCD format): LATEST;TEST;01;ACQUIREDMEDIA
```

5. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Data Provider

For information about the [SS&C Eze](#) data provider, see [SS&C Eze](#) . In local deployments, [universe selection](#) is available if you download the data from the [Dataset Market](#) . The dataset listings show how to download the universe selection data with the CLI. To stay up-to-date, periodically download the new data from QuantConnect Cloud, which you can automate with Python scripts. For example, the following tutorials explain how to download historical data and download daily updates:

- [US Equity Coarse Universe](#)
- [US Equity Option Universe](#)
- [US ETF Constituents](#)
- [US Index Option Universe](#)
- [US Futures Universe](#)
- [US Futures Options Universe](#)

The SS&C Eze data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data, [download the US Futures Security Master](#) .

Brokerages

Trading Technologies

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Trading Technologies brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Trading Technologies integration, see the [Lean.Brokerages.TradingTechnologies repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Trading Technologies brokerage and the Trading Technologies data provider:

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 deploy "<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 deploy "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'
```

4. Enter 11 to select the Trading Technologies brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 11
```

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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

10. Set your initial portfolio holdings.

```

$ lean cloud live deploy "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: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

11. 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 deploy "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

```

12. Enter 11 to select the Trading Technologies data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 11
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Trading Technologies
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Trading Technologies
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 VP83T1ZUHR0L', '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: Trading Technologies
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 11 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 11
```

4. Enter [your Trading Technologies credentials](#) .

```
$ lean live "My Project"
User name: john
Session password: *****
Account name: jane
```

5. Enter the [REST configuration](#) .

```
$ lean live "My Project"
REST app key: my-rest-app-key
REST app secret: *****
REST environment: my-environment
```

6. 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:
```

7. 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:
```

8. Enter whether FIX messages must be logged.

```
$ lean live "My Project"
Log FIX messages (yes/no): yes
```

9. Set your initial cash balance.

```
$ lean live deploy "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. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

11. 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` .

Brokerages

Samco

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Samco brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Samco brokerage integration, see the [Lean.Brokerages.Samco repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Samco brokerage and the Samco data provider:

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 deploy "<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 deploy "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'
```

4. Enter 9 to select the Samco brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 9
```

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
<code>mis</code>	Intraday products
<code>cnc</code>	Delivery products
<code>nrml</code>	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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial cash balance.

```
$ lean cloud live deploy "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 deploy "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: G00G
Symbol ID: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'G00G', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. 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 deploy "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
```

14. Enter 9 to select the Samco data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 9
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Samco
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Samco
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 VP83T1ZUHR0L', '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
```

16. 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: Samco
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 9 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 9
```

4. Enter your Samco credentials.

```
$ lean live "My Project"
Client ID:
Client Password:
```

5. Enter your year of birth.

```
$ lean live "My Project"
Year of Birth:
```

6. Enter the product type.

```
$ lean live "My Project"
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
<code>mis</code>	Intraday products
<code>cnc</code>	Delivery products
<code>nrml</code>	Carry forward products

7. 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
<code>equity</code>	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
<code>commodity</code>	For trading commodities on the Multi Commodity Exchange of India (MCX)

8. Enter 8 to select the Samco data provider.

```
$ lean live deploy "My Project"
Select a live data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma: 8
```

9. 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` .

Data Provider

For more information about the Samco data provider, see [Samco](#) .

Brokerages

Zerodha

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the Zerodha brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the Zerodha brokerage integration, see the [Lean.Brokerages.Zerodha repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Zerodha brokerage and the Zerodha data provider:

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 deploy "<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 deploy "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'
```

4. Enter 8 to select the Zerodha brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 8
```

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
<code>mis</code>	Intraday products
<code>cnc</code>	Delivery products
<code>nrml</code>	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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

11. Set your initial cash balance.

```
$ lean cloud live deploy "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 deploy "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: G00CV VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'G00CV VP83T1ZUHR0L', 'quantity': 10,
'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n
```

13. 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 deploy "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
```

14. Enter 8 to select the Zerodha data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 8
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Zerodha
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Zerodha
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': 'G00CV VP83T1ZUHR0L', '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
```

16. 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: Zerodha
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter the brokerage number, 8 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 8
```

4. Enter your [Kite Connect](#) API key and access token.

```
$ lean live "My Project"
API key: hp9erb9ct0lqaxpm
Access token: *****
```

5. Enter the product type.

```
$ lean live "My Project"  
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
<code>mis</code>	Intraday products
<code>cnc</code>	Delivery products
<code>nrml</code>	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
<code>equity</code>	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
<code>commodity</code>	For trading commodities on the Multi Commodity Exchange of India (MCX)

7. Enter 7 to select the Zerodha data provider.

```
$ lean live deploy "My Project"
Select a live data feed:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma: 7
```

8. Enter whether you have a history API subscription.

```
$ lean live "My Project"
Do you have a history API subscription? (true, false): true
```

9. 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` .

Data Provider

For more information about the Zerodha data provider, see [Zerodha](#) .

Brokerages

CFD and FOREX Brokerages

Introduction

The Lean CLI supports live trading on your local machine or in QuantConnect Cloud, which makes the transfer from backtesting to live trading as seamless as possible. You must [log in](#) using your QuantConnect account to deploy your algorithm on your local machine or in QuantConnect Cloud. This page contains instructions on how to start live trading with the OANDA brokerage. If the [Lean Configuration file](#) in your [organization workspace](#) contains values for some of the command options, the CLI skips some of the prompts.

To view the implementation of the OANDA brokerage integration, see the [Lean.Brokerages.OANDA repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Oanda brokerage and the Oanda data provider:

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 deploy "<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 deploy "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'
```

4. Enter 4 to select the Oanda brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 4
```

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. 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 deploy "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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y

```

10. Set your initial cash balance.

```

$ lean cloud live deploy "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 deploy "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 VP83T1ZUHR0L
Quantity: 10
Average Price: 50
Portfolio Holdings: [{'symbol': 'GOOG', 'symbolId': 'GOOCV VP83T1ZUHR0L', 'quantity': 10, 'averagePrice': 50.0}]
Do you want to add more holdings? [y/N]: n

```

12. 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 deploy "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
```

13. Enter 4 to select the Oanda data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 4
```

If you select one of the following data providers, see the respective page for more instructions:

- [Polygon](#)

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: Oanda
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Oanda
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 VP83T1ZUHR0L', '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: Oanda
Launched: 2021-06-09 15:10:12 UTC
```

Deploy Local Algorithms

Follow these steps to start local live trading 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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / <projectName> and then enter the brokerage number, 4 .

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 4
```

4. Enter the environment to use. Enter **Trade** for fxTrade or **Practice** for fxTrade Practice.

```
$ lean live "My Project"
Environment? (Practice, Trade): Trade
```

5. 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.

6. 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.

7. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

If you select one of the following data providers, see the respective page for more instructions:

- [IQFeed](#)
- [Polygon](#)
- [Theta Data](#)

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` .

Live Trading

Data Providers

Data providers supply data to run your live algorithm using LEAN. You can use multiple data providers in live trading algorithms.

[IQFeed](#)

[US Equities, US Options, Forex, & Futures](#)

[Polygon](#)

[US Equities, US Equity Options, US Indices, & US Index Options](#)

[Theta Data](#)

[US Equities, US Equity Options, US Indices, & US Index Options](#)

See Also

[Brokerages](#)

Data Providers

IQFeed

Introduction

Instead of using the data 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 [IQFeed data provider](#) with the QuantConnect Paper Trading brokerage.

To view the implementation of the IQFeed integration, see the [Lean.DataSource.IQFeed repository](#).

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Local Algorithms

Follow these steps to start local live trading with the IQ Feed data provider:

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 live deploy "<projectName>"` to start a live deployment wizard for the project in `.` / `<projectName>` and then enter a brokerage number.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

5. 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]:
```

6. Enter your IQFeed username and password.

```
$ lean live "My Project"
Username: 123456
Password: *****
```

7. 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
```

8. 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
```

9. 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 the IQFeed data provider.

Supported Assets

Our IQFeed integration supports securities from the following asset classes:

- [US Equity](#)
- [US Equity Options](#)
- [Forex](#) (listed on FXCM)
- [US Futures](#)

The IQFeed data provider serves raw data. To get adjusted Equity data in local deployments, [download the US Equity Security Master](#) . To get [continuous Futures](#) data in local deployments, [download the US Futures Security Master](#) .

Multiple Data Providers

When you [deploy a live algorithm](#) , you can add multiple data providers. If you use multiple data providers, the order you select them in defines their order of precedence in Lean.

Data Providers

Polygon

Introduction

Instead of using the data from your brokerage, you can also use Polygon. This tutorial demonstrates how to set up the [Polygon data provider](#) with the QuantConnect Paper Trading brokerage.

To view the implementation of the Polygon integration, see the [Lean.DataSource.Polygon repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Local Algorithms

Follow these steps to start local live trading with the Polygon data provider:

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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / `<projectName>` and then enter a brokerage number.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

4. Enter the number of the live data provider(s) to use and then follow the steps required for the data

connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

5. Enter your Polygon API key.

```
$ lean live "My Project"
Configure credentials for Polygon

Your Polygon API Key:
```

To get your API key, see the [API Keys page](#) on the Polygon website.

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 QuantConnect Paper Trading brokerage and the Polygon data provider:

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 deploy "<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 deploy "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'
```

4. Enter 1 to select the QuantConnect Paper Trading brokerage.

```
$ lean cloud live deploy "My Project" --push --open
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```

5. 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 deploy "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
```

6. 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 deploy "My Project" --push --open
Do you want to enable automatic algorithm restarting? [Y/n]: y
```

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 deploy "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. Enter 14 to select the Polygon data provider.

```
$ lean live "My Project"
Select a live data feed:
1) QuantConnect
2) Interactive Brokers
3) Tradier
4) Oanda
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Polygon
15) CoinApi
16) Bybit
17) TradeStation
18) Alpaca
19) Tastytrade
20) Eze
To enter multiple options, separate them with comma: 14
```

9. Enter your Polygon API key.

```
$ lean cloud live "My Project" --push --open
Configure credentials for Polygon

Your Polygon API Key:
```

To get your API key, see the [API Keys page](#) on the Polygon website.

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

```
$ lean cloud live deploy "My Project" --push --open
Brokerage: QuantConnect Paper Trading
Project id: 1234567
Environment: Live
Server name: L-MICRO node 89c90172
Server type: L-MICRO
Live Data providers: Polygon
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
```

11. 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 Trading
Launched: 2021-06-09 15:10:12 UTC
```

Supported Assets

Our Polygon integration supports securities from the following asset classes:

- [US Equity](#)
- [US Equity Options](#)
- [US Indices](#)
- [US Index Options](#)

Multiple Data Providers

When you [deploy a live algorithm](#), you can add multiple data providers. If you use multiple data providers, the order you select them in defines their order of precedence in Lean. For example, if you set Polygon as the first provider and IB as the second provider, Lean only uses the IB data provider for securities that aren't available from the Polygon data provider. This configuration makes it possible to use Polygon data provider for Equity and use IB for Futures.

Data Providers

Theta Data

Introduction

Instead of using the data from your brokerage, you can also use Theta Data if you're deploying a local project. To use Theta Data, you need to [install and launch the Theta Terminal](#) . This tutorial demonstrates how to set up the [Theta Data data provider](#) with the QuantConnect Paper Trading brokerage.

To view the implementation of the Theta Data integration, see the [Lean.DataSource.ThetaData repository](#) .

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Deploy Local Algorithms

Follow these steps to start local live trading with the ThetaData data provider:

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 live deploy "<projectName>"` to start a live deployment wizard for the project in . / `<projectName>` and then enter a brokerage number.

```
$ lean live deploy "My Project"
Select a brokerage:
1) Paper Trading
2) Interactive Brokers
3) Tradier
4) OANDA
5) Bitfinex
6) Coinbase Advanced Trade
7) Binance
8) Zerodha
9) Samco
10) Terminal Link
11) Trading Technologies
12) Kraken
13) Charles Schwab
14) Bybit
15) TradeStation
16) Alpaca
17) Tastytrade
18) Eze
Enter an option: 1
```


4. Enter the number of the live data provider(s) to use and then follow the steps required for the data connection.

```
$ lean live deploy "My Project"
Select a live data provider:
1) Interactive Brokers
2) Tradier
3) Oanda
4) Bitfinex
5) Coinbase Advanced Trade
6) Binance
7) Zerodha
8) Samco
9) Terminal Link
10) Trading Technologies
11) Kraken
12) Charles Schwab
13) IQFeed
14) Polygon
15) CoinApi
16) ThetaData
17) Custom data only
18) Bybit
19) TradeStation
20) Alpaca
21) Tastytrade
22) Eze
To enter multiple options, separate them with comma:
```

5. (Optional) Enter the host of the ThetaData Client.

The default host is `ws://host.docker.internal:25520/v1/events` or `http://host.docker.internal:25510` .

```
$ lean live "My Project"
The host of ThetaData Client [ws://host.docker.internal:25520/v1/events]:
The host of ThetaData Client [http://host.docker.internal:25510]:
```

6. Enter your Theta Data subscription plan.

```
$ lean live "My Project"
ThetaData subscription price plan (Free, Value, Standard, Pro):
```

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

The CLI doesn't currently support deploying cloud algorithms with the Theta Data data provider.

Supported Assets

Our Theta Data integration supports securities from the following asset classes:

- [Equity](#)
- [Equity Options](#)
- [Index](#)
- [Index Options](#)

Multiple Data Providers

When you deploy a live algorithm, you can add multiple data providers. If you use multiple data providers, the order you select them in defines their order of precedence in Lean.

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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](#).

Send Commands

To send [commands](#) to your algorithm, open a terminal in the [organization workspace](#) that contains the project and then run `lean live command "My Project" --data "<payload>"`.

```
$ lean live command "My Project" --data '{"ticker': 'AAPL', 'quantity': 1}"
```

The preceding line will run the `on_command` method of your algorithm. If you wrap the logic in a `Command` class in your algorithm, include a `$type` key in the payload and set the value to be the name of the class.

```
$ lean live command "My Project" --data '{"$type': 'MyCommand', 'ticker': 'AAPL', 'quantity': 1}"
```

If you run the command in PowerShell, use ``$type` instead of just `$type`.

