

## **Dockerized Python Application Execution on a Python-less VM**

### **1. Introduction**

Modern applications often depend on specific runtime environments and libraries. Installing and managing these dependencies directly on a system can lead to compatibility issues, security risks, and deployment complexity.

This project demonstrates how **Docker** can be used to package a **Python application along with all its dependencies** into a single container image that can be executed on a **Virtual Machine (VM) without Python installed** on the host operating system.

### **2. Objective**

The main objectives of this project are:

- To create a Docker container that includes:
  - A Python runtime
  - Required Python libraries
  - A custom Python application
- To execute the Python program on a VM where Python is **not installed**
- To demonstrate environment isolation and portability using Docker

### **3. Problem Statement**

Running Python applications on multiple systems requires:

- Installing Python
- Managing package versions
- Handling dependency conflicts

In environments such as:

- Minimal VMs
- Secure systems
- Production servers

installing Python directly may be **undesirable or restricted**.

This project solves the problem by using **containerization**, where the application and its runtime are bundled together.

## **4. Tools and Technologies Used**

### **Tool / Technology Purpose**

Docker	Containerization platform
Python 3.11	Application runtime
Dockerfile	Image build configuration
Virtual Machine	Execution environment
Linux OS	Host operating system

## **5. System Requirements**

### **5.1 Host Machine (VM)**

- Operating System: Linux / Windows
- Docker Installed
- Python: **Not required**

### **5.2 Docker Image**

- Base Image: python:3.11-slim
- Python Runtime: Included inside container
- Required Libraries: Installed via pip

## **6. Project Structure**

```
my-python-app/
|
├── Dockerfile
├── app.py
└── requirements.txt
```

## **7. Python Application Description**

The Python application (app.py) is a simple program designed to demonstrate execution inside a Docker container.

Example functionality:

- Prints a message to verify successful execution
- Can be extended to include networking, IoT logic, or data processing

## **8. Dockerfile Explanation**

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt || true
COPY app.py .
CMD ["python", "app.py"]
```

### **Explanation:**

- **FROM:** Uses an official Python image with minimal OS footprint
- **WORKDIR:** Sets the working directory inside the container
- **COPY:** Transfers application files into the container
- **RUN:** Installs required Python libraries
- **CMD:** Defines the command executed when the container starts

## **9. Docker Image Build Process**

Command used to build the image:

```
docker build -t my-python-container .
```

This command:

- Reads instructions from the Dockerfile
- Downloads the base image
- Installs dependencies

- Packages the application into a single image

## **10. Container Execution**

Command used to run the container:

```
docker run --rm my-python-container
```

- `--rm`: Automatically removes the container after execution
- The Python program runs inside the container environment

## **11. Verification on Python-less VM**

To verify that Python is not installed on the VM:

```
python --version
```

Result:

