TOPICS 蟒蛇 ▼

容器中的Python

(https://github.com/Microsoft/vscode-docs/blob/master/docs/containers/quickstart-python.md)

在本指南中，您将学习如何：

- 创建一个 `Dockerfile` 描述简单Python容器的文件。
- 构建，运行和验证Django (https://www.djangoproject.com/)，Flask (http://flask.pocoo.org/)或General Python应用程序的功能。
- 调试在容器中运行的应用程序。

先决条件

- 必须按照概述 (overview#_installation)中所述安装Docker Desktop和VS Code Docker扩展。
- 对于Python开发，请完成所有Python入门 (/docs/python/python-tutorial)步骤
- 可运行的Python应用程序

创建一个Python项目

如果您还没有Python项目，请在终端上依次执行以下命令：

```
pip install django
django-admin startproject helloworld
cd helloworld
code .
```

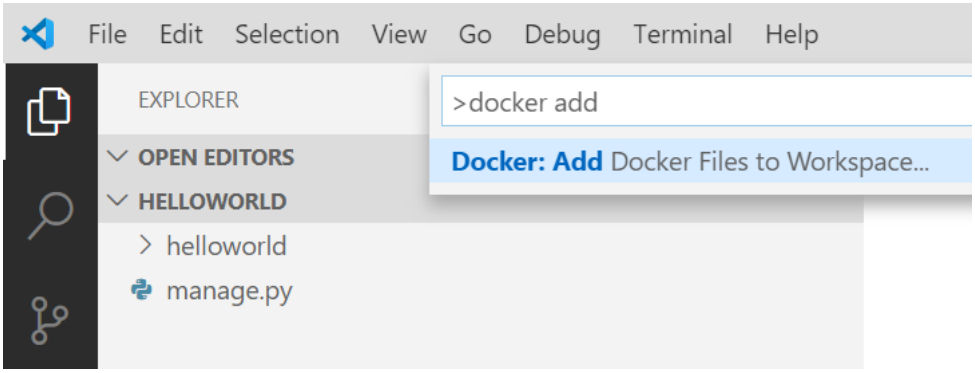
如果要容器化完整的Django或Flask Web应用程序，则可以使用以下示例之一：

- python-sample-vscode-django-tutorial (https://github.com/microsoft/python-sample-vscode-django-tutorial/tree/tutorial)，这是遵循Django教程 (/docs/python/tutorial-django)的结果 (/docs/python/tutorial-django)
- python-sample-vscode-flask-tutorial (https://github.com/microsoft/python-sample-vscode-flask-tutorial/tree/tutorial)，这是遵循Flask教程 (/docs/python/tutorial-flask)的结果 (/docs/python/tutorial-flask)

验证您的应用程序运行正常之后，您现在可以对您的应用程序进行Dockerize。

将Docker文件添加到项目

1. 在VS Code中打开项目文件夹。
2. 打开命令面板（ Ctrl + Shift + P ），然后使用**Docker：将Docker文件添加到工作区**...命令：



3. 出现提示时，选择**Python：Django**，**Python：Flask**或**Python：General**作为应用程序类型。在本教程中，我们将选择**Python：Django**。
4. 当提示您包括Docker Compose (https://docs.docker.com/compose/)文件时，选择**是**或**否**。 (https://docs.docker.com/compose/)
5. 输入应用程序入口点的相对路径。这不包括您从其开始的工作区文件夹。根据Django官方文档 (https://docs.djangoproject.com/en/3.0/intro/tutorial01/#creating-a-project)，此路径通常是 `manage.py` （根文件夹）或 `subfolder_name/manage.py` 。根据Flask的官方文档 (https://flask.palletsprojects.com/en/1.1.x/api/)，这是创建Flask实例的路径。

