QUANTCONNECT DOCUMENTATION

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

Key Concepts

Key Concepts

Getting Started

Introduction

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

Prerequisites

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.

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

Installation

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

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

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

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

Basic Usage

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

Authenticate With the Cloud

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

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

Pull Projects From the Cloud

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

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

Source Data

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

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

Run a Local Backtest

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

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

```
...
```

Push Local Changes to the Cloud

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

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

Run a Cloud Backtest

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

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

LEAN vs LEAN CLI

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

Key Concepts

Troubleshooting

Introduction

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

Common Errors

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

Error Message	Possible Cause and Fix
No such command '<name>' No such option: <option>	The command you tried to run does not exist or it doesn't support the option you provided. If the documentation says it is available you are probably using an outdated version of the CLI. Run `pip install --upgrade lean` to update to the latest version.
Invalid credentials, please log in using `lean login`	You are trying to use a command which communicates with the QuantConnect API and you haven't authenticated yourself yet. Run `lean login` to log in with your API credentials.
Please make sure Docker is installed and running	You are trying to use a command which needs to run the LEAN engine locally, which always happens in a Docker container. Make sure Docker is running if you installed it already, or visit Install Docker if you haven't.
	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 `--lean-config` option.
We couldn't find you account in the given organization, ORG: <32-char-hash>	The organization Id found in the **lean.json** is incorrect. You need to re-install Lean CLI running `lean init` in an empty directory.
Invalid value for 'PROJECT': Path '<path>' does not exist.	You are trying to run an action on a project but specified an invalid project path. Make sure you are running the command from your organization workspace directory and make sure `./<path>` points to an existing directory.
	You provided a path that is not valid for your operating system. This error is most likely to appear on Windows,

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

Report Issues

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

Installation

Installation

Installing pip

Introduction

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

Install on Windows

Follow these steps to install Anaconda on your computer:

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

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

Install on macOS (Intel)

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

1. Download the latest 64-bit Graphical Installer for macOS.
2. Click **Continue** on the Introduction, Read Me, and License pages.
3. Agree to the license by clicking **Agree** .
4. Click **Install** on the Installation Type page to install to start the installation.

5. After the installation has finished, click **Continue** on the PyCharm IDE page and **Close** on the Summary page to close the installer.

Install on macOS (Apple)

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

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

Install on Linux

Follow these steps to install Anaconda on your computer:

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

Installation

Installing Lean CLI

Introduction

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

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

Install Docker

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

Install on Windows

Windows systems must meet the following requirements to install Docker:

- A 64-bit processor
- 4 GB RAM or more
- Windows 10, version 1903 or higher (released May 2019)
- Hardware virtualization enabled in the BIOS
- 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 adminstrator privledges and run:

   ```
   $ wsl --update
   ```

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

Install on macOS

Mac systems must meet the following requirements to install Docker:

- Mac hardware from 2010 or newer with an Intel processor
- macOS 10.14 or newer (Mojave, Catalina, or Big Sur)
- 4 GB RAM or more
- 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 adminstrator for the following commands.
4. Install the LEAN CLI with pip.

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

Next Steps

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

Stay Up To Date

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

Keep the CLI Up-To-Date

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

Keep the Docker Images Up-To-Date

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

Uninstall

To uninstall the Lean CLI, run `pip uninstall lean` . To perform a full uninstallation, you must also delete configuration files that

the CLI generates, which you can find in the following directories:

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

Initialization

Initialization

Authentication

Introduction

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

Log In

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

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

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

Log Out

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

Check Account Status

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

Initialization

Organization Workspaces

Introduction

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

We recommend running all Lean CLI commands in 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.

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.

Global Configuration

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

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

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

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

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

Lean Configuration

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

The Lean configuration file is based on the Launcher / config.json file from the Lean GitHub repository. When you run `lean init` ,

the latest version of this file is downloaded and stored in your organization workspace. Before the file is stored, some properties are automatically removed because the CLI automatically sets them.

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

Name	Description	Default
`show-missing-data-logs`	Log missing data files. This is useful for debugging.	true
`maximum-warmup-history-days-look-back`	The maximum number of days of data the history provider will provide during warm-up in live trading. The history provider expects older data to be on disk.	5
`maximum-chart-series`	The maximum number of chart series you can create in backtests.	30
`maximum-data-points-per-chart-series`	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

Introduction

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

Default Location

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

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

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

Change Location

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

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

Other Data Sources

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

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

Datasets

Generating Data

Introduction

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

Supported Security Types

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

Security Type	Supported Resolutions				
	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

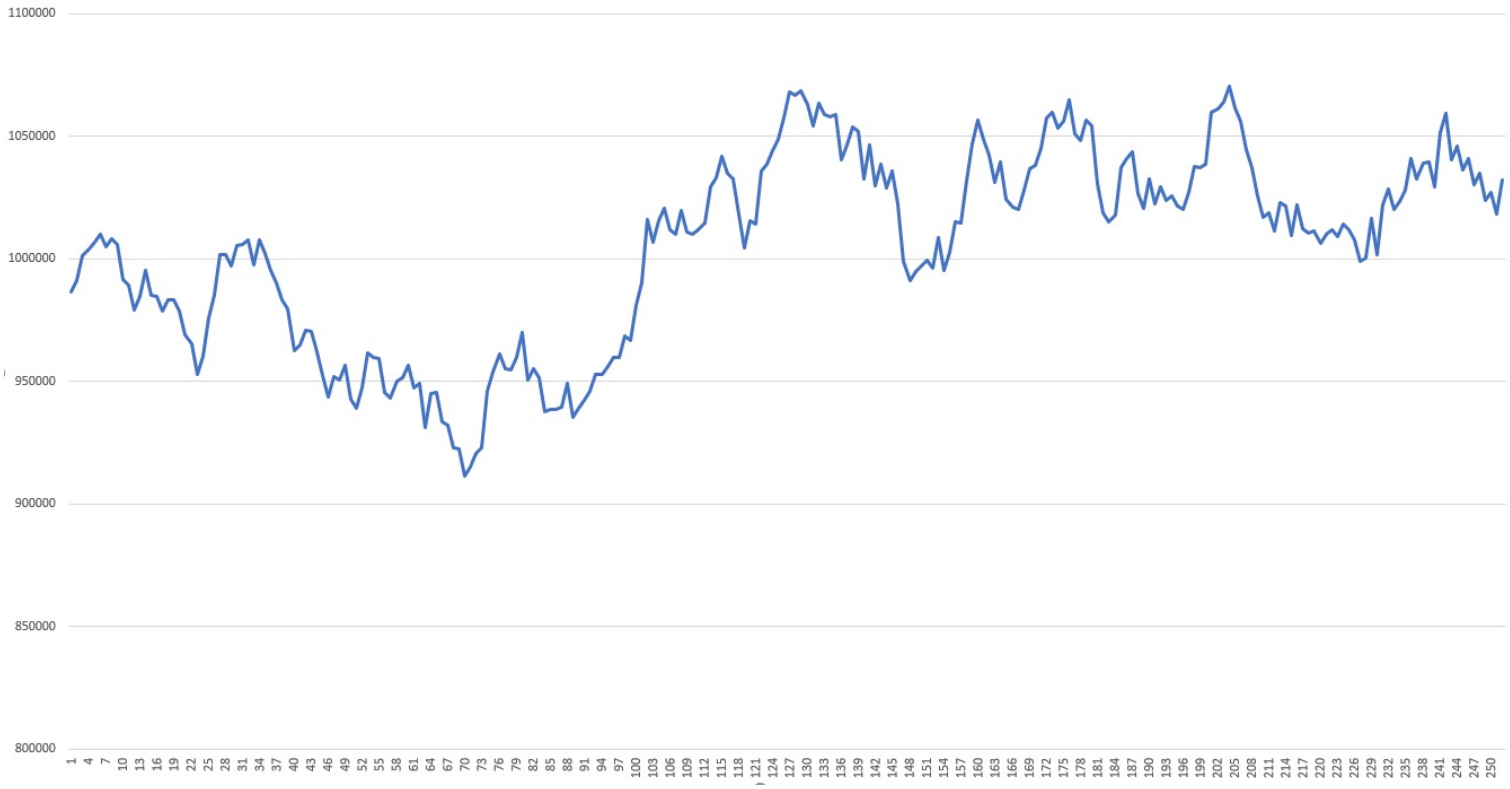
for 10 random symbols.

```
$ 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.

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 `BaseData`) to retrieve data from a local file instead of a remote one:

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

```C#
using System;
using System.IO;
using QuantConnect.Data;

namespace QuantConnect.Algorithm.CSharp
{
    public class Weather : BaseData
    {
        public override SubscriptionDataSource GetSource(SubscriptionDataConfig config, DateTime date, bool
isLive)
        {
            // Old:
            // var source = "https://www.dropbox.com/s/8v6z949n25hyk9o/custom_weather_data.csv?dl=1";
            // return new SubscriptionDataSource(source, SubscriptionTransportMedium.RemoteFile);

            // New:
            // Replace custom_weather_data.csv with the path to your data file in the data directory
            var source = Path.Combine(Globals.DataFolder, "custom_weather_data.csv");
            return new SubscriptionDataSource(source, SubscriptionTransportMedium.LocalFile);
        }
    }
}
```

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.

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         │ 32         │ 800 QCC  │
│                          │          │ Ticker: BTCBUSD          │            │          │
│                          │          │ Resolution: Second       │            │          │
│                          │          │ Start date: 2022-05-05   │            │          │
│                          │          │ End date: 2022-06-05     │            │          │
└──────────────────────────┴──────────┴──────────────────────────┴────────────┴──────────┘
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
```

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

10. Confirm your data purchase.

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

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

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

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

Using Lean

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

Supported Datasets

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

Download By Ticker

Costs

Introduction

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

US Equity

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

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

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

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

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

Universe Type	Required Dataset	File Format	Cost per file
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:

```
C#
namespace QuantConnect.Algorithm.CSharp
{
    public class USEquityDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            UniverseSettings.Asynchronous = true;
            AddUniverse(Universe.DollarVolume.Top(100));
        }
    }
}
```

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

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

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

Equity Options

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

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

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

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

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

C#

```
namespace QuantConnect.Algorithm.CSharp
{
    public class USEquityOptionsDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            UniverseSettings.Asynchronous = true;
            var underlying = AddEquity("GOOG").Symbol;
            AddOption(underlying);
        }
    }
}
```

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

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

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

Crypto

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

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

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

For example, the following algorithm creates a universe of 100 Cryptocurrencies and then subscribes to minute resolution data

for each one in the universe:

```C#
namespace QuantConnect.Algorithm.CSharp
{
    public class CryptoDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            UniverseSettings.Asynchronous = true;
            AddUniverse(new CryptoCoarseFundamentalUniverse(Market.Coinbase, UniverseSettings,
                cryptoCoarse => cryptoCoarse.OrderByDescending(cf => cf.VolumeInUsd).Take(100).Select(x => x.Symbol))
            );
        }
    }
}
```

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

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

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

Forex

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

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

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

```C#
namespace QuantConnect.Algorithm.CSharp
{
    public class ForexDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            AddForex("USDCAD");
        }
    }
}
```

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

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

Futures

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

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

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

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

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

```
C#
namespace QuantConnect.Algorithm.CSharp
{
    public class USFuturesDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            var future = AddFuture(Futures.Indices.SP500EMini,
                dataNormalizationMode: DataNormalizationMode.BackwardsRatio,
                dataMappingMode: DataMappingMode.OpenInterest,
                contractDepthOffset: 0
            );
            future.SetFilter(0, 90);
        }
    }
}
```

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

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

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

Index Options

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

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

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

```C#
namespace QuantConnect.Algorithm.CSharp
{
    public class USIndexOptionsDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            AddIndexOption("VIX");
        }
    }
}
```

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

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

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

CFD

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

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

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

```
C#
namespace QuantConnect.Algorithm.CSharp
{
    public class CFDDataAlgorithm : QCAlgorithm
    {
        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetEndDate(2021, 1, 1);
            AddCfd("XAUUSD");
        }
    }
}
```

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

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

Alternative Data

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

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 ETF Constituents

US Futures

US Index Options

See Also

Datasets

Download in Bulk

CFD Data

Introduction

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

Download History

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

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

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

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

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

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

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

Download Daily Updates

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

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

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

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

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

Size and Format

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

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

Price

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

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

Download in Bulk

FOREX Data

Introduction

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

Download History

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

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

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

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

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

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

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

Download Daily Updates

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

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

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

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

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

Size and Format

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

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

Price

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

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

Download in Bulk

US Equities

Introduction

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

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

Download History

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

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

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

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

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

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

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

To download the US Equity Security Master, run:

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

Download Daily Updates

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

- US Equity Daily Updates by AlgoSeek

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

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

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

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

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

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

Size and Format

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

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

Price

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

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

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

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

Download in Bulk

US Equity Coarse Universe

Introduction

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

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

Download History

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

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

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

To download the US Equity Security Master, run:

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

Download Daily Updates

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

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

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

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

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

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

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

Size and Format

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

Price

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

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

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

Download in Bulk

US Equity Options

Introduction

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

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

Download History

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

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

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

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

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

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

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

To download the US Equity Security Master, run:

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

Download Daily Updates

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

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

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

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

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

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

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

Size and Format

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

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

Price

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

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

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

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

Download in Bulk

US ETF Constituents

Introduction

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

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

Download History

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

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

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

To download the US Equity Security Master, run:

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

Download Daily Updates

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

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

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

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

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

Alternatively, instead of directly calling the `lean data download` command, you can place a Python script in the **data** directory of

your organization workspace and run it to update your data files. The following example script updates all of the new data that's missing from your local copy:

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

Size and Format

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

Price

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

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

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

Download in Bulk

US Futures

Introduction

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

Download History

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

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

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

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

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

   The **Ticker** , **Start Date** , and **End Date** fields are irrelevant for bulk downloads.
3. Open a terminal in your organization workspace and then run the command from the **CLI Command Generator** .

Download Daily Updates

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

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

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

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

and run it to update your data files. The following example script updates all data resolutions and markets:

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

Size and Format

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

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

Price

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

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

Download in Bulk

US Index Options

Introduction

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

Download History

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

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

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

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

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

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

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

Download Daily Updates

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

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

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

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

The preceding script checks the date of the most recent minute resolution data you have for SPX. If there is new minute data

available, it downloads the new data files and overwrites your hourly and daily files. If you don't intend to download all resolutions, adjust this script to your needs.

Size and Format

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

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

Price

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

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

Datasets

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. 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 .

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.

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.

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>` . The paper trading brokerage 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.

Supported Assets

Our Alpha Vantage integration supports US Equity securities.

Datasets

IEX Cloud

Introduction

Instead of using the data from QuantConnect or your brokerage, you can use IEX Cloud if you're deploying a local project. To use IEX Cloud, you need an account subscribed to one or more data bundles . You also must select either the Launch, Grow, or Enterprise subscription tier; this dictates your maximum requests per second. If you don't have an account, register today and get your API key . Pricing information can be found here .

Upon registration, your account will automatically start a 7-day free trial. This trial has access to all API endpoints and a rate limit of 5 requests per second. If you have any questions regarding IEX Cloud's API service, please contact them here: support@iexcloud.io .

To view the implementation of the IEX Cloud integration, see the Lean.DataSource.IEX repository .

Backtesting

To run a local backtest with IEX Cloud data, open a terminal in your organization workspace and then run `lean backtest <projectName> --data-provider-historical IEX --iex-cloud-api-key <apiKey> --iex-price-plan <pricePlan>`.

```
$ lean backtest "My Project" --data-provider-historical IEX --iex-cloud-api-key apiKey --iex-price-plan Grow
```

The `--iex-price-plan` option must be `Launch` , `Grow` , or `Enterprise` . 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.

Research

To access IEX Cloud data from the local Research Environment, open a terminal in your organization workspace and then run `lean research <projectName> --data-provider-historical IEX --iex-cloud-api-key <apiKey> --iex-price-plan <pricePlan>`.

```
$ lean research "My Project" --data-provider-historical IEX --iex-cloud-api-key apiKey --iex-price-plan Grow
```

The `--iex-price-plan` option must be `Launch` , `Grow` , or `Enterprise` . 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.

Optimization

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

1. Add some parameters to your project.
2. Open a terminal in your organization workspace .
3. Run
   `lean optimize <projectName> --data-provider-historical IEX --iex-cloud-api-key <apiKey> --iex-price-plan <pricePlan>`
   .

   ```
   $ lean optimize "My Project" --data-provider-historical IEX --iex-cloud-api-key apiKey --iex-price-plan Grow
   ```

   The `--iex-price-plan` option must be `Launch` , `Grow` , or `Enterprise` .
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 IEX Cloud 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 IEX Cloud as the data provider, open a terminal in your organization workspace and then run
`lean live deploy <projectName> --data-provider-live IEX --iex-cloud-api-key <apiKey> --iex-price-plan <pricePlan> --brokerage <brokerageName> <requiredBrokerageOptions>`
.

```
$ lean live deploy "My Project" --data-provider-live IEX --iex-cloud-api-key apiKey --iex-price-plan Grow --brokerage
"Paper Trading"
```

The `--iex-price-plan` option must be `Launch` , `Grow` , or `Enterprise` . 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 IEX Cloud as the data provider, run
`lean cloud live deploy <projectName> --data-provider-live IEX --iex-cloud-api-key <apiKey> --iex-price-plan <pricePlan> --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 IEX --iex-cloud-api-key apiKey --iex-price-plan Grow --
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 IEX Cloud .

Supported Assets

Our IEX Cloud integration supports US Equity securities.

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. 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 .

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.

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.

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 .

Supported Assets

Our IQFeed integration supports securities from the following asset classes:

- US Equity
- US Equity Options
- Forex (listed on FXCM)
- US Futures

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 .

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.

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.

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 Polgyon 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 .

Supported Assets

Our Polygon integration supports securities from the following asset classes:

- US Equity
- US Equity Options
- US Indices
- US Index Options

Datasets

Terminal Link

Projects

Projects

Project Management

Introduction

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

Create Projects

Follow these steps to create a new Python project:

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

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

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

Follow these steps to create a new C# project:

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

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

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

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

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

Set the Default Language

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

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

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

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

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

Rename Projects

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

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

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

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

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

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

Delete Projects

Follow these steps to delete a project:

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

```
$ lean project-delete "My Project"
```

```
Successfully deleted project 'My Project'
```

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

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

Projects

Cloud Synchronization

Introduction

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

Pulling Cloud Projects

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

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

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

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

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

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

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

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

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

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

Pushing Local Projects

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

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

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

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

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

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

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

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

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):

C#

```
using System;

if (Environment.MachineName == "Computer")
{
    // Running locally
}
else
{
    // Running in the cloud
}
```

Projects

Structure

Introduction

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

Project Structure

New projects have the following structure:

```
.
├── .vscode/
│   └── launch.json
├── config.json
├── <projectName>.csproj
├── Main.cs (only generated by lean project-create)
└── Research.ipynb (only generated by lean project-create)
```

These files contain the following content:

File	Content
.vscode / launch.json	This file contains debug configuration to make debugging with VS Code easier.
config.json	This file contains the project configuration of the created project.
<projectName>.csproj	This file contains project configuration which Visual Studio, Rider, and VS Code can read to provide accurate C# autocomplete.
Main.cs	This file contains a basic C# algorithm to help you get started.
research.ipynb	This file contains a C#-based research notebook which can be opened in a research environment.

Projects

Workflows

Introduction

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

Cloud-focused Workflow

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

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

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

Locally-focused Workflow

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

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

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

Mixed Workflow

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

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

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

Projects

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.

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.