```
command not found
```

Despite this, the container executes the Python program successfully, proving that the runtime is fully encapsulated within the Docker image.

## **12. Advantages of Dockerized Python Applications**

- No dependency installation on host OS
- Consistent behavior across environments
- Improved security through isolation
- Easy deployment and scalability
- Reduced configuration errors

## **13. Limitations**

- Docker must be installed on the host system
- Slight overhead compared to native execution
- Requires basic knowledge of container management

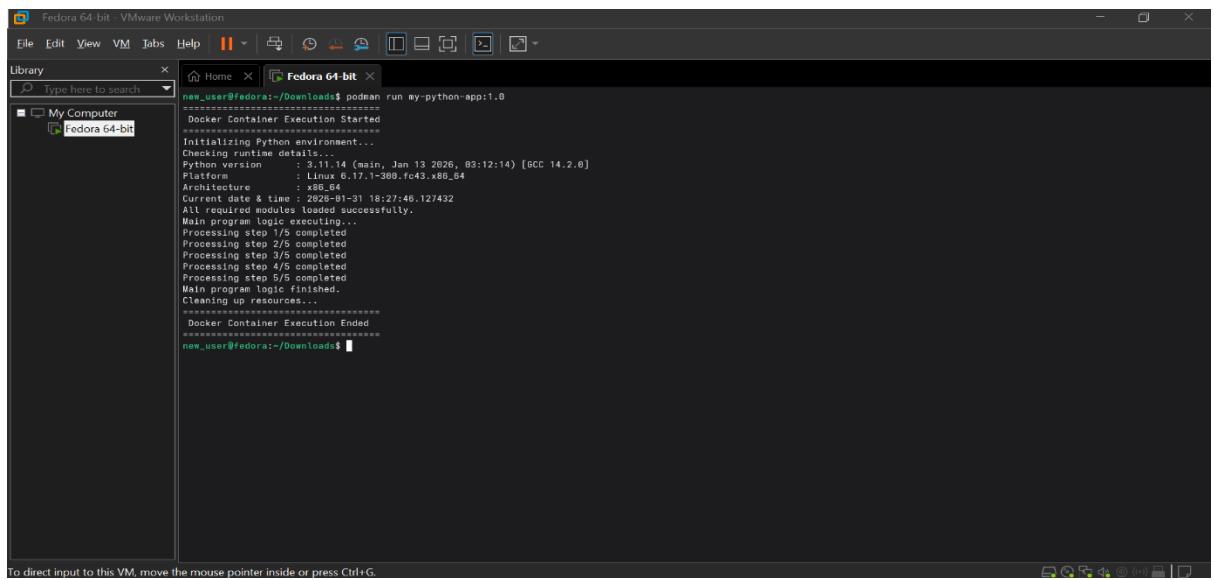
