VSCode and Docker

Container-based Development

- There are advantages to using container-based technology, especially Docker, to significantly improve software quality through repeatable builds.

Prerequisites

- Docker is installed – Install Docker Engine
- The VSCode extension Remote – Containers is installed
- VSCode

Options...

- There are several different approaches to using Dev Containers. Here are three options:
  - Using an existing Docker image from Docker Hub
  - Using a pre-build Microsoft container setup
  - Using a custom Docker image based on a project specific Dockerfile

Knowledge of Docker

- Container
- Container Image
- Container Registry

What Are Containers?

- Containers offer a logical packaging mechanism in which applications are abstracted from the environment in which they actually run.
- This decoupling allows container-based applications to be deployed easily and consistently.
- The target environment may be:
  - private data center
  - public cloud
  - developer's desktop machine or personal laptop
  - QA, staging, production environment
  - moved between cloud providers

A Clean Break

- Containerization provides a clean separation of concerns:
  - Developers can focus on their application logic and dependencies.
  - IT operations teams can focus on deployment and management.
    - (Without bothering with application details such as specific software versions and configurations specific to the application.)

Containers vs Virtualization

- Instead of virtualizing the hardware stack as with the virtual machines approach, containers virtualize at the operating system level, with multiple containers running atop the OS kernel directly.
- Containers are:
  - Far more lightweight since they share the OS kernel
  - Start much faster
  - Use a fraction of the memory compared to booting an entire OS.

Containers vs Virtualization

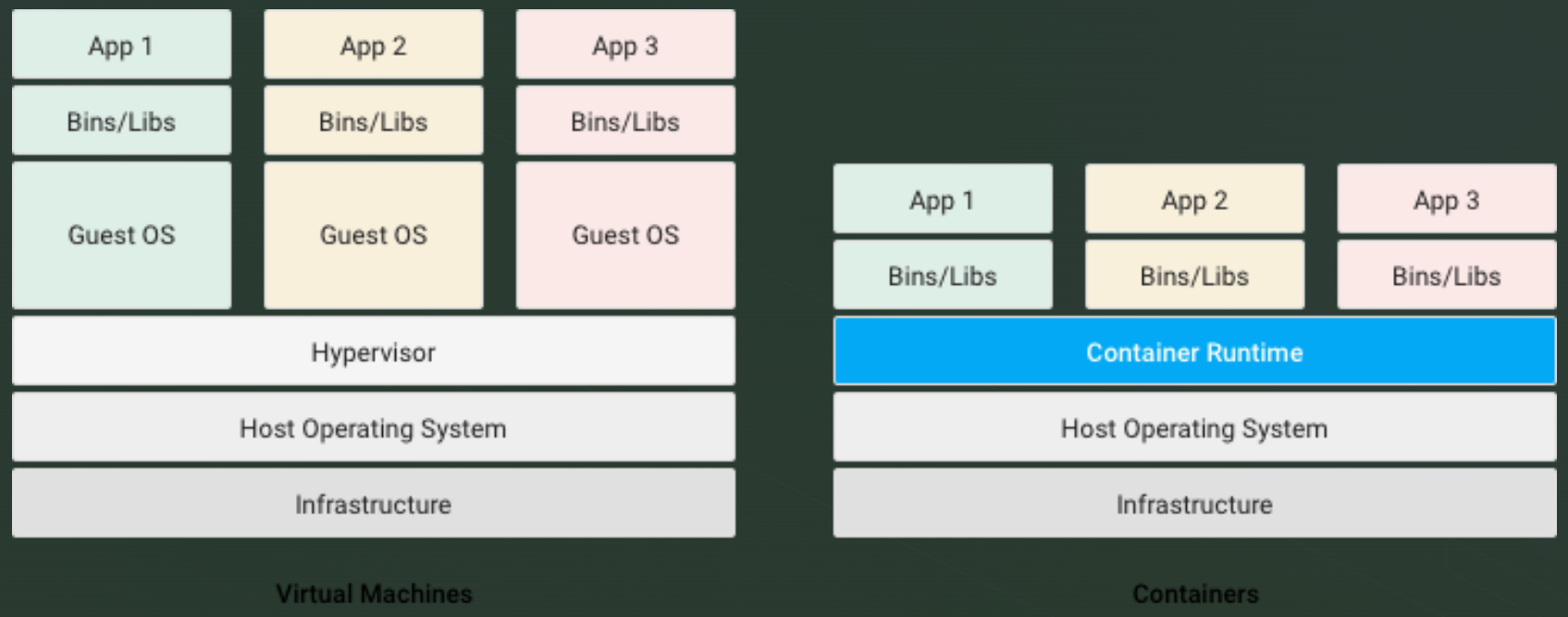


Image from Google Cloud

Developer Advantages

- Jump between Python 2.7 and 3.6
- Install dependent libraries on a clean machine image
- Is not affected by installs made on your local machine (for other projects)
- Can create an image which can be shared among team members – this keeps your local machine free and clear of any downloads or dependencies that may be loaded over time

Container Image

- A container image is an unchangeable, static file that includes executable code so it can run an isolated process on a containerization platform such as Docker.
- The image is comprised of:
  - system libraries
  - system tools
  - platforms settings a software program needs to run
- A container image is compiled from file system layers built onto a parent or base image.

Types of Container Images

- A container image may be hand built.
  - A user creates a container image from scratch with the build command of a container platform, such as Docker.
  - Each command in the Dockerfile creates a new layer in the image.
- Obtain a vendor image.
  - Many software vendors create publicly available images of their products.

Container Image Warning

- You should be aware of the existence of corrupt, fake and malicious publicly available container images, sometimes disguised to resemble official vendors' images.
- The Docker Content Trust feature relies on digital signatures to help verify that images files downloaded from public repositories are original and unaltered.

Docker Desktop

- Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices.
- Docker Desktop includes:
  - Docker Engine
  - Docker CLI client
  - Docker Compose (a tool for defining and running multi-container Docker applications)
  - Notary (a tool for publishing and managing trusted collections of content)
  - Kubernetes (a portable, extensible, open-source platform for managing containerized workloads and services)
  - Credential Helper (a suite of programs to use native stores to keep Docker credentials safe)

Docker Support

https://docs.docker.com/engine/install/

Search the docs

Install Docker Engine

Estimated reading time: 5 minutes

Supported platforms

Docker Engine is available on a variety of Linux platforms, macOS and Windows 10 through Docker Desktop, and as a static binary installation. Find your preferred operating system below.

Desktop

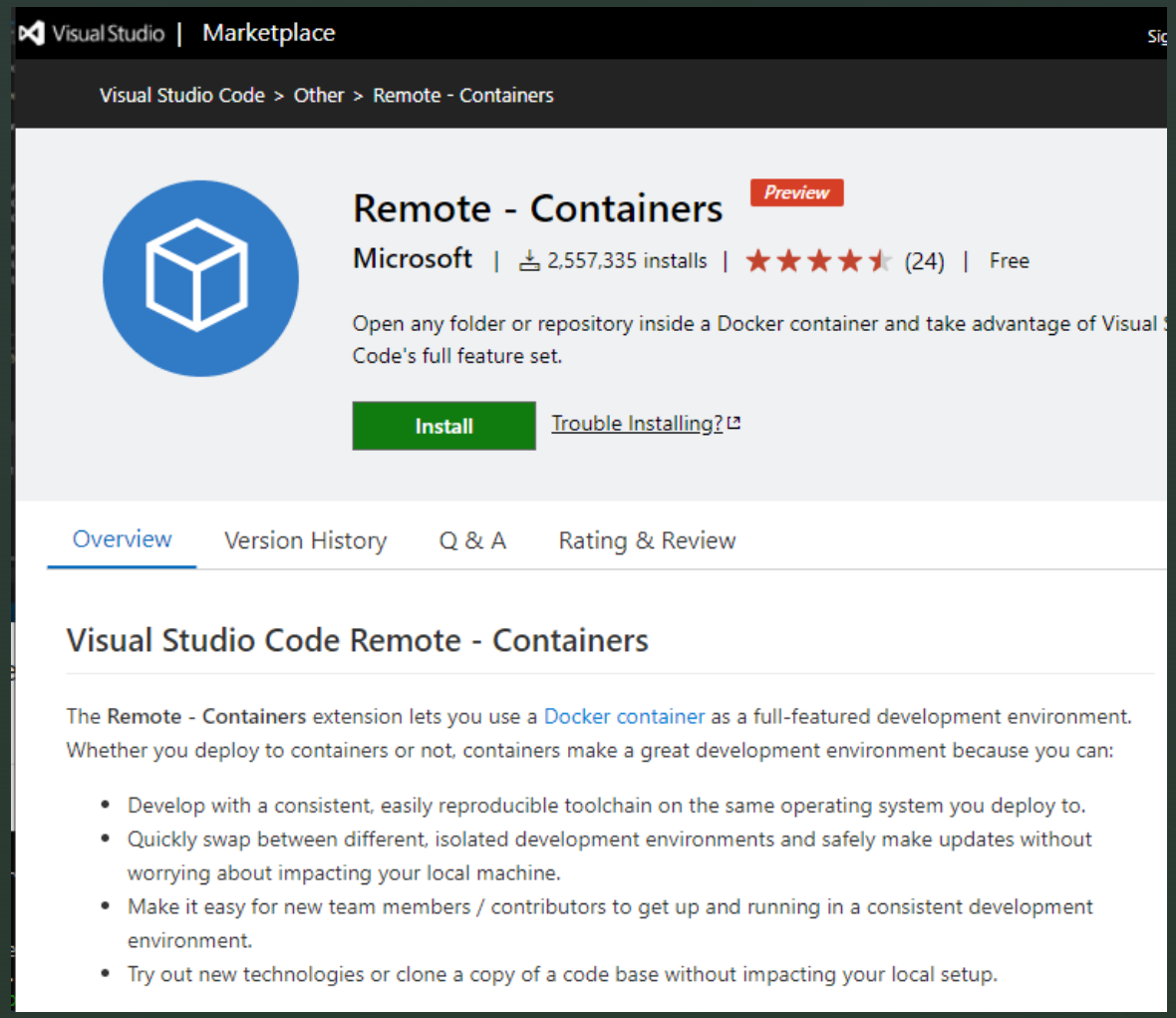
Platform	x86_64 / amd64
Docker Desktop for Mac (macOS)	✓
Docker Desktop for Windows	✓