Properties

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

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

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:

```C#
using System;
using System.Collections;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.Linq;
using QuantConnect;
using QuantConnect.Parameters;
using QuantConnect.Benchmarks;
using QuantConnect.Brokerages;
using QuantConnect.Util;
using QuantConnect.Interfaces;
using QuantConnect.Algorithm;
using QuantConnect.Algorithm.Framework;
using QuantConnect.Algorithm.Framework.Selection;
using QuantConnect.Algorithm.Framework.Alphas;
using QuantConnect.Algorithm.Framework.Portfolio;
using QuantConnect.Algorithm.Framework.Execution;
using QuantConnect.Algorithm.Framework.Risk;
using QuantConnect.Indicators;
using QuantConnect.Data;
using QuantConnect.Data.Consolidators;
using QuantConnect.Data.Custom;
using QuantConnect.Data.Fundamental;
using QuantConnect.Data.Market;
using QuantConnect.Data.UniverseSelection;
using QuantConnect.Notifications;
using QuantConnect.Orders;
using QuantConnect.Orders.Fees;
using QuantConnect.Orders.Fills;
using QuantConnect.Orders.Slippage;
using QuantConnect.Scheduling;
using QuantConnect.Securities;
using QuantConnect.Securities.Equity;
using QuantConnect.Securities.Forex;
using QuantConnect.Securities.Interfaces;
using QuantConnect.Python;
using QuantConnect.Storage;
```

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 `using` statement at the top of your code file.

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

Supported Libraries for AMD64 Systems

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

C#

```
# Name                                Version
Accord                                3.6.0
Accord.Fuzzy                          3.6.0
Accord.MachineLearning                3.6.0
Accord.Math                           3.6.0
Accord.Statistics                     3.6.0
CloneExtensions                       1.3.0
Common.Logging                        3.4.1
Common.Logging.Core                   3.4.1
CsvHelper                             19.0.0
Deedle                                2.1.0
DotNetZip                             1.16.0
DynamicInterop                        0.9.1
fasterflect                           3.0.0
MathNet.Numerics                      5.0.0
McMaster.Extensions.CommandLineUtils 2.6.0
Microsoft.IO.RecyclableMemoryStream  2.3.2
Microsoft.NET.Test.Sdk                16.9.4
Microsoft.TestPlatform.ObjectModel   16.9.4
Moq                                   4.16.1
NetMQ                                 4.0.1.6
Newtonsoft.Json                       13.0.2
NodaTime                              3.0.5
NUnit                                 3.13.3
NUnit3TestAdapter                     4.2.1
protobuf-net                          3.1.33
QLNet                                 1.13.0
QuantConnect.pythonnet                2.0.26
RestSharp                             106.12.0
SharpZipLib                           1.3.3
```

```
System.ComponentModel.Composition    6.0.0
```

Supported Libraries for ARM64 Systems

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

C#

```
// Name                                     Version
Accord                                      3.5.0
Accord.Fuzzy                                3.5.0
Accord.MachineLearning                      3.5.0
Accord.Math                                 3.5.0
Accord.Math.Core                            3.5.0
Accord.Statistics                           3.5.0
AsyncIO                                     0.1.26.0
CloneExtensions                             1.3.0
CoinAPI.WebSocket.V1                        1.6.0
Common.Logging                              3.4.1
Common.Logging.Core                         3.4.1
CSharpAPI                                   1.0.0.0
CsvHelper                                   19.0.0
DotNetZip                                   1.13.3.506
DynamicInterop                              0.9.0
Fasterflect                                 3.0.0.0
FSharp.Core                                 4.5.0.0
ICSharpCode.SharpZipLib                     1.2.0
IQFeed.CSharpApiClient                      2.5.1
LaunchDarkly.EventSource                    3.3.2
MathNet.Numerics                            4.15.0
McMaster.Extensions.CommandLineUtils        2.6.0
Microsoft.IO.RecyclableMemoryStream         1.3.5.0
Microsoft.Win32.SystemEvents                3.1.0
NetMQ                                       4.0.0.1
Newtonsoft.Json                             12.0.3
NodaTime                                    3.0.5
protobuf-net                                3.0.29
protobuf-net.Core                           3.0.29
Python.Runtime                              2.0.1.0
QLNet                                       1.11.3
QuantConnect.Algorithm                      2.5.0.0
QuantConnect.Algorithm.CSharp               2.5.0.0
QuantConnect.Algorithm.Framework            2.5.0.0
QuantConnect.AlgorithmFactory               2.5.0.0
QuantConnect.Api                            2.5.0.0
QuantConnect.Brokerages                     2.5.0.0
QuantConnect.Common                         2.5.0.0
QuantConnect.Compression                    2.5.0.0
QuantConnect.Configuration                  2.5.0.0
QuantConnect.IBAutomater.exe                2.0.17
QuantConnect.Indicators                     2.5.0.0
QuantConnect.Lean.Engine                    2.5.0.0
QuantConnect.Lean.Launcher.exe              2.5.0.0
QuantConnect.Logging                        2.5.0.0
QuantConnect.Messaging                      2.5.0.0
QuantConnect.Queues                         2.5.0.0
QuantConnect.Research                       2.5.0.0
QuantConnect.ToolBox.exe                    2.5.0.0
RDotNet                                     1.9.0
RestSharp                                   106.6.10
SuperSocket.ClientEngine                    0.10.0.0
System.ComponentModel.Composition           5.0.0
System.Configuration.ConfigurationManager   3.1.0
System.Drawing.Common                       4.6.26919.02
System.Private.ServiceModel                 3.1.0
System.Security.Cryptography.Pkcs           4.6.26515.06
System.Security.Cryptography.ProtectedData  3.1.0
System.Security.Cryptography.Xml            4.6.26515.06
System.Security.Permissions                 3.1.0
System.ServiceModel                         3.1.0
System.ServiceModel.Primitives              3.1.0
System.Windows.Extensions                   3.1.0
Utf8Json                                    1.3.7.0
WebSocket4Net                               0.15.2.11
```

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 C# project:

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

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

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

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

Remove Libraries

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

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

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

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

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

Libraries

Project Libraries

Introduction

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

Create Libraries

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

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

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

```
$ lean project-create "Library/MyLibrary" --language python
Successfully created Python 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
   Restoring packages in 'My Project' to provide local autocomplete
   Restored successfully
   ```

3. In your project files, add the library namespace to the top of the page.

   By default, the namespace is `QuantConnect` .

   ```
   using QuantConnect;
   ```
   C#

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

```
var x = new MyLibrary();
var 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.
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"
Removing D:\qc\workspaces\Quant Organization\Library\MyLibrary from 'My Project\My Project.csproj'
Restoring packages in 'My Project'
  Determining projects to restore...
  Restored D:\qc\workspaces\Quant Organization\My Project\My Project.csproj (in 399 ms).
```

Delete Libraries

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

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

Projects

Custom Docker Images

Introduction

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

Using Custom Images

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

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

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

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

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

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

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

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

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

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

Building Custom Images

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

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

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

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

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

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

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.

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 locally with your own data.

Running Local Research Environment

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

Current Configuration

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

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

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

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

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

Old Configurations from QuantConnect

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

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

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

Opening Research Notebooks in PyCharm

Follow these steps to open a research notebook in PyCharm:

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

Opening Research Notebooks in VS Code

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

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

Retrieving Local Backtests

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

```C#
using Newtonsoft.Json;
using QuantConnect.Packets;

var backtestPath = "backtests/2021-03-03_23-46-43/CSharpProject.json";

var json = File.ReadAllText(backtestPath);
var data = JsonConvert.DeserializeObject<BacktestResultParameters>(json);

foreach (var item in data.Statistics) {
    Console.WriteLine($"{item.Key}: {item.Value}");
}
```

Retrieving Cloud Backtests

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

C#

```
var projectName = "Python Template";
var backtestName = "Adaptable Tan Frog";

var project = api.ListProjects().Projects.First(p => p.Name == projectName);
var partialBacktest = api.ListBacktests(project.ProjectId).Backtests.First(b => b.Name == backtestName);
var backtest = api.ReadBacktest(project.ProjectId, partialBacktest.BacktestId);

foreach (var item in backtest.Statistics) {
    Console.WriteLine($"{item.Key}: {item.Value}");
}
```

Backtesting

Backtesting

Deployment

Introduction

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

Run Local Backtests

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

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

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

Regular LEAN Engine

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

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

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

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

```
$ lean backtest "My Project" --output "My Project/custom-output"
```

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

Custom LEAN Engine

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

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

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

US Equity Options Algorithms

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

1. Download the US Equity Security Master dataset.

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

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

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

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

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

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

4. Create a new local project .

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

You can use the following example algorithm:

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

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

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

Run Cloud Backtests

Follow these steps to run a cloud backtest:

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

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

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

Download Datasets During Backtests

An alternative to manually downloading all 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.

Backtesting

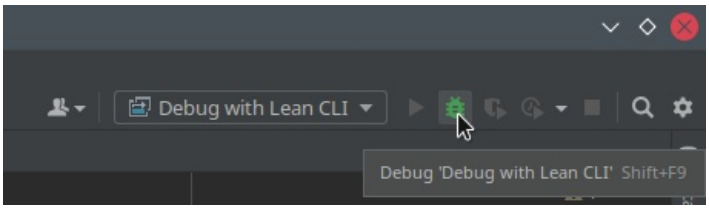
Debugging

Introduction

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

Python and PyCharm

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

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

4. 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.

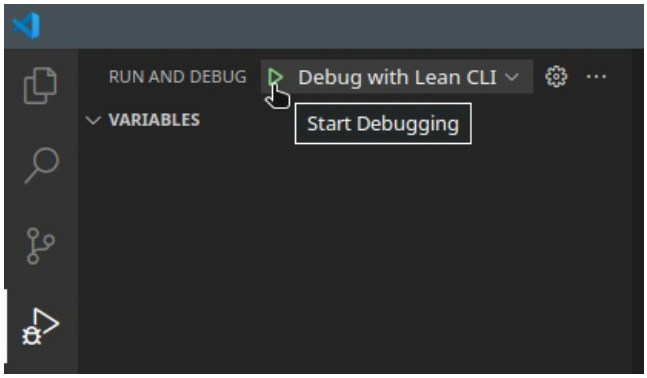
After finishing Python debugging with PyCharm, you will see a message saying "Connection to Python debugger failed". You can safely ignore this message.

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 ptvsd` , and then wait until the CLI tells you to attach to the debugger.

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

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

C# and Visual Studio

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

1. Follow the How to set up local autocomplete for C# in Visual Studio tutorial.
2. Open the project containing the **Main.cs** file in a new Visual Studio window. It is important that you open the project directory itself, not your organization workspace .
3. 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...
```

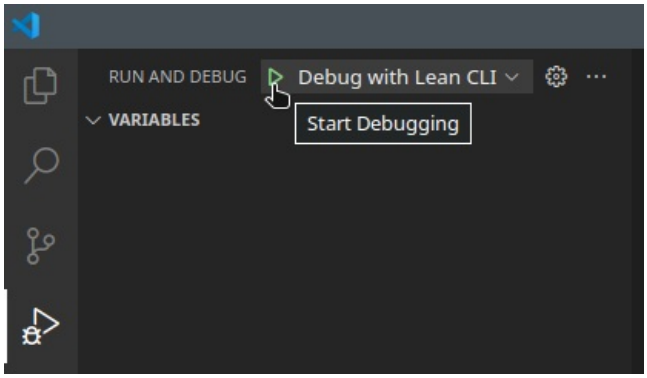
4. In Rider, select **Run > Attach To Remote Process...** .
5. In the pop-up that opens, select the target named **root@localhost:2222** .
6. Wait for Rider to connect and select the process named **dotnet QuantConnect.Lean.Launcher.dll** when a selector pops up to start debugging. You may have to select **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

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

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

Binance

Crypto & Crypto Futures

Bitfinex

Crypto

Bybit

Crypto & Crypto Futures

Coinbase

Crypto

Interactive Brokers

Equities, Options, Forex, Futures, & Future Options

Kraken

Crypto

Oanda

Forex & CFD

Samco

India Equities

TD Ameritrade

Equities

Tradier

Equities & Options

Trading Technologies

Futures

Zerodha

India Equities

Bloomberg EMSX

Equities, Forex, Crypto, Futures, & Options

See Also

IQ Feed
Polygon

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. 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.

Deploy Local Algorithms

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

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

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 1
```

3. Set your initial cash balance.

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

4. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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.

```
SetBenchmark(x => 0);
```
C#

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

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

Deploy Cloud Algorithms

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

1. Log in to the CLI if you haven't done so already.

2. Open a terminal in the organization workspace that contains the project.
3. Run `lean cloud live "<projectName>" --push --open` to push **./<projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

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

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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 "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 "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 "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) TD Ameritrade
   14) IQFeed
   15) Polygon
   16) IEX
   17) CoinApi
   18) Bybit
   To enter multiple options, separate them with comma:
   ```

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

   - IEX Cloud
   - IQFeed
   - Polygon

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

```
$ lean cloud live "My Project" --push --open
Brokerage: QuantConnect Paper 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
```

- 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

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. 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 .

Deploy Local Algorithms

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

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

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 1
```

3. Enter the exchange to use.

```
$ lean live "My Project"
Binance Exchange (Binance, BinanceUS, Binance-USDM-Futures, Binance-COIN-Futures): BinanceUS
```

4. 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 .

5. Enter the environment to use.

```
$ lean live "My Project"
Use the testnet? (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 "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter **7** to select the Binance brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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: wL1waeOC7VD447skCFeiat9pP3r1uKXfYomGg43uyCOgzl8xsI9SZsX97AXP4zWv
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 "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

9. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

10. Set your initial cash balance.

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

11. Set your initial portfolio holdings.

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

12. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

```
$ lean cloud live "My Project" --push --open
Select a node:
```

```
1) L-MICRO node 89c90172 - 1 CPU @ 2.4GHz, 0.5GB Ram
2) L-MICRO node 85a52135 - 1 CPU @ 2.4GHz, 0.5GB Ram
Enter an option: 1
```

13. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **5** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 5
```

3. Enter your API key Id and secret.

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

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

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

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter 5 to select the Bitfinex brokerage.

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

7. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

8. Set your initial cash balance.

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

9. Set your initial portfolio holdings.

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

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 "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 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
```

```
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **14** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 14
```

3. 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.

4. 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.

5. Enter the environment to use.

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

   The paper environment is Bybit Demo Trading .

6. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

- 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 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 cloud live "<projectName>" --push --open` to push **./ <projectName>** to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

4. Enter **14** to select the Bybit brokerage.

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

8. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

9. Set your initial cash balance.

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

10. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

11. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

```
$ lean cloud live "My Project" --push --open
Brokerage: Bybit
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
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
```

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **6** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 6
```

3. Enter your API key and API secret.

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

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

4. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Coinbase 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 cloud live "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

4. Enter 6 to select the Coinbase Advanced Trade brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
Enter an option: 6
```

5. Enter your API key and API secret.

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

To create new API credentials, see the API settings page on the Coinbase 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 "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

7. Enable or disable automatic algorithm restarting.

This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

8. Set your initial cash balance.

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

9. Set your initial portfolio holdings.

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

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 "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 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
```

```
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

If you have an ARM M1 chip, you can't deploy a local live algorithm with the IB brokerage.

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

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

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 2
```

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

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

```
Account password: ***************
```

5. Enter 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.

6. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

- Enter whether you want to enable delayed market data.

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

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

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter **2** to select the Interactive Brokers brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
Enter an option: 2
```

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

10. Enable or disable automatic algorithm restarting.

    This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

11. Set your initial cash balance.

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

12. Set your initial portfolio holdings.

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

13. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

14. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed

- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **12** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 12
```

3. Enter your API key and API secret.

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

To get your API credentials, see the API Management Settings page on the Kraken website.

4. Enter your verification tier.

```
$ lean live "My Project"
Select the Verification Tier (Starter, Intermediate, Pro):
```

   For more information about verification tiers, see Verification levels explained on the Kraken website.

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

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

wizard, and open the results in the browser once the deployment starts.

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

4. Enter **12** to select the Kraken brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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 "My Project" --push --open
Do you want to send notifications on order events? [y/N]: y
Do you want to send notifications on insights? [y/N]: y
Email notifications: None
Webhook notifications: None
SMS notifications: None
Select a notification method:
1) Email
2) Webhook
3) SMS
```

```
4) Telegram
Enter an option: 1
Email address: john.doe@example.com
Subject: Algorithm notification
Email notifications: john.doe@example.com
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

8. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

9. Set your initial cash balance.

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

10. Set your initial portfolio holdings.

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

11. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

12. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

Brokerages

Oanda

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **4** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 4
```

3. Enter the environment to use. Enter `Trade` for fxTrade or `Practice` for fxTrade Practice.

```
$ lean live "My Project"
Environment? (Practice, Trade): Trade
```

4. Enter your OANDA account ID.

```
$ lean live "My Project"
Account id: 001-011-5838423-001
```

To get your account ID, see your Account Statement page on the OANDA website.

5. Enter your OANDA API token.

```
$ lean live "My Project"
API token: ****************
```

To create a token, see the Manage API Access page on the OANDA website.

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

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter **4** to select the Oanda brokerage.

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

9. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

10. Set your initial cash balance.

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

11. Set your initial portfolio holdings.

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

12. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

13. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **9** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 9
```

3. Enter your Samco credentials.

```
$ lean live "My Project"
Client ID:
Client Password:
```

4. Enter your year of birth.

```
$ lean live "My Project"
Year of Birth:
```

5. Enter the product type.

```
$ lean live "My Project"
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

6. Enter the trading segment.

```
$ lean live "My Project"
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

7. Enter **8** to select the Samco data provider.

```
$ lean live "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) TD Ameritrade
```

```
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma: 8
```

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

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

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Samco brokerage 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 "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

4. Enter **9** to select the Samco brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

8. Enter the trading segment.

```
$ lean cloud live "My Project" --push --open
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

9. Configure your notification settings.

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

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

```
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting.

    This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

11. Set your initial cash balance.

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

12. Set your initial portfolio holdings.

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

13. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma: 9
```

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

```
$ lean cloud live "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 VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

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
```

Brokerages

TD Ameritrade

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. This page contains instructions on how to start live trading with the TD Ameritrade 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 TD Ameritrade brokerage integration, see the Lean.Brokerages.TDAmeritrade repository .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **13** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 13
```

3. Enter your TD Ameritrade credentials.

```
$ lean live "My Project"
API key:
OAuth Access Token:
Account number:
```

To get your account credentials, see Account Types .

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

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

```
Successfully compiled project 'My Project'
```

4. Enter **13** to select the TD Ameritrade brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
Enter an option: 13
```

5. Enter your TD Ameritrade credentials.

```
$ lean cloud live "My Project"
API key:
OAuth Access Token:
Account number:
```

   To get your account credentials, see Account Types .

6. Configure your notification settings.

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

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

7. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

8. Set your initial cash balance.

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

9. Set your initial portfolio holdings.

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

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 "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 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) TD Ameritrade
```

```
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

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

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

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

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

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **3** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 3
```

3. Enter your Tradier account Id and access token.

```
$ lean live "My Project"
Account id: VA000001
Access token: ****************
```

To get these credentials, see your Settings/API Access page on the Tradier website.

4. Enter whether the developer sandbox should be used.

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

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

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter **3** to select the Tradier brokerage.