```
$ lean live command "My Project" --data '{"`$type': 'MyCommand', 'ticker': 'AAPL', 'quantity': 1}"
```

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 execution.

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 lets 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](#).

Send Commands

To send [commands](#) to your algorithm, open a terminal in the [organization workspace](#) that contains the project and then run `lean cloud live cloud command "My Project" --data "<payload>"`.

```
$ lean cloud live command "My Project" --data '{"ticker': 'AAPL', 'quantity': 1}"
```

The preceding line will run the `on_command` method of your algorithm. If you wrap the logic in a `Command` class in your algorithm, include a `$type` key in the payload and set the value to be the name of the class.

```
$ lean cloud live command "My Project" --data '{"$type': 'MyCommand', 'ticker': 'AAPL', 'quantity': 1}"
```

If you run the command in PowerShell, use ``$type` instead of just `$type`.

```
$ lean cloud live command "My Project" --data '{"`$type': 'MyCommand', 'ticker': 'AAPL', 'quantity': 1}"
```

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 <code>--strategy-name <value></code> . 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 <code>--strategy-version <value></code> .
Strategy description	This description is displayed underneath the "Strategy Description" header on the first page and can be configured using <code>--strategy-description</code> . 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 <code>--live-results <path></code> . The provided path must point to a JSON file containing live results. For example, <code>--live-results "My Project/live/2022-03-17_10-53-12/L-3578882079.json"</code> .

Key Statistics

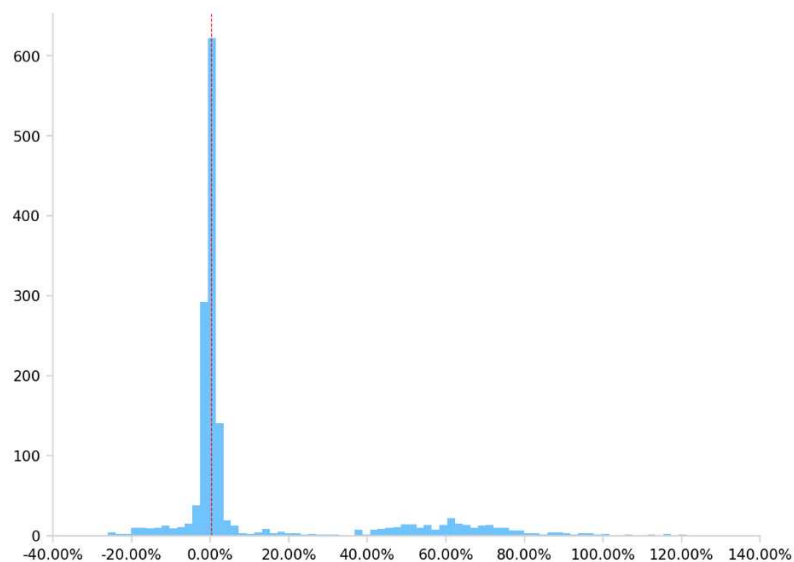
The top of the backtest report displays statistics to summarize your algorithm's performance. The following table describes the key statistics in the report:

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

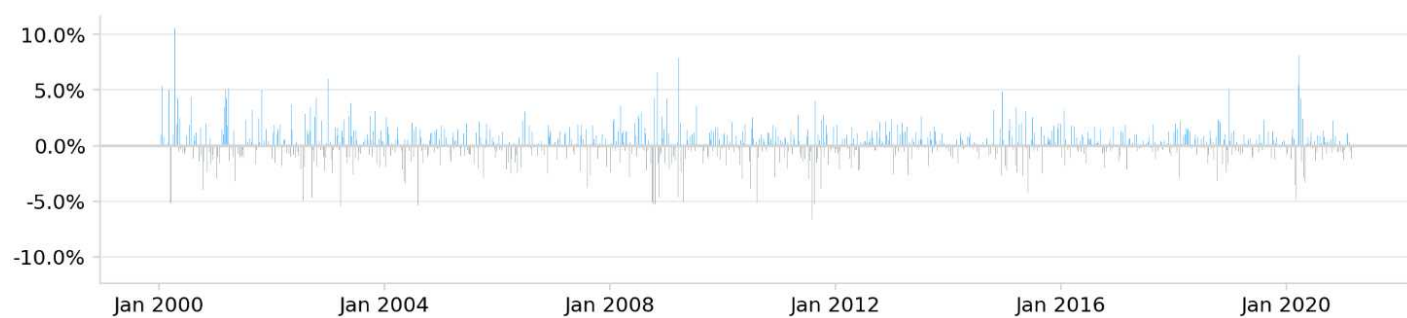
The backtest report displays charts to show the algorithm's returns per trade, per day, per month, per year, and the cumulative returns over the backtest.

Returns per Trade



This chart displays a histogram that shows the distribution of returns per trade over the backtesting period.

Daily Returns



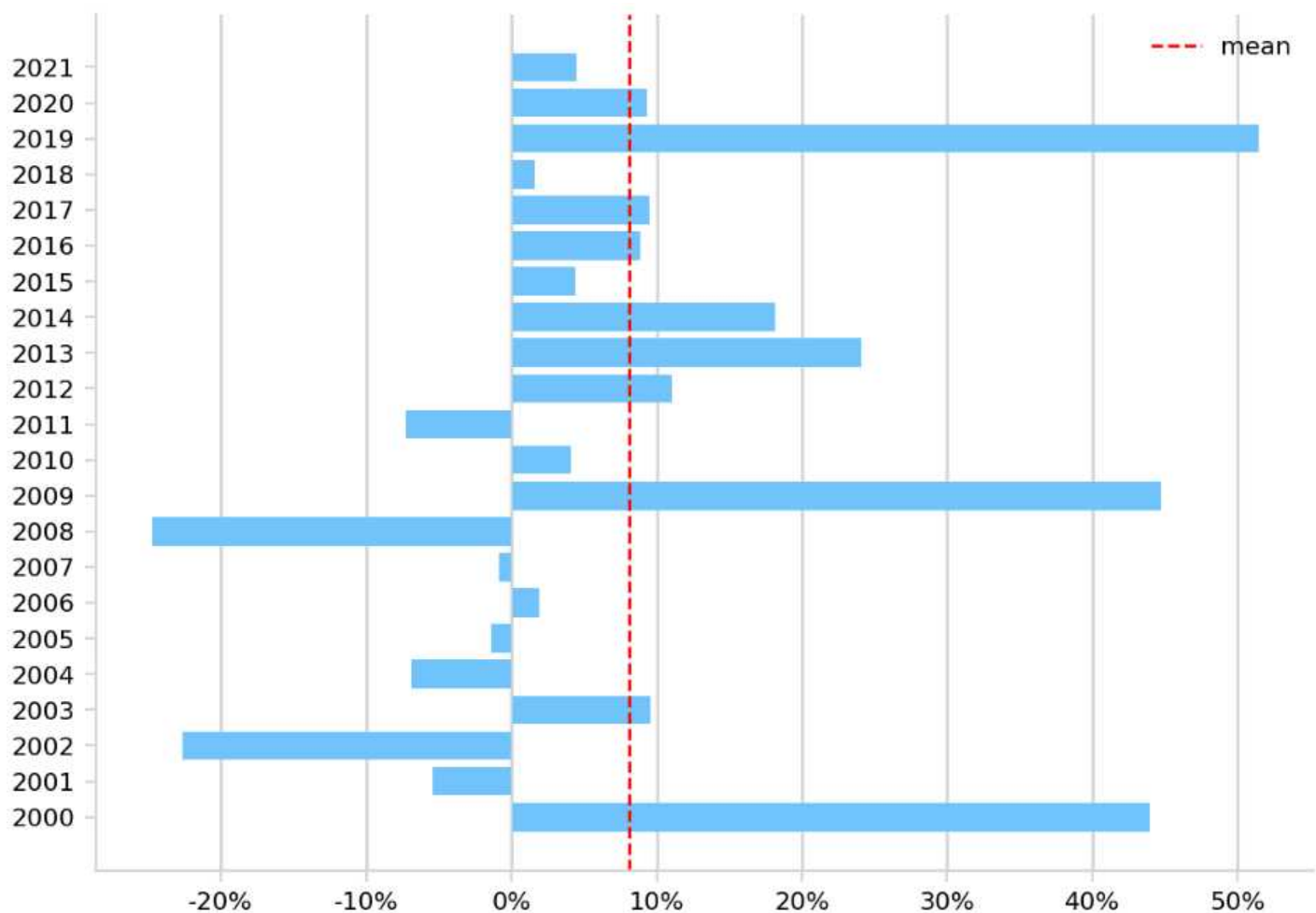
This chart displays the returns of each day. Blue bars represent profitable days and gray bars represent unprofitable days.

Monthly Returns

2000	-2.6	3.1	-2.6	0.1	-10.6	3.3	7.4	4.7	-2.2	9.8	11.7	15.2
2001	-10.2	-3.9	3.9	0.7	-3.7	-3.4	-8.3	-4.0	14.5	-3.5	3.5	9.2
2002	-4.7	18.7	-11.3	5.0	-5.4	-10.5	-8.9	0.2	-4.4	9.8	1.6	-10.0
2003	-2.8	-1.2	-2.1	8.0	1.4	6.3	6.3	6.7	-10.2	-2.1	0.3	-0.9
2004	4.3	-1.5	-2.7	-0.6	-7.2	1.1	-0.2	1.9	-0.3	0.7	1.2	-1.5
2005	-8.5	5.2	-0.3	-3.9	4.7	4.8	1.3	-2.4	-2.6	-1.1	4.1	-6.5
2006	1.1	0.1	-1.4	-9.6	-3.9	6.7	-1.5	2.7	2.3	-0.3	-1.5	4.0
2007	3.7	-1.0	-1.5	3.1	3.9	1.7	-1.9	-6.5	3.2	-0.0	-3.3	-4.0
2008	-4.3	3.4	-0.1	9.4	0.7	-11.3	4.8	1.6	-12.7	-20.6	-5.4	11.9
2009	2.9	-5.7	4.2	5.2	5.9	4.6	9.0	-0.5	6.5	0.1	1.6	8.9
2010	2.6	1.8	-1.0	-1.9	-7.0	0.7	1.7	-8.3	1.4	8.6	5.7	6.2
2011	3.6	4.5	1.1	-0.9	11.2	-9.2	0.4	-11.4	-1.7	6.3	-2.6	-3.8
2012	2.3	4.8	3.6	-3.1	-3.7	-6.0	1.0	6.6	1.6	4.2	2.0	-4.4
2013	-3.0	2.1	5.9	-0.5	2.2	-2.0	5.2	-1.8	3.4	2.1	4.8	3.4
2014	-4.7	5.8	-3.4	-1.2	-0.6	1.2	3.3	1.7	2.3	2.4	1.8	5.7
2015	-0.2	5.4	7.3	2.6	-1.5	-9.8	5.4	-15.4	-1.6	6.0	4.8	4.6
2016	-10.0	-0.9	8.8	-3.8	3.8	6.5	4.2	0.2	-0.9	-4.2	0.2	7.3
2017	-2.3	5.6	1.3	2.1	-2.2	-2.5	-2.4	1.3	-0.0	8.3	-1.4	1.5
2018	4.8	3.8	-6.6	1.2	4.6	-0.4	1.3	-0.6	2.1	-4.8	5.8	-9.8
2019	13.0	5.5	4.4	6.6	-4.2	8.8	3.1	-0.7	3.7	2.5	2.1	0.2
2020	2.7	-7.6	-14.1	16.0	2.6	1.9	3.4	10.8	-1.4	-5.1	8.4	-3.8
2021	3.3	-2.8	4.0									
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec

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



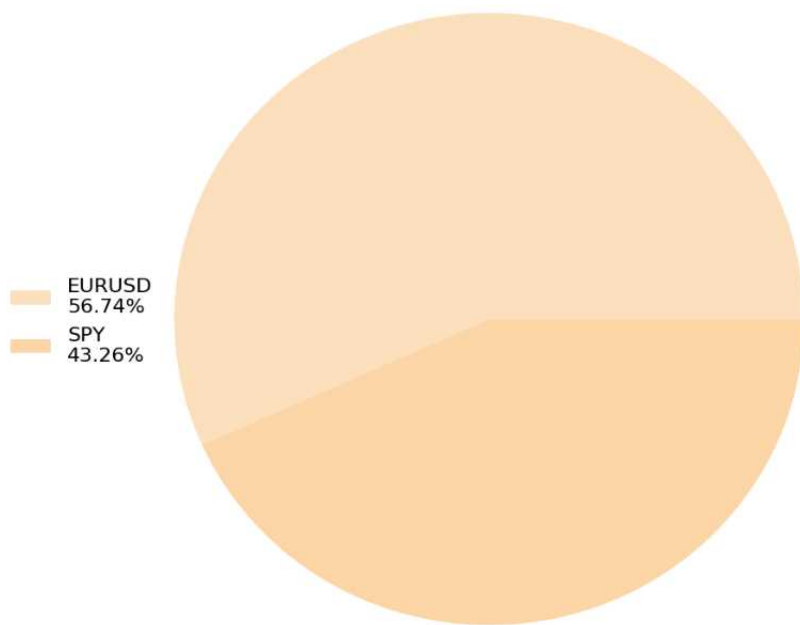
This chart displays the return of each year. We calculate the total return within each year and represent each year with a blue bar. The red dotted line represents the average of the annual returns.

Cumulative Returns



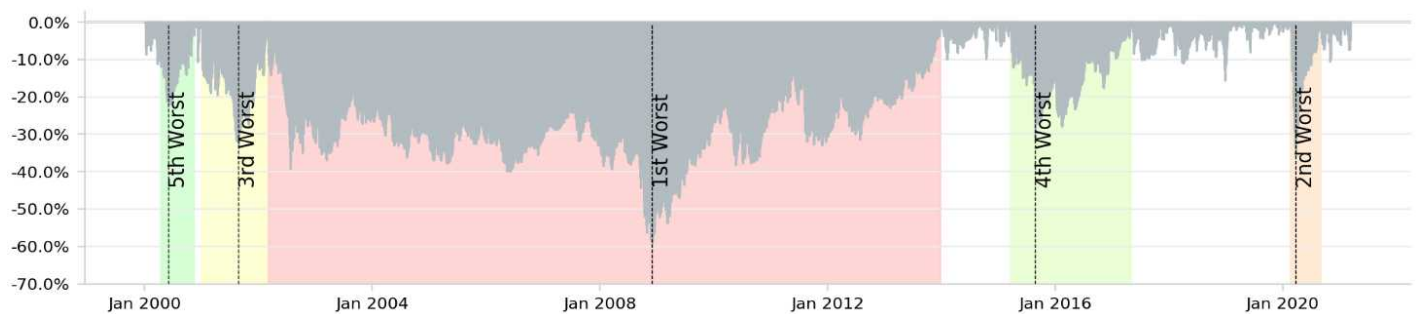
This chart displays the cumulative returns of your algorithm. The blue line represents your algorithm and the gray line represents the benchmark.

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

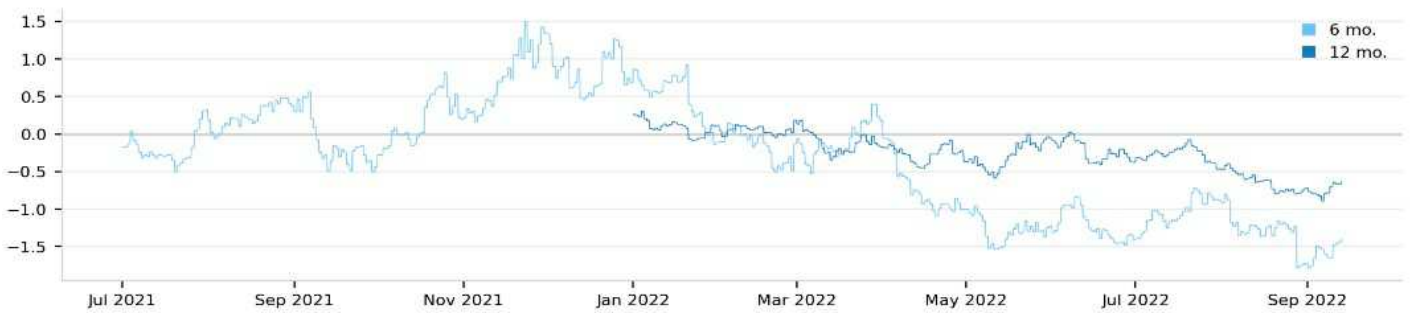
The backtest report displays time series for your portfolio's rolling [beta](#) and [Sharpe ratio](#) .

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

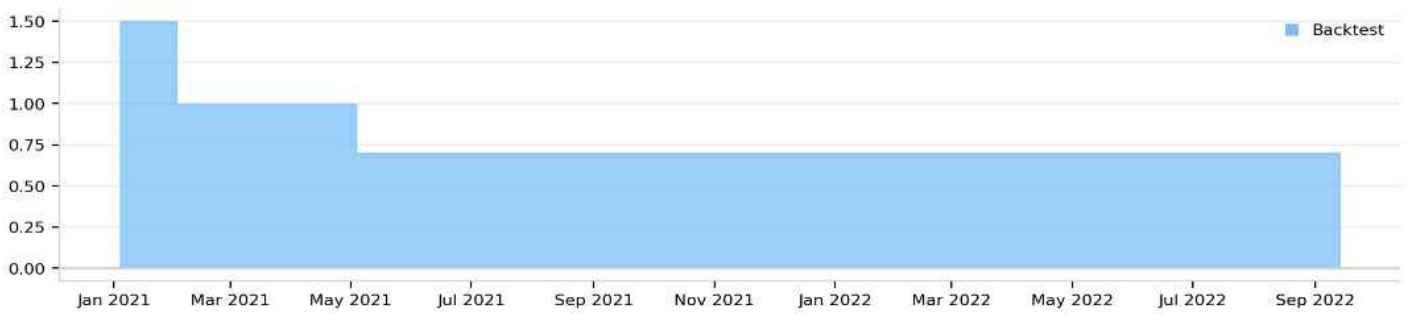


This chart displays the rolling portfolio Sharpe ratio over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Exposure

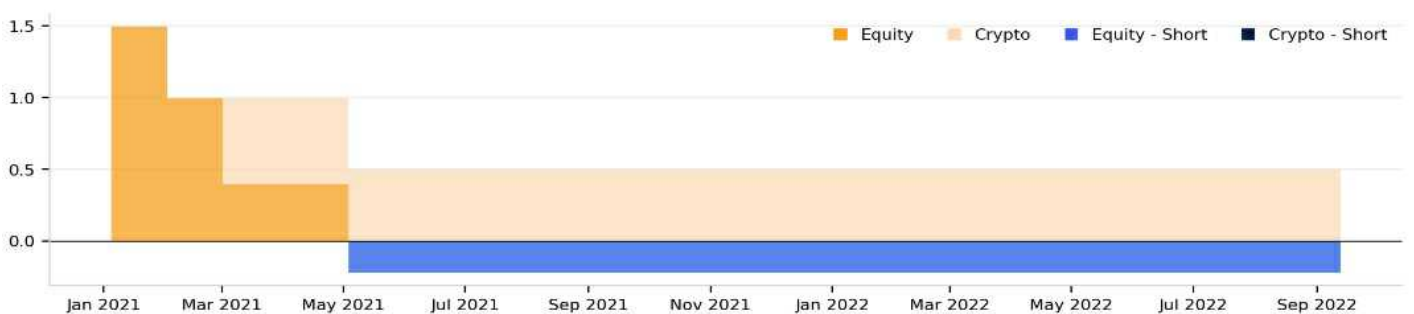
The backtest report displays time series for your portfolio's overall leverage and your portfolio's long-short exposure by asset class.

Leverage



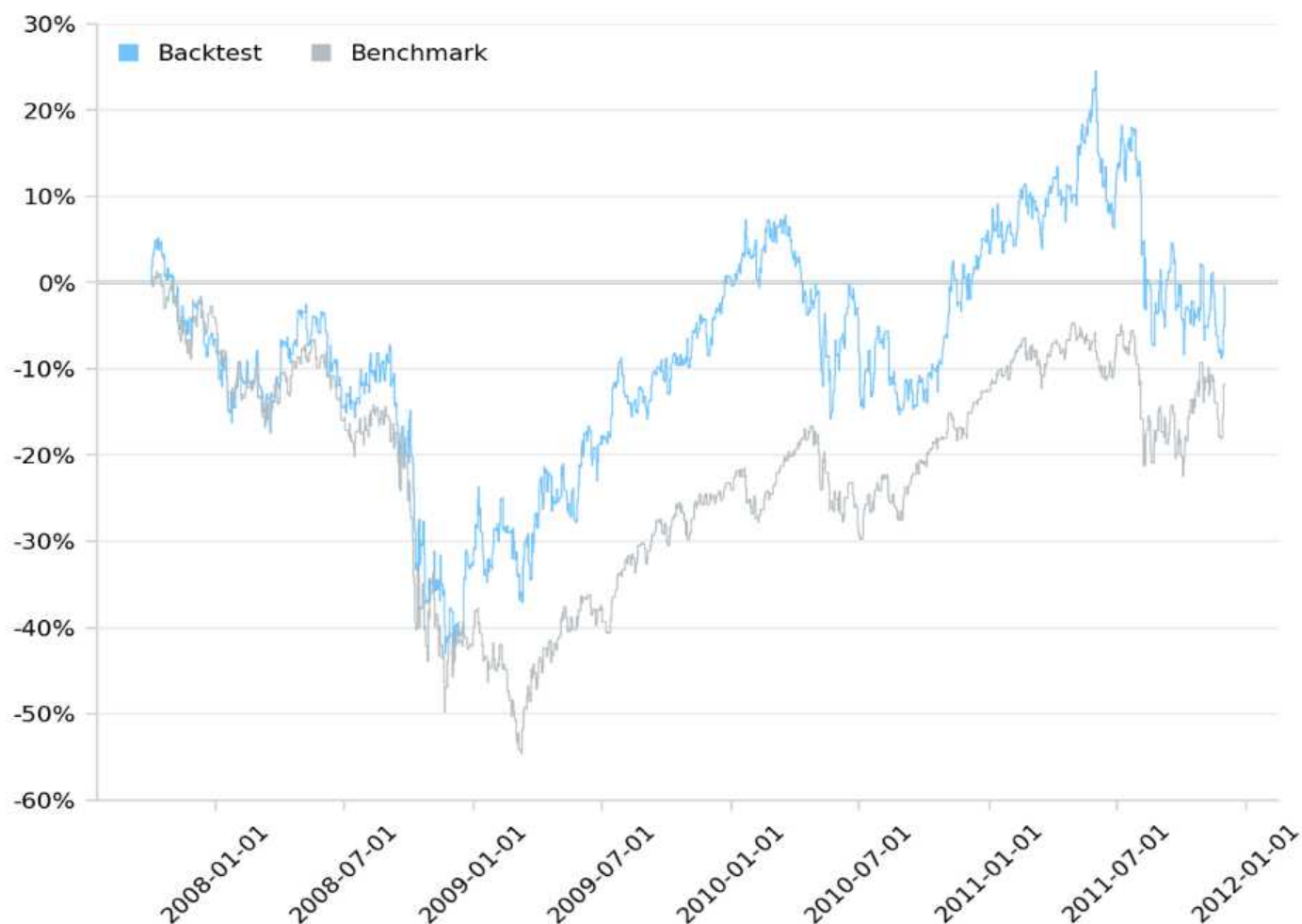
This chart displays your algorithm's utilization of leverage over time.

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
Post-COVID Run-up 2020-2021	4/1/2020	1/1/2022
Meme Season 2021	1/1/2021	5/15/2021
Russia Invades Ukraine 2022-2023	2/1/2022	1/1/2024
AI Boom 2022-Present	11/30/2022	Present

Parameters

This section of the report shows the name and value of all the [parameters](#) in your project.

Customize Reports

To create custom reports, customize the HTML and CSS.

Customize the Report HTML

The [Report / template.html](#) file in the LEAN GitHub repository defines the structure of the reports you generate. To override the HTML file, add a new HTML file to your local machine. If you add it to your [organization workspace](#) , don't name it report.html because that's the default name and location of the reports you generate. To include

some of the information and charts that are in the default report, use the report keys in the [Report / ReportKey.cs](#) file in the LEAN GitHub repository. For example, to add the [Sharpe ratio](#) of your backtest to the custom HTML file, use `{{KPI-SHARPE}}` .

To include the [crisis event plots](#) in your report, add the `{{HTML-CRISIS-PLOTS}}` key and then define the structure of the individual plots inside of `<!--crisis` and `crisis-->` . Inside of this comment, you can utilize the `{{TEXT-CRISIS-TITLE}}` and `{{PLOT-CRISIS-CONTENT}}` keys. For example, the following HTML is the default format for each crisis plot:

```
<!--crisis
<div class="col-xs-4">
  <table class="crisis-chart table compact">
    <thead>
      <tr>
        <th style="display: block; height: 75px;">{{TEXT-CRISIS-TITLE}}</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td style="padding:0;">
          
        </td>
      </tr>
    </tbody>
  </table>
</div>
crisis--&gt;
```

To include the [algorithm parameters](#) in your report, add the `{{PARAMETERS}}` key and then define the HTML element inside of `<!--parameters` and `parameters-->` . Inside of this comment, you can use special keys `{{KEY<parameterIndex>}}` and `{{VALUE<parameterIndex>}}` , which represent the key and value of a single parameter. For example, the following HTML is the default format for the parameters element:

```
<!--parameters
<tr>
  <td class = "title"> {{KEY0}} </td><td> {{VALUE0}} </td>
  <td class = "title"> {{KEY1}} </td><td> {{VALUE1}} </td>
</tr>
parameters--&gt;
```

In the preceding example, `{{KEY0}}` is the name of the first parameter in the algorithm and `{{VALUE0}}` is its value.

To generate the report with your custom HTML file, run `lean report --html <pathToCustomHTMLFile>` .

Customize the Report CSS

The [Report / css / report.css](#) file in the LEAN GitHub repository defines the style of the reports you generate. To override the stylesheet, add a new CSS file to your local machine.

To generate the report with your custom CSS file, run `lean report --css <pathToCustomCSSFile>` .

Optimization

Optimization > Parameters

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.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 `QCAAlgorithm.GetParameter(name)` in your algorithm to retrieve the value of a parameter and use that instead of constant values.

```
class ParameterizedAlgorithm(QCAAlgorithm):  
    def initialize(self) -> None:  
        self.set_start_date(2020, 1, 1)  
        self.set_cash(1000000)  
        self.add_equity("SPY")  
  
        fast_period = self.get_parameter("ema-fast", 10)  
        medium_period = self.get_parameter("ema-medium", 30)  
        slow_period = self.get_parameter("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 you run optimizations in the cloud, you don't need your own powerful machine.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

When you run a local optimization, the default data provider is your local machine. To use data from QuantConnect Cloud, a brokerage, or a third-party data provider, include the `--data-provider-historical` option. To view what brokerages and third-party data providers are available, see [the reference page for this command](#) .

Run Cloud Optimizations

When you run an optimization in QuantConnect Cloud, it uses the data from the [Dataset Market](#) . 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) 02-8 (2 cores, 8 GB RAM) @ $0.15 per hour
2) 04-12 (4 cores, 12 GB RAM) @ $0.30 per hour
3) 08-16 (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.

Object Store

Introduction

The Object Store is an organization-specific key-value storage location to save and retrieve data. Similar to a dictionary or hash table, a key-value store is a storage system that saves and retrieves objects by using keys. A key is a unique string that is associated with a single record in the key-value store and a value is an object being stored. Some common use cases of the Object Store include the following:

- Transporting data between the backtesting environment and the research environment.
- Training machine learning models in the research environment before deploying them to live trading.

The Object Store is shared across the entire organization. Using the same key, you can access data across all projects in an organization.

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Local Storage

To add files to your Object Store, add them to the storage directory of your [organization workspace](#) or call the [Save methods](#) in your algorithm.

To view the contents of the Object Store, run `lean object-store ls`.

```
$ lean object-store ls
```

This command opens File Explorer to show the storage directory. You can delete and rename files in the Object Store directly from the File Explorer. To edit files, open them in a text editor.

Cloud Storage

The CLI enables you to upload files to the Object Store, view a summary of your stored files, and delete files.

Upload Files

To upload files to the Object Store, run `lean cloud object-store set <storage-key> <local-file-path>`.

```
$ lean cloud object-store set 15737956/signals D:\qc\magic-signals.txt
Setting object 15737956/signals in organization d6d62db48592c72e67b534553413b691
```

If the file fails to upload, you may have insufficient [storage space](#). If you need more, [edit your storage plan](#).