Server

Docker provides `.deb` and `.rpm` packages from the following Linux distributions and architectures:

Platform	x86_64 / amd64	ARM	ARM64 / AARCH64
CentOS	✓		✓
Debian	✓	✓	✓
Fedora	✓		✓
Raspbian		✓	✓
Ubuntu	✓	✓	✓

Remote Containers Extensions



VSCode

- Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS.
- Features include:
  - support for debugging
  - syntax highlighting
  - intelligent code completion
  - Snippets
  - code refactoring
  - embedded Git
- Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

VSCode is Popular

- Microsoft has released Visual Studio Code's source code on the microsoft/vscode repository of GitHub, under the permissive MIT License, while the releases by Microsoft are freeware.
- In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents reporting that they use it.

Docker is Running

- Make sure that Docker is running by checking your computer's 'status' area for the Docker container logo.

- Or, execute the 'docker version' command from your command line.

```
C:\Users\Gene>docker version
Client: Docker Engine - Community
 Cloud integration: 1.0.7
 Version:           20.10.2
 API version:       1.41
 Go version:        go1.13.15
 Git commit:        2291f61
 Built:             Mon Dec 28 16:14:16 2020
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true
```

- Otherwise, you will see a connection error.

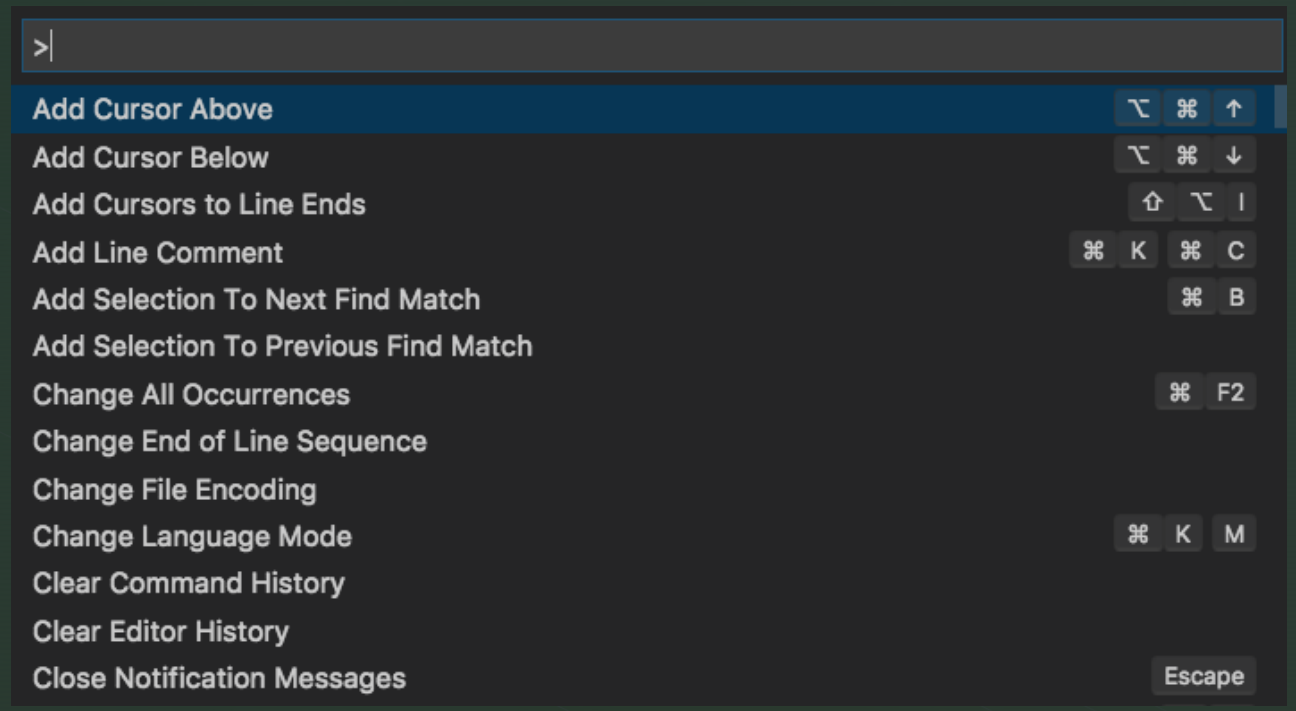
```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon runn
ing?
```

One Last Thing... Git

- Git is the most commonly used version control system.
- Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to.
- Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

Command Palette

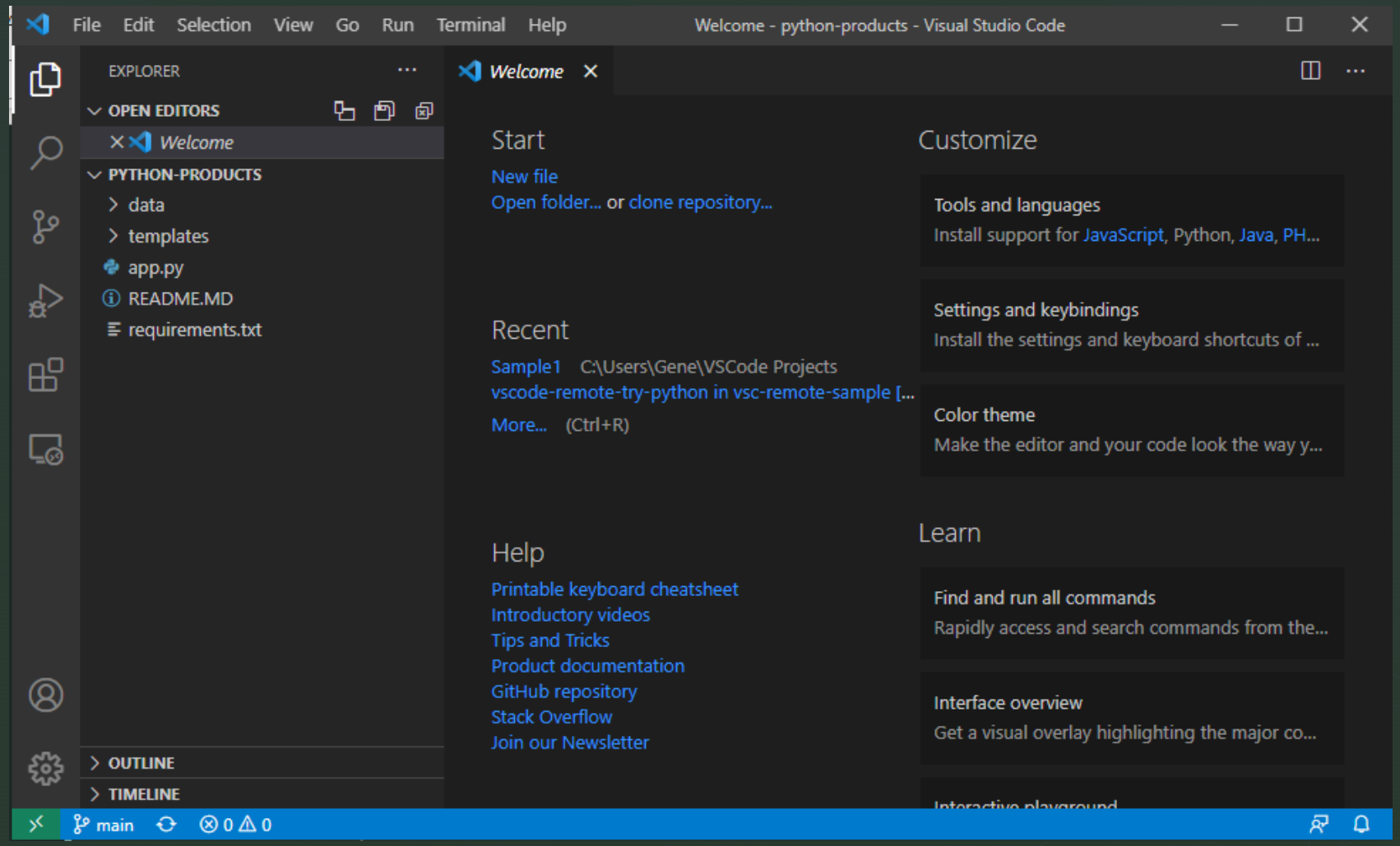
- VS Code can be easily driven from the keyboard. The most important key combination to know is Ctrl+Shift+P, which brings up the **Command Palette**. This gives you access to all of the functionality of VS Code, including keyboard shortcuts for the most common operations.