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

```
Webhook notifications: None
SMS notifications: None
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

8. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

9. Set your initial cash balance.

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

10. Set your initial portfolio holdings.

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

11. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

12. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **11** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 11
```

3. Enter your Trading Technologies credentials.

```
$ lean live "My Project"
User name: john
Session password: ***************
Account name: jane
```

4. Enter the REST configuration.

```
$ lean live "My Project"
REST app key: my-rest-app-key
REST app secret: ******************
REST environment: my-environment
```

5. Enter the market data configuration.

```
$ lean live "My Project"
Market data sender comp id:
Market data target comp id:
Market data host:
Market data port:
```

6. Enter the order routing configuration.

```
$ lean live "My Project"
Order routing sender comp id:
Order routing target comp id:
Order routing host:
Order routing port:
```

7. Enter whether FIX messages must be logged.

```
$ lean live "My Project"
Log FIX messages (yes/no): yes
```

8. Set your initial cash balance.

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

9. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

Deploy Cloud Algorithms

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

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

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

4. Enter **11** to select the Trading Technologies brokerage.

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

9. Enable or disable automatic algorithm restarting.

   This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

10. Set your initial portfolio holdings.

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

11. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

12. 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma:
```

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

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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. 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 .

Deploy Local Algorithms

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

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **8** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 8
```

3. Enter your Kite Connect API key and access token.

```
$ lean live "My Project"
API key: hp9erb9ct0lqaxpm
Access token: ********************
```

4. Enter the product type.

```
$ lean live "My Project"
Product type (mis, cnc, nrml):
```

The following table describes the product types:

Product Type	Description
mis	Intraday products
cnc	Delivery products
nrml	Carry forward products

5. Enter the trading segment.

```
$ lean live "My Project"
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
equity	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
commodity	For trading commodities on the Multi Commodity Exchange of India (MCX)

6. Enter 7 to select the Zerodha data provider.

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma: 7
```

7. Enter whether you have a history API subscription.

```
$ lean live "My Project"
Do you have a history API subscription? (true, false): true
```

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

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

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the Zerodha brokerage 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 "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

4. Enter **8** to select the Zerodha brokerage.

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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
`mis`	Intraday products
`cnc`	Delivery products
`nrml`	Carry forward products

7. Enter the trading segment.

```
$ lean cloud live "My Project" --push --open
Trading segment (equity, commodity):
```

The following table describes when to use each trading segment:

Trading Segment	Description
`equity`	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
`commodity`	For trading commodities on the Multi Commodity Exchange of India (MCX)

8. Enter whether you have a history API subscription.

```
$ lean live "My Project"
Do you have a history API subscription? (true, false): true
```

9. Configure your notification settings.

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

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

```
Telegram notifications: None
Do you want to add another notification method? [y/N]: n
```

10. Enable or disable automatic algorithm restarting.

    This feature attempts to restart your algorithm if it fails due to a runtime error, like a brokerage API disconnection.

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

11. Set your initial cash balance.

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

12. Set your initial portfolio holdings.

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

13. Select the live node that you want to use.

    If you only have one idle live trading node, it is selected automatically and this step is skipped.

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

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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma: 8
```

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

```
$ lean cloud live "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': 'GOOCV VP83T1ZUHROL', 'quantity': 10,
'averagePrice': 50.0}]
Automatic algorithm restarting: Yes
Are you sure you want to start live trading for project 'My Project'? [y/N]: y
```

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
```

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. 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 and UAT paper trading through Bloomberg order routing via the EMSX network.

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`
- `GoodTilCanceled`
- `GoodTilDate`

Get Open Orders

Terminal Link lets you access open orders .

Monitor Fills

Terminal Link allows you to monitor orders as they fill through order events .

Updates

Terminal Link doesn't support order updates .

Cancellations

Terminal Link enables you to cancel open orders .

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 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. Open a terminal in the organization workspace that contains the project.
2. Run `lean live "<projectName>"` to start a live deployment wizard for the project in **. / <projectName>** and then enter the brokerage number, **10** .

```
$ lean live "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) TD Ameritrade
14) Bybit
Enter an option: 10
```

3. Enter the environment to use.

```
$ lean live "My Project"
Environment (Production, Beta): Production
```

4. Enter the host and port of the Bloomberg server.

```
$ lean live "My Project"
Server host: 127.0.0.1
Server port: 8194
```

5. Enter your EMSX configuration

```
$ lean live "My Project"
EMSX broker: someValue
EMSX account:
```

6. Enter your Open FIGI API key.

```
$ lean live "My Project"
Open FIGI API key:
```

7. Enter **9** to select the Terminal Link live data provider.

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma: 9
```

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

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

Deploy Cloud Algorithms

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 :

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

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

4. Enter **10** to select the Terminal Link brokerage.

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

13. Set your initial cash balance.

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

14. Set your initial portfolio holdings.

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

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 "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 **9** to select the Terminal Link live data provider.

```
$ lean live "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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Bybit
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:

- IEX Cloud
- IQFeed
- Polygon

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

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

- 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
```

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.

IEX Cloud

US Equities

IQFeed

US Equities, US Options, Forex, & Futures

Polygon

US Equities, US Equity Options, US Indices, & US Index Options

See Also

Brokerages

Data Providers

IEX Cloud

Introduction

Instead of using the live data from your brokerage, you can also use IEX Cloud. This tutorial demonstrates how to set up the IEX Cloud data provider with the QuantConnect Paper Trading brokerage.

To use IEX Cloud, you need an account subscribed to one or more data bundles . You also must select either the Launch, Grow, or Enterprise subscription tier; this dictates your maximum requests per second. If you don't have an account, register today and get your API key . Pricing information can be found here .

Upon registration, your account will automatically start a 7-day free trial. This trial has access to all API endpoints and a rate limit of 5 requests per second. If you have any questions regarding IEX Cloud's API service, please contact them here: support@iexcloud.io .

To view the implementation of the IEX Cloud integration, see the Lean.DataSource.IEX repository .

Deploy Local Algorithms

Follow these steps to start local live trading with the IEX Cloud data provider:

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

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

3. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

4. Enter your IEX Cloud API Key.

```
$ lean live "My Project"
Configure credentials for IEX
Your iexcloud.io API token publishable key:
```

5. Enter your price plan.

```
$ lean live "My Project"
IEX Cloud Price plan (Launch, Grow, Enterprise):
```

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 IEX Cloud 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 "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment

wizard, and open the results in the browser once the deployment starts.

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

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

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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 "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 "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 "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 **15** to select the IEX Cloud 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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
To enter multiple options, separate them with comma: 15
```

9. Enter your IEX Cloud API Key.

```
$ lean cloud live "My Project" --push --open
Configure credentials for IEX
Your iexcloud.io API token publishable key:
```

10. Enter your price plan.

```
$ lean cloud live "My Project" --push --open
IEX Cloud Price plan (Launch, Grow, Enterprise):
```

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

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

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

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

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

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

Supported Assets

Our IEX Cloud integration supports US Equity securities.

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 .

Deploy Local Algorithms

Follow these steps to start local live trading with the IQ Feed data provider:

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

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

3. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

4. Enter the path to the IQConnect binary.

   The default path is **C: / Program Files (x86) / DTN / IQFeed / iqconnect.exe** if you used the default settings when installing the IQFeed client.

```
$ lean live "My Project"
IQConnect binary location [C:/Program Files (x86)/DTN/IQFeed/iqconnect.exe]:
```

5. Enter your IQFeed username and password.

```
$ lean live "My Project"
Username: 123456
Password: **********
```

6. If you have an IQFeed developer account, enter the product Id and version of your account.

```
$ lean live "My Project"
Product id: <yourID>
Product version: 1.0
```

7. If you don't have an IQFeed developer account, open **iqlink.exe** , log in to IQLink with your username and password, and then enter random numbers for the product id and version.

```
$ lean live "My Project"
Product id: 123
Product version: 1.0
```

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

If you already have a live environment configured in your Lean configuration file , you can skip the interactive wizard by providing

the `--environment <value>` option in step 2. The value of this option must be the name of an environment which has `live-mode` set to `true` .

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

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 .

Deploy Local Algorithms

Follow these steps to start local live trading with the Polygon data provider:

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

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

3. 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 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) TD Ameritrade
13) IQFeed
14) Polygon
15) IEX
16) CoinApi
17) Custom data only
18) Bybit
To enter multiple options, separate them with comma:
```

4. 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.

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

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

Deploy Cloud Algorithms

Follow these steps to start live trading a project in the cloud with the QuantConnect Paper Trading brokerage 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 "<projectName>" --push --open` to push **. / <projectName>** . to the cloud, start a live deployment wizard, and open the results in the browser once the deployment starts.

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

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

```
$ lean cloud live "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) TD Ameritrade
14) Bybit
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 "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 "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 "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) TD Ameritrade
14) IQFeed
15) Polygon
16) IEX
17) CoinApi
18) Bybit
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 "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

Live Trading

Algorithm Control

Introduction

The algorithm control features let you adjust your algorithm while it executes live so that you can perform actions that are not written in the project files. The control features let you intervene in the execution of your algorithm and make adjustments. The control features that are available to you depend on if you deploy the algorithm on your local machine or on the QuantConnect cloud servers.

Control Local Algorithms

While your local algorithms run, you can add security subscriptions, submit orders, adjust orders, and stop their execution.

Add Security Subscriptions

You can manually create security subscriptions for your algorithm instead of calling the `Add securityType` methods in your code files. If you add security subscriptions to your algorithm, you can place manual trades without having to edit and redeploy the algorithm. To add security subscriptions, open a terminal in the organization workspace that contains the project and then run `lean live add-security "My Project"` .

```
$ lean live add-security "My Project" --ticker "SPY" --market "usa" --security-type "equity"
```

For more information about the command options, see Options .

You can't manually remove security subscriptions.

Submit Orders

In local deployments, you can manually place orders instead of calling the automated methods in your project files. You can use any order type that is supported by the brokerage that you used when deploying the algorithm. To view the supported order types of your brokerage, see the **Orders** section of your brokerage model . Some example situations where it may be helpful to place manual orders instead of stopping and redeploying the algorithm include the following:

- Your brokerage account had holdings in it before you deployed your algorithm
- Your algorithm had bugs in it that caused it to purchase the wrong security
- You want to add a hedge to your portfolio without adjusting the algorithm code
- You want to rebalance your portfolio before the rebalance date

To submit orders, open a terminal in the organization workspace that contains the project and then run `lean live submit-order "My Project"` .

```
$ lean live submit-order "My Project" --ticker "SPY" --market "usa" --security-type "equity" --order-type "market" --
quantity 10
```

For more information about the command options, see Options .

Update Orders

To update an existing order, open a terminal in the organization workspace that contains the project and then run `lean live update-order "My Project"` .

```
$ lean live update-order "My Project" --order-id 1 --quantity 5
```

For more information about the command options, see Options .

Cancel Orders

To cancel an existing order, open a terminal in the organization workspace that contains the project and then run `lean live cancel-order "My Project"` .

```
$ lean live cancel-order "My Project" --order-id 1
```

For more information about the command options, see Options .

Liquidate Positions

To liquidate a specific asset in your algorithm, open a terminal in the organization workspace that contains the project and then run `lean live liquidate "My Project"` .

```
$ lean live liquidate "My Project" --ticker "SPY" --market "usa" --security-type "equity"
```

When you run the command, if the market is open for the asset, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market on open orders.

For more information about the command options, see Options .

Stop Algorithms

The `lean live stop` command immediately stops your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Update the settings you entered into the deployment command
- Place manual orders through your brokerage account

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean live stop "My Project"` .

```
$ lean live stop "My Project"
```

For more information about the command options, see Options .

Control Cloud Algorithms

While your cloud algorithms run, you can liquidate their positions and stop their exeuction.

Liquidate Positions

The `lean cloud live liquidate` command acts as a "kill switch" to sell all of your portfolio holdings. If your algorithm has a bug in it that caused it to purchase a lot of securities that you didn't want, this command let's you easily liquidate your portfolio instead of placing many manual trades. When you run the command, if the market is open for an asset you hold, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market on open orders. After the algorithm submits the liquidation orders, it stops executing.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean cloud live liquidate "My Project"` .

```
$ lean cloud live liquidate "My Project"
```

For more information about the command options, see Options .

Stop Algorithms

The `lean live stop` command immediately stops your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Update the settings you entered into the deployment command
- Place manual orders through your brokerage account

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

To stop an algorithm, open a terminal in the organization workspace that contains the project and then run `lean cloud live stop "My Project"` .

```
$ lean cloud live stop "My Project"
```

For more information about the command options, see Options .

Reports

Introduction

The `lean report` command in the Lean CLI is a wrapper around the LEAN Report Creator. The LEAN Report Creator is a program included with LEAN which allows you to quickly generate polished, professional-grade reports of your backtests and live trading results. We hope that you can use these reports to share your strategy performance with prospective investors.

Generate Reports

Follow these steps to generate a report of a trading algorithm:

1. Open a terminal in the organization workspace that contains the project.
2. Run `lean report` to generate a report of the most recent backtest.

   ```
   $ lean report
   20210322 20:03:48.718 TRACE:: QuantConnect.Report.Main(): Parsing source files...backtest-data-source-file.json,
   20210322 20:03:51.602 TRACE:: QuantConnect.Report.Main(): Instantiating report...
   Successfully generated report to './report.html'
   ```

   By default, the generated report is saved to **./report.html**, although you can change this by providing a custom path with the `--report-destination <path>` option. To generate a report of a backtest that is not the most recent one, you can use the `--backtest-results <path>` option to specify the path to the backtest results JSON file to generate a report for it.
3. Open the generated report in the browser and inspect its results.

You can also configure the following optional details:

Detail	Description
Strategy name	This name is displayed in the top-right corner of each page and can be configured using `--strategy-name <value>`. This value defaults to the name of the project directory.
Strategy version	This version is displayed next to the strategy name and can be configured using `--strategy-version <value>`.
Strategy description	This description is displayed underneath the "Strategy Description" header on the first page and can be configured using `--strategy-description`. This value defaults to the description stored in the project's configuration.
Live results	These results are displayed over the backtest results and can be configured using `--live-results <path>`. The provided path must point to a JSON file containing live results. For example, `--live-results "My Project/live/2022-03-17_10-53-12/L-3578882079.json"`.

Customize the Report HTML

The Report / template.html file in the LEAN GitHub repository defines the stucture 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;">{{$TEXT-CRISIS-TITLE}}</th>
        </tr>
        </thead>
        <tbody>
        <tr>
            <td style="padding:0;">
                <img src="{{$PLOT-CRISIS-CONTENT}}">
            </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>` .

Key Statistics

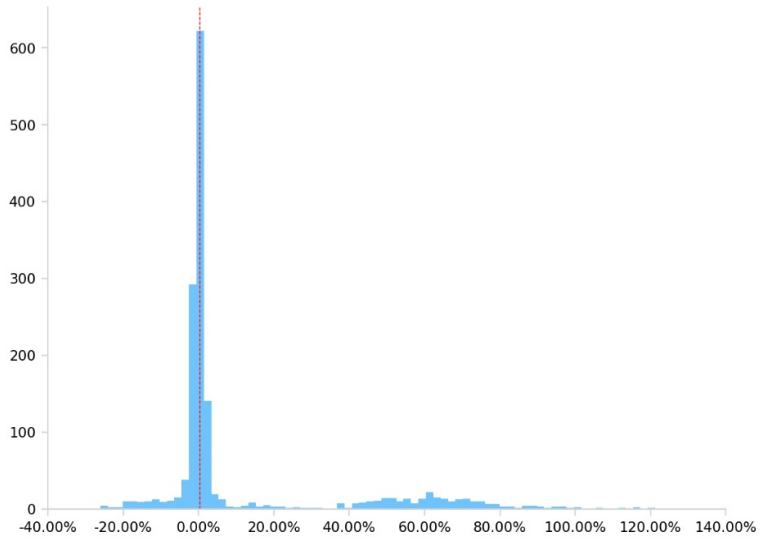
The top of the backtest report displays statistics to summarize your algorithm's performance. The following table describes the key statistics in the report:

Statistic	Description
Runtime Days	The number of days in the backtest or live trading period.
Turnover	The percentage of the algorithm's portfolio that was replaced in a given year.
CAGR	The annual percentage return that would be required to grow a portfolio from its starting value to its ending value.
Markets	The asset classes that the algorithm trades.
Trades per day	The total number of trades during the backtest divided by the number of days in the backtest. Trades per day is an approximation of the algorithm's trading frequency.
Drawdown	The largest peak to trough decline in an algorithm's equity curve.
Probabilistic SR	The probability that the estimated Sharpe ratio of an algorithm is greater than a benchmark (1).
Sharpe Ratio	A measure of the risk-adjusted return, developed by William Sharpe.
Information Ratio	The amount of excess return from the risk-free rate per unit of systematic risk.
Strategy Capacity	The maximum amount of money an algorithm can trade before its performance degrades from market impact.

Returns

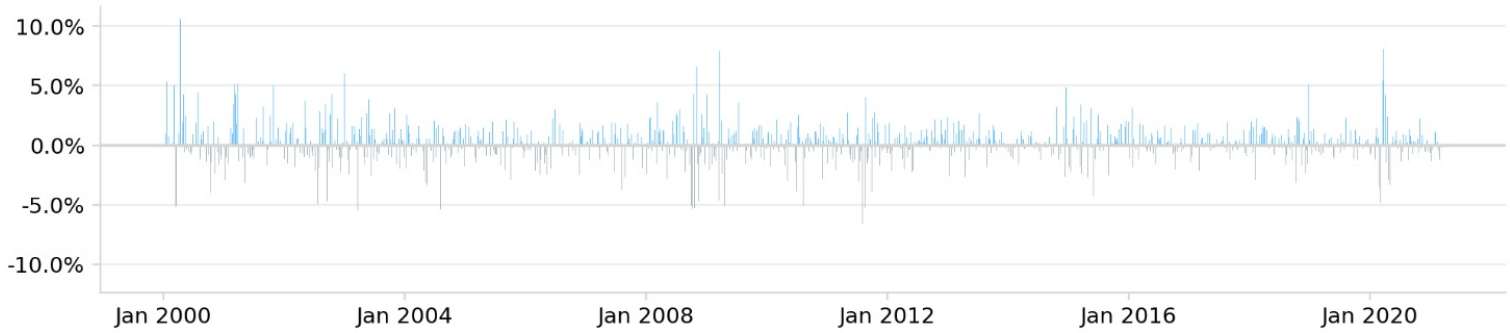
The backtest report displays charts to show the algorithm's returns per trade, per day, per month, per year, and the cumulative returns over the backtest.