List Directory Contents

To view all of the files and folders in the root directory of the Object Store, run `lean cloud object-store ls`.

```
$ lean cloud object-store ls
Key          Bytes  Folder  Filename
/15710069    None   True    15710069
/15727540    None   True    15727540
/15727540-1  None   True    15727540-1
/15730221    89649  False   15730221
/15730422    89748  False   15730422
```

To view all of the files and folders that are inside of one of the directories, run

`lean cloud object-store ls <folder-name>`.

```
$ lean cloud object-store ls 15710069
Key          Bytes  Folder  Filename
15710069/adjusted  60024  False   adjusted
15710069/raw      60143  False   raw
```

Get File Metadata

To view the metadata of a file in the Object Store, run `lean cloud object-store properties <path/to/file>`.

```
$ lean cloud object-store properties 15710069/adjusted
Bytes  Modified          Filename          Preview
60024  2023-08-30 23:08:23  15710069/adjusted  {"12723264
```

Delete Content

To delete a file or directory in the Object Storage, run `lean cloud object-store delete <key>`.

```
$ lean cloud object-store delete 15710069/adjusted
```

Download Files

Permissioned [Institutional](#) clients can build derivative data such as machine learning models and download it from the Object Store. [Contact us](#) to unlock this feature for your account.

To download a file or directory from the Object Store, run `lean cloud object-store get <key>`.

```
$ lean cloud object-store get 15710069/adjusted
Fetching object store download url
Unzipping object store keys values into: <Current Directory>
```

Bulk File Upload

To upload all files from a directory to the Object Store, run the following Python script:

```
from os import listdir, path
from subprocess import run
SRC = "" # Source directory
DST = "" # Destination directory (empty is root)
if __name__ == '__main__':

    command = [ "lean", "cloud", "object-store", "set" ]

    for key in listdir(SRC):
        fullname = path.join(SRC, key)
        if not path.isfile(fullname):
            continue
        key = DST + '/' + key
        args = [key, fullname]
        print(' '.join(command + args))
        run(command + args)
```

Live Trading Considerations

When you deploy a live algorithm, you can access the data within minutes of modifying the Object Store. Ensure your algorithm is able to handle a changing dataset.

API Reference

The Lean CLI is a cross-platform CLI which makes it easier to develop with the LEAN engine locally and in the cloud. You can use any of the following commands. Click one to learn more.

[lean backtest](#)

[lean build](#)

[lean cloud backtest](#)

[lean cloud live](#)

[lean cloud live command](#)

[lean cloud live deploy](#)

[lean cloud live liquidate](#)

[lean cloud live stop](#)

[lean cloud object-store delete](#)

[lean cloud object-store get](#)

[lean cloud object-store list](#)

[lean cloud object-store ls](#)

[lean cloud object-store properties](#)

[lean cloud object-store set](#)

[lean cloud optimize](#)

[lean cloud pull](#)

[lean cloud push](#)

[lean cloud status](#)

[lean config get](#)

[lean config list](#)

[lean config set](#)

[lean config unset](#)

[lean create-project](#)

[lean data download](#)

[lean data generate](#)

[lean decrypt](#)

lean delete-project

lean encrypt

lean init

lean library add

lean library remove

lean live

lean live add-security

lean live cancel-order

lean live command

lean live deploy

lean live liquidate

lean live stop

lean live submit-order

lean live update-order

lean login

lean logout

lean logs

lean object-store delete

lean object-store get

lean object-store list

lean object-store ls

lean object-store properties

lean object-store set

lean optimize

lean private-cloud add-compute

lean private-cloud start

lean private-cloud stop

lean project-create

lean project-delete

lean report

lean research

lean whoami

API Reference

lean backtest

Introduction

Backtest a project locally using Docker.

```
$ lean backtest [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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-historical` 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. The following table shows the available data providers and their required options in non-interactive mode:

<code>--data-provider-historical</code>	Required Options
Alpaca	<code>--alpaca-environment</code>
AlphaVantage	<code>--alpha-vantage-api-key</code>
	<code>--alpha-vantage-price-plan</code>
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>

	--binance-api-secret or --binanceus-api-secret
Bybit	--bybit-api-key
	--bybit-api-secret
CoinApi	--coinapi-api-key
	--coinapi-product
"Coinbase Advanced Trade"	--coinbase-api-name
	--coinbase-api-private-key
FactSet	--factset-auth-config-file
IEX	--iex-cloud-api-key
	--iex-price-plan
"Interactive Brokers"	--ib-user-name
	--ib-account
	--ib-password
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Kraken	--kraken-api-key
	--kraken-api-secret
	--kraken-verification-tier
Local	N/A
Oanda	--oanda-account-id
	--oanda-access-token
	--oanda-environment

Polygon	--polygon-api-key
QuantConnect	N/A
"Terminal Link"	--terminal-link-connection-type
	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
	--terminal-link-server-auth-id if you use --terminal-link-connection-type SAPI
ThetaData	--thetadata-subscription-plan
TradeStation	N/A

You can use the `--download-data` flag as an alias for `--data-provider-historical QuantConnect`. 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 won't have to download it again. If the file contain data for multiple days (for example, daily Equity price data files), the `ApiDataProvider` re-downloads the file if your local version is at least 7 days old. To adjust this setting, update the `downloader-data-update-period` value in your [Lean configuration](#) file.

You can also use 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
<code><project></code>	The path to the project directory or algorithm file to backtest.

Options

The `lean backtest` command supports the following options:

Option	Description
<code>--output <directory></code>	Directory to store results in (defaults to PROJECT/backtests/TIMESTAMP)
<code>-d, --detach</code>	Run the backtest in a detached Docker container and return immediately
<code>--debug <enum: pycharm ptvsd debugpy vsdbg rider local-platform></code>	Enable a certain debugging method (see --help for more information)

<pre>--data-provider-historical <enum: Interactive Brokers Oanda Bitfinex Coinbase Advanced Trade Binance Kraken CharlesSchwab IQFeed Polygon FactSet AlphaVantage CoinApi ThetaData QuantConnect Local Terminal Link Bybit TradeStation Alpaca Tastytrade Eze></pre>	<p>Update the Lean configuration file to retrieve data from the given historical provider</p>
<pre>--ib-user-name <string></pre>	Your Interactive Brokers username
<pre>--ib-account <string></pre>	Your Interactive Brokers account id
<pre>--ib-password <string></pre>	Your Interactive Brokers password
<pre>--ib-weekly-restart-utc-time <string></pre>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)
<pre>--oanda-account-id <string></pre>	Your OANDA account id
<pre>--oanda-access-token <string></pre>	Your OANDA API token

<code>--oanda-environment <enum: Practice Trade></code>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<code>--bitfinex-api-key <string></code>	Your Bitfinex API key
<code>--bitfinex-api-secret <string></code>	Your Bitfinex API secret
<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file
<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binance exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--kraken-api-key <string></code>	Your Kraken API key
<code>--kraken-api-secret <string></code>	Your Kraken API secret
<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--iqfeed-iqconnect <string></code>	The path to the IQConnect binary
<code>--iqfeed-username <string></code>	Your IQFeed username
<code>--iqfeed-password <string></code>	Your IQFeed password
<code>--iqfeed-version <string></code>	The product version of your IQFeed developer account
<code>--iqfeed-host <string></code>	The IQFeed host address (Optional)
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--factset-auth-config-file <file></code>	The path to the FactSet authentication configuration file
<code>--alpha-vantage-api-key <string></code>	Your Alpha Vantage Api Key

<code>--alpha-vantage-price-plan <enum: Free Plan30 Plan75 Plan150 Plan300 Plan600 Plan1200></code>	Your Alpha Vantage Premium API Key plan
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key
<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--thetadatas-ws-url <string></code>	The ThetaData host address (Optional)
<code>--thetadatas-rest-url <string></code>	The ThetaData host address (Optional)
<code>--thetadatas-subscription-plan <enum: Free Value Standard Pro></code>	Your ThetaData subscription price plan
<code>--terminal-link-connection-type <enum: DAPI S API></code>	Terminal Link Connection Type [DAPI, SAPI]
<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in
<code>--terminal-link-server-host <string></code>	The host of the TerminalLink server
<code>--terminal-link-server-port <integer></code>	The port of the TerminalLink server
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--bybit-api-key <string></code>	Your Bybit API key
<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--trade-station-environment <enum: live paper ></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number

<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password
<code>--download-data</code>	Update the Lean configuration file to download data from the QuantConnect API, alias for <code>--data-provider-historical</code> QuantConnect
<code>--data-purchase-limit <integer></code>	The maximum amount of QCC to spend on downloading data during the backtest when using QuantConnect as historical data provider
<code>--release</code>	Compile C# projects in release configuration instead of debug
<code>--image <string></code>	The LEAN engine image to use (defaults to quantconnect/lean:latest)
<code>--python-venv <string></code>	The path of the python virtual environment to be used
<code>--update</code>	Pull the LEAN engine image before running the backtest
<code>--backtest-name <string></code>	Backtest name
<code>--extra-docker-config <string></code>	Extra docker configuration as a JSON string. For more information https://docker-py.readthedocs.io/en/stable/containers.html
<code>--no-update</code>	Use the local LEAN engine image instead of pulling the latest version
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean backtest</code> command and exit

API Reference

lean build

Introduction

Build Docker images of your own version of LEAN.

```
$ lean build [OPTIONS] [ROOT]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code><lean></code>	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
<code>--tag <string></code>	The tag to apply to custom images (defaults to latest)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean build</code> command and exit

API Reference

lean cloud backtest

Introduction

Backtest a project in the cloud.

```
$ lean cloud backtest [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--name <string></code>	The name of the backtest (a random one is generated if not specified)
<code>--push</code>	Push local modifications to the cloud before running the backtest
<code>--open</code>	Automatically open the results in the browser when the backtest is finished
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud backtest</code> command and exit

API Reference

lean cloud live

Introduction

Interact with the QuantConnect cloud live deployments.

```
$ lean cloud live [OPTIONS] COMMAND [ARGS]...
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean cloud live` command supports the following options:

Option	Description
<code>--help</code>	Display the help text of the <code>lean cloud live</code> command and exit

Commands

The `lean cloud live` command expects the following arguments:

Argument	Description
<code>command</code>	Send a command to a running cloud live trading project
<code>deploy</code>	Start live trading for a project in the cloud
<code>liquidate</code>	Stops live trading and liquidates existing positions for a certain project
<code>stop</code>	Stops live trading for a certain project without liquidating existing positions

API Reference

lean cloud live command

Introduction

Send a command to a running cloud live trading project.

```
$ lean cloud live command [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean cloud live command` command supports the following options:

Option	Description
<code>--data <string></code>	The command to send, 'str' representation of a 'dict' e.g. <code>{"target\":"BTCUSD\","\$type\":"MyCommand\}"</code>
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud live command</code> command and exit

API Reference

lean cloud live deploy

Introduction

Start live trading for a project in the cloud.

```
$ lean cloud live deploy [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Starts live trading for a cloud project. Before starting live trading, the CLI shows an interactive wizard letting you configure the brokerage, data provider, live node, and 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` and `--data-provider-live` options, 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`. 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 following options are required for each brokerage in non-interactive mode:

<code>--brokerage</code>	Required Options
"Paper Trading"	N/A
Alpaca	<code>--alpaca-environment</code>
	<code>--alpaca-api-key</code>
	<code>--alpaca-api-secret</code>
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>
	<code>--binance-api-secret</code> or <code>--binanceus-api-secret</code>

	--binance-use-testnet
Bitfinex	--bitfinex-api-key
	--bitfinex-api-secret
Bybit	--bybit-api-key
	--bybit-api-secret
	--bybit-vip-level
"Coinbase Advanced Trade"	--coinbase-api-name
	--coinbase-api-private-key
"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

"Terminal Link"	--terminal-link-server-auth-id
	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-account
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
TradeStation	--trade-station-environment
	--trade-station-account-type
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-order-routing-sender-comp-id
Zerodha	--zerodha-api-key
	--zerodha-access-token
	--zerodha-product-type
	--zerodha-trading-segment
	--zerodha-history-subscription

The `--data-provider-live` option is required. The following table shows the available live data providers and their

required options in non-interactive mode. To select multiple data providers, separate them with a comma. The order you select them in defines the order of precedence.

<code>--data-provider-live</code>	Required Options
Alpaca	All options required by <code>--brokerage Alpaca</code> .
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>
	<code>--binance-api-secret</code> or <code>--binanceus-api-secret</code>
Bitfinex	All options required by <code>--brokerage Bitfinex</code> .
Bybit	All options required by <code>--brokerage Bybit</code> .
CoinApi	<code>--coinapi-api-key</code>
	<code>--coinapi-product</code>
"Coinbase Advanced Trade"	<code>--coinbase-api-name</code>
	<code>--coinbase-api-private-key</code>
IEX	<code>--iex-cloud-api-key</code>
	<code>--iex-price-plan</code>
"Interactive Brokers"	All options required by <code>--brokerage "Interactive Brokers"</code> .
Kraken	All options required by <code>--brokerage Kraken</code> .
Oanda	<code>--oanda-account-id</code>
	<code>--oanda-access-token</code>
Polygon	<code>--polygon-api-key</code>
QuantConnect	N/A
Samco	All options required by <code>--brokerage Samco</code> .
TD Ameritrade	All options required by <code>--brokerage TD Ameritrade</code> .
"Terminal Link"	All options required by <code>--brokerage "Terminal Link"</code> .
TradeStation	All options required by <code>--brokerage TradeStation</code> .

Tradier	--tradier-account-id
	--tradier-access-token
"Trading Technologies"	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key
	--tt-rest-app-secret
	--tt-rest-environment
	--tt-order-routing-sender-comp-id
Zerodha	All options required by --brokerage Zerodha .
	--zerodha-history-subscription

If you omit some of the required 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 deploy "My Project" \
  --brokerage "Paper Trading" \
  --data-provider-live QuantConnect \
  --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 deploy` command expects the following arguments:

Argument	Description
<project>	The name or Id of the project to start live trading.