Find a Project

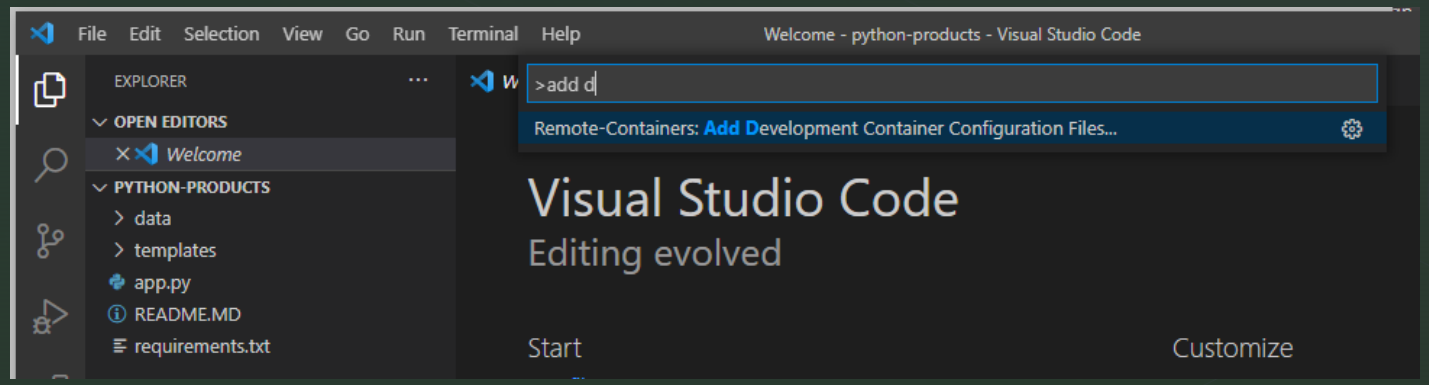
- Clone a project from Github using the Command Palette.
- Note: If you haven't already logged into Git (or installed the Git Extension- VSCode) will prompt you for Git credentials and/or permission to install the extension.
- In this example I will be using a repository used by a Microsoft container tutorial: https://github/burkeholland/python-products

Open the Folder



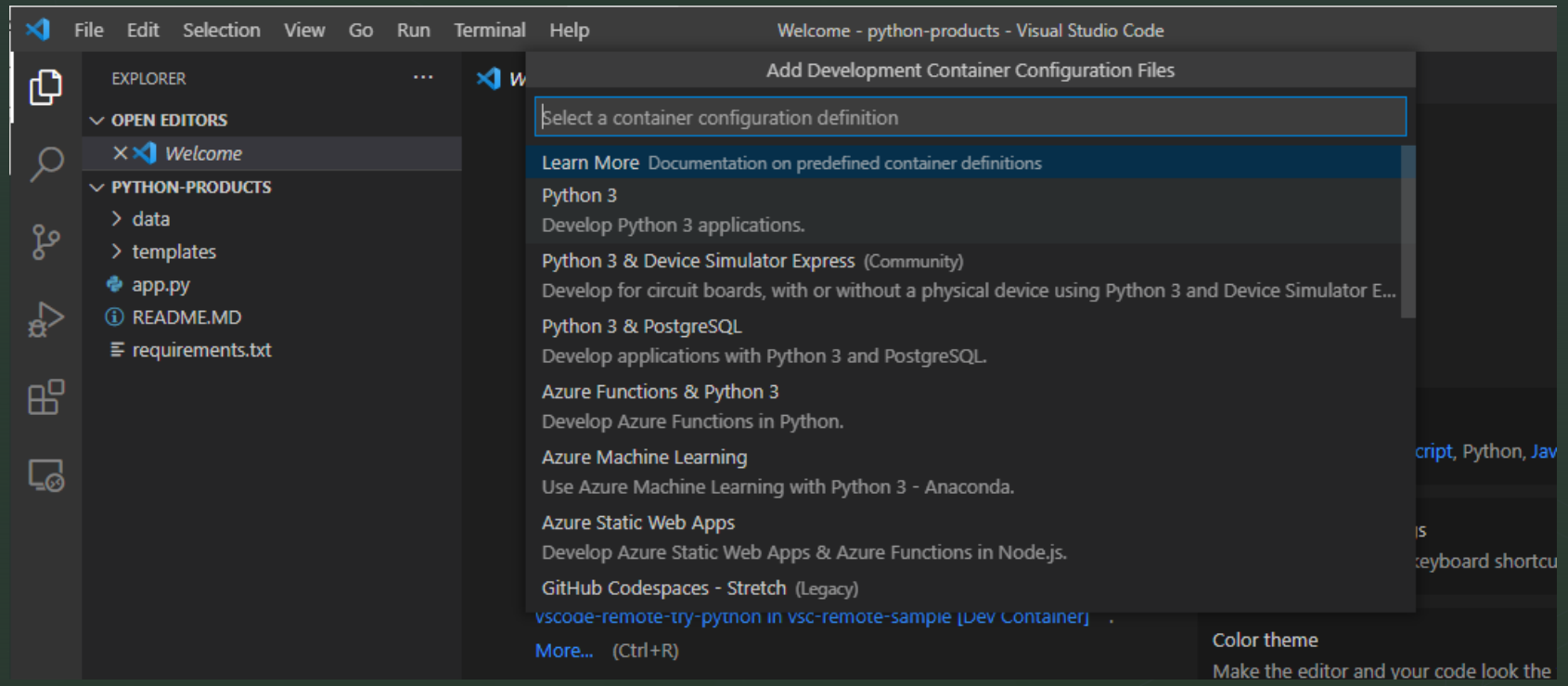
Create a Container Config

- Add a container configuration to the cloned and downloaded project.
- Open Command Palette and search for "Add Development Container Configuration Files"



