

Installation/Deployment Manual for UnitPylot

Table of Contents

1. [Introduction](#)
 2. [Prerequisites](#)
 3. [Installation Process](#)
 4. [Deploying to the Marketplace](#)
 5. [Launching the Extension](#)
 6. [Example Codebases](#)
 7. [Steps to get started with UnitPylot](#)
-

1. Introduction

This document outlines the steps required to install/deploy **UnitPylot**. Follow these instructions to ensure a smooth installation/deployment process.

2. Prerequisites

- **npm** installed.
- Access to the Visual Studio Code Marketplace.
- Access to Copilot: To use any GitHub Copilot extension in Visual Studio Code, you need either an active Copilot subscription (such as Copilot Pro, Copilot Enterprise, or Copilot Business).
- Visual Studio Code should be installed.
- **Copilot in Visual Studio Code**: Follow this if not yet set-up - [GitHub Copilot Setup Guide](#)

2.1 Overview

1. Make sure you have the GitHub Copilot activated in VS Code Follow the link above if you need assistance.
 2. Add our extension by searching for "**UnitPylot**" in the **VS Code Extension Marketplace** or from Source.
 3. Download an **example codebase** as below to try out our extension!
-

3. Installation Process

3.1 From the Marketplace

1. Open VS Code.

2. Search for **UnitPylot** in the VS Code Marketplace.
3. Click Install to add the extension.

3.2 Build From Source

To build UnitPylot from source, follow these steps:

1. Clone the repository:

```
git clone https://github.com/ucl-syseng-tools-for-vscode/UnitPylot.git
cd UnitPylot
```

2. Install the dependencies:

```
npm install
```

3. Compile the extension:

```
npm run compile
```

4. Open the project in Visual Studio Code:

```
code .
```

Please note you may encounter something similar to the image below during the `npm install` step. However, this can be ignored.

```
64 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (4 moderate, 1 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

4. Deploying to the Marketplace

Publishing the Extension From Source Code

Please note that UnitPylot is already available on the VS Code Extension Marketplace. However, if you need to publish an extension like UnitPylot, follow these instructions:

To package the extension:

1. Run `vsce package` This should create a .vsix file which can be installed in VS Code.

To publish the extension:

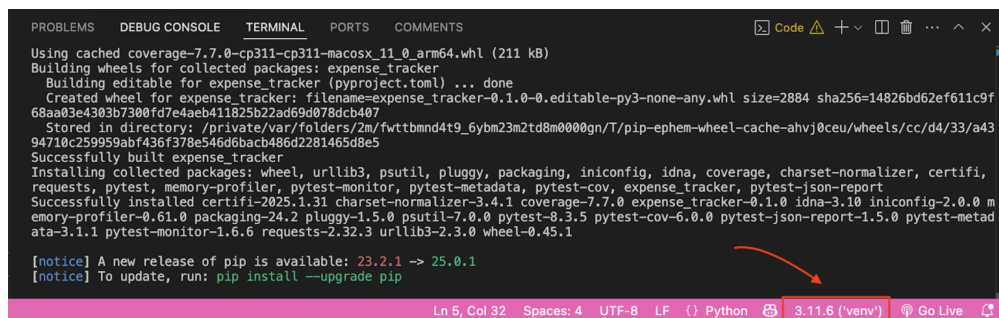
1. Link the azure devops organisation using your personal access token by running `vsce login <publisher_id>`
2. Run `vsce publish` This should push the extension to the marketplace.

For further and more comprehensive instructions, please use this link: [Publishing an Extension](#)

5. Launching the Extension

To begin using UnitPylot, follow these steps:

1. Press `F5` OR open the **Command Palette (Shift + Command + P)** and run **Debug: Start Debugging**.
2. Open one of the projects within the **example-codebases** folder.
3. Run the `make.sh` file to create a **virtual environment (venv)** to run the project within OR ensure that you have the necessary dependencies installed by running: `pip install pytest pytest-cov pytest-json-report pytest-monitor`.