Options

The `lean cloud live deploy` command supports the following options:

Option	Description
<code>--brokerage <enum: Paper Trading Interactive Brokers Tradier Oanda Bitfinex Coinbase Advanced Trade Binance Zerodha Samco Terminal Link Trading Technologies Kraken CharlesSchwab Bybit TradeStation Alpaca Tastytrade Eze></code>	The brokerage to use
<code>--data-provider-live <enum: QuantConnect Interactive Brokers Tradier Oanda Bitfinex Coinbase Advanced Trade Binance Zerodha Samco Terminal Link Trading Technologies Kraken CharlesSchwab Polygon CoinApi Bybit TradeStation Alpaca Tastytrade Eze></code>	The live data provider to use
<code>--ib-user-name <string></code>	Your Interactive Brokers username
<code>--ib-account <string></code>	Your Interactive Brokers account id
<code>--ib-password <string></code>	Your Interactive Brokers password
<code>--ib-weekly-restart-utc-time <string></code>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)

<code>--ib-financial-advisors-group-filter <string></code>	Use this field to filter data by a specific financial advisor group within your Interactive Brokers account. Leave blank to include all groups. (Optional)
<code>--tradier-account-id <string></code>	Your Tradier account id
<code>--tradier-access-token <string></code>	Your Tradier access token
<code>--tradier-environment <enum: live paper></code>	Whether the developer sandbox should be used
<code>--oanda-account-id <string></code>	Your OANDA account id
<code>--oanda-access-token <string></code>	Your OANDA API token
<code>--oanda-environment <enum: Practice Trade></code>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<code>--bitfinex-api-key <string></code>	Your Bitfinex API key
<code>--bitfinex-api-secret <string></code>	Your Bitfinex API secret
<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file
<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binance exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--binance-use-testnet <enum: live paper></code>	Whether the testnet should be used
<code>--zerodha-api-key <string></code>	Your Kite Connect API key
<code>--zerodha-access-token <string></code>	Your Kite Connect access token
<code>--zerodha-product-type <enum: mis cnc nrml></code>	MIS if you are targeting intraday products, CNC if you are targeting delivery products, NRML if you are targeting carry forward products
<code>--zerodha-trading-segment <enum: equity commodity></code>	EQUITY if you are trading equities on NSE or BSE, COMMODITY if you are trading commodities on MCX

<code>--zerodha-history-subscription <enum: true false></code>	Whether you have a history API subscription for Zerodha
<code>--samco-client-id <string></code>	Your Samco account Client ID
<code>--samco-client-password <string></code>	Your Samco account password
<code>--samco-year-of-birth <string></code>	Your year of birth (YYYY) registered with Samco
<code>--samco-product-type <enum: mis cnc nrml></code>	MIS if you are targeting intraday products, CNC if you are targeting delivery products, NRML if you are targeting carry forward products
<code>--samco-trading-segment <enum: equity commodity></code>	EQUITY if you are trading equities on NSE or BSE, COMMODITY if you are trading commodities on MCX
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in
<code>--terminal-link-server-host <string></code>	The host of the SAPI server
<code>--terminal-link-server-port <integer></code>	The port of the SAPI server
<code>--terminal-link-emsx-broker <string></code>	The EMSX broker to use (Optional)
<code>--terminal-link-emsx-account <string></code>	The EMSX account to use (Optional)
<code>--terminal-link-emsx-team <string></code>	The EMSX team to receive order events from (Optional)
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--tt-user-name <string></code>	Your Trading Technologies username
<code>--tt-session-password <string></code>	Your Trading Technologies session password
<code>--tt-account-name <string></code>	Your Trading Technologies account name
<code>--tt-rest-app-key <string></code>	Your Trading Technologies REST app key
<code>--tt-rest-app-secret <string></code>	Your Trading Technologies REST app secret
<code>--tt-rest-environment <string></code>	The REST environment to run in
<code>--tt-order-routing-sender-comp-id <string></code>	The order routing sender comp id to use
<code>--kraken-api-key <string></code>	Your Kraken API key

<code>--kraken-api-secret <string></code>	Your Kraken API secret
<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--bybit-api-key <string></code>	Your Bybit API key
<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--bybit-vip-level <enum: VIP0 VIP1 VIP2 VIP3 VIP4 VIP5 SupremeVIP Pro1 Pro2 Pro3 Pro4 Pro5></code>	Your Bybit VIP Level
<code>--trade-station-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number
<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password
<code>--eze-trading-account <string></code>	Format: BANK;BRANCH;CUSTOMER;DEPOSIT
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key

<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--alpaca-api-key <string></code>	Your Alpaca Api Key
<code>--alpaca-api-secret <string></code>	Your Alpaca Api Secret
<code>--node <string></code>	The name or id of the live node to run on
<code>--auto-restart <boolean></code>	Whether automatic algorithm restarting must be enabled
<code>--notify-order-events <boolean></code>	Whether notifications must be sent for order events
<code>--notify-insights <boolean></code>	Whether notifications must be sent for emitted insights
<code>--notify-emails <string></code>	A comma-separated list of 'email:subject' pairs configuring email-notifications
<code>--notify-webhooks <string></code>	A comma-separated list of 'url:HEADER_1=VALUE_1:HEADER_2=VALUE_2:etc' pairs configuring webhook-notifications
<code>--notify-sms <string></code>	A comma-separated list of phone numbers configuring SMS-notifications
<code>--notify-telegram <string></code>	A comma-separated list of 'user/group id:token(optional)' pairs configuring telegram-notifications
<code>--live-cash-balance <string></code>	A comma-separated list of currency:amount pairs of initial cash balance
<code>--live-holdings <string></code>	A comma-separated list of symbol:symbolId:quantity:averagePrice of initial portfolio holdings
<code>--push</code>	Push local modifications to the cloud before starting live trading
<code>--open</code>	Automatically open the live results in the browser once the deployment starts
<code>--show-secrets</code>	Show secrets as they are input
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud live deploy</code> command and exit

API Reference

lean cloud live liquidate

Introduction

Stops live trading and liquidates existing positions for a certain project.

```
$ lean cloud live liquidate [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Arguments

The `lean cloud live liquidate` command expects the following arguments:

Argument	Description
<code><project></code>	The name or Id of the project to liquidate.

Options

The `lean cloud live liquidate` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud live liquidate</code> command and exit

API Reference

lean cloud live stop

Introduction

Stops live trading for a certain project without liquidating existing positions.

```
$ lean cloud live stop [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud live stop</code> command and exit

API Reference

lean cloud object-store delete

Introduction

Delete a value from the organization's cloud object store.

```
$ lean cloud object-store delete [OPTIONS] KEY
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Arguments

The `lean cloud object-store delete` command expects the following arguments:

Argument	Description
<key>	The key to a value in the Object Store.

Options

The `lean cloud object-store delete` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store delete</code> command and exit

API Reference

lean cloud object-store get

Introduction

Download an object store value to disk from the organization's cloud object store.

```
$ lean cloud object-store get [OPTIONS] [KEY]...
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Arguments

The `lean cloud object-store get` command expects the following arguments:

Argument	Description
<key>	The key to a file in the Object Store.

Options

The `lean cloud object-store get` command supports the following options:

Option	Description
<code>--destination-folder <string></code>	The destination folder to download the object store values, if not provided will use to current directory
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store get</code> command and exit

API Reference

lean cloud object-store list

Introduction

List all values for the given root key in the organization's cloud object store.

```
$ lean cloud object-store list [OPTIONS] [KEY]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Arguments

The `lean cloud object-store list` command expects the following arguments:

Argument	Description
(Optional) <code><key></code>	The path to a directory in the Object Store.

If you don't provide the `<key>`, `lean cloud object-store list` lists the contents of the root directory.

Options

The `lean cloud object-store list` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store list</code> command and exit

API Reference

lean cloud object-store ls

Introduction

Alias for 'list'

```
$ lean cloud object-store ls [OPTIONS] [KEY]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Arguments

The `lean cloud object-store ls` command expects the following arguments:

Argument	Description
(Optional) <code><key></code>	The path to a directory in the Object Store.

If you don't provide the `<key>`, `lean cloud object-store ls` lists the contents of the root directory.

Options

The `lean cloud object-store ls` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store ls</code> command and exit

API Reference

lean cloud object-store properties

Introduction

Get a value properties from the organization's cloud object store.

```
$ lean cloud object-store properties [OPTIONS] KEY
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean cloud object-store properties` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store properties</code> command and exit

API Reference

lean cloud object-store set

Introduction

Sets the data to the given key in the organization's cloud object store.

```
$ lean cloud object-store set [OPTIONS] KEY PATH
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Arguments

The `lean cloud object-store set` command expects the following arguments:

Argument	Description
<key>	The key under which to save the file.
<path>	Path to the local file to upload.

Options

The `lean cloud object-store set` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud object-store set</code> command and exit

API Reference

lean cloud optimize

Introduction

Optimize a project in the cloud.

```
$ lean cloud optimize [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 04-12 \  
  --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
<code><project></code>	The name or Id of the project to optimize.

Options

The `lean cloud optimize` command supports the following options:

Option	Description
<code>--target <string></code>	The target statistic of the optimization
<code>--target-direction <enum: min max></code>	Whether the target must be minimized or maximized
<code>--parameter <<string> <float> <float> <float> >...</code>	The 'parameter min max step' pairs configuring the parameters to optimize
<code>--constraint <string></code>	The 'statistic operator value' pairs configuring the constraints of the optimization
<code>--node <enum: 02-8 04-12 08-16></code>	The node type to run the optimization on
<code>--parallel-nodes <integer></code>	The number of nodes that may be run in parallel
<code>--name <string></code>	The name of the optimization (a random one is generated if not specified)
<code>--push</code>	Push local modifications to the cloud before starting the optimization
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud optimize</code> command and exit

API Reference

lean cloud pull

Introduction

Pull projects from QuantConnect to the local drive.

```
$ lean cloud pull [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--project <string></code>	Name or id of the project to pull (all cloud projects if not specified)
<code>--pull-bootcamp</code>	Pull Boot Camp projects (disabled by default)
<code>--encrypt</code>	Pull your cloud files and encrypt them before saving on your local drive
<code>--decrypt</code>	Pull your cloud files and decrypt them before saving on your local drive
<code>--key <file></code>	Path to the encryption key to use
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud pull</code> command and exit

API Reference

lean cloud push

Introduction

Push local projects to QuantConnect.

```
$ lean cloud push [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Pushes local projects to QuantConnect while preserving the directory structure of your local projects. The project's [supported 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
<code>--project <directory></code>	Path to the local project to push (all local projects if not specified)
<code>--encrypt</code>	Push your local files and encrypt them before saving on the cloud
<code>--decrypt</code>	Push your local files and decrypt them before saving on the cloud
<code>--key <file></code>	Path to the encryption key to use
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud push</code> command and exit

API Reference

lean cloud status

Introduction

Show the live trading status of a project in the cloud.

```
$ lean cloud status [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code><project></code>	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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean cloud status</code> command and exit

API Reference

lean config get

Introduction

Get the current value of a configurable option.

```
$ lean config get [OPTIONS] KEY
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code><key></code>	The key of the value to retrieve. Run <code>lean config list</code> to get a list of all available keys.

Options

The `lean config get` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean config get</code> command and exit

API Reference

lean config list

Introduction

List the configurable options and their current values.

```
$ lean config list [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean config list</code> command and exit

API Reference

lean config set

Introduction

Set a configurable option.

```
$ lean config set [OPTIONS] KEY VALUE
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>user-id</code>	The user Id used when making authenticated requests to the QuantConnect API.
<code>api-token</code>	The API token used when making authenticated requests to the QuantConnect API.
<code>default-language</code>	The default language used when creating new projects (must be either <code>python</code> or <code>csharp</code>).
<code>engine-image</code>	The Docker image used when running the LEAN engine (<code>quantconnect/lean:latest</code> if not set).
<code>research-image</code>	The Docker image used when running the research environment (<code>quantconnect/research:latest</code> if not set).

Arguments

The `lean config set` command expects the following arguments:

Argument	Description
<code><key></code>	The key of the option to update.
<code><value></code>	The new value for the option with the given key.

Options

The `lean config set` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean config set</code> command and exit

API Reference

lean config unset

Introduction

Unset a configurable option.

```
$ lean config unset [OPTIONS] KEY
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code><key></code>	The key of the option to unset.

Options

The `lean config unset` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean config unset</code> command and exit

API Reference

lean create-project

Introduction

Alias for 'project-create'

```
$ lean create-project [OPTIONS] NAME
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 create-project** 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 create-project` command supports the following options:

Option	Description
-l, --language <enum: python csharp>	The language of the project to create
--verbose	Enable debug logging
--help	Display the help text of the <code>lean create-project</code> command and exit

API Reference

lean data download

Introduction

Purchase and download data directly from QuantConnect or download from supported data providers

```
$ lean data download [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Lets you purchase and download historical data from the [QuantConnect Dataset Market](#) , a brokerage, or a third-party data provider.

Download from QuantConnect

When you download data from the QuantConnect Dataset Market, 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.

Data from the Dataset Market 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 "BTCUSD" \
  --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.

To automatically confirm payment confirmation prompts, use the `--yes` flag.

Download from Brokerages and Third-Party Providers

To download data from a brokerage or third-party data provider, run this command and follow the prompts to select the data you need. To view all the available data providers, see the `--data-provider-historical` option description. After you know what options you need to provide to download data from your specific brokerage or third-party data provider, you can run the command in a script to automate the process of downloading new data.