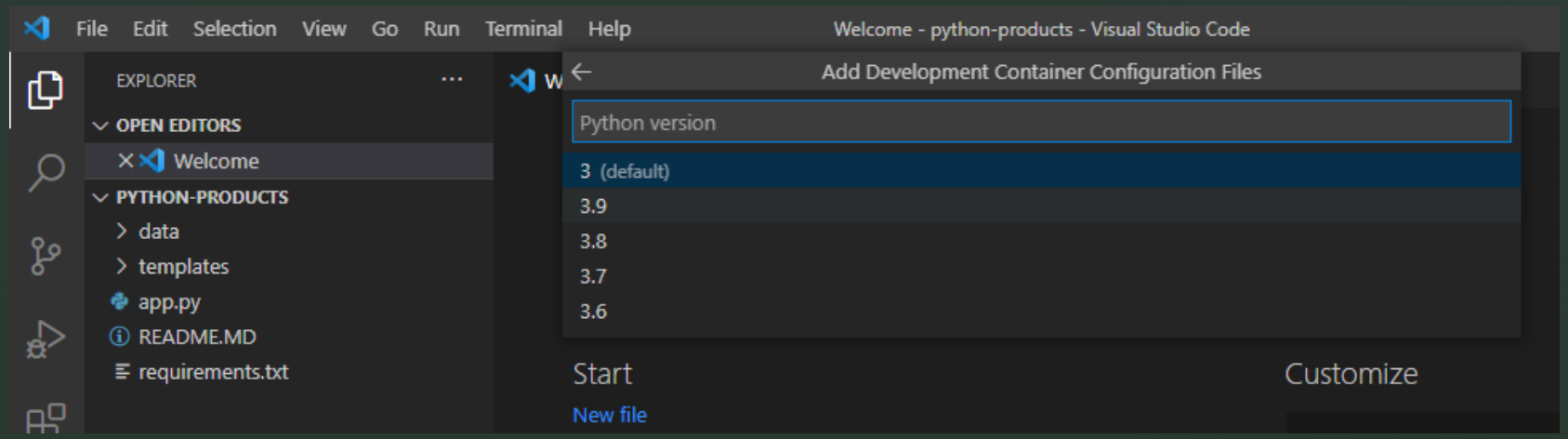
Container Recommendation

- VSCode will make recommendations based on the files in the open folder.



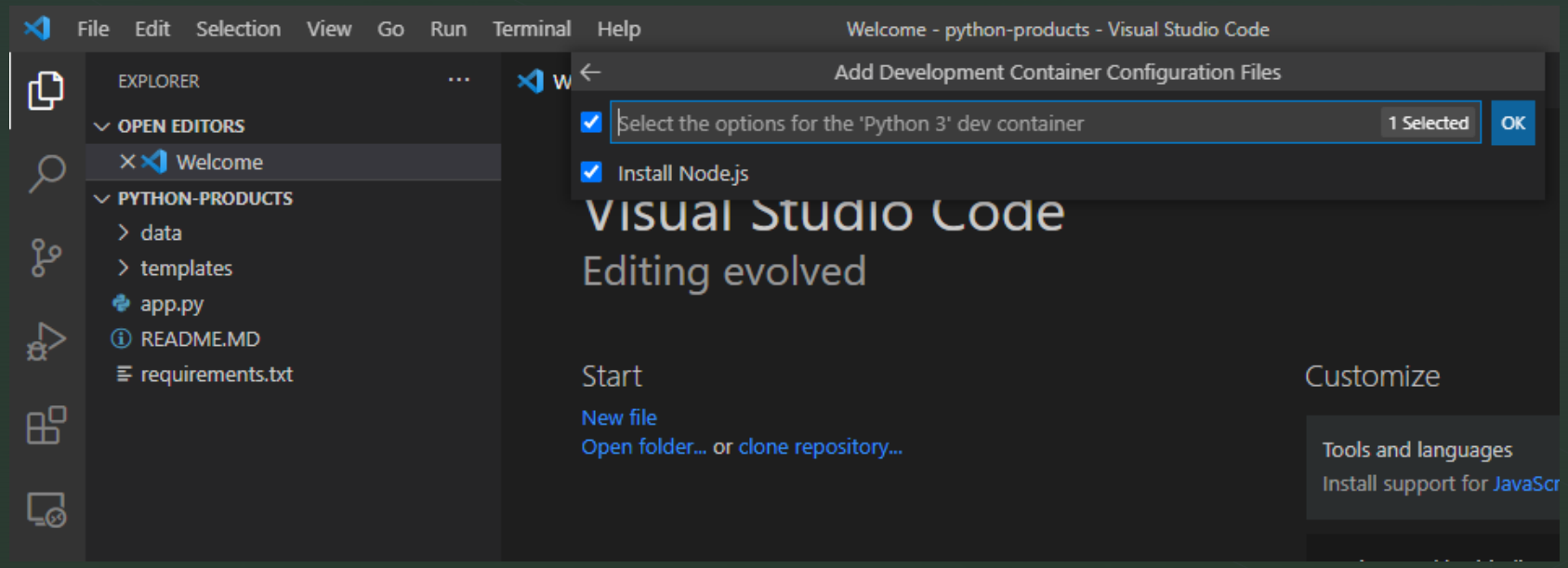
Python and Version

- Select Python 3
- The VSCode extension will prompt the version of Python to use.



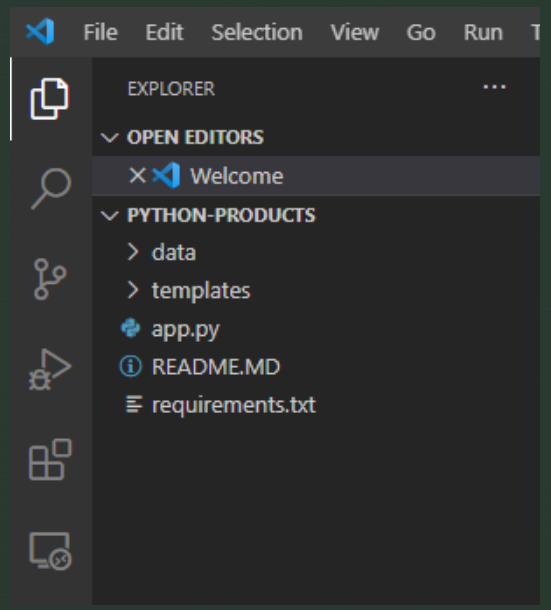
Container Options

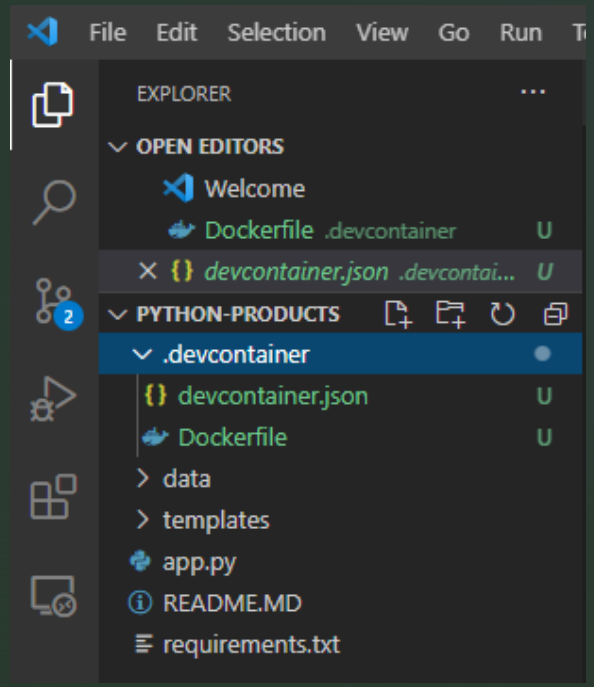
- The extension will then prompt to install any additional options.
- Deselected Node.js



.devcontainer Folder

- The container extension has created a .devcontainer folder containing two files.