Returns per Trade



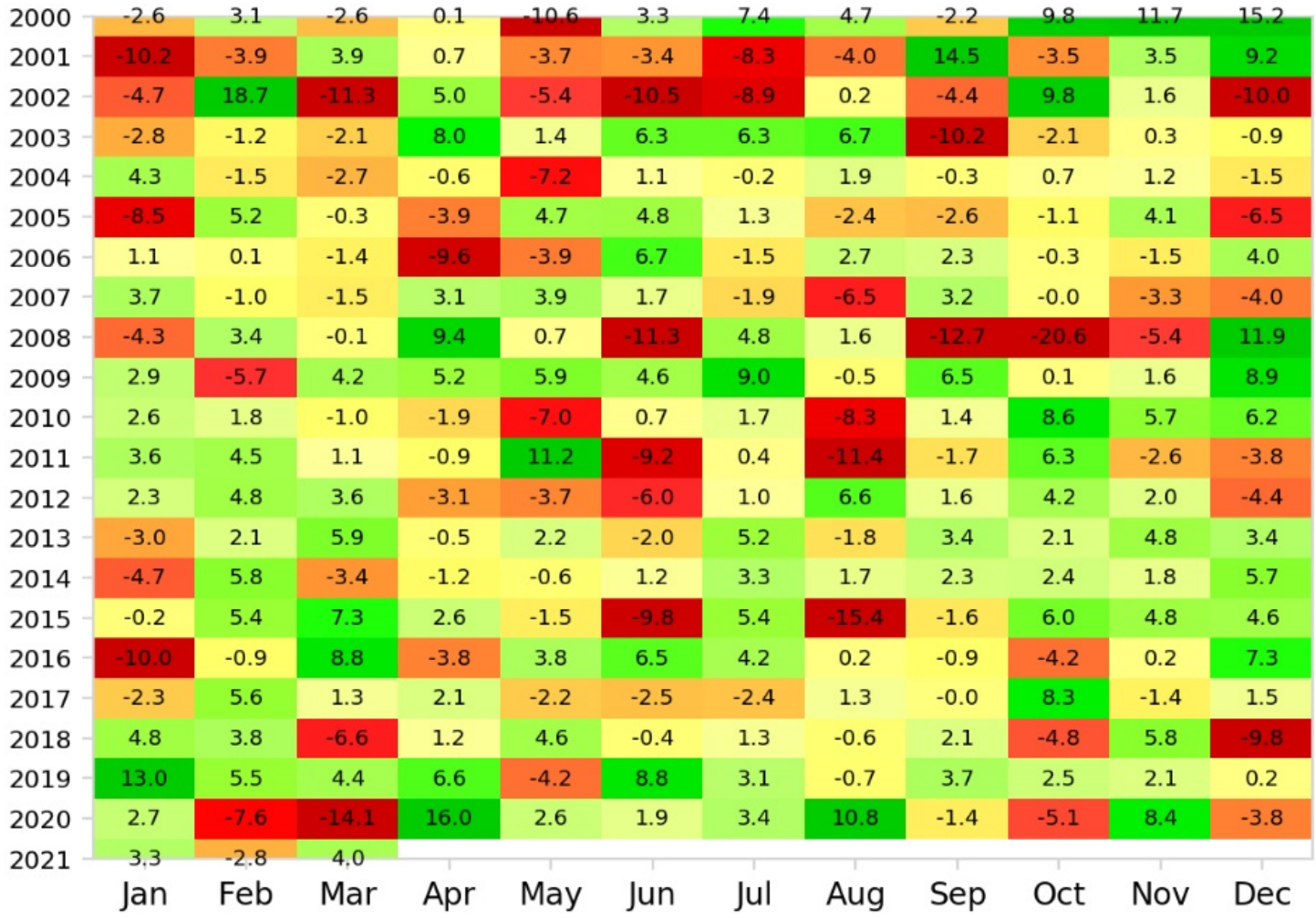
This chart displays a histogram that shows the distribution of returns per trade over the backtesting period.

Daily Returns



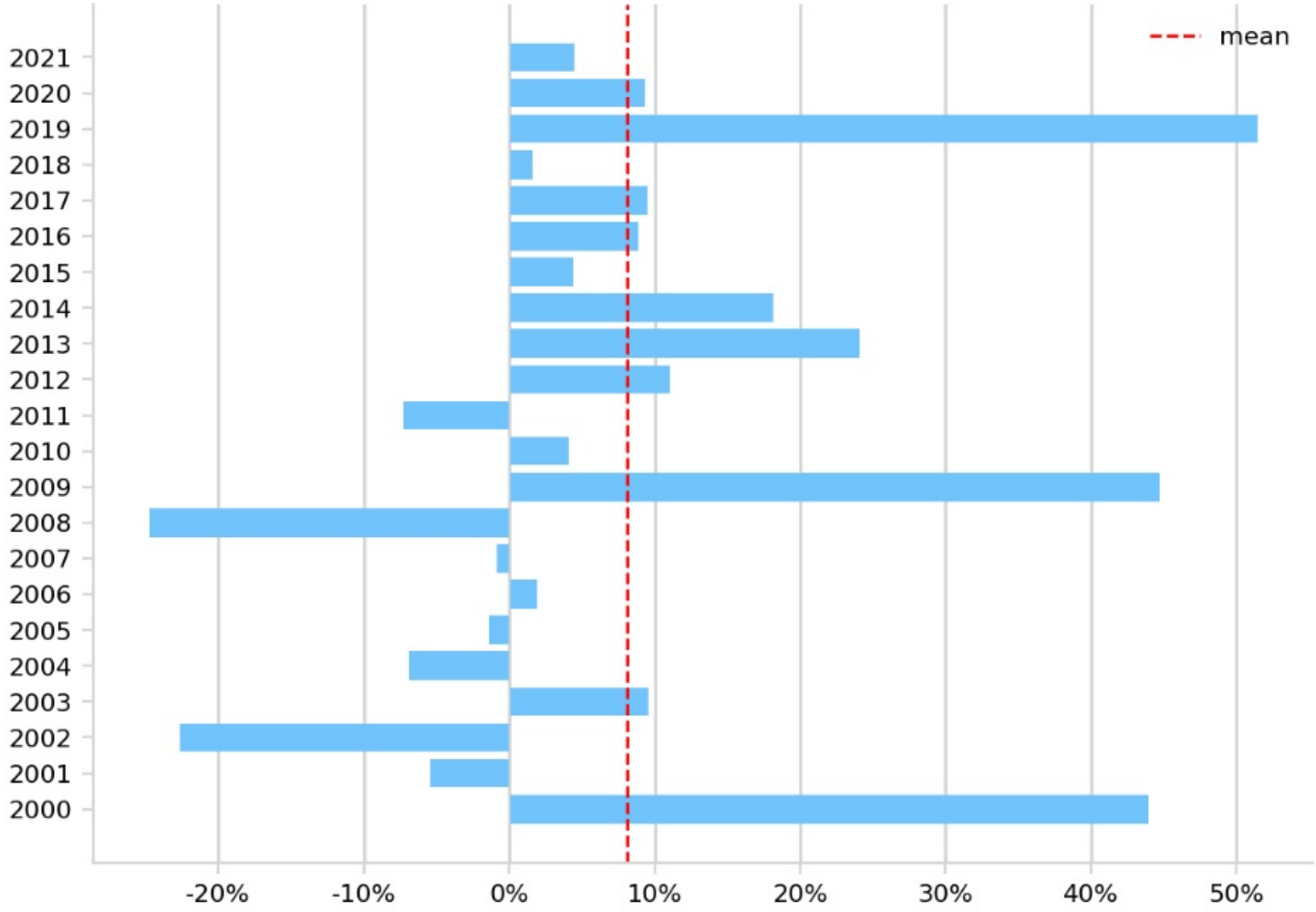
This chart displays the returns of each day. Blue bars represent profitable days and gray bars represent unprofitable days.

Monthly Returns



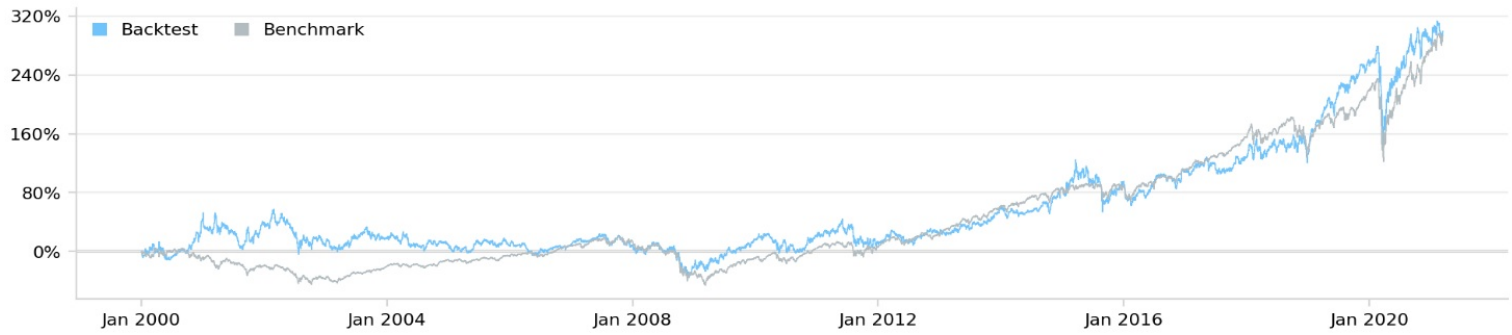
This chart displays the return of each month. We convert the original equity curve series into a monthly series and calculate the returns of each month. Green cells represent months with a positive return and red cells represent months with a negative return. Months that have a greater magnitude of returns are represented with darker cells. Yellow cells represent months with a relatively small gain or loss. White rectangles represent months that are not included in the backtest period. The values in the cells are percentages.

Annual Returns



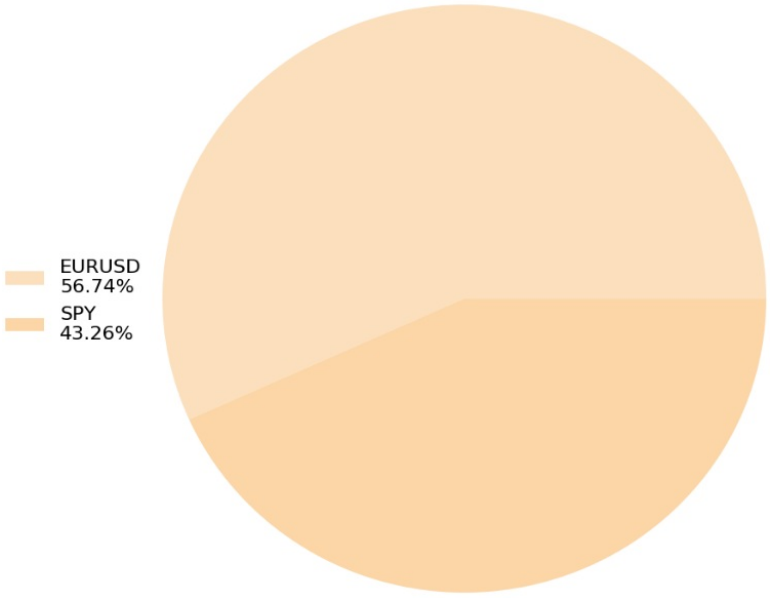
This chart displays the return of each year. We calculate the total return within each year and represent each year with a blue bar. The red dotted line represents the average of the annual returns.

Cumulative Returns



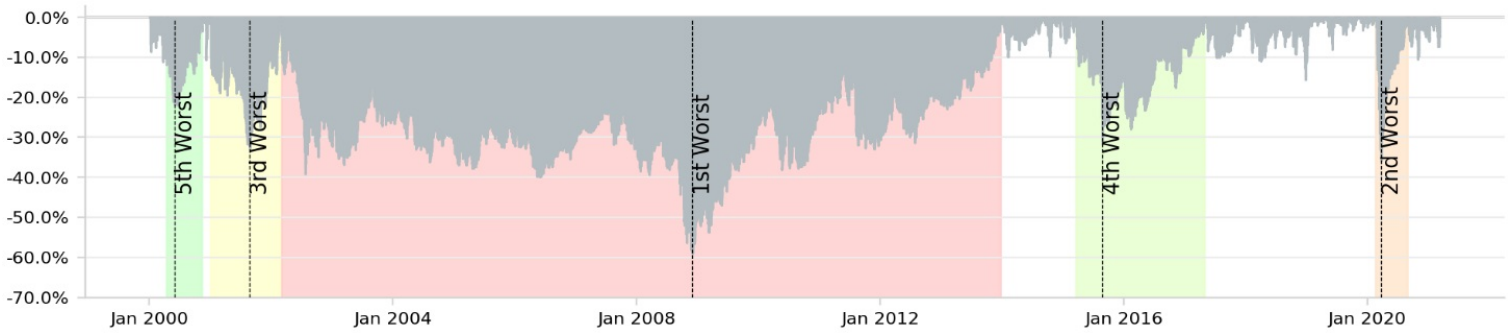
This chart displays the cumulative returns of your algorithm. The blue line represents your algorithm and the gray line represents the benchmark.

Asset Allocation



This chart displays a time-weighted average of the absolute holdings value for each asset that entered your portfolio during the backtest. When an asset has a percentage that is too small to be shown in the pie chart, it is incorporated into an "Others" category.

Drawdown

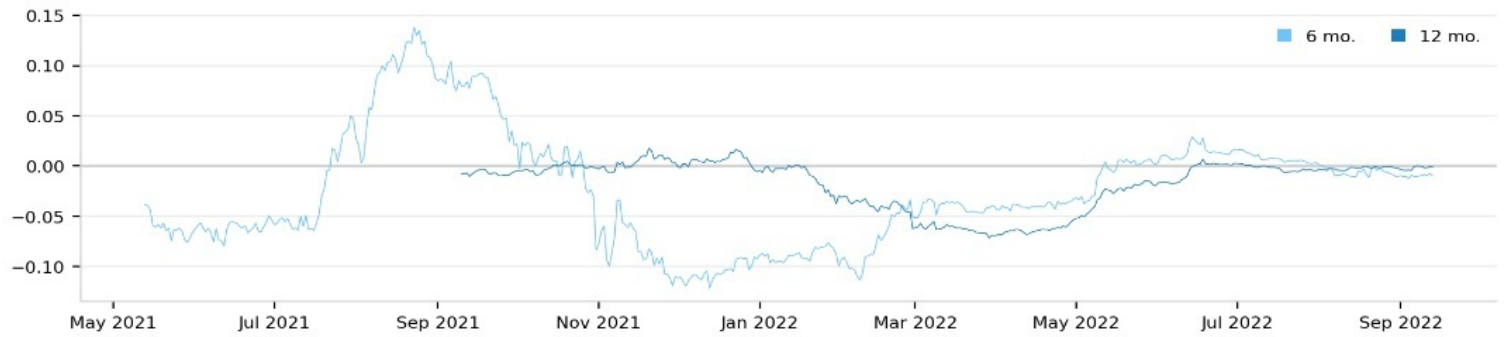


This chart displays the peak-to-trough drawdown of your portfolio's equity throughout the backtest period. The drawdown of each day is defined as the percentage loss since the maximum equity value before the current day. The drawdowns are calculated based on daily data. The top 5 drawdown periods are marked in the chart with different colors.

Rolling Statistics

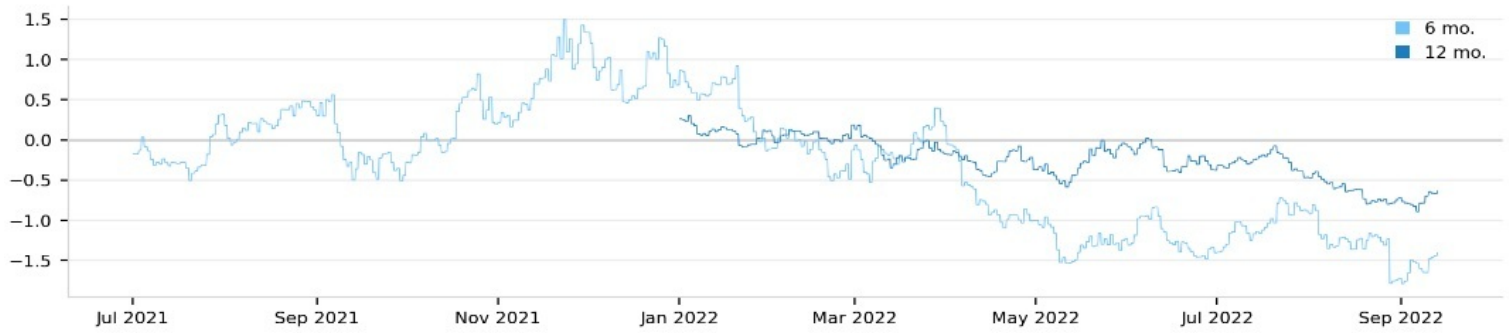
The backtest report displays time series for your portfolio's rolling beta and Sharpe ratio .

Rolling Portfolio Beta



This chart displays the rolling portfolio beta over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Rolling Sharpe Ratio

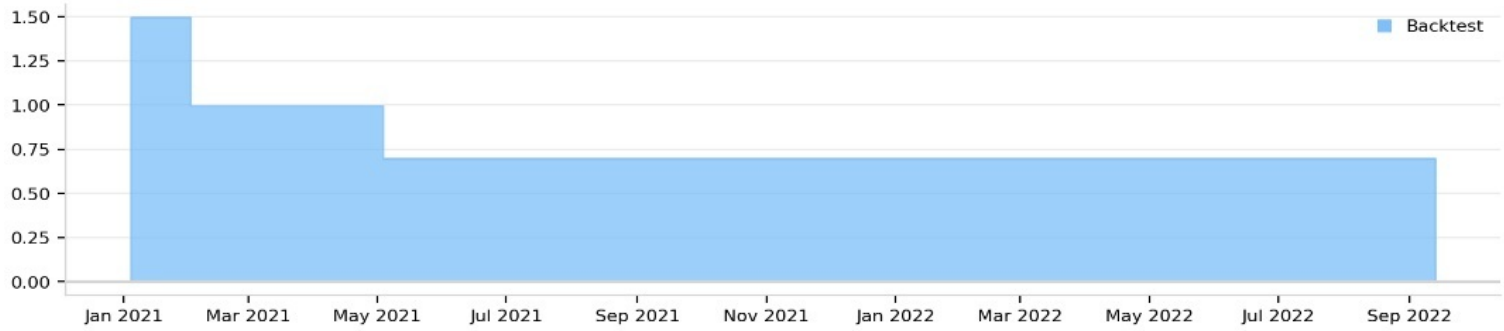


This chart displays the rolling portfolio Sharpe ratio over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Exposure

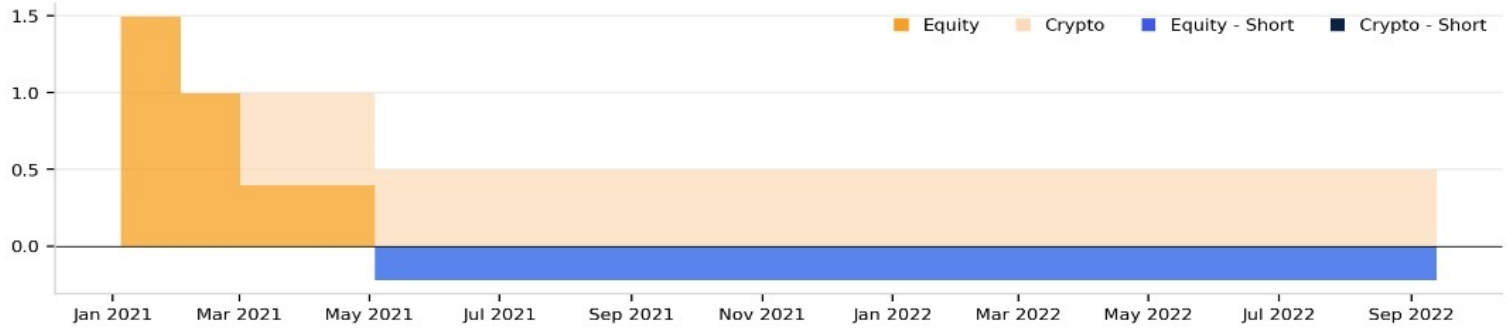
The backtest report displays time series for your portfolio's overall leverage and your portfolio's long-short exposure by asset class.

