How to run Python scripts using Docker

Running Python scripts is one of the most common tasks in automation. However, managing dependencies across multiple systems can be a challenge. That's where Docker comes in.

Running a Python script is one of the most common tasks in automation. However, managing dependencies across different systems can be a challenge. That's where Docker comes in. Docker allows you to package a Python script along with all the necessary dependencies into a container, ensuring it runs the same way on every machine. Today's article will walk you through the process of creating an actual Python script and running it inside a Docker container.

Write Python script

Create a project directory to store your Python script and Dockerfile. Once created, navigate to this directory using the cd command :

```
mkdir docker_file_organizer cd docker_file_organizer
```

Create a script called 'organize_files.py' to scan a directory and group files into directories based on their extensions:

```
nano organize_files.py
```

Paste the following code into the 'organize_file.py' file. Here, we use two built-in Python modules, named **os** and **shutil** , to handle files and create directories dynamically:

```
import os import shutil SOURCE_DIR = "/files" def organize_by_extension(directory
?
 {ext}/") except Exception as e: print(f"Error organizing files: {e}") if __name_
```

In this script, we sort the files in a given directory based on their extensions. We use the **os** module to list the files, check if each entry is a file, extract the file extensions, and create directories named after those extensions (if they don't already exist). Then, we use the **shutil** module to move each file into its corresponding directory. For each move, a message showing the new location of the file is output.

Create Dockerfile

Now, let's create a Dockerfile to define the environment in which the script will run:

```
FROM python:latest LABEL maintainer="you@example.com" WORKDIR /usr/src/app COPY
```

We use this Dockerfile to create a container in Python , add a script to it, and make sure the script runs automatically when the container starts:

Build Docker image

Before you can build a Docker image, you first need to install Docker. Then, run the following command to package everything into a Docker image:

```
sudo docker build -t file-organizer .
```

It reads the Dockerfile, combines the Python setup and scripts so they are ready to run in a single container image:

Create a sample folder with the files

To see the script in action, create a test directory called 'sample_files' with a few files of different types. Create these files just to clutter the directory a bit and see how the Python script handles it:

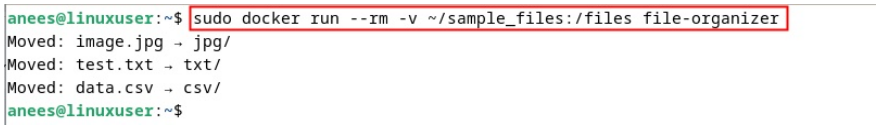
```
mkdir ~/sample_files touch ~/sample_files/test.txt touch ~/sample_files/image.jp
```

Run scripts inside Docker

Finally, run the Docker container and mount the sample directory into it. The **-v** flag mounts the local '~/sample_files' directory into the '/files' directory in the container, allowing the Python script to read and sort the files on your host:

```
docker run --rm -v ~/sample_files:/files file-organizer
```

Here, the article uses the **--rm** option to automatically delete the container after it finishes running, helping to save disk space:



Finally, use the tree command to check if the files are sorted into directories based on their extensions:

```
tree sample_files
```

Note : The tree command does not come pre-installed on most systems. You can easily install it using a package manager like apt on Ubuntu , brew on macOS, etc.

You finished reading the article "**How to run Python scripts using Docker**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com