```

PROBLEMS  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
Using cached coverage-7.7.0-cp311-cp311-macosx_11_0_arm64.whl (211 kB)
Building wheels for collected packages: expense_tracker
  Building editable for expense_tracker (pyproject.toml) ... done
  Created wheel for expense_tracker: filename=expense_tracker-0.1.0-0.editable-py3-none-any.whl size=2884 sha256=14826bd62ef611c9f68aa03e4303b7300fd7e4aeb411825b22ad69d078dcb407
  Stored in directory: /private/var/folders/2m/fwtbtbmd4t9_6ybm23m2td8m0000gn/T/pip-ephem-wheel-cache-ahvj0ceu/wheels/cc/d4/33/a4394710c259959abf436f378e546d6bacb486d2281465d8e5
Successfully built expense_tracker
Installing collected packages: wheel, urllib3, psutil, pluggy, packaging, iniconfig, idna, coverage, charset-normalizer, certifi, requests, pytest, memory-profiler, pytest-monitor, pytest-metadata, pytest-cov, expense_tracker, pytest-json-report
Successfully installed certifi-2025.1.31 charset-normalizer-3.4.1 coverage-7.7.0 expense_tracker-0.1.0 idna-3.10 iniconfig-2.0.0 memory-profiler-0.61.0 packaging-24.2 pluggy-1.5.0 psutil-7.0.0 pytest-8.3.5 pytest-cov-6.0.0 pytest-json-report-1.5.0 pytest-metadata-3.1.1 pytest-monitor-1.6.6 requests-2.32.3 urllib3-2.3.0 wheel-0.45.1