Leverage



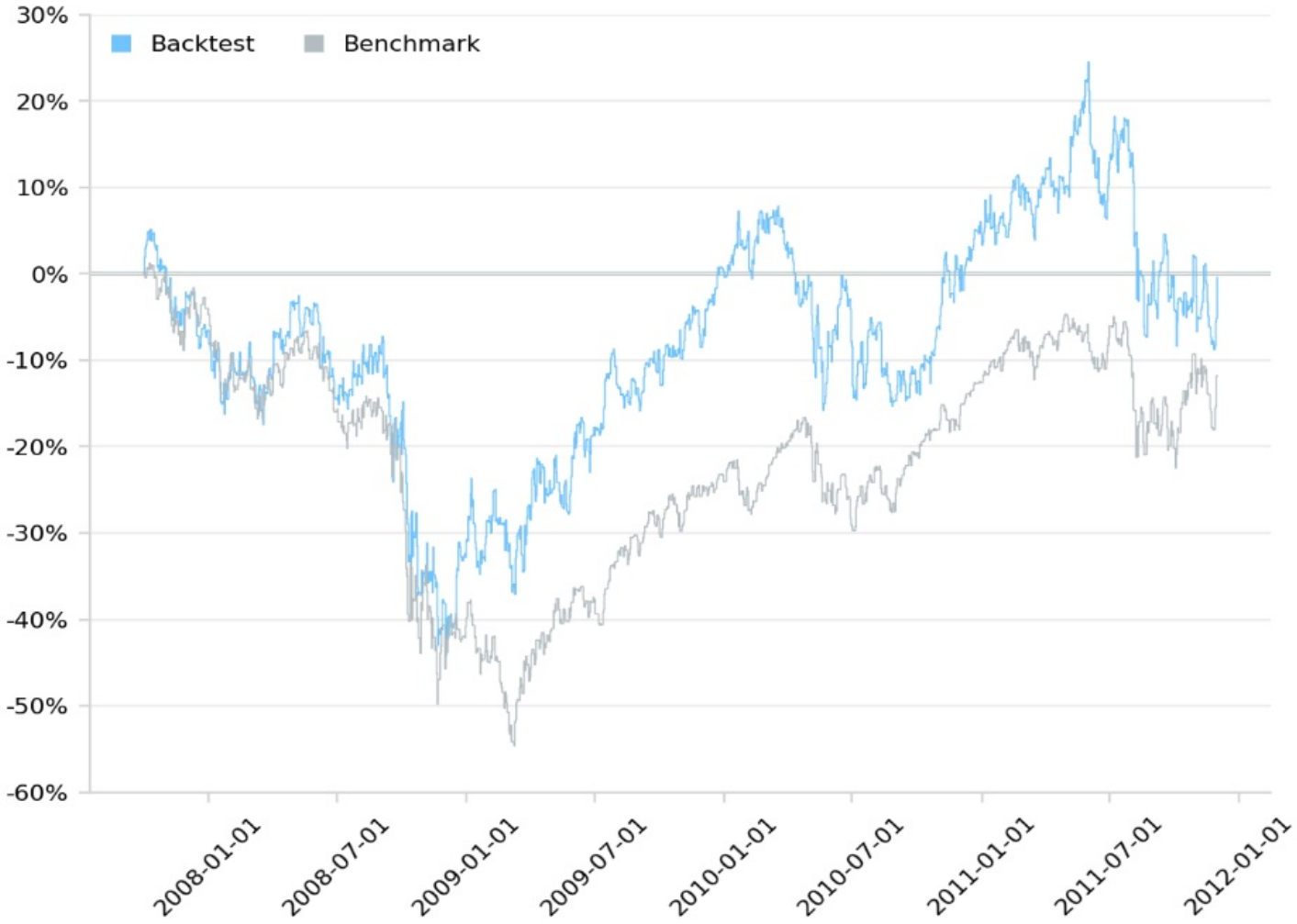
This chart displays your algorithm's utilization of leverage over time.

Long-Short Exposure By Asset Class



This chart displays your algorithm's long-short exposure by asset class over time.

Crisis Events



This set of charts displays the cumulative returns of your algorithm and the benchmark during various historical periods. The blue line represents the cumulative returns of your algorithm and the grey line represents the cumulative return of the benchmark. The report only contains the crisis event that occurred during your algorithm's backtest period. The following table shows the crisis events that may be included in your backtest report:

Crisis Name	Start Date	End Date
DotCom Bubble 2000	2/26/2000	9/10/2000
September 11, 2001	9/5/2001	10/10/2001
U.S. Housing Bubble 2003	1/1/2003	2/20/2003
Global Financial Crisis 2007	10/1/2007	12/1/2011
Flash Crash 2010	5/1/2010	5/22/2010
Fukushima Meltdown 2011	3/1/2011	4/22/2011
U.S. Credit Downgrade 2011	8/5/2011	9/1/2011
ECB IR Event 2012	9/5/2012	10/12/2012
European Debt Crisis 2014	10/1/2014	10/29/2014
Market Sell-Off 2015	8/10/2015	10/10/2015
Recovery 2010-2012	1/1/2010	10/1/2012
New Normal 2014-2019	1/1/2014	1/1/2019
COVID-19 Pandemic 2020	2/10/2020	9/20/2020

Parameters

This section of the report shows the name and value of all the parameters in your project.

Optimization

Optimization

Parameters

Introduction

Project parameters are parameters that are defined in your project's configuration file . These parameters are a replacement for constants in your algorithm and can be optimized using one of LEAN's optimization strategies either locally or in the cloud.

Configure Project Parameters

Follow these steps to make your algorithm use project parameters instead of constant values:

1. Open your project in your preferred editor.
2. Open the project's **config.json** file.
3. Add the required parameters in the `parameters` property. All keys and values of this object must be strings. Example:

```
{
    "parameters": {
        "ema-fast": "10",
        "ema-medium": "30",
        "ema-slow": "50"
    }
}
```

4. Open your algorithm in the editor.
5. Call `QCAlgorithm.GetParameter(name)` in your algorithm to retrieve the value of a parameter and use that instead of constant values.

C#

```
namespace QuantConnect.Algorithm.CSharp
{
    public class ParameterizedAlgorithm : QCAlgorithm
    {
        private ExponentialMovingAverage _fast;
        private ExponentialMovingAverage _medium;
        private ExponentialMovingAverage _slow;

        public override void Initialize()
        {
            SetStartDate(2020, 1, 1);
            SetCash(100000);
```

```
            AddEquity("SPY");

            var fastPeriod = GetParameter("ema-fast", 10);
            var mediumPeriod = GetParameter("ema-medium", 30);
            var slowPeriod = GetParameter("ema-slow", 50);

            _fast = EMA("SPY", fastPeriod);
            _medium = EMA("SPY", mediumPeriod);
            _slow = EMA("SPY", slowPeriod);
        }
    }
}
```

Optimization

Deployment

Introduction

The Lean CLI supports optimizing a project's parameters on your local machine or in the cloud using LEAN's powerful optimization strategies. Optimization is helpful when you want to find the best combination of parameters to minimize or maximize a certain statistic, like the algorithm's Sharpe ratio or drawdown . If your run optimizations in the cloud, you don't need your own powerful machine or data library.

Run Local Optimizations

Follow these steps to run a local optimization:

1. Set up your local data for all the data required by your project.
2. Convert your project to use project parameters instead of constants for all values that must be optimized.
3. Open a terminal in the organization workspace that contains the project.
4. Run `lean optimize "<projectName>"` to start optimizing the project in **. / <projectName>** . This command starts an interactive wizard which lets you configure the optimizer.

```
$ lean optimize "My Project"
Select the optimization strategy to use:
1) Grid Search
2) Euler Search
Enter an option:
```

5. Enter the number of the optimization strategy to use. You can either choose for Grid Search, which runs through all possible combinations of parameters, or for Euler Search, which performs an Euler-like which gradually works towards smaller optimizations.

```
$ lean optimize "My Project"
Select the optimization strategy to use:
1) Grid Search
2) Euler Search
Enter an option: 1
```

6. Enter the number of the optimization target to use. The target specifies what statistic you want to optimize and whether you want to minimize or maximize it.

```
$ lean optimize "My Project"
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
```

```
3) Compounding Annual Return (min)
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option: 2
```

7. For each parameter, enter whether you want to optimize it and what its values can be.

```
$ lean optimize "My Project"
Should the 'ema-fast' parameter be optimized? [Y/n]: y
Minimum value for 'ema-fast': 5
Maximum value for 'ema-fast': 10
Step size for 'ema-fast' [1.0]: 1
Should the 'ema-medium' parameter be optimized? [Y/n]: y
Minimum value for 'ema-medium': 25
Maximum value for 'ema-medium': 30
Step size for 'ema-medium' [1.0]: 1
Should the 'ema-slow' parameter be optimized? [Y/n]: y
Minimum value for 'ema-slow': 45
Maximum value for 'ema-slow': 50
Step size for 'ema-slow' [1.0]: 1
```

8. Enter the constraints of the optimization. An example optimization is "Drawdown <= 0.25", which discards all parameter combinations resulting in a drawdown higher than 25%.

```
$ lean optimize "My Project"
Current constraints: None
Do you want to add a constraint? [y/N]: y
Select a constraint target:
1) Sharpe Ratio
2) Compounding Annual Return
3) Probabilistic Sharpe Ratio
4) Drawdown
Enter an option: 4
Select a constraint operator (<value> will be asked after this):
1) Less than <value>
2) Less than or equal to <value>
3) Greater than <value>
4) Greater than or equal to <value>
5) Equal to <value>
6) Not equal to <value>
Enter an option: 2
Set the <value> for the selected operator: 0.25
Current constraints: TotalPerformance.PortfolioStatistics.Drawdown <= 0.25
Do you want to add a constraint? [y/N]: n
```

After configuring the constraints the optimizer starts running.

9. View the results in the terminal after the optimizer finished. The logs contains the optimal parameter combination.

```
$ lean optimize "My Project"
20220223 18:26:20.000 TRACE:: Program.Main(): Exiting Lean...
20220223 18:26:20.079 TRACE:: LeanOptimizer.TriggerOnEndEvent(OID 2313bbba-9c71-4b9e-8b91-c70cc117b0c7):
Optimization has ended. Result for Target: ['TotalPerformance'].['PortfolioStatistics'].['SharpeRatio'] at:
3.6205: was reached using ParameterSet: (ema-slow:47,ema-medium:26,ema-fast:5) backtestId '21c30000-dc5a-4dec-
b75a-da5b1796ccba'. Constraints: (['TotalPerformance'].['PortfolioStatistics'].['Drawdown'] 'LessOrEqual' 0.25)
Optimal parameters: ema-slow: 47, ema-medium: 26, ema-fast: 5
Successfully optimized 'My Project' and stored the output in 'My Project/optimizations/2021-03-24_00-22-15'
```

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

```
$ lean optimize "My Project" --output "My Project/custom-output"
20220223 18:28:20.000 TRACE:: Program.Main(): Exiting Lean...
20220223 18:28:20.079 TRACE:: LeanOptimizer.TriggerOnEndEvent(OID 1ac5e638-aae0-4aa9-80d4-02c51bb7b84d):
Optimization has ended. Result for Target: ['TotalPerformance'].['PortfolioStatistics'].['SharpeRatio'] at:
3.6205: was reached using ParameterSet: (ema-slow:47,ema-medium:26,ema-fast:5) backtestId 'e2aa3abf-bb60-4e91-
a281-59c882ada62f'. Constraints: (['TotalPerformance'].['PortfolioStatistics'].['Drawdown'] 'LessOrEqual' 0.25)
Optimal parameters: ema-slow: 47, ema-medium: 26, ema-fast: 5
Successfully optimized 'My Project' and stored the output in 'My Project/custom-output'
```

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

Run Cloud Optimizations

Follow these steps to run a cloud optimization:

1. Log in to the CLI if you haven't done so already.
2. Convert your project to use project parameters instead of constants for all values that must be optimized.
3. Open a terminal in the organization workspace that contains the project.
4. Run `lean cloud optimize "<projectName>" --push` to push **. / <projectName>** to the cloud and start optimizing the project in the cloud.

```
$ lean cloud optimize "My Project" --push
[1/1] Pushing 'My Project'
Successfully updated cloud file 'My Project/main.py'
Started compiling project 'My Project'
Successfully compiled project 'My Project'
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
3) Compounding Annual Return (min)
```

```
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option:
```

5. Enter the number of the optimization target to use. The target specifies what statistic you want to optimize and whether you want to minimize or maximize it.

```
$ lean cloud optimize "My Project" --push
Select an optimization target:
1) Sharpe Ratio (min)
2) Sharpe Ratio (max)
3) Compounding Annual Return (min)
4) Compounding Annual Return (max)
5) Probabilistic Sharpe Ratio (min)
6) Probabilistic Sharpe Ratio (max)
7) Drawdown (min)
8) Drawdown (max)
Enter an option: 2
```

6. For each parameter, enter whether you want to optimize it and what its values can be.

```
$ lean cloud optimize "My Project" --push
Should the 'ema-fast' parameter be optimized? [Y/n]: y
Minimum value for 'ema-fast': 1
Maximum value for 'ema-fast': 10
Step size for 'ema-fast' [1.0]: 1
Should the 'ema-slow' parameter be optimized? [Y/n]: y
Minimum value for 'ema-slow': 21
Maximum value for 'ema-slow': 30
Step size for 'ema-slow' [1.0]: 1
```

7. Enter the constraints of the optimization. An example optimization is "Drawdown <= 0.25", which discards all parameter combinations resulting in a drawdown higher than 25%.

```
$ lean cloud optimize "My Project" --push
Current constraints: None
Do you want to add a constraint? [y/N]: y
Select a constraint target:
1) Sharpe Ratio
2) Compounding Annual Return
3) Probabilistic Sharpe Ratio
4) Drawdown
Enter an option: 4
Select a constraint operator (<value> will be asked after this):
1) Less than <value>
2) Less than or equal to <value>
```

```
3) Greater than <value>
4) Greater than or equal to <value>
5) Equal to <value>
6) Not equal to <value>
Enter an option: 2
Set the <value> for the selected operator: 0.25
Current constraints: TotalPerformance.PortfolioStatistics.Drawdown <= 0.25
Do you want to add a constraint? [y/N]: n
```

8. Enter the number of the optimization node type to use.

```
$ lean cloud optimize "My Project" --push
Select the optimization node type:
1) O2-8 (2 cores, 8 GB RAM) @ $0.15 per hour
2) O4-12 (4 cores, 12 GB RAM) @ $0.30 per hour
3) O8-16 (8 cores, 16 GB RAM) @ $0.60 per hour
Enter an option: 2
```

9. Enter the number of nodes that should run in parallel.

```
$ lean cloud optimize "My Project" --push
How many nodes should run in parallel (1-12) [6]: 10
```

10. Confirm the given input to start the optimizer.

```
$ lean cloud optimize "My Project" --push
Estimated number of backtests: 100
Estimated batch time: 8 minutes
Estimated batch cost: $0.38
Organization balance: 173,368 QCC ($1,733.68)
Do you want to start the optimization on the selected node type? [Y/n]: y
```

11. Inspect the optimal parameter combination and the full statistics of the backtest that ran with this combination at the bottom of the logs when the optimizer has finished.

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.

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 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 get <path/to/file>`.

```
$ lean cloud object-store get 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
```

Bulk File Upload

To upload all files from a directory to the Object Store, run the following Python script:

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

API Reference

lean backtest

Introduction

Backtest a project locally using Docker.

```
$ lean backtest <project> [options]
```

Description

Runs a local backtest in a Docker container using the quantconnect/lean Docker image. The logs of the backtest are shown in real-time and the full results are stored in the **<project> / backtest / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to backtest. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to be the algorithm to run.

If the `--debug` option is given, this command configures the Docker container in such a way to allow debugging using your editor's debugger. The exact ways to get local debugging to work depends on your editor and language, see Debugging for more information on how to set this up.

You can use the `--data-provider-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:

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:

--data-provider-historical	Required Options
AlphaVantage	--alpha-vantage-api-key
	--alpha-vantage-price-plan
IEX	--iex-cloud-api-key
	--iex-price-plan
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Local	N/A
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

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
`<project>`	The path to the project directory or algorithm file to backtest.

Options

The `lean backtest` command supports the following options:

Option	Description
`--output <path>`	Directory to store results in (defaults to **<project> / backtests / <timestamp>**).
`--detach`, `-d`	Run the backtest in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--debug`	Enable a certain debugging method, see Debugging for more information.
`--data-provider-historical`	The historical data source.
`--iqfeed-iqconnect <path>`	The path to your IQConnect binary.
`--iqfeed-username <value>`	Your IQFeed username.
`--iqfeed-password <value>`	Your IQFeed password.
`--iqfeed-version <value>`	The product version of your IQFeed developer account.
`--iqfeed-host`	The IQFeed host address.
`--polygon-api-key <value>`	Your Polygon.io API Key.
`--iex-cloud-api-key`	Your IEX Cloud API Key.
`--iex-price-plan`	Your IEX Cloud price plan. `Launch`, `Grow`, or `Enterprise`.
`--alpha-vantage-api-key`	Your Alpha Vantage API Key.
`--alpha-vantage-price-plan`	Your Alpha Vantage price plan. `Free`, `Plan30`, `Plan75`, `Plan150`, `Plan300`, `Plan600`, or `Plan1200`.
`--coinapi-api-key`	
`--coinapi-product`	

`--terminal-link-connection-type <value>`	The Terminal Link connection type, which must be `SAPI` or `DAPI` .
`--terminal-link-server-auth-id <value>`	Your unique user identifier (UUID). The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--download-data`	Update the Lean configuration file to download data from the QuantConnect API, alias for `--data-provider-historical QuantConnect` .
`--data-purchase-limit <value>`	The maximum amount of QCC to spend on downloading data during the backtest when using QuantConnect as data provider.
`--release`	Compile C# projects in release configuration instead of debug.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--python-venv <value>`	The path of the python virtual environment to use.
`--update`	Pull the LEAN engine image before running the backtest.
`--backtest-name`	
`--extra-docker-config`	
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean backtest` command and exit.

API Reference

lean build

Introduction

Build Docker images of your own version of LEAN.

```
$ lean build [root] [options]
```

Description

Builds local Docker images of a local version of LEAN and sets them up for local usage with the CLI. After running this command, all commands that run the LEAN engine or the research environment use your custom images. This command performs the following actions:

1. The `lean-cli/foundation:latest` image is built from **Lean / DockerfileLeanFoundation** (if you're using an AMD64-based system) or **Lean / DockerfileLeanFoundationARM** (if you're using an ARM64-based system).
2. LEAN is compiled in a Docker container using the `lean-cli/foundation:latest` image.
3. The `lean-cli/engine:latest` image is built from **Lean / Dockerfile** using `lean-cli/foundation:latest` as the base image.
4. The `lean-cli/research:latest` image is built from **Lean / DockerfileJupyter** using `lean-cli/engine:latest` as the base image.
5. The default engine image is set to `lean-cli/engine:latest` .
6. The default research image is set to `lean-cli/research:latest` .

When the foundation Dockerfile is the same as the one used for the official foundation image, step 1 is skipped and `quantconnect/lean:foundation` is used instead of `lean-cli/foundation:latest` .

Arguments

The `lean build` command expects the following arguments:

Argument	Description
`<lean>`	The path to the directory containing the LEAN repository. Defaults to the current working directory.

Options

The `lean build` command supports the following options:

Option	Description
`--tag <value>`	The tag to apply to custom images (defaults to `latest`).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean build` command and exit.

API Reference

lean cloud backtest

Introduction

Backtest a project in the cloud.

```
$ lean cloud backtest <project> [options]
```

Description

Runs a backtest for a cloud project. While running the backtest, a progress bar shows to keep you up-to-date on the status of the backtest. After running the backtest, the resulting statistics and a link to the full results on QuantConnect are logged. You can use the `--open` option to automatically open the full results in the browser after the backtest has finished.

If you have a local copy of the cloud project, you can use the `--push` option to push local modifications to the cloud before running the backtest.

Arguments

The `lean cloud backtest` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project for which to run a backtest.

Options

The `lean cloud backtest` command supports the following options:

Option	Description
`--name <value>`	The name of the backtest (a random one is generated if not specified).
`--push`	Push local modifications to the cloud before running the backtest.
`--open`	Automatically open the results in the browser when the backtest is finished.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud backtest` command and exit.