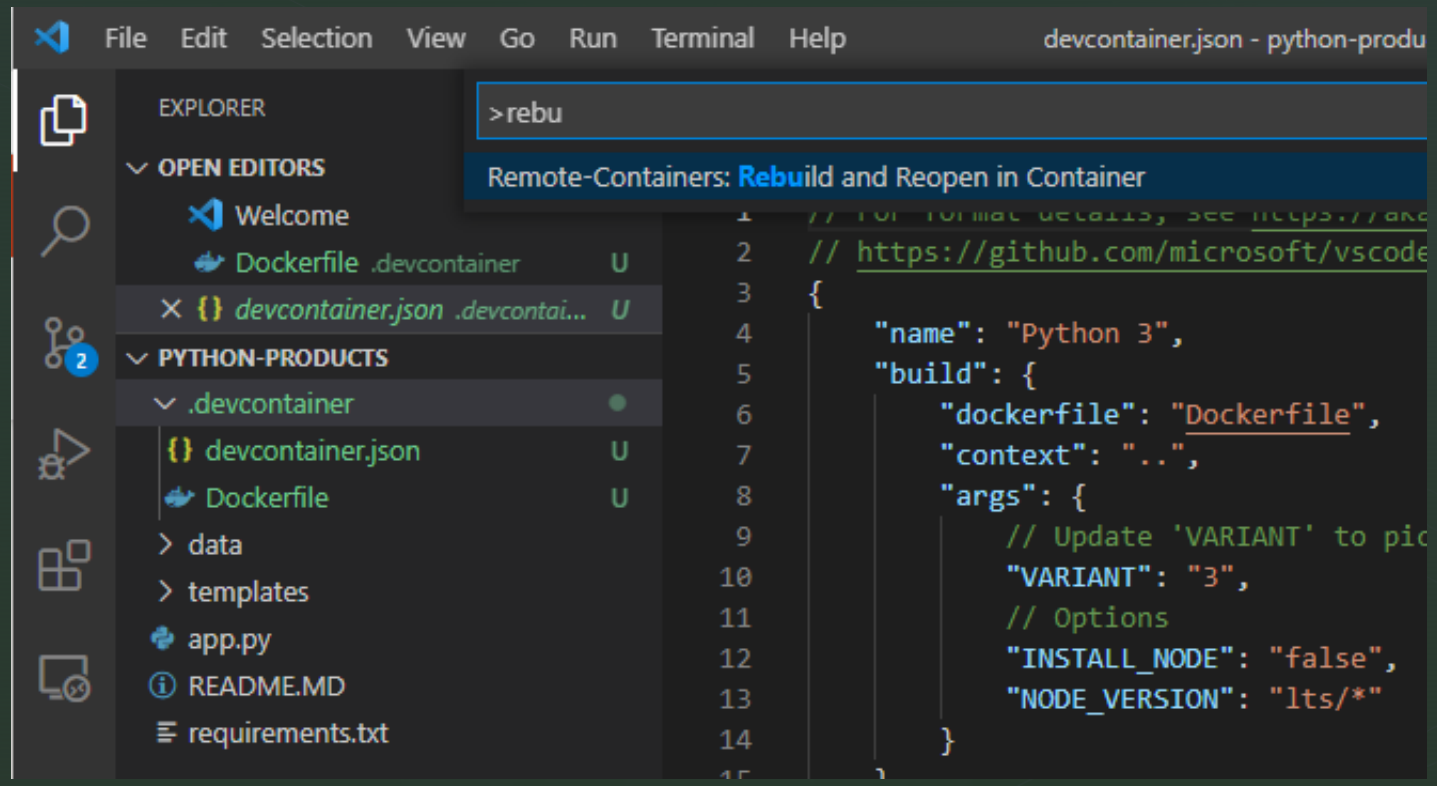


Containerized Project

- The presence of the .devcontainer folder with the two files committed to source control will allow other developers, who have VSCode with Remote Container Extensions and Docker installed, to open, execute and debug this project inside a container.

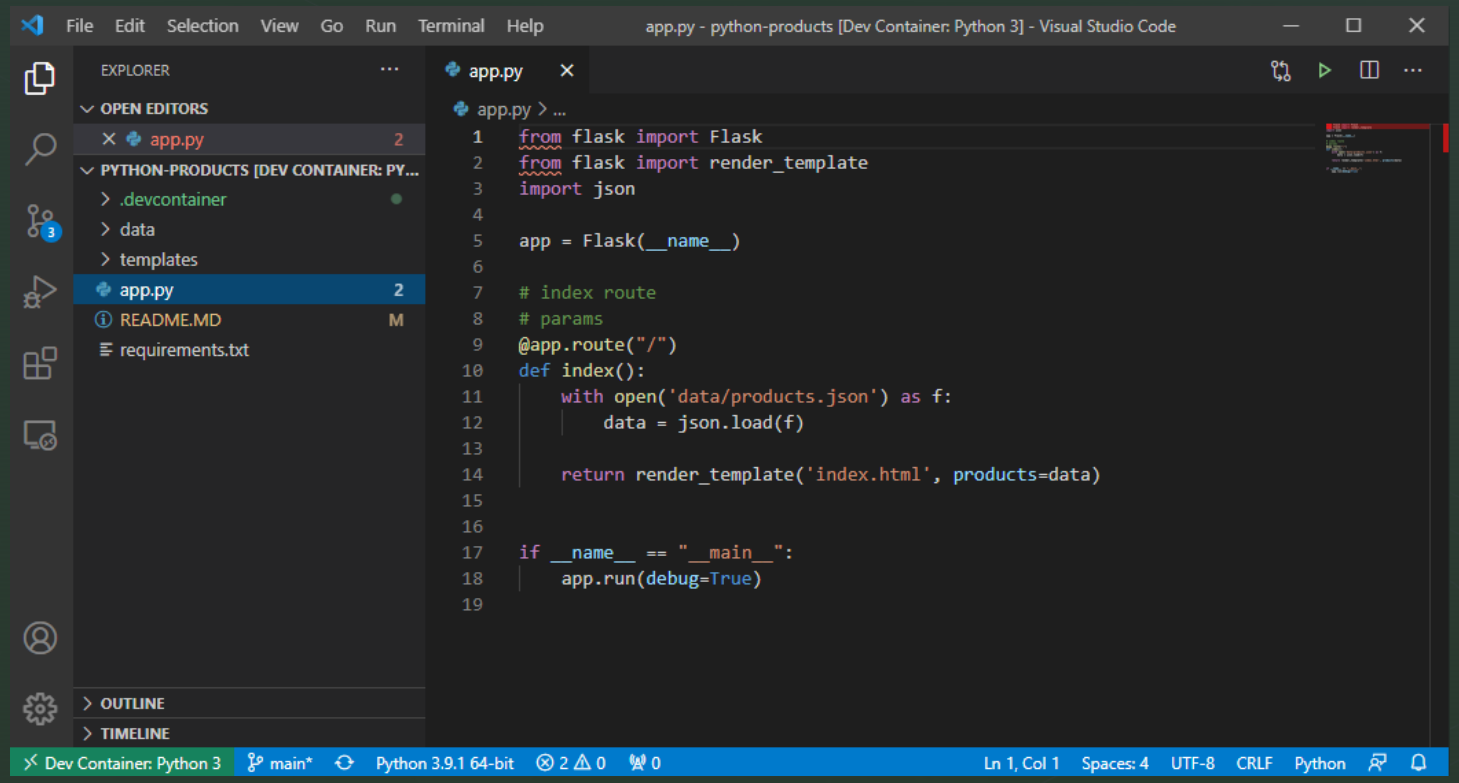
Build and Open

- Next, we have to build the container and open the project in the container.



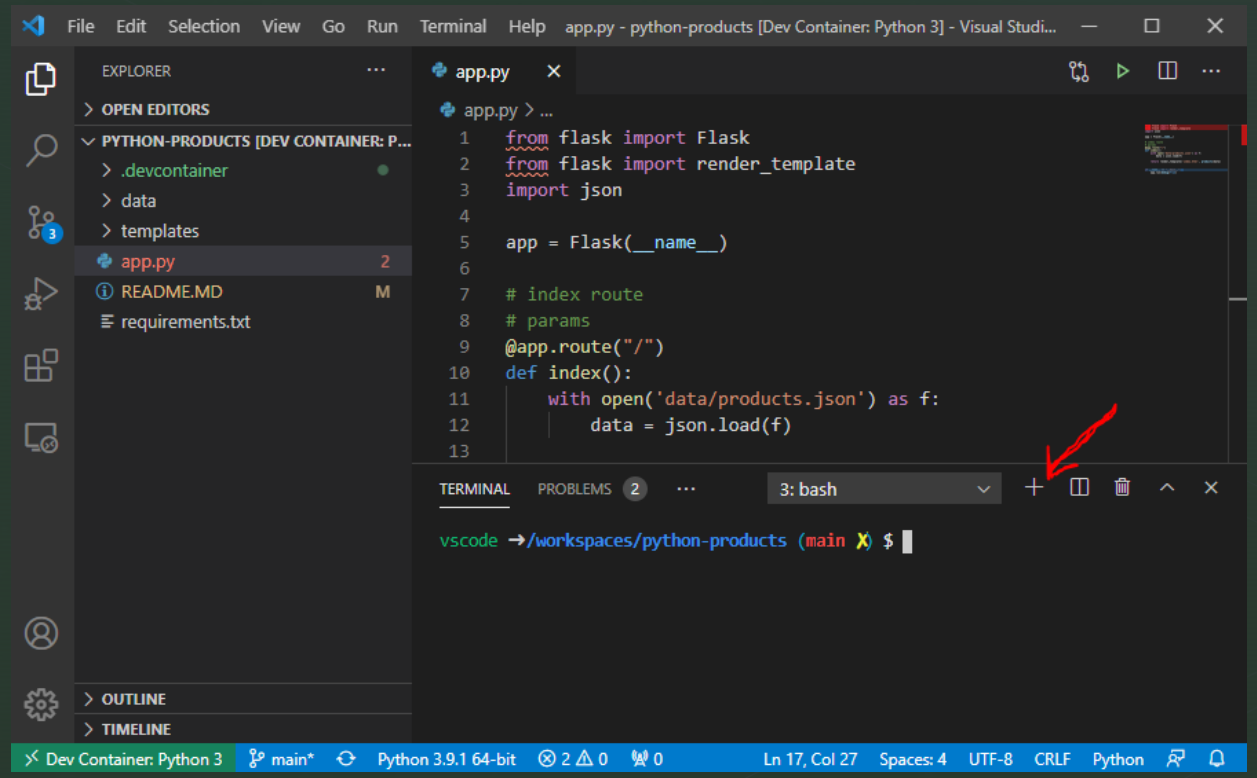
Working in the Container

- The container with which VSCode is working is shown in the the new 'green' status indicator at the bottom left of the editor.