   > **提示**：您也可以输入文件夹名称的路径，只要该文件夹包含 `__main__.py` 文件即可。

6. 如果选择了**Python：Django**或**Python：Flask**，请为本地开发指定应用程序端口。Django的默认端口为8000，而Flask的默认端口为5000；但是，任何未使用的端口都可以使用。我们建议选择端口1024或更高版本，以减轻以root用户身份运行时的 (/docs/containers/python-user-rights)安全隐患。
7. 利用所有这些信息，Docker扩展将创建以下文件：
   - 一 `Dockerfile` 。要在此文件中了解有关Intellisense的更多信息，请参阅概述 (/docs/containers/overview)。
   - 一个 `.dockerignore` 文件，通过排除不需要的如文件和文件夹，以减少图像尺寸 `.git` ， `.vscode` 以及 `__pycache__` 。
   - 如果选择了Docker Compose，则为一个 `docker-compose.yml` 文件。

- 如果尚不存在， `requirements.txt` 则为捕获所有应用程序依赖项的文件。

> **重要提示**：要使用我们的设置，Python的框架（Django的/瓶）和Gunicorn **必须包含**中 `requirements.txt` 的文件。如果虚拟环境/主机已经安装了这些先决条件，并且应该与容器环境相同，请确保通过 `pip freeze > requirements.txt` 在终端中运行来移植应用程序依赖项。**这将覆盖您当前的 `requirements.txt` 文件。**

Django / Flask应用程序的文件修改

为了给Python Web开发人员一个很好的起点，我们选择使用Gunicorn (https://gunicorn.org/#docs)作为默认的Web服务器。由于在默认Dockerfile中引用了它，因此它作为依赖项包含在 `requirements.txt` 文件中。

> **注意**：要将Gunicorn用作Web服务器，必须将其 `requirements.txt` 作为应用程序依赖项包含在文件中。不需要将其安装在虚拟环境/主机中。

DJANGO应用程式

要使用Gunicorn，它必须绑定到可调用的应用程序（应用程序服务器用来与您的代码进行通信的内容）作为入口点。该可调用对象在 `wsgi.py` Django应用程序的文件中声明。为了完成此绑定，Dockerfile中的最后一行说：

```
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "{workspace_folder_name}.wsgi"]`
```

如果您的项目不遵循Django的默认项目结构（即，一个工作空间文件夹和一个与该工作空间相同的子文件夹中的wsgi.py (http://wsgi.py)文件），则必须覆盖Dockerfile中的Gunicorn入口点以找到正确的 `wsgi.py` 文件。

> **提示**：如果您的 `wsgi.py` 文件位于根文件夹中，则上面命令中的最后一个参数将为 `"wsgi"` 。在子文件夹中，参数为 `"subfolder1_name.subfolder2_name.wsgi"` 。

烧瓶应用

要使用Gunicorn，它必须绑定到可调用的应用程序（应用程序服务器用来与您的代码进行通信的内容）作为入口点。该可调用对象与您创建的Flask实例的**文件位置**和**变量名称**相对应。根据Flask官方文档 (https://flask.palletsprojects.com/en/1.1.x/api/)，用户通常 `__init__.py` 以这种方式在主模块或软件包文件中创建Flask实例：

```
from flask import Flask
app = Flask(__name__) # Flask instance named app
```

为了完成此绑定，Dockerfile中的最后一行说：

```
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "{subfolder}.{module_file}:app"]
```

在**Docker：Add Docker Files to Workspace ...**命令中，您配置Flask实例的路径，但是Docker扩展假定您的Flask实例变量名为 `app` 。如果不是这种情况，则必须在Dockerfile中更改变量名称。

> **提示**：如果您的主模块以名为的文件位于根文件夹中， `main.py` 并且具有一个Flask实例变量 `myapp` ，则上述命令中的最后一个参数将为 `"main:myapp"` 。在子文件夹中，参数为 `"subfolder1_name.subfolder2_name.main:myapp"` 。