[notice] A new release of pip is available: 23.2.1 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
Ln 5, Col 32  Spaces: 4  UTF-8  LF  ( ) Python  3.11.6 ('venv')  Go Live

```

Ensure that the project is running in the created venv after running the `make.sh` bash script.

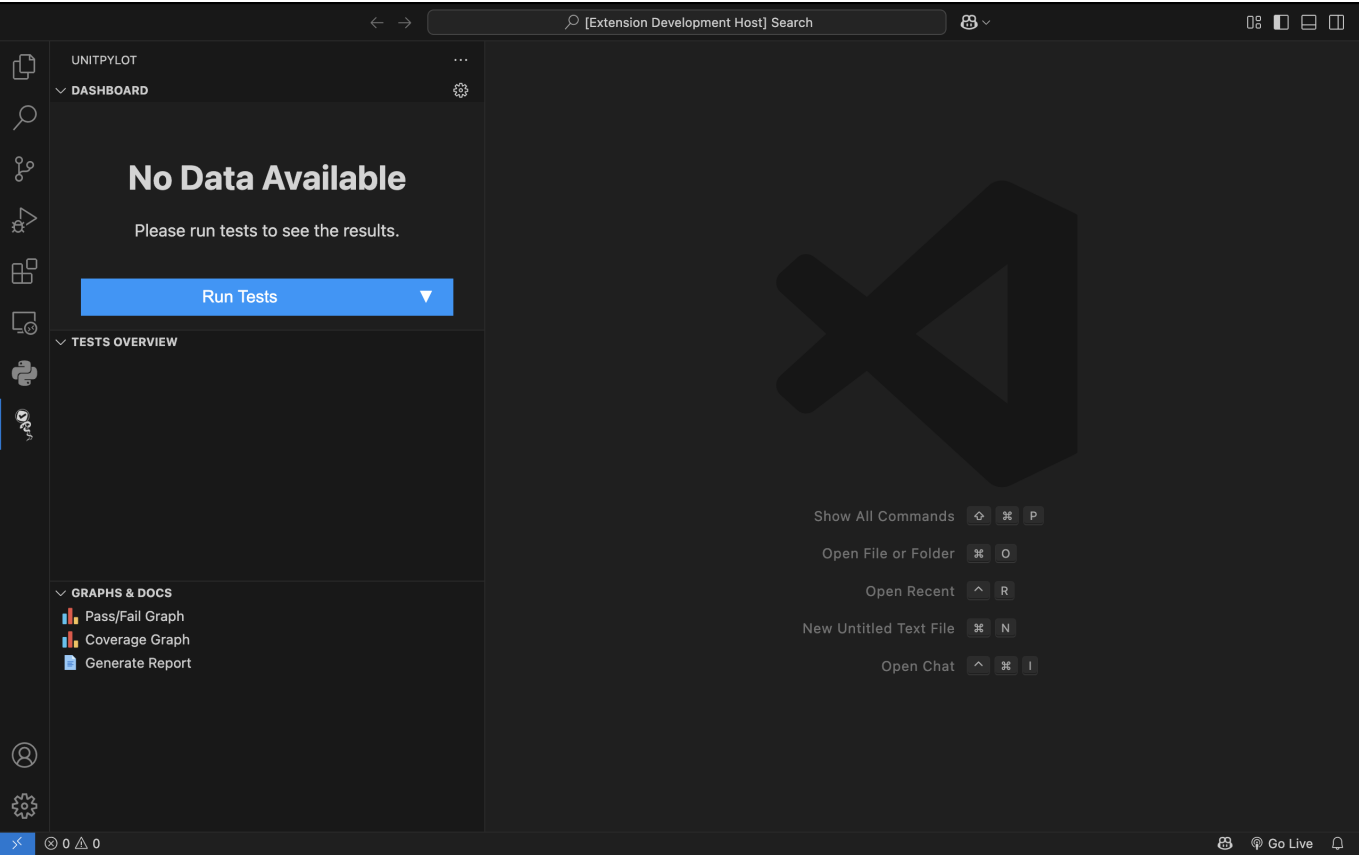
6. Testing out the Extension: Clone the example-codebases Repository

You can use some of our **example codebases** to see UnitPylot in action. Clone this repo below to do so.

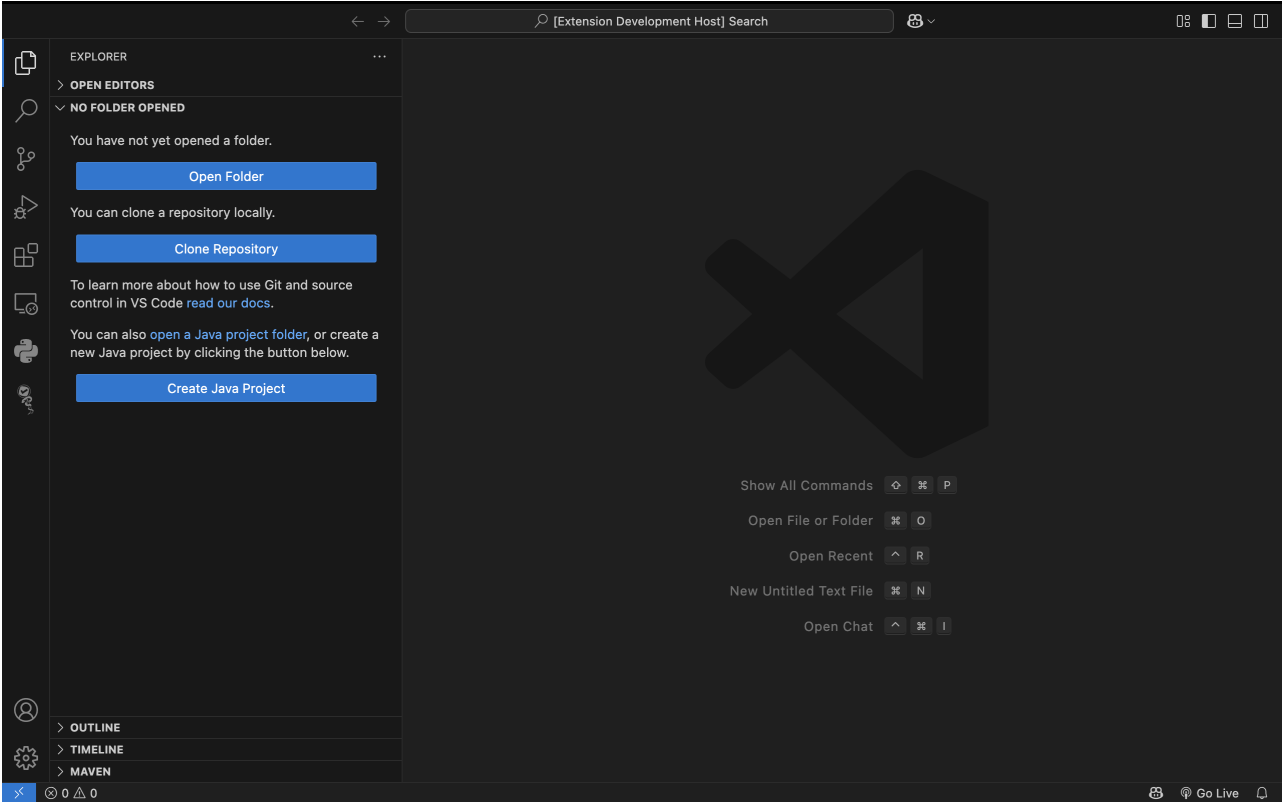
```
git clone https://github.com/ucl-syseng-tools-for-vscode/example-codebases.git
```