## **14. Future Enhancements**

- Add logging and monitoring support
- Convert project to Docker Compose
- Implement non-root container execution
- Deploy container to cloud platforms
- Add CI/CD pipeline for automated builds

## 15. Conclusion

This project successfully demonstrates the use of Docker to execute a Python application on a system without Python installed. By containerizing the application and its dependencies, the solution ensures portability, consistency, and ease of deployment across different environments.

## 16. Screenshots

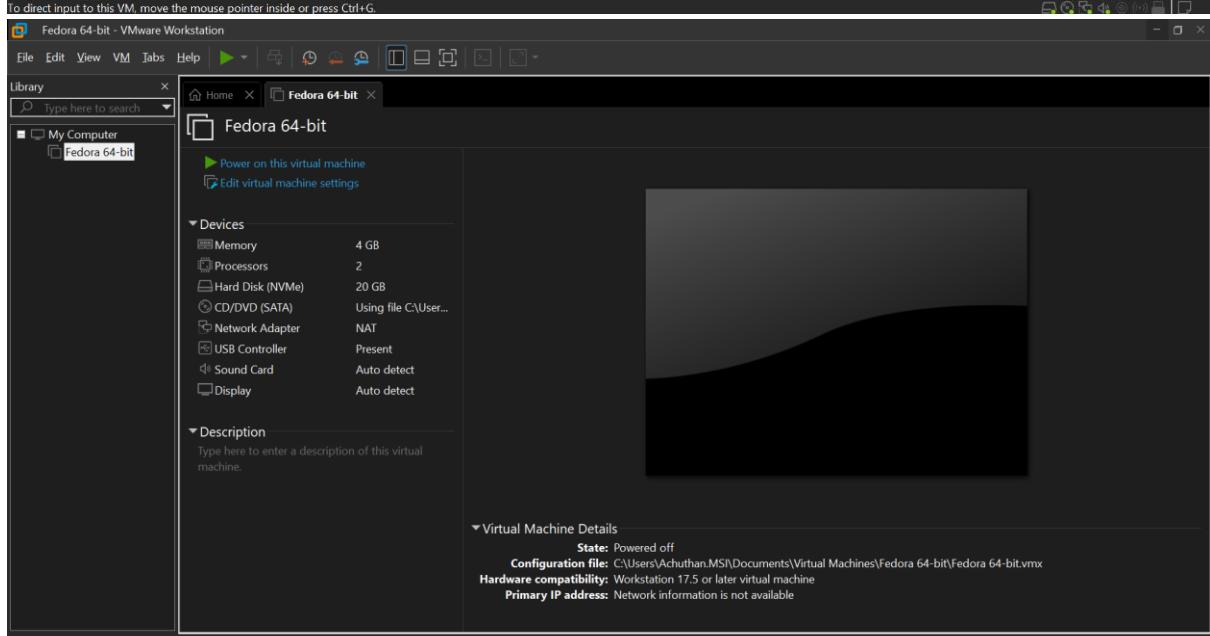
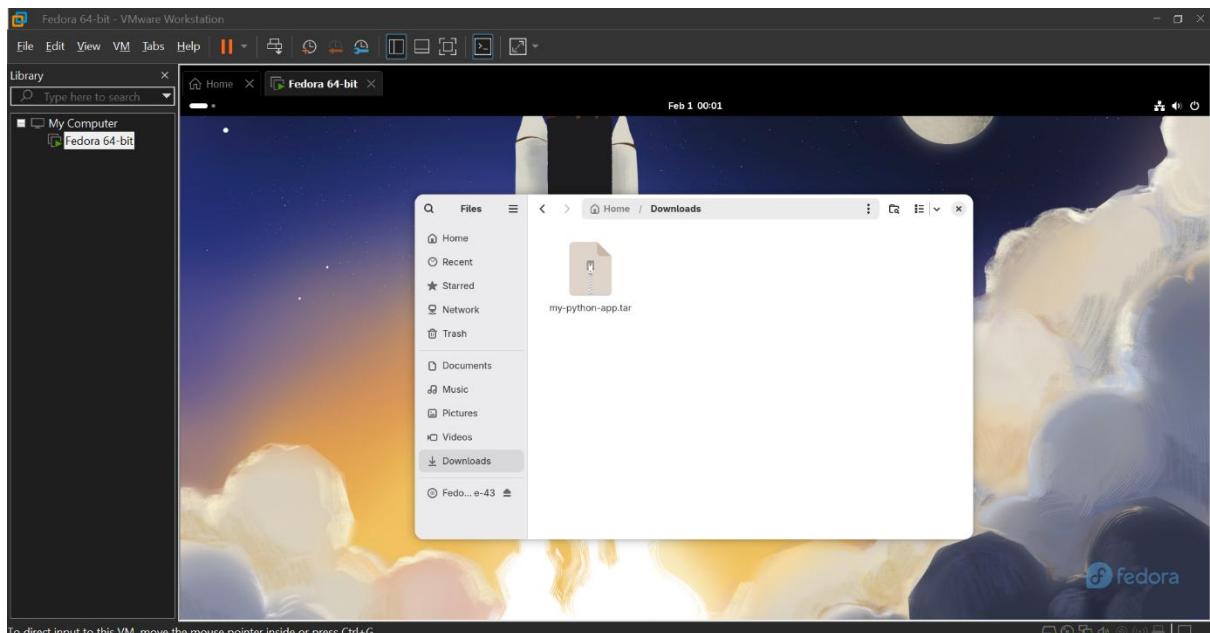