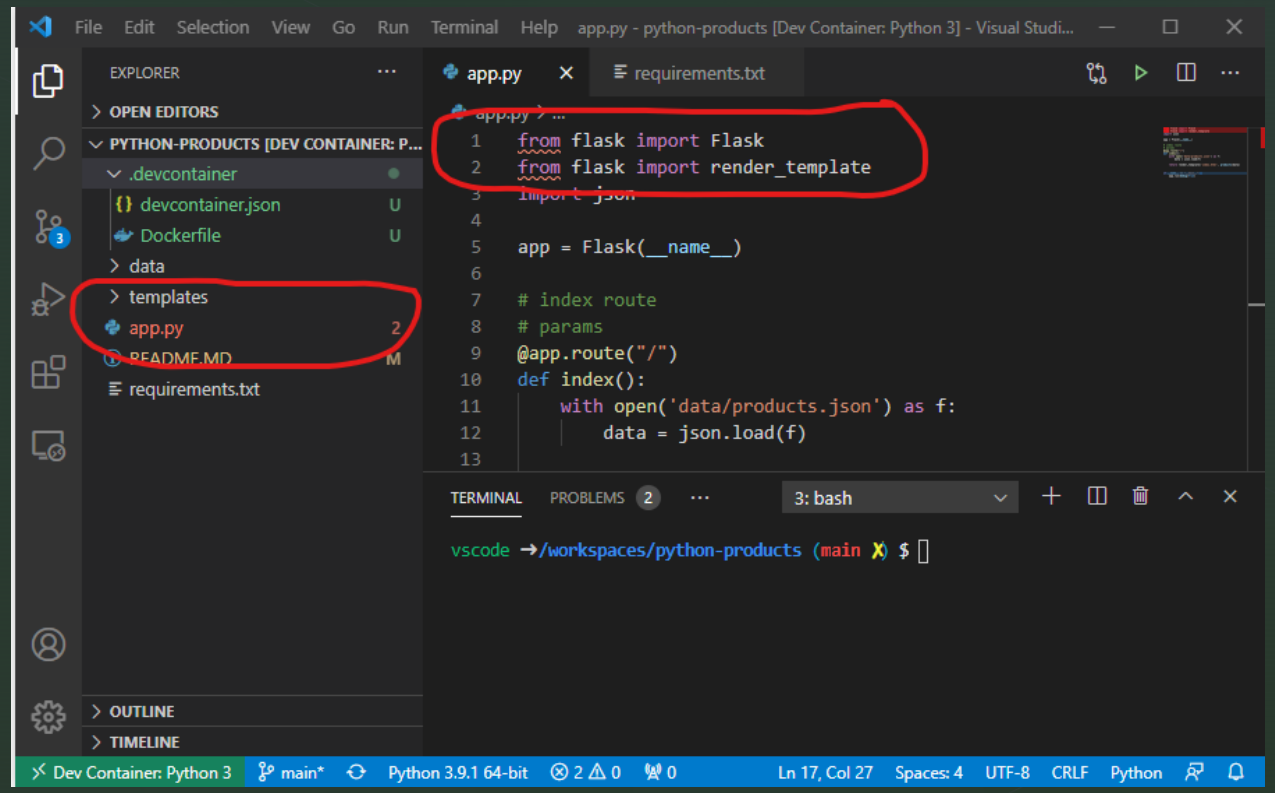
Open a Terminal

- Click on the '+' button in the lower pane to open a new terminal.
- The terminal is in the docker container, we are not on our local machine.



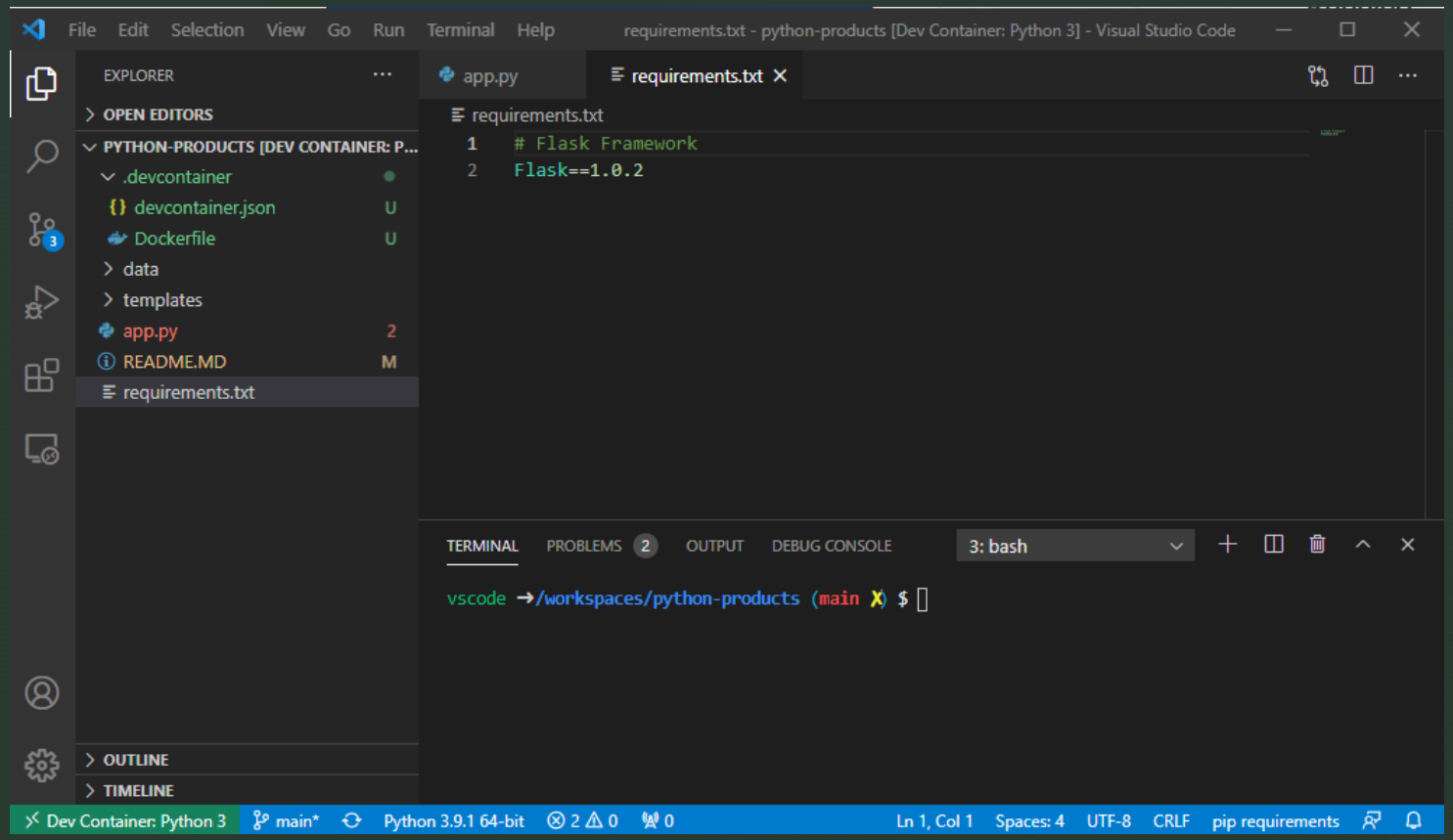
Build Errors

- Compiling the small project has revealed two errors. The Flask framework is required.