API Reference

lean cloud live

Introduction

Interact with the QuantConnect Cloud live deployments. Alias for lean cloud live deploy .

```
$ lean cloud live [command] <project> [options]
```

Commands

The `lean cloud live` command expects the following commands:

Argument	Description
`deploy`	Start live trading for a project in the cloud.
`liquidate`	Stops live trading and liquidates existing positions for a certain project.
`stop`	Stops live trading for a certain project without liquidating existing positions.

The default command is `deploy` , and `lean cloud live` becomes an alias for `lean cloud live deploy` .

Options

The `lean cloud live` command supports the following options:

Option	Description
`--help`	Display the help text of the `lean cloud live` command and exit.

API Reference

lean cloud live deploy

Introduction

Start live trading for a project in the cloud.

```
$ lean cloud live deploy <project> [options]
```

Description

Starts live trading for a cloud project. Before starting live trading, the CLI shows an interactive wizard letting you configure the brokerage, 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:

`--brokerage`	Required Options
`"Paper Trading"`	N/A
`Binance`	`--binance-exchange-name`
	`--binance-api-key` or `--binanceus-api-key`
	`--binance-api-secret` or `--binanceus-api-secret`
	`--binance-use-testnet`
`Bitfinex`	`--bitfinex-api-key`
	`--bitfinex-api-secret`
`Bybit`	`--bybit-api-key`
	`--bybit-api-secret`
	`--bybit-vip-level`

"Coinbase Advanced Trade"	--coinbase-api-key
	--coinbase-api-secret
"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-broker
	--terminal-link-openfigi-api-key
Tradier	--tradier-account-id
	--tradier-access-token
	--tradier-environment

"Trading Technologies"	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key
	--tt-rest-app-secret
	--tt-rest-environment
	--tt-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, seperate them with a comma. The order you select them in defines the order of precedence.

--data-provider-live	Required Options
Binance	--binance-exchange-name
	--binance-api-key or --binanceus-api-key
	--binance-api-secret or --binanceus-api-secret
Bitfinex	All options required by --brokerage Bitfinex .
Bybit	All options required by --brokerage Bybit .
"Coinbase Advanced Trade"	--coinbase-api-key
	--coinbase-api-secret
IEX	--iex-cloud-api-key
	--iex-price-plan
"Interactive Brokers"	All options required by --brokerage "Interactive Brokers" .
Kraken	All options required by --brokerage Kraken .
	--oanda-account-id

`Oanda`	
	`--oanda-access-token`
`Polygon`	`--polygon-api-key`
`QuantConnect`	N/A
`Samco`	All options required by `--brokerage Samco`.
`TDAmeritrade`	All options required by `--brokerage TDAmeritrade`.
`"Terminal Link"`	All options required by `--brokerage "Terminal Link"`.
`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.

--data-provider-historical	Required Options
AlphaVantage	--alpha-vantage-api-key
	--alpha-vantage-price-plan
IEX	--iex-cloud-api-key
	--iex-price-plan
Polygon	--polygon-api-key
QuantConnect	N/A

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
--brokerage <value>	The brokerage to use when running in non-interactive mode.
--data-provider-live <value>	The live data source.
--data-provider-historical <value>	The historical data source.
--ib-user-name <value>	Your Interactive Brokers username (example: trader777).

`--ib-account <value>`	Your Interactive Brokers account Id (example: DU1234567).
`--ib-password <value>`	Your Interactive Brokers password.
`--ib-weekly-restart-utc-time`	
`--tradier-account-id <value>`	Your Tradier account id, which you can find on your Settings > API Access page on the Tradier website.
`--tradier-access-token <value>`	Your Tradier access token.
`--tradier-environment <value>`	`live` to use the live environment or `paper` to use the developer sandbox.
`--oanda-account-id <value>`	Your OANDA account id, which you can find on your Account Statement page on the OANDA website.
`--oanda-access-token <value>`	Your OANDA API token, which you can generate on the Manage API Access page on the OANDA website.
`--oanda-environment <value>`	`Practice` to trade on fxTrade Practice or `Trade` to trade on fxTrade.
`--bitfinex-api-key <value>`	Your Bitfinex API key, which you can generate on the API Management page on the Bitfinex website.
`--bitfinex-api-secret <value>`	Your Bitfinex API secret.
`--coinbase-api-key <value>`	Your Coinbase API key, which you can generate on the API settings page on the Coinbase website.
`--coinbase-api-secret <value>`	Your Coinbase API secret.
`--binance-exchange-name <value>`	`Binance` , `BinanceUS` , `Binance-USDM-Futures` , or `Binance-COIN-Futures`
`--binance-api-key <value>`	Your Binance API key, which you can generate on the API Management page on the Binance website.
`--binanceus-api-key <value>`	Your Binance US API key, which you can generate on the API Management page on the Binance US website.
`--binance-api-secret <value>`	Your Binance API secret.
`--binanceus-api-secret <value>`	Your Binance US API secret.
`--binance-use-testnet <value>`	`live` to use the production environment or `paper` to use the testnet.
`--zerodha-api-key <value>`	Your Kite Connect API key.
`--zerodha-access-token <value>`	Your Zerodha access token.
`--zerodha-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.

`--zerodha-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--zerodha-history-subscription <boolean>`	Whether you have a history API subscription for Zerodha.
`--samco-client-id <value>`	Your Samco account Client ID.
`--samco-client-password <value>`	Your Samco account password.
`--samco-year-of-birth <value>`	Your year of birth (YYYY) registered with Samco.
`--samco-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--samco-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--terminal-link-server-auth-id <value>`	Your unique user identifier (UUID). The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.
`--terminal-link-environment <value>`	The environment to run in. `Production` or `Beta` .
`--terminal-link-server-host <value>`	The public IP address of the SAPI AWS server.
`--terminal-link-server-port <value>`	The port where SAPI is listening. The default port is 8194.
`--terminal-link-emsx-broker <value>`	The EMSX broker to use.
`--terminal-link-emsx-account`	
`--terminal-link-openfigi-api-key <value>`	The OpenFIGI API key to use for mapping options.
`--tt-user-name <value>`	Your Trading Technologies username.
`--tt-session-password <value>`	Your Trading Technologies session password.
`--tt-account-name <value>`	Your Trading Technologies account name.
`--tt-rest-app-key <value>`	Your Trading Technologies REST app key.
`--tt-rest-app-secret <value>`	Your Trading Technologies REST app secret.
`--tt-rest-environment <value>`	The REST environment in which to run.
`--tt-order-routing-sender-comp-id <value>`	The order routing sender comp Id to use.
`--kraken-api-key <value>`	Your Kraken API key, which you can find on the API Management Settings page on the Kraken website.
`--kraken-api-secret <value>`	Your Kraken API secret.
	Your Kraken verification tier (`Starter` , `Intermediate` , or

`--kraken-verification-tier <value>`	`Pro`). For more information about verification tiers, see Verification levels explained on the Kraken website.
`--tdameritrade-api-key <value>`	Your TDAmeritrade API key.
`--tdameritrade-access-token <value>`	Your TDAmeritrade OAuth Access Token.
`--tdameritrade-account-number <value>`	Your TDAmeritrade account number.
`--bybit-api-key <value>`	Your Bybit API key, which you can generate by following the How to Create Your API Key page on the Bybit website.
`--bybit-api-secret <value>`	Your Bybit API secret.
`--bybit-vip-level <value>`	Your Bybit VIP level (`VIP0` , `VIP1` , `VIP2` , `VIP3` , `VIP4` , `VIP5` , `SupremeVIP` , `Pro1` , `Pro2` , `Pro3` , `Pro4` , or `Pro5`). For more information about the levels, see FAQ — Bybit VIP Program on the Bybit website.
`--polygon-api-key <value>`	Your Polygon API Key.
`--iex-cloud-api-key`	Your IEX Cloud API Key.
`--iex-price-plan`	Your IEX Cloud price plan. `Launch` , `Grow` , or `Enterprise` .
`--coinapi-api-key`	
`--coinapi-product`	
`--node <value>`	The name or Id of the live node to run on.
`--auto-restart <boolean>`	Whether automatic algorithm restarting must be enabled.
`--notify-order-events <boolean>`	Whether notifications must be sent for order events.
`--notify-insights <boolean>`	Whether notifications must be sent for emitted insights.
`--notify-emails <email> <subject>`	A comma-separated list of "email:subject" pairs configuring email-notifications.
`--notify-webhooks <url> <headers>`	A comma-separated list of "url:HEADER_1=VALUE_1:HEADER_2=VALUE_2:etc" pairs configuring webhook-notifications.
`--notify-sms <value>`	A comma-separated list of phone numbers configuring SMS notifications.
`--notify-telegram <value>`	A comma-separated list of "user/group Id:token(optional)" pairs configuring telegram notifications.
`--live-cash-balance <value>`	A comma-separated list of "currency:amount" pairs that define the initial cash balance.
`--live-holdings <value>`	A comma-separated list of "symbol:symbolId:quantity:averagePrice" pairs that define the initial portfolio holdings. For example, `"GOOG:GOOCV VP83T1ZUHROL:10:50.0"` .

`--push`	Push local modifications to the cloud before starting live trading.
`--open`	Automatically open the live results in the browser once the deployment starts.
`--show-secrets`	
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live deploy` command and exit.

API Reference

lean cloud live liquidate

Introduction

Stop live trading and liquidate existing positions for a certain project.

```
$ lean cloud live liquidate <project> [options]
```

Description

Arguments

The `lean cloud live liquidate` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to liquidate.

Options

The `lean cloud live liquidate` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live liquidate` command and exit.

API Reference

lean cloud live stop

Introduction

Stop live trading a certain project without liquidating existing positions.

```
$ lean cloud live stop <project> [options]
```

Description

Arguments

The `lean cloud live stop` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to stop live trading.

Options

The `lean cloud live stop` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud live stop` command and exit.

API Reference

lean cloud object-store delete

Introduction

Delete a key-value pair from the organization's Object Store in QuantConnect Cloud.

```
$ lean cloud object-store delete <key> [options]
```

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
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud object-store delete` command and exit.

API Reference

lean cloud object-store get

Introduction

Get the metadata of a file (including the size, timestamp of last modification, name, and a preview of the content) in the organization's Object Store in QuantConnect Cloud.

```
$ lean cloud object-store get <key> [options]
```

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
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud object-store get` command and exit.

API Reference

lean cloud object-store list

Introduction

List the metadata (including the key, size, and file name) for the items of a directory in the organization's Object Store in QuantConnect Cloud.

```
$ lean cloud object-store list <key> [options]
```

Arguments

The `lean cloud object-store list` command expects the following arguments:

Argument	Description
(Optional) `<key>`	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
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud object-store list` command and exit.

API Reference

lean cloud object-store ls

Introduction

List the metadata (including the key, size, and file name) for the items of a directory in the organization's Object Store in QuantConnect Cloud. Alias for `lean cloud object-store list`.

```
$ lean cloud object-store ls <key> [options]
```

Arguments

The `lean cloud object-store ls` command expects the following arguments:

Argument	Description
(Optional) `<key>`	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
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud object-store ls` command and exit.

API Reference

lean cloud object-store set

Introduction

Uploads a file to the organization's Object Store in QuantConnect Cloud under a given key.

```
$ lean cloud object-store set <key> <path> [options]
```

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
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud object-store set` command and exit.

API Reference

lean cloud optimize

Introduction

Optimize a project in the cloud.

```
$ lean cloud optimize <project> [options]
```

Description

Runs an optimization for a cloud project. While running the optimization, a progress bar shows to keep you up-to-date on the status of the optimization. After running the optimization, the optimal parameters and the statistics of the backtest with the optimal parameters are logged.

By default, an interactive wizard is shown, letting you configure the target, the parameters, the constraints, the node type, and the number of parallel nodes. When `--target` is given the command runs in non-interactive mode and does not prompt for input or confirmation.

When `--target` is given, the optimizer configuration is read from the command-line options. This means the `--target` , `--target-direction` , `--parameter` , `--node` , and `--parallel-nodes` options become required. Additionally, you can also use `--constraint` to specify optimization constraints.

In non-interactive mode, the parameters can be configured using the `--parameter` option. This option takes the following values: the name of the parameter, its minimum value, its maximum value, and its step size. You can provide this option multiple times to configure multiple parameters.

In non-interactive mode, the constraints can be configured using the `--constraint` option. This option takes a "statistic operator value" string as value, where the statistic must be a path to a property in a backtest's output file, like "TotalPerformance.PortfolioStatistics.SharpeRatio". This statistic can also be shortened to "SharpeRatio" or "Sharpe Ratio", in which case, the command automatically converts it to the longer version. The value must be a number and the operator must be `<` , `>` , `<=` , `>=` , `==` , or `==` . You can provide this option multiple times to configure multiple constraints.

Example non-interactive usage:

```
$ lean cloud optimize "My Project" \
    --target "Sharpe Ratio" \
    --target-direction "max" \
    --parameter my-first-parameter 1 10 0.5 \
    --parameter my-second-parameter 20 30 5 \
    --constraint "Drawdown < 0.5" \
    --constraint "Sharpe Ratio >= 1" \
    --node O4-12 \
    --parallel-nodes 12 \
    --push
```

If you have a local copy of the cloud project, you can use the `--push` option to push local modifications to the cloud before

starting the optimization.

Arguments

The `lean cloud optimize` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the project to optimize.

Options

The `lean cloud optimize` command supports the following options:

Option	Description
`--target <value>`	The path to the property in the backtest's output file to target in non-interactive mode, like "TotalPerformance.PortfolioStatistics.SharpeRatio". May also be a shortened version, like "SharpeRatio" or "Sharpe Ratio".
`--target-direction <value>`	`min` if the target must be minimized, `max` if it must be maximized.
`--parameter <name> <min> <max> <step>`	The 'parameter min max step' pairs configuring the parameters to optimize. May be used multiple times.
`--constraint <value>`	The 'statistic operator value' pairs configuring the constraints of the optimization. May be used multiple times.
`--node <value>`	The node to optimize on, must `O2-8` , `O4-12` , or `O8-16` .
`--parallel-nodes <value>`	The number of nodes to run in parallel.
`--name <value>`	The name of the optimization (a random one is generated if not specified).
`--push`	Push local modifications to the cloud before starting the optimization.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud optimize` command and exit.

API Reference

lean cloud pull

Introduction

Pull projects from QuantConnect to the local drive.

```
$ lean cloud pull [options]
```

Description

Pulls projects from QuantConnect to your local directory while preserving the directory structure of your projects on QuantConnect. The project's files, description, and parameters are pulled from the cloud. By default, all cloud projects are pulled from the organization that's linked to your current organization workspace . If you provide a `--project` option, you only pull a single project from the cloud.

Before pulling a cloud project, the CLI checks if the local directory with the same path already exists. If it does and the local directory is not linked to the cloud project (because of an earlier `lean cloud pull` or `lean cloud push`), the CLI skips pulling the cloud project and logs a descriptive warning message.

If you have a local copy of a project when you pull it from the cloud, local files that don't exist in the cloud are not deleted, but the configuration values of your cloud project overwrite the configuration values of the local version . If you have renamed the project in the cloud, when you pull the project from the cloud, the local project is renamed to match the name of the cloud project.

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

Options

The `lean cloud pull` command supports the following options:

Option	Description
`--project <value>`	The name or Id of the cloud project to pull (all your cloud projects in the organization if not specified).
`--pull-bootcamp`	Pull Boot Camp projects (disabled by default).
`--encrypt`	Encrypt the local project files that you pull from QuantConnect Cloud.
`--decrypt`	Decrypt the local project files that you pull from QuantConnect Cloud.
`--key <path>`	Path to file that contains the encryption key to use.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud pull` command and exit.

API Reference

lean cloud push

Introduction

Push local projects to QuantConnect.

```
$ lean cloud push [options]
```

Description

Pushes local projects to QuantConnect while preserving the directory structure of your local projects. The project's 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
`--project <path>`	Path to the local project to push (all local projects in your current organization workspace if not specified).
`--encrypt`	Encrypt the cloud project files that you push to QuantConnect Cloud.
`--decrypt`	Decrypt the cloud project files that you push to QuantConnect Cloud.
`--key <path>`	Path to file that contains the encryption key to use.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud push` command and exit.

API Reference

lean cloud status

Introduction

Show the live trading status of a project in the cloud.

```
$ lean cloud status <project> [options]
```

Description

Shows the live status of a cloud project. Displays the project id, project name, project URL, and live status. If the project has been deployed before, it also shows the deployment id, results URL, brokerage, and when it launched. If the project has been stopped, it also shows when it was stopped and the error that caused it, if there is one.

Arguments

The `lean cloud status` command expects the following arguments:

Argument	Description
`<project>`	The name or Id of the cloud project for which to show the status.

Options

The `lean cloud status` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean cloud status` command and exit.

API Reference

lean config get

Introduction

Get the current value of a configurable option.

```
$ lean config get <key> [options]
```

Description

Prints the value of the requested option or aborts if the value is not set.

This command doesn't print the values of credentials for security reasons. Open the **credentials** file in your global configuration directory to see your stored credentials.

Arguments

The `lean config get` command expects the following arguments:

Argument	Description
`<key>`	The key of the value to retrieve. Run `lean config list` to get a list of all available keys.

Options

The `lean config get` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config get` command and exit.

API Reference

lean config list

Introduction

List the configurable options and their current values.

```
$ lean config list [options]
```

Description

Displays a table containing all configurable options and their current values. Credentials are masked with asterisks for security reasons.

Options

The `lean config list` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config list` command and exit.

API Reference

lean config set

Introduction

Set a configurable option.

```
$ lean config set <key> <value> [options]
```

Description

Updates the value of a configurable option in your global configuration directory. The command aborts with a descriptive error message if the given value is invalid for the given key.

The following keys can be set:

Key	Description
`user-id`	The user Id used when making authenticated requests to the QuantConnect API.
`api-token`	The API token used when making authenticated requests to the QuantConnect API.
`default-language`	The default language used when creating new projects (must be either `python` or `csharp`).
`engine-image`	The Docker image used when running the LEAN engine (`quantconnect/lean:latest` if not set).
`research-image`	The Docker image used when running the research environment (`quantconnect/research:latest` if not set).

Arguments

The `lean config set` command expects the following arguments:

Argument	Description
`<key>`	The key of the option to update.
`<value>`	The new value for the option with the given key.

Options

The `lean config set` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config set` command and exit.

API Reference

lean config unset

Introduction

Unset a configurable option.

```
$ lean config unset <key> [options]
```

Description

Unsets the value of a configurable option in your global configuration directory. The command aborts with a descriptive error message if the given value is invalid for the given key.

Arguments

The `lean config unset` command expects the following arguments:

Argument	Description
`<key>`	The key of the option to unset.

Options

The `lean config unset` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean config unset` command and exit.

API Reference

lean create-project

Introduction

Create a new project containing starter code. Alias for `lean project-create` .

```
$ lean create-project <name> [options]
```

Description

Creates a new project with some basic starter code. If the language is set to `python` , this generates a **main.py** file, a Python-based research notebook, a project configuration file, and editor configuration files for PyCharm and VS Code.

If the language is set to `csharp` this generates a **Main.cs** file, a C#-based research notebook, a project configuration file, and editor configuration files for Visual Studio, Rider, and VS Code.

A full list of the created files can be found on the Projects > Structure page.

If no `--language` is given, the default language saved in the global configuration is used. You can update the default language to Python by running `lean config set default-language python` or to C# by running `lean config set default-language csharp` .

If the given project name contains slashes, the name is parsed as a path and the project is created in a subdirectory. Any subdirectories that don't exist yet are created automatically.