Options

The `lean data download` command supports the following options:

Option	Description
<code>--data-provider-historical</code> <enum: Interactive Brokers Oanda Bitfinex Coinbase Advanced Trade Binance Kraken CharlesSchwab IQFeed Polygon FactSet AlphaVantage CoinApi ThetaData QuantConnect Local Terminal Link Bybit TradeStation Alpaca Tastytrade Eze>	The name of the downloader data provider
<code>--ib-user-name</code> <string>	Your Interactive Brokers username
<code>--ib-account</code> <string>	Your Interactive Brokers account id

<code>--ib-password <string></code>	Your Interactive Brokers password
<code>--ib-weekly-restart-utc-time <string></code>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)
<code>--oanda-account-id <string></code>	Your OANDA account id
<code>--oanda-access-token <string></code>	Your OANDA API token
<code>--oanda-environment <enum: Practice Trade></code>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<code>--bitfinex-api-key <string></code>	Your Bitfinex API key
<code>--bitfinex-api-secret <string></code>	Your Bitfinex API secret
<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file
<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binance exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--kraken-api-key <string></code>	Your Kraken API key
<code>--kraken-api-secret <string></code>	Your Kraken API secret
<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--iqfeed-iqconnect <string></code>	The path to the IQConnect binary
<code>--iqfeed-username <string></code>	Your IQFeed username
<code>--iqfeed-password <string></code>	Your IQFeed password

<code>--iqfeed-version <string></code>	The product version of your IQFeed developer account
<code>--iqfeed-host <string></code>	The IQFeed host address (Optional)
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--factset-auth-config-file <file></code>	The path to the FactSet authentication configuration file
<code>--alpha-vantage-api-key <string></code>	Your Alpha Vantage Api Key
<code>--alpha-vantage-price-plan <enum: Free Plan30 Plan75 Plan150 Plan300 Plan600 Plan1200></code>	Your Alpha Vantage Premium API Key plan
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key
<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--thetadatas-ws-url <string></code>	The ThetaData host address (Optional)
<code>--thetadatas-rest-url <string></code>	The ThetaData host address (Optional)
<code>--thetadatas-subscription-plan <enum: Free Value Standard Pro></code>	Your ThetaData subscription price plan
<code>--terminal-link-connection-type <enum: DAPI S API></code>	Terminal Link Connection Type [DAPI, SAPI]
<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in
<code>--terminal-link-server-host <string></code>	The host of the TerminalLink server
<code>--terminal-link-server-port <integer></code>	The port of the TerminalLink server
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--bybit-api-key <string></code>	Your Bybit API key

<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--trade-station-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number
<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password
<code>--dataset <string></code>	The name of the dataset to download non-interactively
<code>--overwrite</code>	Overwrite existing local data
<code>-y, --yes</code>	Automatically confirm payment confirmation prompts
<code>--data-type <enum: Trade Quote Bulk Universe OpenInterest></code>	Specify the type of historical data
<code>--resolution <enum: Tick Second Minute Hour Daily></code>	Specify the resolution of the historical data
<code>--security-type <enum: Equity Index Forex Cfd Future Crypto CryptoFuture Option IndexOption Commodity FutureOption></code>	Specify the security type of the historical data

<code>--market <string></code>	Specify the market name for tickers (e.g., 'USA', 'NYMEX', 'Binance') (if not provided or empty the default market for the requested security type will be used)
<code>--ticker <string></code>	Specify comma separated list of tickers to use for historical data request
<code>--start <string></code>	Specify the start date for the historical data request in the format yyyyMMdd
<code>--end <string></code>	Specify the end date for the historical data request in the format yyyyMMdd. (defaults to today)
<code>--image <string></code>	The LEAN engine image to use (defaults to quantconnect/lean:latest)
<code>--update</code>	Pull the LEAN engine image before running the Downloader Data Provider
<code>--no-update</code>	Use the local LEAN engine image instead of pulling the latest version
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean data download</code> command and exit

API Reference

lean data generate

Introduction

Generate random market data.

```
$ lean data generate [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--start <yyyyMMdd></code>	[required]
<code>--end <yyyyMMdd></code>	End date for the data to generate in yyyyMMdd format (defaults to today)
<code>--symbol-count <integer> <range></code>	[x>=0]
<code>--tickers <string></code>	Comma separated list of tickers to use for generated data
<code>--security-type <enum: Equity Forex Cfd Future Crypto Option></code>	The security type to generate data for (defaults to Equity)

<code>--resolution <enum: Tick Second Minute Hour Daily></code>	The resolution of the generated data (defaults to Minute)
<code>--data-density <enum: Dense Sparse VerySparse></code>	The density of the generated data (defaults to Dense)
<code>--include-coarse <boolean></code>	Whether coarse universe data should be generated for Equity data (defaults to True)
<code>--market <string></code>	The market to generate data for (defaults to standard market for the security type)
<code>--quote-trade-ratio <float></code>	The ratio of generated quotes to generated trades. Values larger than 1 mean more quotes than trades. Only used for Option, Future and Crypto (defaults to 1)
<code>--random-seed <integer> <range></code>	The random number generator seed. Defaults to None, which means no seed will be used [x>=0]
<code>--ipo-percentage <float></code>	The probability each equity generated will have an IPO event. Note that this is not the total probability for all symbols generated. Only used for Equity (defaults to 5.0)
<code>--rename-percentage <float></code>	The probability each equity generated will have a rename event. Note that this is not the total probability for all symbols generated. Only used for Equity (defaults to 30.0)
<code>--splits-percentage <float></code>	The probability each equity generated will have a stock split event. Note that this is not the total probability for all symbols generated. Only used for Equity (defaults to 15.0)
<code>--dividends-percentage <float></code>	The probability each equity generated will have dividends. Note that this is not the probability for all symbols generated. Only used for Equity (defaults to 60.0)
<code>--dividend-every-quarter-percentage <float></code>	The probability each equity generated will have a dividend event every quarter. Note that this is not the total probability for all symbols generated. Only used for Equity (defaults to 30.0)
<code>--option-price-engine <string></code>	The stochastic process, and returns new pricing engine to run calculations for that option (defaults to BaroneAdesiWhaleyApproximationEngine)

<code>--volatility-model-resolution <enum: Tick Second Minute Hour Daily></code>	The volatility model period span (defaults to Daily)
<code>--chain-symbol-count <integer> <range></code>	[x>=0]
<code>--image <string></code>	The LEAN engine image to use (defaults to quantconnect/lean:latest)
<code>--update</code>	Pull the LEAN engine image before running the generator
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean data generate</code> command and exit

API Reference

lean decrypt

Introduction

Decrypt your local project using the specified decryption key.

```
$ lean decrypt [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean decrypt` command supports the following options:

Option	Description
<code>--key <file></code>	Path to the decryption key to use
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean decrypt</code> command and exit

API Reference

lean delete-project

Introduction

Alias for 'project-delete'

```
$ lean delete-project [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 delete-project` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean delete-project</code> command and exit

API Reference

lean encrypt

Introduction

Encrypt your local project using the specified encryption key.

```
$ lean encrypt [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean encrypt` command supports the following options:

Option	Description
<code>--key <file></code>	Path to the encryption key to use
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean encrypt</code> command and exit

API Reference

lean init

Introduction

Scaffold a Lean configuration file and data directory.

```
$ lean init [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--organization <string></code>	The name or id of the organization the Lean CLI will be scaffolded for
<code>-l, --language <enum: python csharp></code>	The default language to use for new projects
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean init</code> command and exit

API Reference

lean library add

Introduction

Add a custom library to a project.

```
$ lean library add [OPTIONS] PROJECT NAME
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 3.11 (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 3.11 cannot be installed in the official Docker images.

If a project is [encrypted](#) , you can only add a project library to it if the project library is unencrypted or is encrypted with the same key as the project. When you add a project library to a 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
<code>--version <string></code>	The version of the library to add (defaults to latest compatible version)
<code>--no-local</code>	Skip making changes to your local environment
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean library add</code> command and exit

API Reference

lean library remove

Introduction

Remove a custom library from a project.

```
$ lean library remove [OPTIONS] PROJECT NAME
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--no-local</code>	Skip making changes to your local environment
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean library remove</code> command and exit

API Reference

lean live

Introduction

Interact with the local machine.

```
$ lean live [OPTIONS] COMMAND [ARGS]...
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean live` command supports the following options:

Option	Description
<code>--help</code>	Display the help text of the <code>lean live</code> command and exit

Commands

The `lean live` command expects the following arguments:

Argument	Description
<code>add-security</code>	Represents a command to add a security to the algorithm
<code>cancel-order</code>	Represents a command to cancel a specific order by id
<code>command</code>	Send a command to a local running live trading project
<code>deploy</code>	Start live trading a project locally using Docker
<code>liquidate</code>	Liquidate the given symbol from the latest deployment of the given project
<code>stop</code>	Stop an already running local live trading project
<code>submit-order</code>	Represents a command to submit an order to the algorithm
<code>update-order</code>	Represents a command to update a specific order by id

API Reference

lean live add-security

Introduction

Represents a command to add a security to the algorithm.

```
$ lean live add-security [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--ticker <string></code>	[required]
<code>--market <string></code>	[required]
<code>--security-type <string></code>	[required]
<code>--resolution <string></code>	The resolution of the symbol to add
<code>--fill-data-forward</code>	The fill forward behavior, true to fill forward, false otherwise - defaults to true
<code>--leverage <float></code>	The leverage for the security, defaults to 2 for equity, 50 for forex, and 1 for everything else
<code>--extended-market-hours</code>	The extended market hours flag, true to allow pre/post market data, false for only in market data
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live add-security</code> command and exit

API Reference

lean live cancel-order

Introduction

Represents a command to cancel a specific order by id.

```
$ lean live cancel-order [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Arguments

The `lean live cancel-order` command expects the following arguments:

Argument	Description
<code><project></code>	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
<code>--order-id <integer></code>	[required]
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live cancel-order</code> command and exit

API Reference

lean live command

Introduction

Send a command to a local running live trading project.

```
$ lean live command [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean live command` command supports the following options:

Option	Description
<code>--data <string></code>	The command to send, 'str' representation of a 'dict' e.g. <code>{"target\":"BTCUSD\","\$type\":"MyCommand\}"</code>
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live command</code> command and exit

API Reference

lean live deploy

Introduction

Start live trading a project locally using Docker.

```
$ lean live deploy [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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 provider to use. When you provide `--environment` or both `--brokerage` and `--data-provider-live`, 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` and `--data-provider-live` 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 provider.

The following options are required for each brokerage in non-interactive mode:

<code>--brokerage</code>	Required Options
<code>"Paper Trading"</code>	N/A
	<code>--alpaca-environment</code>

Alpaca	--alpaca-api-key
	--alpaca-api-secret
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
Bybit	--bybit-api-key
	--bybit-api-secret
	--bybit-vip-level
	--bybit-use-testnet
"Coinbase Advanced Trade"	--coinbase-api-name
	--coinbase-api-private-key
"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-client-id

Samco	--samco-client-password
	--samco-year-of-birth
	--samco-product-type
	--samco-trading-segment
TD Ameritrade	--tdameritrade-api-key
	--tdameritrade-access-token
	--tdameritrade-account-number
"Terminal Link"	--terminal-link-connection-type
	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-account
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
	--terminal-link-server-auth-id if you use --terminal-link-connection-type SAPI
TradeStation	--trade-station-environment
	--trade-station-account-type
Tradier	--tradier-account-id
	--tradier-access-token
	--tradier-environment
	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key

"Trading Technologies"	--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
Zerodha	--zerodha-api-key
	--zerodha-access-token
	--zerodha-product-type
	--zerodha-trading-segment

The `--data-provider-live` option is required. The following table shows the available live data providers and their required options in non-interactive mode. To select multiple data providers, separate them with a comma. The order you select them in defines the order of precedence.

<code>--data-provider-live</code>	Required Options
Alpaca	All options required by <code>--brokerage Alpaca</code> .
Binance	--binance-exchange-name
	--binance-api-key or --binanceus-api-key
	--binance-api-secret or --binanceus-api-secret
Bitfinex	All options required by <code>--brokerage Bitfinex</code> .
Bybit	--bybit-api-key
	--bybit-api-secret

	--bybit-vip-level
CoinApi	--coinapi-api-key
	--coinapi-product
"Coinbase Advanced Trade"	All options required by --brokerage "Coinbase Advanced Trade" .
"Custom data only"	N/A
IEX	--iex-cloud-api-key
	--iex-price-plan
"Interactive Brokers"	All options required by --brokerage "Interactive Brokers" .
	--ib-enable-delayed-streaming-data
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Kraken	All options required by --brokerage Kraken .
Oanda	All options required by --brokerage Oanda .
Polygon	--polygon-api-key
Samco	All options required by --brokerage Samco .
TD Ameritrade	All options required by --brokerage TD Ameritrade .
"Terminal Link"	All options required by --brokerage "Terminal Link" .
ThetaData	--thetadata-subscription-plan
TradeStation	All options required by --brokerage TradeStation .
Tradier	--tradier-account-id
	--tradier-access-token

"Trading Technologies"	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key
	--tt-rest-app-secret
	--tt-rest-environment
	--tt-order-routing-sender-comp-id
Zerodha	All options required by --brokerage Zerodha .
	--zerodha-history-subscription

The `--data-provider-historical` option specifies the source of historical data. The following table shows the available historical data providers and their required options in non-interactive mode. If the live data provider you set also provides historical data and you omit the `--data-provider-historical` option, it defaults to the same value as the `--data-provider-live` option. If the live data provider you set doesn't provide historical data and you omit the `--data-provider-historical` option, it defaults to the `Local` data provider.

<code>--data-provider-historical</code>	Required Options
Alpaca	All options required by --brokerage Alpaca .
AlphaVantage	--alpha-vantage-api-key
	--alpha-vantage-price-plan
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 .
Bybit	--bybit-api-key
	--bybit-api-secret
	--bybit-vip-level

CoinApi	--coinapi-api-key
	--coinapi-product
"Coinbase Advanced Trade"	All options required by --brokerage "Coinbase Advanced Trade" .
FactSet	--factset-auth-config-file
IEX	--iex-cloud-api-key
	--iex-price-plan
"Interactive Brokers"	All options required by --brokerage "Interactive Brokers" .
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Kraken	All options required by --brokerage Kraken .
Local	N/A
Oanda	All options required by --brokerage Oanda .
Polygon	--polygon-api-key
QuantConnect	N/A
ThetaData	--thetadata-subscription-plan
TradeStation	All options required by --brokerage TradeStation .
Zerodha	All options required by --brokerage Zerodha .
	--zerodha-history-subscription

If you omit some of the required brokerage or data provider options when running in non-interactive mode, the CLI uses the option values in your [LEAN configuration file](#) .