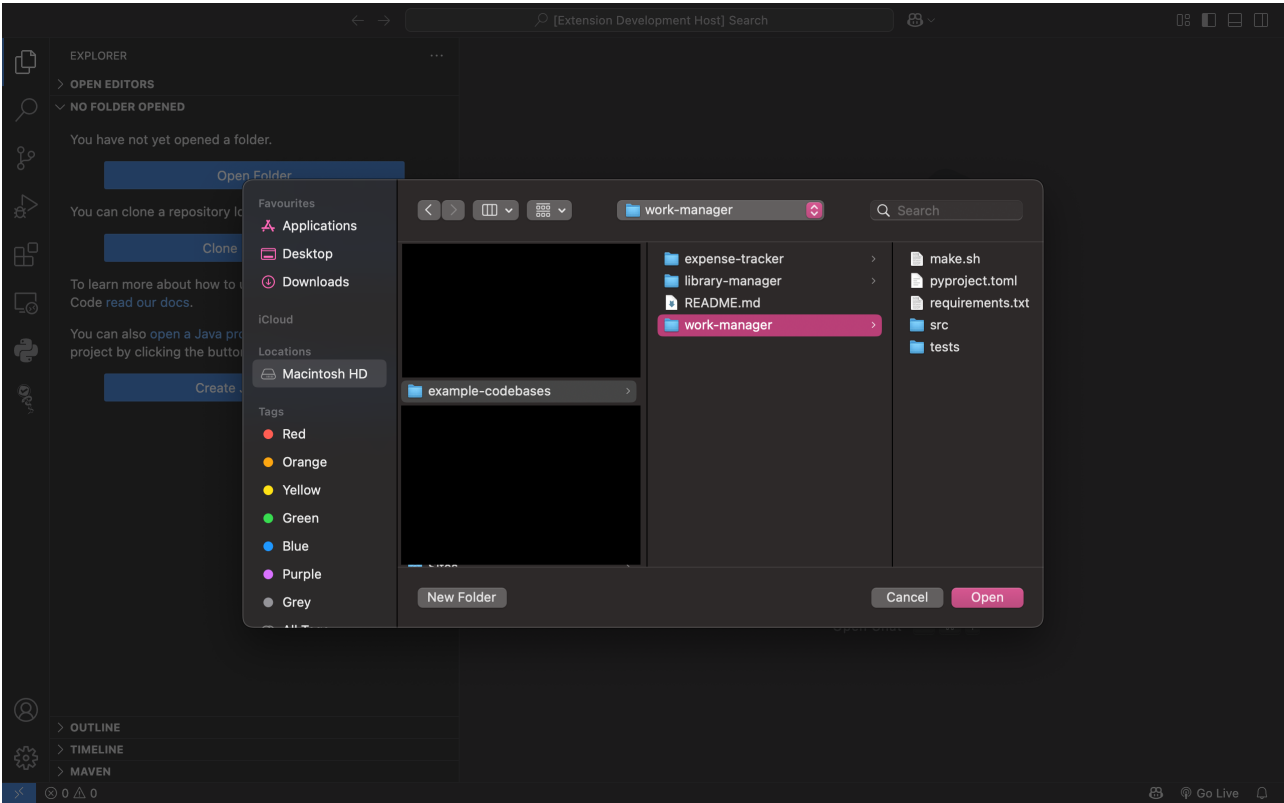
7. Steps to get started with UnitPylot

When you open UnitPylot, you will see something like this:

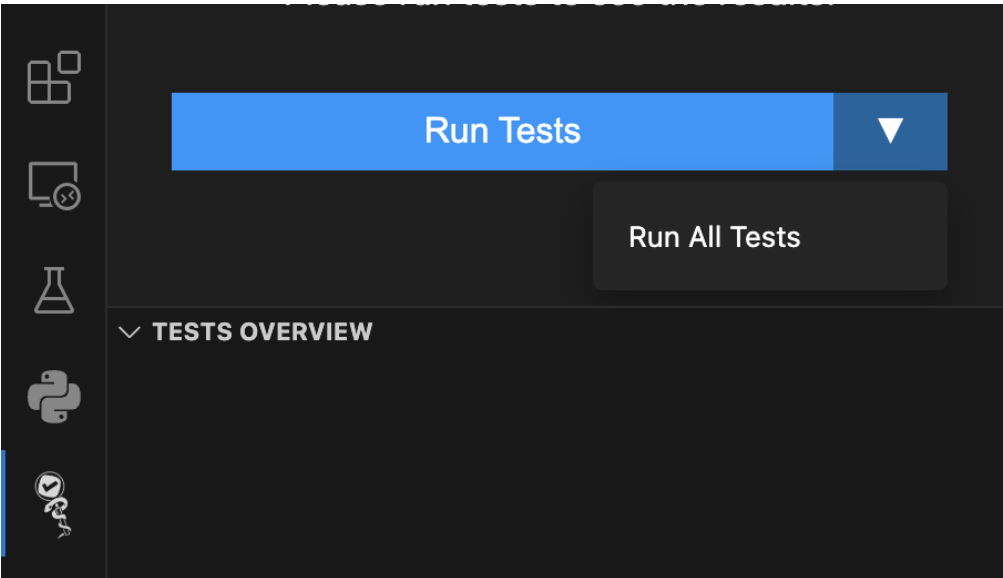


- 1. To use UnitPylot on your codebase, you **must** first open a folder in the workspace.

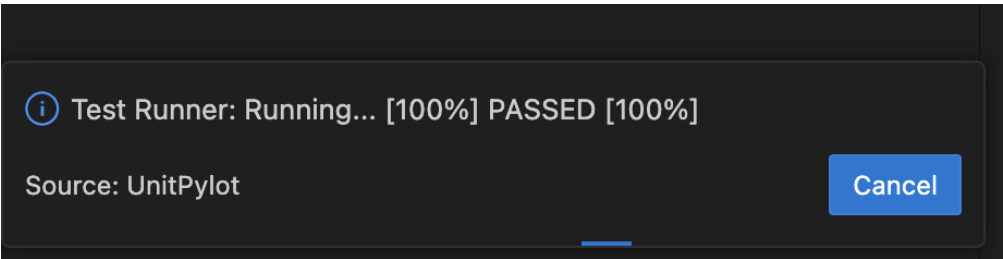




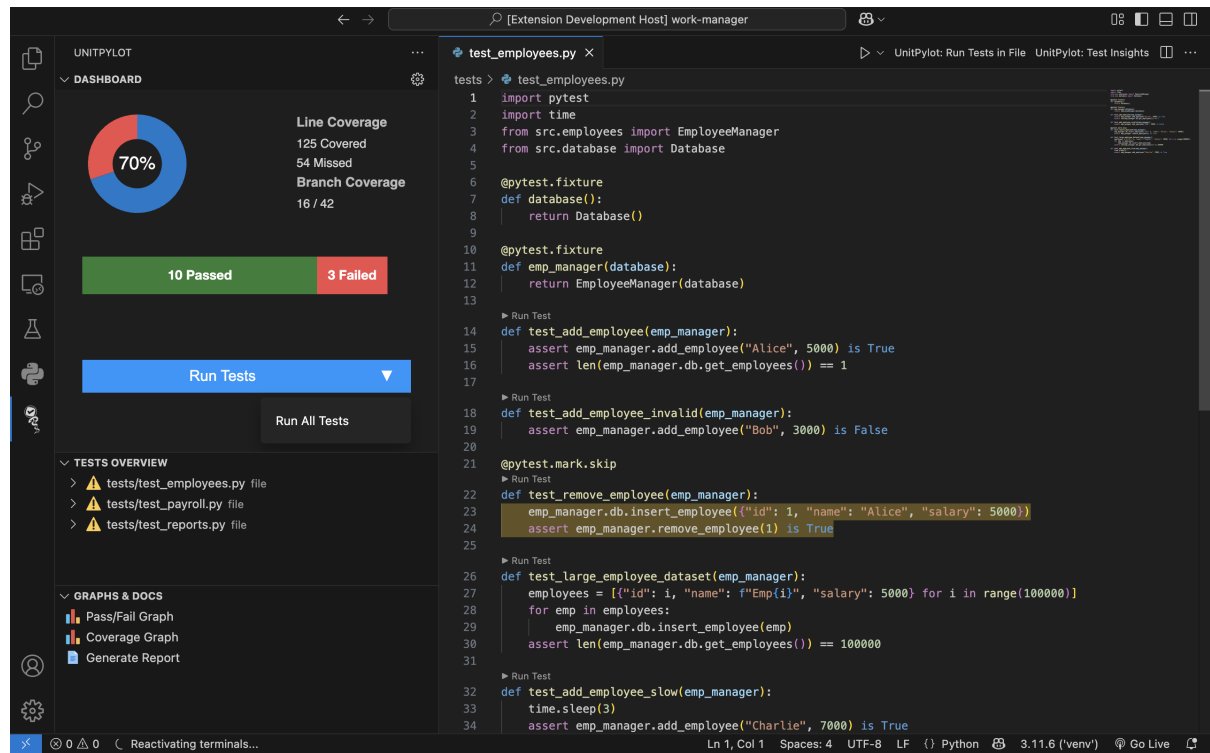
2. You must then click **Run all tests** in order for the Dashboard to correctly load.



3. Once completed, you will see a notification to alert you all tests have successfully finished and the Dashboard should be filled!



Progress notification after running all tests



The screenshot displays the UnitPylot IDE interface. On the left, a sidebar contains icons for Explorer, Search, Run and Debug, Source Control, and Test Explorer. The main area is divided into three sections: a Dashboard, Tests Overview, and Graphs & Docs.

Dashboard: A donut chart shows 70% Line Coverage (125 Covered, 54 Missed) and 16/42 Branch Coverage. Below the chart, a green bar indicates 10 Passed tests and a red bar indicates 3 Failed tests. A large blue button labeled "Run Tests" is present, along with a smaller "Run All Tests" button.

TESTS OVERVIEW: Lists three test files with warning icons: tests/test_employees.py, tests/test_payroll.py, and tests/test_reports.py.

GRAPHS & DOCS: Includes links for "Pass/Fail Graph", "Coverage Graph", and "Generate Report".

The right side of the interface shows a code editor with the file `test_employees.py`. The code includes imports for `pytest` and `time`, and defines several test functions: `test_add_employee`, `test_add_employee_invalid`, `test_remove_employee`, `test_large_employee_dataset`, and `test_add_employee_slow`. The status bar at the bottom indicates the current line and column (Ln 1, Col 1), encoding (UTF-8), and the Python environment (3.11.6 'venv').

Now you are ready to use UnitPylot!