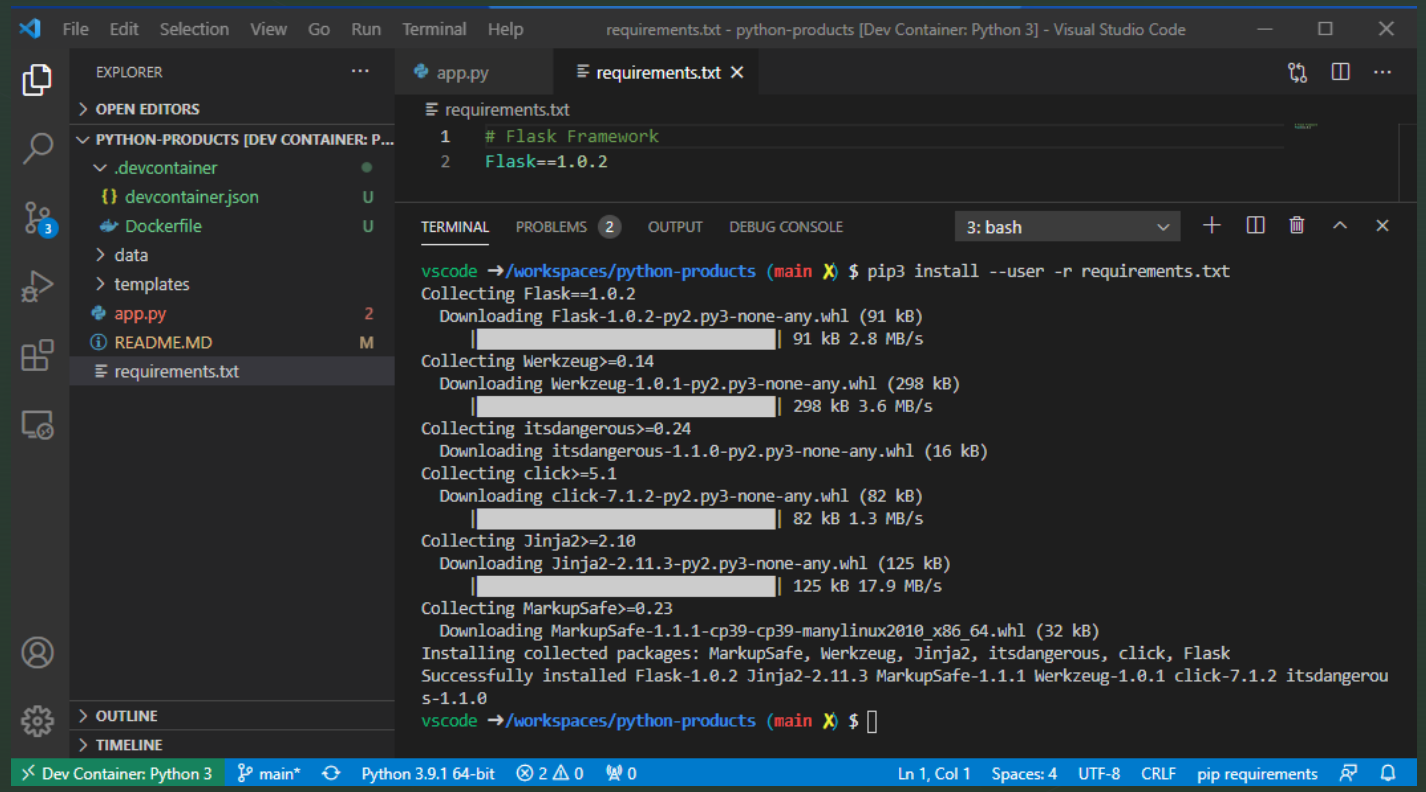
Dependencies

- The sample project contains a 'requirements.txt' file.



Pip3

- Use Pip to install the dependencies identified in the requirements.txt file.



Run It!

```
vscode ➜/workspaces/python-products (main ✗) $ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 813-955-307
127.0.0.1 - - [17/Feb/2021 01:15:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Feb/2021 01:15:16] "GET /favicon.ico HTTP/1.1" 404 -
```

Your service running on port 5000 is available. See all forwarded ports

Open in Browser

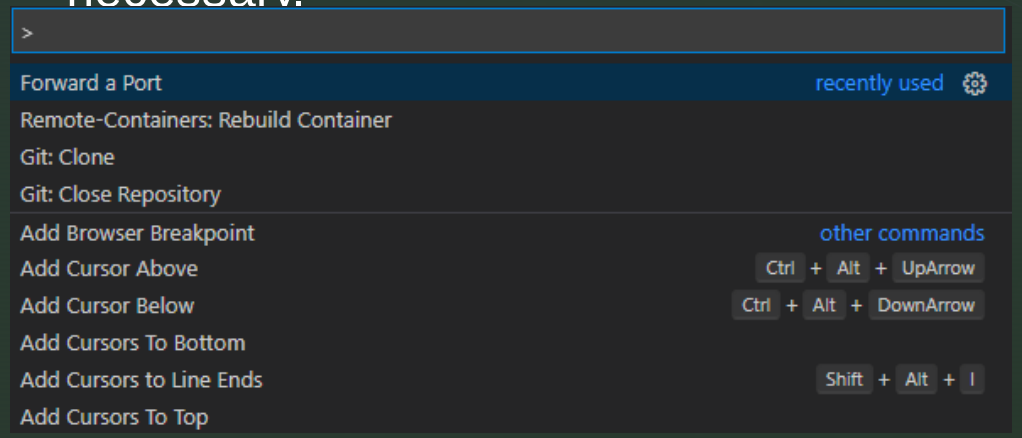
127.0.0.1:5000

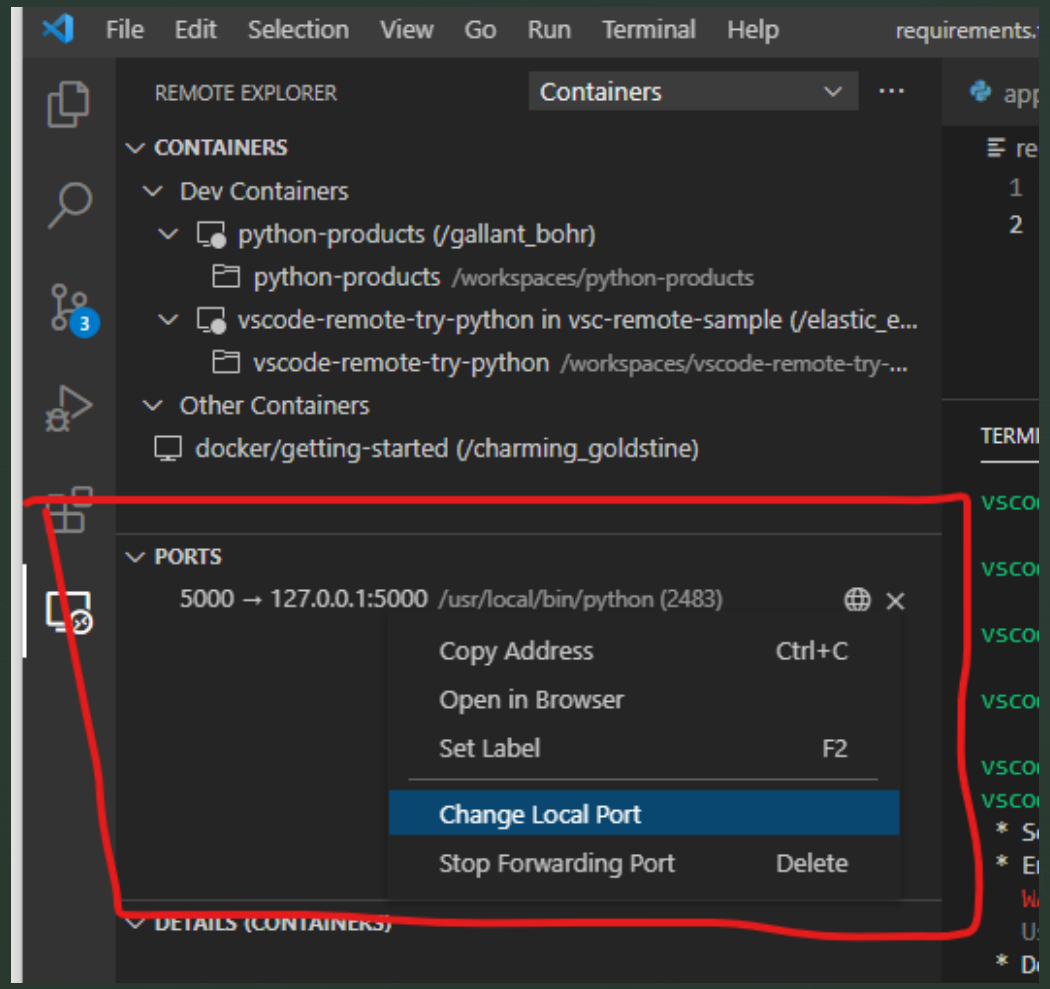
Products

Name	Brand	Price	Units in Stock
Single red garden gnome	Home & Pro tools	56	98
Two red garden gnomes	Home & Pro tools	92	4
One sat gnome	Home & Pro tools	34	34
One sat on shoe gnome	Home & Pro tools	54	54
One barrow gnome	Home & Pro tools	29	23
One glasses gnome	Home & Pro tools	54	94
One smiling gnome	Home & Pro tools	43	45
Two singing gnomes	Home & Pro tools	65	78
Two sleeping gnomes	Home & Pro tools	32	12

Port Forwarding

- Each container is an isolated environment.
- In order to acess any service or web page in the container, you need to forward the port to your local machine (the host).
- Use the Command Palette to open the 'Ports' view.
- You can re-map to a different local port as necessary.