Example non-interactive usage:

```
$ lean live deploy "My Project" \  
  --brokerage "Paper Trading" \  
  --data-provider-live "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 deploy` command expects the following arguments:

Argument	Description
<code><project></code>	The path to the project directory or algorithm file to start local live trading.

Options

The `lean live deploy` command supports the following options:

Option	Description
<code>--environment <string></code>	The environment to use
<code>--output <directory></code>	Directory to store results in (defaults to PROJECT/live/TIMESTAMP)
<code>-d, --detach</code>	Run the live deployment in a detached Docker container and return immediately

<pre>--brokerage <enum: Paper Trading Interactive Brokers Tradier Oanda Bitfinex Coinbase Advanced Trade Binance Zerodha Samco Terminal Link Trading Technologies Kraken CharlesSchwab Bybit TradeStation Alpaca Tastytrade Eze></pre>	<p>The brokerage to use</p>
--	-----------------------------

<pre>--data-provider-live <enum: Interactive Brokers Tradier Oanda Bitfinex Coinbase Advanced Trade Binance Zerodha Samco Terminal Link Trading Technologies Kraken CharlesSchwab IQFeed Polygon CoinApi ThetaData Custom data only Bybit TradeStation Alpaca Tastytrade Eze></pre>	<p>The live data provider to use</p>
---	--------------------------------------

<pre>--data-provider-historical <enum: Interactive Brokers Oanda Bitfinex Coinbase Advanced Trade Binance Kraken CharlesSchwab IQFeed Polygon FactSet AlphaVantage CoinApi ThetaData QuantConnect Local Bybit TradeStation Alpaca Tastytrade Eze></pre>	Update the Lean configuration file to retrieve data from the given historical provider
<pre>--ib-user-name <string></pre>	Your Interactive Brokers username
<pre>--ib-account <string></pre>	Your Interactive Brokers account id
<pre>--ib-password <string></pre>	Your Interactive Brokers password
<pre>--ib-weekly-restart-utc-time <string></pre>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)
<pre>--ib-financial-advisors-group-filter <string></pre>	Use this field to filter data by a specific financial advisor group within your Interactive Brokers account. Leave blank to include all groups. (Optional)
<pre>--tradier-account-id <string></pre>	Your Tradier account id
<pre>--tradier-access-token <string></pre>	Your Tradier access token
<pre>--tradier-environment <enum: live paper></pre>	Whether the developer sandbox should be used
<pre>--oanda-account-id <string></pre>	Your OANDA account id
<pre>--oanda-access-token <string></pre>	Your OANDA API token
<pre>--oanda-environment <enum: Practice Trade></pre>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<pre>--bitfinex-api-key <string></pre>	Your Bitfinex API key
<pre>--bitfinex-api-secret <string></pre>	Your Bitfinex API secret

<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file
<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binance exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--binance-use-testnet <enum: live paper></code>	Whether the testnet should be used
<code>--zerodha-api-key <string></code>	Your Kite Connect API key
<code>--zerodha-access-token <string></code>	Your Kite Connect access token
<code>--zerodha-product-type <enum: mis cnc nrml></code>	MIS if you are targeting intraday products, CNC if you are targeting delivery products, NRML if you are targeting carry forward products
<code>--zerodha-trading-segment <enum: equity commodity></code>	EQUITY if you are trading equities on NSE or BSE, COMMODITY if you are trading commodities on MCX
<code>--zerodha-history-subscription <enum: true false></code>	Whether you have a history API subscription for Zerodha
<code>--samco-client-id <string></code>	Your Samco account Client ID
<code>--samco-client-password <string></code>	Your Samco account password
<code>--samco-year-of-birth <string></code>	Your year of birth (YYYY) registered with Samco
<code>--samco-product-type <enum: mis cnc nrml></code>	MIS if you are targeting intraday products, CNC if you are targeting delivery products, NRML if you are targeting carry forward products
<code>--samco-trading-segment <enum: equity commodity></code>	EQUITY if you are trading equities on NSE or BSE, COMMODITY if you are trading commodities on MCX
<code>--terminal-link-connection-type <enum: DAPI SAPI></code>	Terminal Link Connection Type [DAPI, SAPI]
<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in

<code>--terminal-link-server-host <string></code>	The host of the TerminalLink server
<code>--terminal-link-server-port <integer></code>	The port of the TerminalLink server
<code>--terminal-link-emsx-broker <string></code>	The EMSX broker to use (Optional)
<code>--terminal-link-emsx-account <string></code>	The EMSX account to use (Optional)
<code>--terminal-link-emsx-team <string></code>	The EMSX team to receive order events from (Optional)
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--tt-user-name <string></code>	Your Trading Technologies username
<code>--tt-session-password <string></code>	Your Trading Technologies session password
<code>--tt-account-name <string></code>	Your Trading Technologies account name
<code>--tt-rest-app-key <string></code>	Your Trading Technologies REST app key
<code>--tt-rest-app-secret <string></code>	Your Trading Technologies REST app secret
<code>--tt-rest-environment <string></code>	The REST environment to run in
<code>--tt-order-routing-sender-comp-id <string></code>	The order routing sender comp id to use
<code>--tt-market-data-sender-comp-id <string></code>	The market data sender comp id to use
<code>--tt-market-data-target-comp-id <string></code>	The market data target comp id to use
<code>--tt-market-data-host <string></code>	The host of the market data server
<code>--tt-market-data-port <string></code>	The port of the market data server
<code>--tt-order-routing-target-comp-id <string></code>	The order routing target comp id to use
<code>--tt-order-routing-host <string></code>	The host of the order routing server
<code>--tt-order-routing-port <string></code>	The port of the order routing server
<code>--tt-log-fix-messages <boolean></code>	Whether FIX messages should be logged (Optional)
<code>--kraken-api-key <string></code>	Your Kraken API key
<code>--kraken-api-secret <string></code>	Your Kraken API secret

<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--bybit-api-key <string></code>	Your Bybit API key
<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--bybit-vip-level <enum: VIP0 VIP1 VIP2 VIP3 VIP4 VIP5 SupremeVIP Pro1 Pro2 Pro3 Pro4 Pro5></code>	Your Bybit VIP Level
<code>--bybit-use-testnet <enum: live paper></code>	Whether the testnet should be used
<code>--trade-station-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number
<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password
<code>--eze-trading-account <string></code>	Format: BANK;BRANCH;CUSTOMER;DEPOSIT
<code>--ib-enable-delayed-streaming-data <boolean></code>	Whether delayed data may be used when your algorithm subscribes to a security you don't have a market data subscription for (Optional)
<code>--charles-schwab-enable-delayed-streaming-data <boolean></code>	Whether delayed data may be used when your algorithm subscribes to a security you don't have a market data subscription for (Optional)

<code>--iqfeed-iqconnect <string></code>	The path to the IQConnect binary
<code>--iqfeed-username <string></code>	Your IQFeed username
<code>--iqfeed-password <string></code>	Your IQFeed password
<code>--iqfeed-version <string></code>	The product version of your IQFeed developer account
<code>--iqfeed-host <string></code>	The IQFeed host address (Optional)
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key
<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--thetadata-ws-url <string></code>	The ThetaData host address (Optional)
<code>--thetadata-rest-url <string></code>	The ThetaData host address (Optional)
<code>--thetadata-subscription-plan <enum: Free Value Standard Pro></code>	Your ThetaData subscription price plan
<code>--trade-station-enable-delayed-streaming-data <boolean></code>	Whether delayed data may be used when your algorithm subscribes to a security you don't have a market data subscription for (Optional)
<code>--alpaca-api-key <string></code>	Your Alpaca Api Key
<code>--alpaca-api-secret <string></code>	Your Alpaca Api Secret
<code>--factset-auth-config-file <file></code>	The path to the FactSet authentication configuration file
<code>--alpha-vantage-api-key <string></code>	Your Alpha Vantage Api Key

<pre>--alpha-vantage-price-plan <enum: Free Plan30 Plan75 Plan150 Plan300 Plan600 Plan1200></pre>	<p>Your Alpha Vantage Premium API Key plan</p>
<pre>--release</pre>	<p>Compile C# projects in release configuration instead of debug</p>
<pre>--image <string></pre>	<p>The LEAN engine image to use (defaults to quantconnect/lean:latest)</p>
<pre>--python-venv <string></pre>	<p>The path of the python virtual environment to be used</p>
<pre>--live-cash-balance <string></pre>	<p>A comma-separated list of currency:amount pairs of initial cash balance</p>
<pre>--live-holdings <string></pre>	<p>A comma-separated list of symbol:symbolId:quantity:averagePrice of initial portfolio holdings</p>
<pre>--update</pre>	<p>Pull the LEAN engine image before starting live trading</p>
<pre>--show-secrets</pre>	<p>Show secrets as they are input</p>
<pre>--extra-docker-config <string></pre>	<p>Extra docker configuration as a JSON string. For more information https://docker-py.readthedocs.io/en/stable/containers.html</p>
<pre>--no-update</pre>	<p>Use the local LEAN engine image instead of pulling the latest version</p>
<pre>--lean-config <file></pre>	<p>The Lean configuration file that should be used (defaults to the nearest lean.json)</p>
<pre>--verbose</pre>	<p>Enable debug logging</p>
<pre>--help</pre>	<p>Display the help text of the <code>lean live deploy</code> command and exit</p>

API Reference

lean live liquidate

Introduction

Liquidate the given symbol from the latest deployment of the given project.

```
$ lean live liquidate [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Arguments

The `lean live liquidate` command expects the following arguments:

Argument	Description
<code><project></code>	The path to the project directory or algorithm file to liquidate.

Options

The `lean live liquidate` command supports the following options:

Option	Description
<code>--ticker <string></code>	The ticker of the symbol to liquidate
<code>--market <string></code>	The market of the symbol to liquidate
<code>--security-type <string></code>	The security type of the symbol to liquidate
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live liquidate</code> command and exit

API Reference

lean live stop

Introduction

Stop an already running local live trading project.

```
$ lean live stop [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Arguments

The `lean live stop` command expects the following arguments:

Argument	Description
<code><project></code>	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
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live stop</code> command and exit

API Reference

lean live submit-order

Introduction

Represents a command to submit an order to the algorithm.

```
$ lean live submit-order [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--ticker <string></code>	[required]
<code>--market <string></code>	[required]
<code>--security-type <string></code>	[required]
<code>--order-type <string></code>	[required]
<code>--quantity <float></code>	[required]
<code>--limit-price <float></code>	The limit price of the order be submitted
<code>--stop-price <float></code>	The stop price of the order to be submitted
<code>--tag <string></code>	The tag to be attached to the order
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live submit-order</code> command and exit

API Reference

lean live update-order

Introduction

Represents a command to update a specific order by id.

```
$ lean live update-order [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Description

Arguments

The `lean live update-order` command expects the following arguments:

Argument	Description
<code><project></code>	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
<code>--order-id <integer></code>	[required]
<code>--quantity <float></code>	The number of units to be updated (directional)
<code>--limit-price <float></code>	The limit price of the order to be updated
<code>--stop-price <float></code>	The stop price of the order to be updated
<code>--tag <string></code>	The tag to be attached to the order
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean live update-order</code> command and exit

API Reference

lean login

Introduction

Log in with a QuantConnect account.

```
$ lean login [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>-u, --user-id <string></code>	QuantConnect user id
<code>-t, --api-token <string></code>	QuantConnect API token
<code>--show-secrets</code>	Show secrets as they are input
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean login</code> command and exit

API Reference

lean logout

Introduction

Log out and remove stored credentials.

```
$ lean logout [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean logout</code> command and exit

API Reference

lean logs

Introduction

Display the most recent backtest/live/optimization logs.

```
$ lean logs [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--backtest</code>	Display the most recent backtest logs (default)
<code>--live</code>	Display the most recent live logs
<code>--optimization</code>	Display the most recent optimization logs
<code>--project <directory></code>	The project to get the most recent logs from
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean logs</code> command and exit

API Reference

lean object-store delete

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store delete [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store delete` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store delete</code> command and exit

API Reference

lean object-store get

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store get [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store get` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store get</code> command and exit

API Reference

lean object-store list

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store list [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store list` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store list</code> command and exit

API Reference

lean object-store ls

Introduction

Alias for 'list'

```
$ lean object-store ls [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store ls` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store ls</code> command and exit

API Reference

lean object-store properties

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store properties [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store properties` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store properties</code> command and exit

API Reference

lean object-store set

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store set [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean object-store set` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean object-store set</code> command and exit

API Reference

lean optimize

Introduction

Optimize a project's parameters locally using Docker.

```
$ lean optimize [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

You can use the `--data-provider-historical` 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. The following table shows the available data providers and their required options in non-interactive mode:

<code>--data-provider-historical</code>	Required Options
Alpaca	<code>--alpaca-environment</code>
AlphaVantage	<code>--alpha-vantage-api-key</code>
	<code>--alpha-vantage-price-plan</code>
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>
	<code>--binance-api-secret</code> or <code>--binanceus-api-secret</code>
Bybit	<code>--bybit-api-key</code>
	<code>--bybit-api-secret</code>
CoinApi	<code>--coinapi-api-key</code>
	<code>--coinapi-product</code>
"Coinbase Advanced Trade"	<code>--coinbase-api-name</code>
	<code>--coinbase-api-private-key</code>
FactSet	<code>--factset-auth-config-file</code>
IEX	<code>--iex-cloud-api-key</code>
	<code>--iex-price-plan</code>
"Interactive Brokers"	<code>--ib-user-name</code>
	<code>--ib-account</code>
	<code>--ib-password</code>

IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Kraken	--kraken-api-key
	--kraken-api-secret
	--kraken-verification-tier
Local	N/A
Oanda	--oanda-account-id
	--oanda-access-token
	--oanda-environment
Polygon	--polygon-api-key
QuantConnect	N/A
"Terminal Link"	--terminal-link-connection-type
	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
	--terminal-link-server-auth-id if you use --terminal-link-connection-type SAPI
ThetaData	--thetadata-subscription-plan
TradeStation	N/A

You can use the `--download-data` flag as an alias for `--data-provider-historical QuantConnect`. 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 won't have to download it again. If the file contain data for multiple days (for example, daily Equity price data files), the `ApiDataProvider` re-downloads the file if your local version is at least 7 days old. To adjust this setting, update the `downloader-data-update-period` value in your [Lean configuration](#) file.

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
<code><project></code>	The path to the project directory or algorithm file to optimize.