生成，运行和调试容器

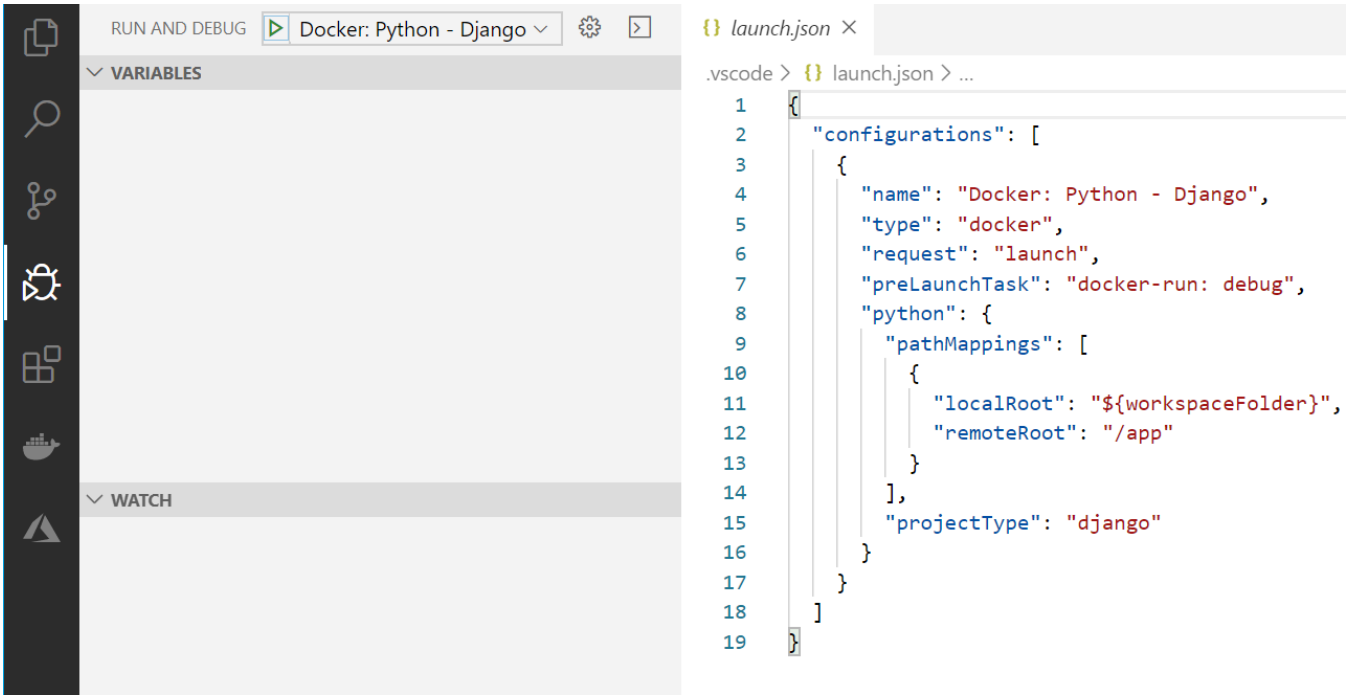
该**泊坞窗：添加泊坞文件到工作区**...命令会自动创建一个泊坞窗启动配置来构建和调试模式下运行的容器。要调试您的Python应用容器：

1. 导航到 `manage.py` 文件并在此行上设置断点：

   ```
   os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'helloworld.settings')
   ```