Rebuild and Redeploy Container

- Rebuild and redeploy the container.
- Run the application.

```
TERMINAL   PROBLEMS   ···   1: bash

vscode ➜ /workspaces/python-products (main ✗) $ python app.py
Traceback (most recent call last):
  File "/workspaces/python-products/app.py", line 1, in <module>
    from flask import Flask
ModuleNotFoundError: No module named 'flask'
vscode ➜ /workspaces/python-products (main ✗) $
```

Add a Layer to the Image

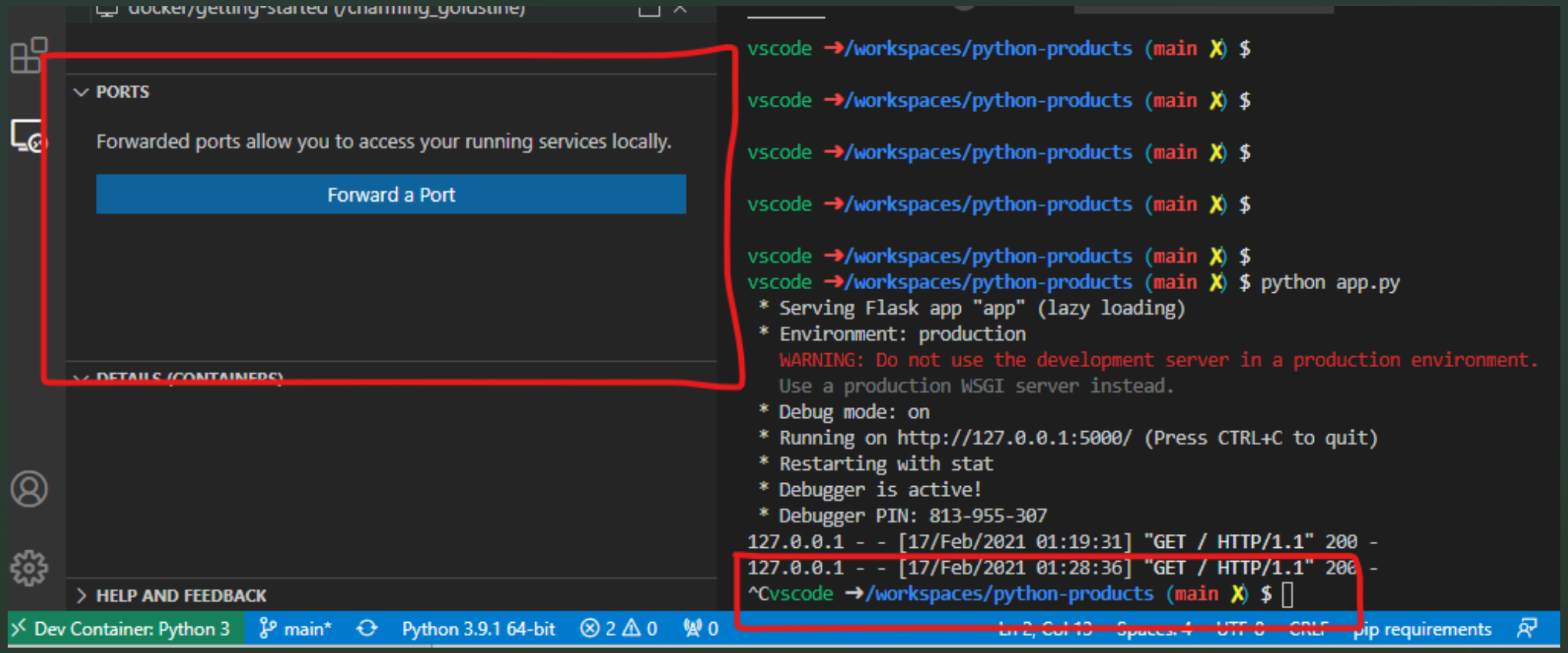
- You can modify the Dockerfile to add a layer to the image which installs the Flask framework.

Dockerfile ×

.devcontainer > Dockerfile

```
1   # See here for image contents: https://github.com/microsoft/vscode-dev-containers/tree/v0.158.0/
2
3   # [Choice] Python version: 3, 3.9, 3.8, 3.7, 3.6
4   ARG VARIANT="3"
5   FROM mcr.microsoft.com/vscode/devcontainers/python:0-${VARIANT}
6
7   # [Option] Install Node.js
8   ARG INSTALL_NODE="true"
9   ARG NODE_VERSION="lts/*"
10  RUN if [ "${INSTALL_NODE}" = "true" ]; then su vscode -c "umask 0002 && . /usr/local/share/nvm/r
11
12  RUN pip3 install Flask==1.0.2
13
14  # [Optional] If your pip requirements rarely change, uncomment this section to add them to the i
15  # COPY requirements.txt /tmp/pip-tmp/
16  # RUN pip3 --disable-pip-version-check --no-cache-dir install -r /tmp/pip-tmp/requirements.txt \
17  #    && rm -rf /tmp/pip-tmp
18
19  # [Optional] Uncomment this section to install additional OS packages.
20  # RUN apt-get update && export DEBIAN_FRONTEND=noninteractive \
21  #     && apt-get -y install --no-install-recommends <your-package-list-here>
```

Stop the Application and the Port is No Longer Forwarded



devcontainer.json

- Contains properties of the containerized project and allows you to modify aspects of the project and the environment without modifying the Docker file.

```
{
    "name": "Python 3",
    "build": {
        "dockerfile": "Dockerfile",
        "context": "..",
        "args": {
            // Update 'VARIANT' to pick a Python version: 3, 3.6, 3.7, 3.8, 3.9
            "VARIANT": "3",
            // Options
            "INSTALL_NODE": "false",
            "NODE_VERSION": "lts/*"
        }
    },

    // Set *default* container specific settings.json values on container create.
    "settings": {
        "terminal.integrated.shell.linux": "/bin/bash",
        "python.pythonPath": "/usr/local/bin/python",
        "python.linting.enabled": true,
        "python.linting.pylintEnabled": true,
        "python.formatting.autopep8Path": "/usr/local/py-utils/bin/autopep8",
        "python.formatting.blackPath": "/usr/local/py-utils/bin/black",
        "python.formatting.yapfPath": "/usr/local/py-utils/bin/yapf",
        "python.linting.banditPath": "/usr/local/py-utils/bin/bandit",
        "python.linting.flake8Path": "/usr/local/py-utils/bin/flake8",
        "python.linting.mypyPath": "/usr/local/py-utils/bin/mypy",
        "python.linting.pycodestylePath": "/usr/local/py-utils/bin/pycodestyle",
        "python.linting.pydocstylePath": "/usr/local/py-utils/bin/pydocstyle",
        "python.linting.pylintPath": "/usr/local/py-utils/bin/pylint"
    },

    // Add the IDs of extensions you want installed when the container is created.
    "extensions": [
        "ms-python.python"
    ],

    // Use 'forwardPorts' to make a list of ports inside the container available locally.
    // "forwardPorts": [],

    // Use 'postCreateCommand' to run commands after the container is created.
    // "postCreateCommand": "pip3 install --user -r requirements.txt",

    // Comment out connect as root instead. More info: https://aka.ms/vscode-
remote/containers/non-root.
    "remoteUser": "vscode"
}
```