Options

The `lean optimize` command supports the following options:

Option	Description
<code>--output <directory></code>	Directory to store results in (defaults to PROJECT/optimizations/TIMESTAMP)
<code>-d, --detach</code>	Run the optimization in a detached Docker container and return immediately

<code>--optimizer-config <file></code>	The optimizer configuration file that should be used
<code>--strategy <enum: Grid Search Euler Search></code>	The optimization strategy to use
<code>--target <string></code>	The target statistic of the optimization
<code>--target-direction <enum: min max></code>	Whether the target must be minimized or maximized
<code>--parameter <<string> <float> <float> <float> >...</code>	The 'parameter min max step' pairs configuring the parameters to optimize
<code>--constraint <string></code>	The 'statistic operator value' pairs configuring the constraints of the optimization
<code>--data-provider-historical <enum: Interactive Brokers Oanda Bitfinex Coinbase Advanced Trade Binance Kraken CharlesSchwab IQFeed Polygon FactSet AlphaVantage CoinApi ThetaData QuantConnect Local Terminal Link Bybit TradeStation Alpaca Tastytrade Eze></code>	Update the Lean configuration file to retrieve data from the given historical provider
<code>--download-data</code>	Update the Lean configuration file to download data from the QuantConnect API, alias for <code>--data-provider-historical QuantConnect</code>
<code>--data-purchase-limit <integer></code>	The maximum amount of QCC to spend on downloading data during the backtest when using QuantConnect as historical data provider
<code>--release</code>	Compile C# projects in release configuration instead of debug
<code>--image <string></code>	The LEAN engine image to use (defaults to quantconnect/lean:latest)
<code>--update</code>	Pull the LEAN engine image before running the optimizer
<code>--estimate</code>	Estimate optimization runtime without running it

<code>--max-concurrent-backtests <integer> <range></code>	[x>=1]
<code>--extra-docker-config <string></code>	Extra docker configuration as a JSON string. For more information https://docker-py.readthedocs.io/en/stable/containers.html
<code>--no-update</code>	Use the local LEAN engine image instead of pulling the latest version
<code>--ib-user-name <string></code>	Your Interactive Brokers username
<code>--ib-account <string></code>	Your Interactive Brokers account id
<code>--ib-password <string></code>	Your Interactive Brokers password
<code>--ib-weekly-restart-utc-time <string></code>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)
<code>--oanda-account-id <string></code>	Your OANDA account id
<code>--oanda-access-token <string></code>	Your OANDA API token
<code>--oanda-environment <enum: Practice Trade></code>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<code>--bitfinex-api-key <string></code>	Your Bitfinex API key
<code>--bitfinex-api-secret <string></code>	Your Bitfinex API secret
<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file
<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binance exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--kraken-api-key <string></code>	Your Kraken API key

<code>--kraken-api-secret <string></code>	Your Kraken API secret
<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--iqfeed-iqconnect <string></code>	The path to the IQConnect binary
<code>--iqfeed-username <string></code>	Your IQFeed username
<code>--iqfeed-password <string></code>	Your IQFeed password
<code>--iqfeed-version <string></code>	The product version of your IQFeed developer account
<code>--iqfeed-host <string></code>	The IQFeed host address (Optional)
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--factset-auth-config-file <file></code>	The path to the FactSet authentication configuration file
<code>--alpha-vantage-api-key <string></code>	Your Alpha Vantage Api Key
<code>--alpha-vantage-price-plan <enum: FreePlan30 Plan75 Plan150 Plan300 Plan600 Plan1200></code>	Your Alpha Vantage Premium API Key plan
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key
<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--thetadata-ws-url <string></code>	The ThetaData host address (Optional)
<code>--thetadata-rest-url <string></code>	The ThetaData host address (Optional)
<code>--thetadata-subscription-plan <enum: Free Value Standard Pro></code>	Your ThetaData subscription price plan
<code>--terminal-link-connection-type <enum: DAPI SAPI></code>	Terminal Link Connection Type [DAPI, SAPI]

<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in
<code>--terminal-link-server-host <string></code>	The host of the TerminalLink server
<code>--terminal-link-server-port <integer></code>	The port of the TerminalLink server
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--bybit-api-key <string></code>	Your Bybit API key
<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--trade-station-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number
<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean optimize</code> command and exit

API Reference

lean private-cloud add-compute

Introduction

Add private cloud compute

```
$ lean private-cloud add-compute [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean private-cloud add-compute` command supports the following options:

Option	Description
<code>--token <string></code>	The master server token
<code>--master-domain, --master-ip <string></code>	The master server domain
<code>--master-port <integer></code>	The master server port
<code>--slave-domain, --slave-ip <string></code>	The slave server domain
<code>--update</code>	Pull the latest image before starting
<code>--no-update</code>	Do not update to the latest version
<code>--compute <string></code>	Compute configuration to use
<code>--extra-docker-config <string></code>	Extra docker configuration as a JSON string
<code>--stop</code>	Stop any existing deployment
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean private-cloud add-compute</code> command and exit

API Reference

lean private-cloud start

Introduction

Start a new private cloud

```
$ lean private-cloud start [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean private-cloud start` command supports the following options:

Option	Description
<code>--master</code>	Run in master mode
<code>--slave</code>	Run in slave mode
<code>--token <string></code>	The master server token
<code>--master-domain, --master-ip <string></code>	The master server domain
<code>--master-port <integer></code>	The master server port
<code>--slave-domain, --slave-ip <string></code>	The slave server domain
<code>--update</code>	Pull the latest image before starting
<code>--no-update</code>	Do not update to the latest version
<code>--compute <string></code>	Compute configuration to use
<code>--extra-docker-config <string></code>	Extra docker configuration as a JSON string
<code>--stop</code>	Stop any existing deployment
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean private-cloud start</code> command and exit

API Reference

lean private-cloud stop

Introduction

Stops a running private cloud

```
$ lean private-cloud stop [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

Options

The `lean private-cloud stop` command supports the following options:

Option	Description
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean private-cloud stop</code> command and exit

API Reference

lean project-create

Introduction

Create a new project containing starter code.

```
$ lean project-create [OPTIONS] NAME
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
-l, --language <enum: python csharp>	The language of the project to create
--verbose	Enable debug logging
--help	Display the help text of the <code>lean project-create</code> command and exit

API Reference

lean project-delete

Introduction

Delete a project locally and in the cloud if it exists.

```
$ lean project-delete [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean project-delete</code> command and exit

API Reference

lean report

Introduction

Generate a report of a backtest.

```
$ lean report [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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.

To view the content of the default reports, see [Reports](#). To create custom reports, see [Generate Reports](#).

Options

The `lean report` command supports the following options:

Option	Description
<code>--backtest-results <file></code>	Path to the JSON file containing the backtest results
<code>--live-results <file></code>	Path to the JSON file containing the live trading results
<code>--report-destination <file></code>	Path where the generated report is stored as HTML (defaults to ./report.html)
<code>--css <file></code>	Path where the CSS override file is stored
<code>--html <file></code>	Path where the custom HTML template file is stored
<code>-d, --detach</code>	Run the report creator in a detached Docker container and return immediately
<code>--strategy-name <string></code>	Name of the strategy, will appear at the top-right corner of each page
<code>--strategy-version <string></code>	Version number of the strategy, will appear next to the project name
<code>--strategy-description <string></code>	Description of the strategy, will appear under the 'Strategy Description' section
<code>--overwrite</code>	Overwrite --report-destination if it already contains a file
<code>--image <string></code>	The LEAN engine image to use (defaults to quantconnect/lean:latest)
<code>--update</code>	Pull the LEAN engine image before running the report creator
<code>--pdf</code>	Create a PDF version along with the HTML version of the report
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest lean.json)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean report</code> command and exit

API Reference

lean research

Introduction

Run a Jupyter Lab environment locally using Docker.

```
$ lean research [OPTIONS] PROJECT
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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-historical` 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. The following table shows the available data providers and their required options in non-interactive mode:

<code>--data-provider-historical</code>	Required Options
Alpaca	<code>--alpaca-environment</code>
AlphaVantage	<code>--alpha-vantage-api-key</code>
	<code>--alpha-vantage-price-plan</code>
Binance	<code>--binance-exchange-name</code>
	<code>--binance-api-key</code> or <code>--binanceus-api-key</code>
	<code>--binance-api-secret</code> or <code>--binanceus-api-secret</code>
Bybit	<code>--bybit-api-key</code>
	<code>--bybit-api-secret</code>

CoinApi	--coinapi-api-key
	--coinapi-product
"Coinbase Advanced Trade"	--coinbase-api-name
	--coinbase-api-private-key
FactSet	--factset-auth-config-file
IEX	--iex-cloud-api-key
	--iex-price-plan
"Interactive Brokers"	--ib-user-name
	--ib-account
	--ib-password
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Kraken	--kraken-api-key
	--kraken-api-secret
	--kraken-verification-tier
Local	N/A
Oanda	--oanda-account-id
	--oanda-access-token
	--oanda-environment
Polygon	--polygon-api-key
QuantConnect	N/A
	--terminal-link-connection-type

"Terminal Link"	--terminal-link-environment
	--terminal-link-server-host
	--terminal-link-server-port
	--terminal-link-emsx-broker
	--terminal-link-openfigi-api-key
	--terminal-link-server-auth-id if you use --terminal-link-connection-type SAPI
ThetaData	--thetadatasubscription-plan
TradeStation	N/A

You can use the `--download-data` flag as an alias for `--data-provider-historical 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
--------	-------------

<code>--port <integer></code>	The port to run Jupyter Lab on (defaults to 8888)
<code>--data-provider-historical <enum: Interactive Brokers Oanda Bitfinex Coinbase Advanced Trade Binance Kraken CharlesSchwab IQFeed Polygon FactSet AlphaVantage CoinApi ThetaData QuantConnect Local Terminal Link Bybit TradeStation Alpaca Tastytrade Eze></code>	Update the Lean configuration file to retrieve data from the given historical provider
<code>--ib-user-name <string></code>	Your Interactive Brokers username
<code>--ib-account <string></code>	Your Interactive Brokers account id
<code>--ib-password <string></code>	Your Interactive Brokers password
<code>--ib-weekly-restart-utc-time <string></code>	Weekly restart UTC time (hh:mm:ss). Each week on Sunday your algorithm is restarted at this time, and will require 2FA verification. This is required by Interactive Brokers. Use this option explicitly to override the default value. (Optional)
<code>--oanda-account-id <string></code>	Your OANDA account id
<code>--oanda-access-token <string></code>	Your OANDA API token
<code>--oanda-environment <enum: Practice Trade></code>	The environment to run in, Practice for fxTrade Practice, Trade for fxTrade
<code>--bitfinex-api-key <string></code>	Your Bitfinex API key
<code>--bitfinex-api-secret <string></code>	Your Bitfinex API secret
<code>--coinbase-api-name <string></code>	Your Coinbase Advanced Trade API name from file
<code>--coinbase-api-private-key <string></code>	Your Coinbase Advanced Trade API private key from file

<code>--binance-exchange-name <enum: Binance BinanceUS Binance-USDM-Futures Binance-COIN-Futures></code>	Binace exchange name [Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures]
<code>--binance-api-key <string></code>	Your Binance API key
<code>--binanceus-api-key <string></code>	Your Binance API key
<code>--binance-api-secret <string></code>	Your Binance API secret
<code>--binanceus-api-secret <string></code>	Your Binance API secret
<code>--kraken-api-key <string></code>	Your Kraken API key
<code>--kraken-api-secret <string></code>	Your Kraken API secret
<code>--kraken-verification-tier <enum: Starter Intermediate Pro></code>	Your Kraken Verification Tier
<code>--charles-schwab-account-number <string></code>	The CharlesSchwab account number
<code>--iqfeed-iqconnect <string></code>	The path to the IQConnect binary
<code>--iqfeed-username <string></code>	Your IQFeed username
<code>--iqfeed-password <string></code>	Your IQFeed password
<code>--iqfeed-version <string></code>	The product version of your IQFeed developer account
<code>--iqfeed-host <string></code>	The IQFeed host address (Optional)
<code>--polygon-api-key <string></code>	Your Polygon.io API Key
<code>--factset-auth-config-file <file></code>	The path to the FactSet authentication configuration file
<code>--alpha-vantage-api-key <string></code>	Your Alpha Vantage Api Key
<code>--alpha-vantage-price-plan <enum: Free Plan30 Plan75 Plan150 Plan300 Plan600 Plan1200></code>	Your Alpha Vantage Premium API Key plan
<code>--coinapi-api-key <string></code>	Your coinapi.io Api Key

<code>--coinapi-product <enum: Free Startup Streamer Professional Enterprise></code>	CoinApi pricing plan (https://www.coinapi.io/market-data-api/pricing)
<code>--thetadate-ws-url <string></code>	The ThetaData host address (Optional)
<code>--thetadate-rest-url <string></code>	The ThetaData host address (Optional)
<code>--thetadate-subscription-plan <enum: Free Value Standard Pro></code>	Your ThetaData subscription price plan
<code>--terminal-link-connection-type <enum: DAPI S API></code>	Terminal Link Connection Type [DAPI, SAPI]
<code>--terminal-link-environment <enum: Production Beta></code>	The environment to run in
<code>--terminal-link-server-host <string></code>	The host of the TerminalLink server
<code>--terminal-link-server-port <integer></code>	The port of the TerminalLink server
<code>--terminal-link-openfigi-api-key <string></code>	The Open FIGI API key to use for mapping options
<code>--terminal-link-server-auth-id <string></code>	The Auth ID of the TerminalLink server
<code>--bybit-api-key <string></code>	Your Bybit API key
<code>--bybit-api-secret <string></code>	Your Bybit API secret
<code>--trade-station-environment <enum: live paper ></code>	Whether Live or Paper environment should be used
<code>--trade-station-account-id <string></code>	The TradeStation account Id
<code>--alpaca-environment <enum: live paper></code>	Whether Live or Paper environment should be used
<code>--tastytrade-account-number <string></code>	The Tastytrade account number
<code>--eze-domain <string></code>	Examples (broker provided): LIGHTSPEED, LIGHTSPEEDDELAY
<code>--eze-locale <string></code>	Examples (broker provided): GLOBAL, AMERICAS, LIVE, DELAYED
<code>--eze-user-name <string></code>	Your Eze Brokers username
<code>--eze-password <string></code>	Your Eze Brokers password

<code>--download-data</code>	Update the Lean configuration file to download data from the QuantConnect API, alias for <code>--data-provider-historical</code> QuantConnect
<code>--data-purchase-limit <integer></code>	The maximum amount of QCC to spend on downloading data during the research session when using QuantConnect as historical data provider
<code>-d, --detach</code>	Run Jupyter Lab in a detached Docker container and return immediately
<code>--no-open</code>	Don't open the Jupyter Lab environment in the browser after starting it
<code>--image <string></code>	The LEAN research image to use (defaults to <code>quantconnect/research:latest</code>)
<code>--update</code>	Pull the LEAN research image before starting the research environment
<code>--extra-docker-config <string></code>	Extra docker configuration as a JSON string. For more information https://docker-py.readthedocs.io/en/stable/containers.html
<code>--no-update</code>	Use the local LEAN research image instead of pulling the latest version
<code>--lean-config <file></code>	The Lean configuration file that should be used (defaults to the nearest <code>lean.json</code>)
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean research</code> command and exit

API Reference

lean whoami

Introduction

Display who is logged in.

```
$ lean whoami [OPTIONS]
```

To use the CLI, you must be a member in an [organization](#) on a paid tier.

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
<code>--verbose</code>	Enable debug logging
<code>--help</code>	Display the help text of the <code>lean whoami</code> command and exit