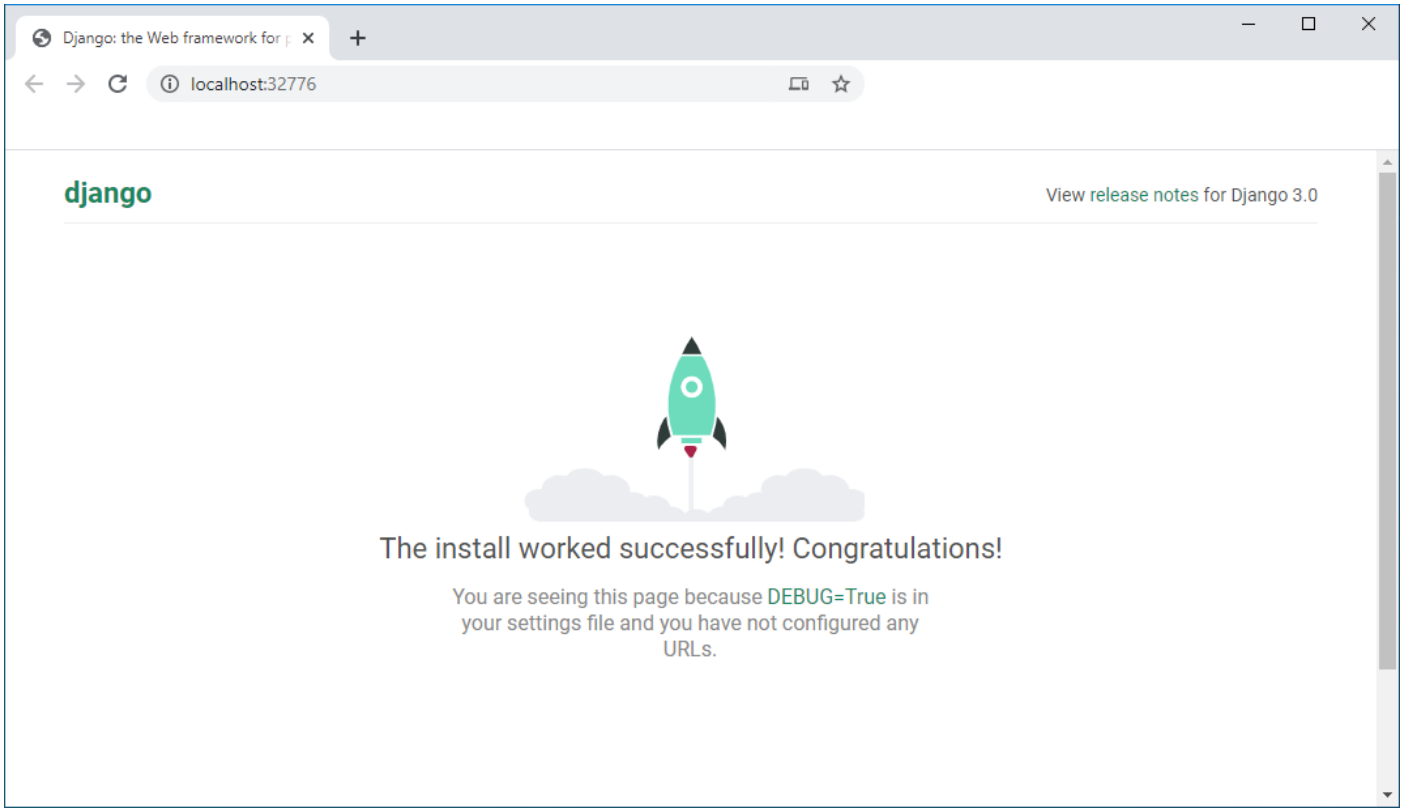
   > **注意**：如果已按照Django教程的“ 创建Django应用程序” (https://code.visualstudio.com/docs/python/tutorial-django#_create-a-django-app)部分中的**说明**创建了应用程序项目，则可以在任意位置 `views.py` 或任意位置设置断点。

2. 导航至“**运行和调试**”，然后选择“ **Docker：Python-Django**”。



3. 使用 F5 键开始调试。
    - Docker镜像构建。
    - Docker容器运行。
    - python调试器命中断点 `manage.py` 。
4. 跨过此行一次。
5. 导航到**调试控制台**并输入 `os.environ["DJANGO_SETTINGS_MODULE"]`
6. 查看输出后，单击继续。

Docker扩展会将您的浏览器启动到随机映射的端口：



提示：要修改Docker构建设置，例如更改image标签，请导航至任务中 `.vscode -> tasks.json` 的 `dockerBuild` 属性下方 `docker-build` 。在文件（ Ctrl + Space ）中使用IntelliSense 来显示所有其他有效指令。

下一步

你完成了！现在您的容器已准备就绪，您可能需要：

- 了解有关在容器中调试Python的信息 (/docs/containers/debug-python)