Arguments

The `lean 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
`--language <value>`	The language of the project to create, must be either `python` or `csharp` .
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean create-project` command and exit.

API Reference

lean data download

Introduction

Purchase and download data from QuantConnect Datasets.

```
$ lean data download [options]
```

Description

Lets you purchase and download data from QuantConnect Datasets. This command performs the following actions:

1. An interactive wizard is shown allowing you to configure exactly which data you want to download.
2. You are asked to accept the CLI API Access and Data Agreement in your web browser.
3. You are asked to confirm the purchase one last time.
4. The CLI downloads the requested files while charging the organization that's linked to your current organization workspace.

The available datasets and their pricing can be found in QuantConnect Datasets . Data is priced on a per-file per-download basis, meaning you pay a certain number of QuantConnect Credits (QCC) for each file you download. The required QCC is deducted from your organization's QCC balance when a file is downloaded. If you force-quit the command before it downloaded all requested files, you are not charged for the files you didn't download.

When `--dataset` is given, the command runs in non-interactive mode and several steps in the preceding list are skipped. Instead, the command reads the downloading configuration from the given command-line options. In this mode, the `--dataset` option is required, as well as all options specific to the selected dataset.

Example non-interactive usage:

```
$ lean data download \
    --dataset "Bitfinex Crypto Price Data" \
    --data-type "Trade" \
    --ticker "BTCUSDT" \
    --resolution "Daily" \
    --start "20201208" \
    --end "20221208"
```

In case the local data already exists, a warning is logged and you are given the choice of whether you want to enable overwriting existing data or not. Use the `--overwrite` flag to override this behavior and enable overwriting existing data in all such cases.

To automatically confirm payment confirmation prompts, use the `--yes` flag.

Options

The `lean data download` command supports the following options:

Option	Description
`--dataset <value>`	The name of the dataset to download data from in non-interactive mode.
`--overwrite`	Overwrite existing local data.
`-y` / `--yes`	Automatically confirm payment confirmation prompts.
`--lean-config`	
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean data download` command and exit.

API Reference

lean data generate

Introduction

Generate realistic market data.

```
$ lean data generate [options]
```

Description

Runs the random data generator in the LEAN ToolBox to generate realistic market data using a Brownian motion model. Requires `--start <yyyyMMdd>` and `--symbol-count <amount>` to be set. The rest of the options have default values.

If `--end <yyyyMMdd>` isn't set, data is generated from the start date until the current date. If the `--end` option is set, data is generated between the given `--start` and `--end` values (inclusive).

By default, dense data is generated, which means the generated data contains at least one data point per resolution step. You can use `--data-density Sparse` to change this to at least one data point per 5 resolution steps, or `--data-density VerySparse` to change it to at least one data point per 50 resolution steps.

If the security type is set to `Equity` , this command will automatically generate map files, factor files, and coarse universe data. To not generate coarse universe data, set the `--include-coarse` option to `false` .

The following combinations of security types and resolutions are supported:

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

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>` . The image is pulled before running the random data generator if it doesn't exist locally yet or if you pass the `--update` flag.

Options

The `lean data generate` command supports the following options:

Option	Description
`--start <yyyyMMdd>`	The inclusive start date for the data to generate in `yyyyMMdd` format. Must be at least `19980101` .
`--end <yyyyMMdd>`	The inclusive end date for the data to generate in `yyyyMMdd` format (defaults to today).
`--symbol-count <value>`	The number of symbols for which to generate data. This value is ignored if you provide the `--tickers` option.
`--tickers <value>`	Comma-separated list of tickers to use for generating data.
`--security-type <value>`	The security type for which to generate data (defaults to `Equity`). Must be `Equity` , `Forex` , `Cfd` , `Future` , `Crypto` or `Option` .
`--resolution <value>`	The resolution of the generated data (defaults to `Minute`). Must be `Tick` , `Second` , `Minute` , `Hour` or `Daily` . See the description for the supported combinations of security types and resolutions.
`--data-density <value>`	The density of the generated data (defaults to `Dense`). Must be `Dense` (at least one data point per resolution step), `Sparse` (at least one data point per 5 resolution steps), or `VerySparse` (at least one data point per 50 resolution steps).
`--include-coarse <true/false>`	Whether coarse universe data must be generated for Equity data (defaults to `true`).
`--market <market>`	The market for which to generate data. This option defaults to LEAN's default market for the requested security type.
`--quote-trade-ratio <value>`	The ratio of generated quotes to generated trades (defaults to 1). Values larger than 1 mean more quotes than trades. This option is only used for Option, Future and Crypto.
`--random-seed <value>`	The random number generator seed. If you set a value, it must be a non-negative integer. If you don't set a value, the genrator won't use a random seed.
`--ipo-percentage <value>`	The probability each generated Equity will have an IPO event (defaults to 5.0). Note this is not the total probability for all generated symbols. This option is only used for Equity.
`--rename-percentage <value>`	The probability each generated Equity will have a rename event (defaults to 30.0). Note that this is not the total probability for all generated symbols. This option is only used for Equity.
`--splits-percentage <value>`	The probability each generated Equity will have a stock split event (defaults to 15.0). Note that this is not the total probability for all generated symbols. This option is only used for Equity.
`--dividends-percentage <value>`	The probability each generated Equity will have dividends (defaults to 60.0). Note that this is not the total probability for all generated symbols. This option is only used for

	Equity.
`--dividend-every-quarter-percentage <value>`	The probability each generated Equity will have a dividend event every quarter (defaults to 30.0). Note that this is not the total probability for all generated symbols. This option is only used for Equity.
`--option-price-engine <value>`	The pricing engine to run calculations for Options (defaults to `BaroneAdesiWhaleyApproximationEngine`).
`--volatility-model-resolution <value>`	The volatility model period span (defaults to `Daily`). Must be `Tick` , `Second` , `Minute` , `Hour` , or `Daily`
`--chain-symbol-count <value>`	The size of the Option chain (defaults to 10). This value must be a non-negative integer.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the generator.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean data generate` command and exit.

API Reference

lean decrypt

Introduction

Decrypt your local project using the specified decryption key.

```
$ lean decrypt [OPTIONS] PROJECT
```

Options

The `lean decrypt` command supports the following options:

Option	Description
`--key FILE`	Path to file that contains the encryption key to use.
`--verbose`	Enable debug logging
`--help`	Display the help text of the `lean decrypt` command and exit.

API Reference

lean delete-project

Introduction

Delete a project on your local machine and in the cloud. Alias for `lean project-delete` .

```
$ lean delete-project <project> [options]
```

Description

Deletes a project from your local machine and in the cloud. If you are a collaborator on the project, this command doesn't delete the project for the other collaborators, but it removes you as a collaborator.

Options

The `lean delete-project` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean delete-project` command and exit.

API Reference

lean encrypt

Introduction

Encrypt your local project using the specified encryption key.

```
$ lean encrypt [OPTIONS] PROJECT
```

Options

The `lean encrypt` command supports the following options:

Option	Description
`--key FILE`	Path to file that contains the encryption key to use.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean encrypt` command and exit.

API Reference

lean init

Introduction

Scaffold a Lean configuration file and data directory.

```
$ lean init [options]
```

Description

Fills the current directory with all the files needed to get going. It'll create a Lean configuration file and a data directory containing some sample data. To view a full list of the created files, see Directory Structure .

Options

The `lean init` command supports the following options:

Option	Description
`--organization &ltvalue>`	The name or Id of the organization to link with the organization workspace.
`--language &ltvalue>` , `-l &ltvalue>`	The default language of new projects (`python` or `csharp`).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean init` command and exit.

API Reference

lean library add

Introduction

Add a custom library to a project.

```
$ lean library add <project> <name> [options]
```

Description

Adds a third-party or project library to a project so you can use it in local backtesting, local live trading, local optimizations, and the local research environment. Additionally, this command updates your local environment so you get autocomplete on the libraries.

C# libraries are added to your C# project file (the file ending in **.csproj**). If `dotnet` is on your `PATH` and `--no-local` is not given, the CLI also restores all dependencies using `dotnet restore` to make local autocomplete work.

Third-party Python libraries are added to your project's **requirements.txt** file. If `pip` is on your `PATH` and `--no-local` is not given, the CLI also installs the Python package in your local Python environment to make local autocomplete work.

If `--version` is not given, the package is pinned to the latest compatible version. For C# projects, this is the latest available version. For Python projects, this is the latest version compatible with Python 3.8 (which is what the Docker images use).

If `--version` is given and the project is a Python project, the CLI will additionally check whether the given version is compatible with Python 3.6. If this is not the case, the command aborts because libraries incompatible with Python 3.8 cannot be installed in the official Docker images.

If 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
`--version <value>`	The version of the package to add to the project (defaults to the latest compatible version).
`--no-local`	Skip making changes to the local environment.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean library add` command and exit.

API Reference

lean library remove

Introduction

Remove a custom library from a project.

```
$ lean library remove <project> <name> [options]
```

Description

Removes a library from a project. This command can remove libraries that are added using `lean library add` , as well as libraries that were manually added to the C# project file or a Python project's **requirements.txt** file.

C# libraries are removed from your C# project file (the file ending in **.csproj**). If `dotnet` is on your `PATH` and `--no-local` is not given, the CLI also restores all dependencies using `dotnet restore` .

Third-party Python libraries are removed from your project's **requirements.txt** file.

Project libraries are removed from your project configuration file .

Arguments

The `lean library remove` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory.
`<name>`	For third-party C# libraries, the name of the NuGet package to remove. For third-party Python libraries, the name of the PyPI package to remove. For project libraries, the path to the library in the **Library** directory of your organization workspace .

Options

The `lean library remove` command supports the following options:

Option	Description
`--no-local`	Skip making changes to the local environment.
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean library remove` command and exit.

API Reference

lean live

Introduction

Interact with the local live deployments using Docker. Alias for `lean live deploy`.

```
$ lean live [command] <project> [options]
```

Commands

The `lean live` command expects the following commands:

Argument	Description
`add-security`	Represents a command to add a security to the algorithm.
`cancel-order`	Represents a command to cancel a specific order by id.
`deploy`	Start live trading a project locally using Docker.
`liquidate`	Liquidate the given symbol from the latest deployment of the given project.
`stop`	Stop an already running local live trading project.
`submit-order`	Represents a command to submit an order to the algorithm.
`update-order`	Represents a command to update a specific order by id.

The default command is `deploy`, and `lean live` becomes an alias for `lean live deploy`.

Options

The `lean cloud live` command supports the following options:

Option	Description
`--help`	Display the help text of the `lean cloud live` command and exit.

API Reference

lean live add-security

Introduction

Add a security subscription to a local live algorithm.

```
$ lean live add-security <project> [options]
```

Description

Arguments

The `lean live add-security` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to which you want to add the security.

Options

The `lean live add-security` command supports the following options:

Option	Description
`--ticker <value>`	The ticker of the symbol to add.
`--market <value>`	The market of the symbol to add.
`--security-type <value>`	The security type of the symbol to add.
`--resolution <value>`	The resolution of the symbol to add. Defaults to `"Minute"`.
`--fill-data-forward <boolean>`	The fill forward behavior. `true` to fill forward or `false` to not fill forward. Defaults to `true`.
`--leverage <value>`	The leverage for the security. Defaults to 2 for Equity, 50 for Forex, and 1 for everything else.
`--extended-market-hours <boolean>`	The extended market hours flag. `true` to allow pre/post market data or `false` for only in market data. Defaults to `false`
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live add-security` command and exit.

API Reference

lean live cancel-order

Introduction

Cancel an order with a specific Id in a local live trading project.

```
$ lean live cancel-order <project> [options]
```

Description

Arguments

The `lean live cancel-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file that contains the order you want to cancel.

Options

The `lean live cancel-order` command supports the following options:

Option	Description
`--order-id <value>`	The Id of the order to cancel.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live cancel-order` command and exit.

API Reference

lean live deploy

Introduction

Start live trading a project locally using Docker.

```
$ lean live deploy <project> [options]
```

Description

Starts local live trading in a Docker container using the quantconnect/lean Docker image. The logs of the algorithm are shown in real-time and the full results are stored in the **<project> / live / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to backtest. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to contain the algorithm to run.

By default, an interactive wizard is shown letting you configure the brokerage and data 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:

`--brokerage`	Required Options
`"Paper Trading"`	N/A
`Binance`	`--binance-exchange-name`
	`--binance-api-key` or `--binanceus-api-key`
	`--binance-api-secret` or `--binanceus-api-secret`
	`--binance-use-testnet`
`Bitfinex`	`--bitfinex-api-key`

Bitfinex	--bitfinex-api-secret
Bybit	--bybit-api-key
	--bybit-api-secret
	--bybit-vip-level
	--bybit-use-testnet
Coinbase	--coinbase-api-key
	--coinbase-api-secret
"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-connection-type
	--terminal-link-environment
	--terminal-link-server-host

"Terminal Link"	--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
Tradier	--tradier-account-id
	--tradier-access-token
	--tradier-environment
"Trading Technologies"	--tt-user-name
	--tt-session-password
	--tt-account-name
	--tt-rest-app-key
	--tt-rest-app-secret
	--tt-rest-environment
	--tt-market-data-sender-comp-id
	--tt-market-data-target-comp-id
	--tt-market-data-host
	--tt-market-data-port
	--tt-order-routing-sender-comp-id
	--tt-order-routing-target-comp-id
	--tt-order-routing-host
	--tt-order-routing-port
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, seperate them with a comma. The order you select them in defines the order of precedence.

--data-provider-live	Required Options
Binance	--binance-exchange-name
	--binance-api-key or --binanceus-api-key
	--binance-api-secret or --binanceus-api-secret
Bitfinex	All options required by --brokerage Bitfinex .
Bybit	All options required by --brokerage Bybit .
"Coinbase Advanced Trade"	--coinbase-api-key
	--coinbase-api-secret
"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	--oanda-account-id
	--oanda-access-token
Polygon	--polygon-api-key
Samco	All options required by --brokerage Samco .
TDAmeritrade	All options required by --brokerage TDAmeritrade .
"Terminal Link"	All options required by --brokerage "Terminal Link" .
Tradier	--tradier-account-id
	--tradier-access-token
	--tt-user-name

	--tt-user-name
	--tt-session-password
	--tt-account-name
"Trading Technologies"	--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.

--data-provider-historical	Required Options
AlphaVantage	--alpha-vantage-api-key
	--alpha-vantage-price-plan
IEX	--iex-cloud-api-key
	--iex-price-plan
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Local	N/A
Polygon	--polygon-api-key
QuantConnect	N/A

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
`<project>`	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
`--environment <value>`	The name of the environment in the Lean configuration file to use.
`--output <path>`	Directory to store results in (defaults to **<project> / live / <timestamp>**).
`--detach` , `-d`	Run the live deployment in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--brokerage <value>`	The brokerage to use when running in non-interactive mode.
`--data-provider-live`	The live data source.
`--data-provider-historical`	The historical data source.
`--ib-user-name <value>`	Your Interactive Brokers username (example: trader777).
`--ib-account <value>`	Your Interactive Brokers account Id (example: DU1234567).
`--ib-password <value>`	Your Interactive Brokers password.

`--ib-weekly-restart-utc-time`	
`--tradier-account-id <value>`	Your Tradier account id, which you can find on your Settings > API Access page on the Tradier website.
`--tradier-access-token <value>`	Your Tradier access token.
`--tradier-environment <value>`	`live` to use the live environment or `paper` to use the developer sandbox.
`--oanda-account-id <value>`	Your OANDA account id, which you can find on your Account Statement page on the OANDA website.
`--oanda-access-token <value>`	Your OANDA API token, which you can generate on the Manage API Access page on the OANDA website.
`--oanda-environment <value>`	`Practice` to trade on fxTrade Practice or `Trade` to trade on fxTrade.
`--bitfinex-api-key <value>`	Your Bitfinex API key, which you can generate on the API Management page on the Bitfinex website.
`--bitfinex-api-secret <value>`	Your Bitfinex API secret.
`--coinbase-api-key <value>`	Your Coinbase API key, which you can generate on the API settings page on the Coinbase website.
`--coinbase-api-secret <value>`	Your Coinbase API secret.
`--binance-exchange-name <value>`	`Binance` , `BinanceUS` , `Binance-USDM-Futures` , or `Binance-COIN-Futures`
`--binance-api-key <value>`	Your Binance API key, which you can generate on the API Management page on the Binance website.
`--binanceus-api-key <value>`	Your Binance US API key, which you can generate on the API Management page on the Binance US website.
`--binance-api-secret <value>`	Your Binance API secret.
`--binanceus-api-secret <value>`	Your Binance US API secret.
`--binance-use-testnet <value>`	`live` to use the production environment or `paper` to use the testnet.
`--zerodha-api-key <value>`	Your Kite Connect API key.
`--zerodha-access-token <value>`	Your Zerodha access token.
`--zerodha-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--zerodha-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--zerodha-history-subscription <boolean>`	Whether you have a history API subscription for Zerodha.

`--samco-client-id <value>`	Your Samco account Client ID.
`--samco-client-password <value>`	Your Samco account password.
`--samco-year-of-birth <value>`	Your year of birth (YYYY) registered with Samco.
`--samco-product-type <value>`	The product type, which must be `mis` if you are targeting intraday products, `cnc` if you are targeting delivery products, or `nrml` if you are targeting carry forward products.
`--samco-trading-segment <value>`	The trading segment, which must be `equity` if you are trading equities on NSE or BSE, or `commodity` if you are trading commodities on MCX.
`--terminal-link-connection-type <value>`	The Terminal Link connection type, which must be `SAPI` or `DAPI` .
`--terminal-link-server-auth-id <value>`	Your unique user identifier (UUID). The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-emsx-broker <value>`	The EMSX broker to use.
`--terminal-link-emsx-account <value>`	The EMSX account to use (optional).
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--tt-user-name <value>`	Your Trading Technologies username.
`--tt-session-password <value>`	Your Trading Technologies session password.
`--tt-account-name <value>`	Your Trading Technologies account name.
`--tt-rest-app-key <value>`	Your Trading Technologies REST app key.
`--tt-rest-app-secret <value>`	Your Trading Technologies REST app secret.
`--tt-rest-environment <value>`	The REST environment in which to run.
`--tt-market-data-sender-comp-id <value>`	The market data sender comp Id to use.
`--tt-market-data-target-comp-id <value>`	The market data target comp Id to use.
`--tt-market-data-host <value>`	The host of the market data server.
`--tt-market-data-port <value>`	The port of the market data server.

`--tt-order-routing-sender-comp-id <value>`	The order routing sender comp Id to use.
`--tt-order-routing-target-comp-id <value>`	The order routing target comp Id to use.
`--tt-order-routing-host <value>`	The host of the order routing server.
`--tt-order-routing-port <value>`	The port of the order routing server.
`--tt-log-fix-messages <boolean>`	Whether FIX messages should be logged.
`--kraken-api-key <value>`	Your Kraken API key, which you can find on the API Management Settings page on the Kraken website.
`--kraken-api-secret <value>`	Your Kraken API secret.
`--kraken-verification-tier <value>`	Your Kraken verification tier (`Starter` , `Intermediate` , or `Pro`). For more information about verification tiers, see Verification levels explained on the Kraken website.
`--tdameritrade-api-key <value>`	Your TDAmeritrade API key.
`--tdameritrade-access-token <value>`	Your TDAmeritrade OAuth Access Token.
`--tdameritrade-account-number <value>`	Your TDAmeritrade account number.
`--bybit-api-key <value>`	Your Bybit API key, which you can generate by following the How to Create Your API Key page on the Bybit website.
`--bybit-api-secret <value>`	Your Bybit API secret.
`--bybit-vip-level <value>`	Your Bybit VIP level (`VIP0` , `VIP1` , `VIP2` , `VIP3` , `VIP4` , `VIP5` , `SupremeVIP` , `Pro1` , `Pro2` , `Pro3` , `Pro4` , or `Pro5`). For more information about the levels, see FAQ — Bybit VIP Program on the Bybit website.
`--bybit-use-testnet <value>`	`live` to use the production environment or `paper` to use the Bybit Demo Trading environment.
`--ib-enable-delayed-streaming-data <boolean>`	Whether delayed data may be used when your algorithm subscribes to a security for which you don't have a market data subscription.
`--iqfeed-iqconnect <path>`	The path to your IQConnect binary.
`--iqfeed-username <value>`	Your IQFeed username.
`--iqfeed-password <value>`	Your IQFeed password.
`--iqfeed-version <value>`	The product version of your IQFeed developer account.
`--iqfeed-host`	The IQFeed host address.
`--polygon-api-key <value>`	Your Polygon API Key.
`--iex-cloud-api-key`	Your IEX Cloud API Key.