```

Fedora 64-bit - VMware Workstation
File Edit View VM Tabs Help || Type here to search
Library Home Fedora 64-bit
My Computer
new_user@fedora:~/Downloads$ podman run my-python-app:1.0
=====
Docker Container Execution Started
Initializing Python environment...
Dockerizing runtime dependencies...
Python version      : 3.11.14 (main, Jan 13 2026, 03:12:14) [GCC 14.2.0]
Platform           : Linux 6.17.1-300.fc43.x86_64
Architecture       : x86_64
Current date & time: 2026-01-31 18:27:46.127432
All required modules loaded successfully.
Main program logic executing...
Processing step 1/5 completed
Processing step 2/5 completed
Processing step 3/5 completed
Processing step 4/5 completed
Processing step 5/5 completed
Main program logic finished.
Cleaning up resources...
=====
Docker Container Execution Ended
=====
new_user@fedora:~/Downloads$ 

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.



```
PS D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container> dir

Directory: D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container

Mode                LastWriteTime       Length Name
----                <-----           ----- 
-a----   31-01-2026     17:09            914 app.py
-a----   31-01-2026     17:19             84 Dockerfile
```

```

PS D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container> docker build -t my-python-app:1.0 .
[+] Building 13.8s (8/8) FINISHED
--> [internal] Load build definition from Dockerfile
--> => transferring dockerfile: 121B
--> [internal] Load metadata for docker.io/library/python:3.11-slim
--> [internal] Load .dockerignore
--> => transferring context: 2B
--> [1/3] FROM docker.io/library/python:3.11-slim@sha256:5be45dbade29beb6886af6b438fd7e0b4eb7b611f39ba62b430263f82de36d2
--> => resolve docker.io/library/python:3.11-slim@sha256:5be45dbade29beb6886af6b438fd7e0b4eb7b611f39ba62b430263f82de36d2
--> => sha256:3e731abb5c1dd05aeef62585d392d31ad26089dc4c031730e5ab0225aef80b3f2 14.36MB / 14.36MB
--> => sha256:5b09819094bb89d5b2416ff2fb03f68666a5372c358cf22f2b62d7f6660d906 1.29MB / 1.29MB
--> => sha256:119d43eec815e5f9a47da3a7d59454581ble204b0c34db86f171b7ceb3336533 29.77MB / 29.77MB
--> => extracting sha256:119d43eec815e5f9a47da3a7d59454581ble204b0c34db86f171b7ceb3336533
--> => extracting sha256:5b09819094bb89d5b2416ff2fb03f68666a5372c358cf22f2b62d7f6660d906
--> => extracting sha256:3e731abb5c1dd05aeef62585d392d31ad26089dc4c031730e5ab0225aef80b3f2
--> => extracting sha256:0b2bf04f68e9f306a8a83f57c6ced322a23968bf3d5acebc07e055c090240826
--> [internal] Load build context
--> => transferring context: 949B
--> [2/3] WORKDIR /app
--> [3/3] COPY app.py .
--> exporting to image
--> => exporting layers
--> => exporting manifest sha256:a36ee690cd4ac72758167072cf15746a0bde96f865db7dd8e316ac5673183e6b
--> => exporting config sha256:9677c28348acd9233eda9b9b95d9c2aa758e290ba29b18ec756fd7f435d01d
--> => exporting attestation manifest sha256:d6e29142dfea6aa35a377b10369a4cd8c0d59da49c0624c18b72c2cff9d99
--> => exporting manifest list sha256:a3b9ccff1a1le33477139064f282d4c7ad85569c5aaef5ec3c2a854986935ae82
--> => naming to docker.io/library/my-python-app:1.0
--> => unpacking to docker.io/library/my-python-app:1.0

```

```

PS D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container> docker images
IMAGE          ID                  DISK USAGE   CONTENT SIZE  EXTRA
my-python-app:1.0  a3b9ccf1a1le    186MB        45.4MB
PS D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container> docker run my-python-app:1.0
=====
Docker Container Execution Started
=====
Initializing Python environment...
Checking runtime details...
Python version      : 3.11.14 (main, Jan 13 2026, 03:12:14) [GCC 14.2.0]
Platform           : Linux 6.6.87.2-microsoft-standard-WSL2
Architecture        : x86_64
Current date & time : 2026-01-31 12:37:46.639621
All required modules loaded successfully.
Main program logic executing...
Processing step 1/5 completed
Processing step 2/5 completed
Processing step 3/5 completed
Processing step 4/5 completed
Processing step 5/5 completed
Main program logic finished.
Cleaning up resources...
=====
Docker Container Execution Ended
=====
PS D:\Assignments\SEM 6\DSC\ASSIGNMENT_1\DOCKER\my-python-container> docker save my-python-app:1.0 -o my-python-app.tar

```