`--iex-price-plan`	Your IEX Cloud price plan. `Launch` , `Grow` , or `Enterprise` .
`--coinapi-api-key`	
`--coinapi-product`	
`--alpha-vantage-api-key`	Your Alpha Vantage API Key.
`--alpha-vantage-price-plan`	Your Alpha Vantage price plan. `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` .
`--release`	Compile C# projects in release configuration instead of debug.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--python-venv <value>`	The path of the python virtual environment to use.
`--live-cash-balance <value>`	A comma-separated list of currency:amount pairs that define the initial cash balance.
`--live-holdings <value>`	A comma-separated list of "symbol:symbolId:quantity:averagePrice" pairs that define the initial portfolio holdings. For example, `"GOOG:GOOCV VP83T1ZUHROL:10:50.0"` .
`--update`	Pull the LEAN engine image before starting live trading.
`--show-secrets`	
`--extra-docker-config`	
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live deploy` command and exit.

API Reference

lean live liquidate

Introduction

Liquidate a specific symbol from the latest local deployment of a project.

```
$ lean live liquidate <project> [options]
```

Description

Arguments

The `lean live liquidate` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to liquidate.

Options

The `lean live update-order` command supports the following options:

Option	Description
`--ticker <value>`	The ticker of the symbol to liquidate.
`--market <value>`	The market of the symbol to liquidate.
`--security-type <value>`	The security type of the symbol to liquidate.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live liquidate` command and exit.

API Reference

lean live stop

Introduction

Stop a local live trading algorithm.

```
$ lean live stop <project> [options]
```

Description

Arguments

The `lean live stop` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file to stop live trading.

Options

The `lean live stop` command supports the following options:

Option	Description
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live stop` command and exit.

API Reference

lean live submit-order

Introduction

Submit an order in a local live trading project.

```
$ lean live submit-order <project> [options]
```

Description

Arguments

The `lean live submit-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file in which you want to submit the order.

Options

The `lean live submit-order` command supports the following options:

Option	Description
`--ticker <value>`	The ticker for the order.
`--market <value>`	The market for the order.
`--security-type <value>`	The security type for the order.
`--order-type <value>`	The type of the order.
`--quantity <value>`	The number of units to be ordered (directional).
`--limit-price <value>`	The limit price of the order.
`--stop-price <value>`	The stop price of the order.
`--tag <value>`	The order tag.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live submit-order` command and exit.

API Reference

lean live update-order

Introduction

Update an order with a specific Id in a local live trading project.

```
$ lean live update-order <project> [options]
```

Description

Arguments

The `lean live update-order` command expects the following arguments:

Argument	Description
`<project>`	The path to the project directory or algorithm file that contains the order you want to update.

Options

The `lean live update-order` command supports the following options:

Option	Description
`--order-id <value>`	The Id of the order to update.
`--quantity <value>`	The number of units to be updated (directional).
`--limit-price <value>`	The limit price of the order to be updated.
`--stop-price <value>`	The stop price of the order to be updated.
`--tag <value>`	The new order tag.
`--lean-config <value>`	The Lean configuration file that should be used (defaults to the nearest **lean.json**).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean live update-order` command and exit.

API Reference

lean login

Introduction

Log in with a QuantConnect account.

```
$ lean login [options]
```

Description

Lets you log in with your QuantConnect API credentials and stores the given values in the **credentials** file in your global configuration directory.

If `--user-id` or `--api-token` is not provided, an interactive prompt is shown and the missing values are read from stdin.

You can request your user Id and API token on your Account page.

Options

The `lean login` command supports the following options:

Option	Description
`-u, --user-id <value>`	The QuantConnect user Id to log in with.
`-t, --api-token <value>`	The QuantConnect API token to log in with.
`--show-secrets`	
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean login` command and exit.

API Reference

lean logout

Introduction

Log out and remove stored credentials.

```
$ lean logout [options]
```

Description

Removes the credentials stored in the **credentials** file in your global configuration directory.

Options

The `lean logout` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean logout` command and exit.

API Reference

lean logs

Introduction

Display the most recent backtest/live/optimization logs.

```
$ lean logs [options]
```

Description

Displays the most recent backtest/live/optimization logs. By default, the most recent backtest logs are shown unless `--live` or `--optimization` is given. You can pass in a project with `--project <directory>` to display the most recent logs from a specific project.

Options

The `lean logs` command supports the following options:

Option	Description
`--backtest`	Display the most recent backtest logs (default).
`--live`	Display the most recent live logs.
`--optimization`	Display the most recent optimization logs.
`--project <directory>`	The project of which to show the most recent logs of.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean logs` command and exit.

API Reference

lean object-store delete

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store delete [OPTIONS]
```

Options

The `lean object-store delete` command supports the following options:

Option	Description
`--verbose`	Enable debug logging
`--help`	Show this message and exit.

API Reference

lean object-store get

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store get [OPTIONS]
```

Options

The `lean object-store get` command supports the following options:

Option	Description
`--verbose`	Enable debug logging
`--help`	Show this message and exit.

API Reference

lean object-store list

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store list [OPTIONS]
```

Options

The `lean object-store list` command supports the following options:

Option	Description
`--verbose`	Enable debug logging
`--help`	Show this message and exit.

API Reference

lean object-store ls

Introduction

Alias for 'list'

```
$ lean object-store ls [OPTIONS]
```

Options

The `lean object-store ls` command supports the following options:

Option	Description
`--verbose`	Enable debug logging
`--help`	Show this message and exit.

API Reference

lean object-store set

Introduction

Opens the local storage directory in the file explorer.

```
$ lean object-store set [OPTIONS]
```

Options

The `lean object-store set` command supports the following options:

Option	Description
`--verbose`	Enable debug logging
`--help`	Show this message and exit.

API Reference

lean optimize

Introduction

Optimize a project's parameters locally using Docker.

```
$ lean optimize <project> [options]
```

Description

Runs a local optimization in a Docker container using the quantconnect/lean Docker image. The logs of the optimizer are shown in real-time and the full results of the optimizer and all executed backtests are stored in the **<project> / optimizations / <timestamp>** directory. You can use the `--output` option to change the output directory.

The given `<project>` argument must be either a project directory or a file containing the algorithm to optimize. If it is a project directory, the CLI looks for a **main.py** or **Main.cs** file, assuming the first file it finds to contain the algorithm to optimize.

By default, an interactive wizard is shown letting you configure the optimizer. When `--optimizer-config` or `--strategy` is given, the command runs in non-interactive mode and doesn't prompt for input.

When the `--optimizer-config <config file>` option is given, the specified config file is used. This option must point to a file containing a full optimizer config (the `algorithm-type-name` , `algorithm-language` and `algorithm-location` properties may be omitted). See the Optimizer.Launcher / config.example.json file in the LEAN repository for an example optimizer configuration file, which also contains documentation on all the required properties.

When `--strategy` is given, the optimizer configuration is read from the command-line options. This means the `--strategy` , `--target` , `--target-direction` , and `--parameter` options become required. Additionally, you can also use `--constraint` to specify optimization constraints.

In non-interactive mode, the parameters can be configured using the `--parameter` option. This option takes the following values: the name of the parameter, its minimum value, its maximum value, and its step size. You can provide this option multiple times to configure multiple parameters.

In non-interactive mode, the constraints can be configured using the `--constraint` option. This option takes a "statistic operator value" string as value, where the statistic must be a path to a property in a backtest's output file, like "TotalPerformance.PortfolioStatistics.SharpeRatio". This statistic can also be shortened to "SharpeRatio" or "Sharpe Ratio", in which case the command automatically converts it to the longer version. The value must be a number and the operator must be `<` , `>` , `<=` , `>=` , `==` , or `==` . You can provide this option multiple times to configure multiple constraints.

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:

--data-provider-historical	Required Options
AlphaVantage	--alpha-vantage-api-key
	--alpha-vantage-price-plan
IEX	--iex-cloud-api-key
	--iex-price-plan
IQFeed	--iqfeed-iqconnect
	--iqfeed-username
	--iqfeed-password
	--iqfeed-version
	--iqfeed-host
Local	N/A
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

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
`<project>`	The path to the project directory or algorithm file to optimize.

Options

The `lean optimize` command supports the following options:

Option	Description
`--output <path>`	Directory to store results in (defaults to **<project> / optimizations / <timestamp>**).
`--detach` , `-d`	Run the optimization in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--optimizer-config <path>`	The optimizer configuration file that should be used (the `algorithm-type-name` , `algorithm-language` and `algorithm-location` properties may be omitted). See the Optimizer.Launcher / config.example.json file in the LEAN repository for an example optimizer config file.
`--strategy <value>`	The optimization strategy to use in non-interactive mode. Must be `Grid Search` or `Euler Search` .
`--target <value>`	The path to the property in the backtest's output file to target, like "TotalPerformance.PortfolioStatistics.SharpeRatio". May also be a shortened version, like "SharpeRatio" or "Sharpe Ratio".
`--target-direction <value>`	`min` if the target must be minimized, `max` if it must be

	maximized.
`--parameter <name> <min> <max> <step>`	The 'parameter min max step' pairs configuring the parameters to optimize. May be used multiple times.
`--constraint <value>`	The 'statistic operator value' pairs configuring the constraints of the optimization. May be used multiple times.
`--data-provider-historical`	The historical data source.
`--download-data`	Update the Lean configuration file to download data from the QuantConnect API, alias for `--data-provider-historical QuantConnect` .
`--data-purchase-limit`	
`--release`	Compile C# projects in release configuration instead of debug.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the optimizer.
`--estimate`	Estimate optimization runtime without running it.
`--max-concurrent-backtests <value>`	Maximum number of concurrent backtests to run (x >= 1).
`--extra-docker-config`	
`--no-update`	Use the local LEAN engine image instead of pulling the latest version.
`--iqfeed-iqconnect <path>`	The path to your IQConnect binary.
`--iqfeed-username <value>`	Your IQFeed username.
`--iqfeed-password <value>`	Your IQFeed password.
`--iqfeed-version <value>`	The product version of your IQFeed developer account.
`--iqfeed-host`	The IQFeed host address.
`--polygon-api-key <value>`	Your Polygon.io API Key.
`--iex-cloud-api-key`	Your IEX Cloud API Key.
`--iex-price-plan`	Your IEX Cloud price plan. `Launch` , `Grow` , or `Enterprise` .
`--alpha-vantage-api-key`	Your Alpha Vantage API Key.
`--alpha-vantage-price-plan`	Your Alpha Vantage price plan. `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` .
`--coinapi-api-key`	
`--coinapi-product`	

`--terminal-link-connection-type <value>`	The Terminal Link connection type, which must be `SAPI` or `DAPI` .
`--terminal-link-server-auth-id <value>`	Your unique user identifier (UUID). The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .
`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-openfigi-api-key <value>`	The Open FIGI API key to use for mapping Options.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean optimize` command and exit.

API Reference

lean project-create

Introduction

Create a new project containing starter code.

```
$ lean project-create <name> [options]
```

Description

Creates a new project with some basic starter code. If the language is set to `python` , this generates a **main.py** file, a Python-based research notebook, a project configuration file, and editor configuration files for PyCharm and VS Code.

If the language is set to `csharp` this generates a **Main.cs** file, a C#-based research notebook, a project configuration file, and editor configuration files for Visual Studio, Rider, and VS Code.

A full list of the created files can be found on the Projects > Structure page.

If no `--language` is given, the default language saved in the global configuration is used. You can update the default language to Python by running `lean config set default-language python` or to C# by running `lean config set default-language csharp` .

If the given project name contains slashes, the name is parsed as a path and the project is created in a subdirectory. Any subdirectories that don't exist yet are created automatically.

Arguments

The `lean project-create` command expects the following arguments:

Argument	Description
`<name>`	The name of the project to create. This name may contain slashes to create a project in a subdirectory. The project name must only contain `-` , `_` , letters, numbers, and spaces. The project name can't start with a space or be any of the following reserved names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, or LPT9. If the project is a Python library, the library name can only contain letters (a-z), numbers (0-9), and underscores (_). Python library names can't contain spaces or start with a number.

Options

The `lean project-create` command supports the following options:

Option	Description
`--language <value>`	The language of the project to create, must be either `python` or `csharp` .
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean project-create` command and exit.

API Reference

lean project-delete

Introduction

Delete a project on your local machine and in the cloud.

```
$ lean project-delete <project> [options]
```

Description

Deletes a project from your local machine and in the cloud. If you are a collaborator on the project, this command doesn't delete the project for the other collaborators, but it removes you as a collaborator.

Options

The `lean project-delete` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean project-delete` command and exit.

API Reference

lean report

Introduction

Generate a report of a backtest or live trading algorithm.

```
$ lean report [options]
```

Description

Runs the LEAN Report Creator in a Docker container using the quantconnect/lean Docker image. By default, this command generates a report of the most recent backtest. This behavior can be overridden by using `--backtest-results <path>` and providing the path to the backtest results JSON file. If `--live-results <path>` is also given, the generated report will contain both the backtest and the live results.

The `--strategy-name` , `--strategy-version` , and `--strategy-description` options can be used to set the name, version, and description that are shown in the report. The name and version are shown in the top-right corner of each page, while the description is shown on the top of the first page. These fields are blank by default.

When the given backtest results are stored in a project directory or one of its subdirectories, the default name is the name of the project directory and the default description is the description in the project's **config.json** file.

By default, the official LEAN engine image is used. You can override this using the `--image <value>` option. Alternatively, you can set the default engine image for all commands using `lean config set engine-image <value>` . The image is pulled before running the report creator if it doesn't exist locally yet or if you pass the `--update` flag.

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
`--backtest-results <path>`	The path to the JSON file containing the backtest results. Defaults to the most recent backtest results file in the current working directory or one of its subdirectories.
`--live-results <path>`	The path to the JSON file containing the live trading results.
`--report-destination <path>`	The path where the generated report is stored as HTML (defaults to **./report.html**).
`--css`	Path where the CSS override file is stored.
`--html`	Path where the custom HTML template file is stored.
`--detach`, `-d`	Run the report creator in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--strategy-name <value>`	The name of the strategy. It will appear at the top-right corner of each page.
`--strategy-version <value>`	The version of the strategy. It will appear next to the project name.
`--strategy-description <value>`	The description of the strategy. It will appear under the **Strategy Description** section on the first page of the report.
`--overwrite`	Overwrite the given `--report-destination` if it already contains a file.
`--image <value>`	The LEAN engine image to use (defaults to `quantconnect/lean:latest`).
`--update`	Pull the LEAN engine image before running the report creator.
`--pdf`	Create a PDF version along with the HTML version of the report.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean report` command and exit.

API Reference

lean research

Introduction

Run a Jupyter Lab environment locally using Docker.

```
$ lean research <project> [options]
```

Description

Runs a local Jupyter Lab environment in a Docker container using the quantconnect/research Docker image. The project directory is mounted in the Docker container and the Jupyter Lab instance is exposed on a local port. After the Jupyter Lab instance has started, the browser automatically opens.

By default, Jupyter Lab is exposed on port 8888. To use a custom port, you can use the `--port` option, which is required to run two Jupyter Lab instances side-by-side.

You can use the `--data-provider-historical` option to change where data is retrieved. This option updates the Lean configuration file , so you don't need to use this option multiple times for the same data provider if you are not switching between them. If you use the The following table shows the available data providers and their required options in non-interactive mode:

`--data-provider-historical`	Required Options
`AlphaVantage`	`--alpha-vantage-api-key`
	`--alpha-vantage-price-plan`
`IEX`	`--iex-cloud-api-key`
	`--iex-price-plan`
`IQFeed`	`--iqfeed-iqconnect`
	`--iqfeed-username`
	`--iqfeed-password`
	`--iqfeed-version`
	`--iqfeed-host`
`Local`	N/A
`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`

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
`--port <value>`	The port to run Jupyter Lab on (defaults to 8888).
`--data-provider-historical`	The historical data source.
`--iqfeed-iqconnect <path>`	The path to your IQConnect binary.
`--iqfeed-username <value>`	Your IQFeed username.
`--iqfeed-password <value>`	Your IQFeed password.
`--iqfeed-version <value>`	The product version of your IQFeed developer account.
`--iqfeed-host`	The IQFeed host address.
`--polygon-api-key <value>`	Your Polygon.io API Key
`--iex-cloud-api-key`	Your IEX Cloud API Key.
`--iex-price-plan`	Your IEX Cloud price plan. `Launch` , `Grow` , or `Enterprise` .
`--alpha-vantage-api-key`	Your Alpha Vantage API Key.
`--alpha-vantage-price-plan`	Your Alpha Vantage price plan. `Free` , `Plan30` , `Plan75` , `Plan150` , `Plan300` , `Plan600` , or `Plan1200` .
`--coinapi-api-key`	
`--coinapi-product`	
`--terminal-link-connection-type <value>`	The Terminal Link connection type, which must be `SAPI` or `DAPI` .
`--terminal-link-server-auth-id <value>`	Your unique user identifier (UUID). The UUID is a unique integer identifier that's assigned to each Bloomberg Anywhere user. If you don't know your UUID, contact Bloomberg.
`--terminal-link-environment <value>`	The environment to run in, which must be `Production` or `Beta` .

`--terminal-link-server-host <value>`	The host on which the Terminal Link server is running.
`--terminal-link-server-port <value>`	The port on which the Terminal Link server is running.
`--terminal-link-openfigi-api-key <value>`	The OpenFIGI API key to use for mapping Options.
`--download-data`	Update the Lean configuration file to download data from the QuantConnect API, alias for `--data-provider-historical QuantConnect`.
`--data-purchase-limit <value>`	The maximum amount of QCC to spend on downloading data during the research session when using QuantConnect as data provider.
`--detach`, `-d`	Run Jupyter Lab in a detached Docker container and return immediately. The name of the Docker container is shown before the command ends. You can use Docker's own commands to manage the detached container.
`--no-open`	Don't open the Jupyter Lab environment in the browser after starting it.
`--image <value>`	The LEAN research image to use (defaults to `quantconnect/research:latest`).
`--update`	Pull the LEAN research image before starting the research environment.
`--extra-docker-config`	
`--no-update`	Use the current LEAN research image.
`--lean-config <path>`	The Lean configuration file that should be used (defaults to the nearest **lean.json** file).
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean research` command and exit.

API Reference

lean whoami

Introduction

Display who is logged in.

```
$ lean whoami [options]
```

Description

Displays the name and the email address of the user who is currently logged in or "You are not logged in" if no-one is logged in.

Options

The `lean whoami` command supports the following options:

Option	Description
`--verbose`	Enable debug logging.
`--help`	Display the help text of the `lean whoami` command and exit.