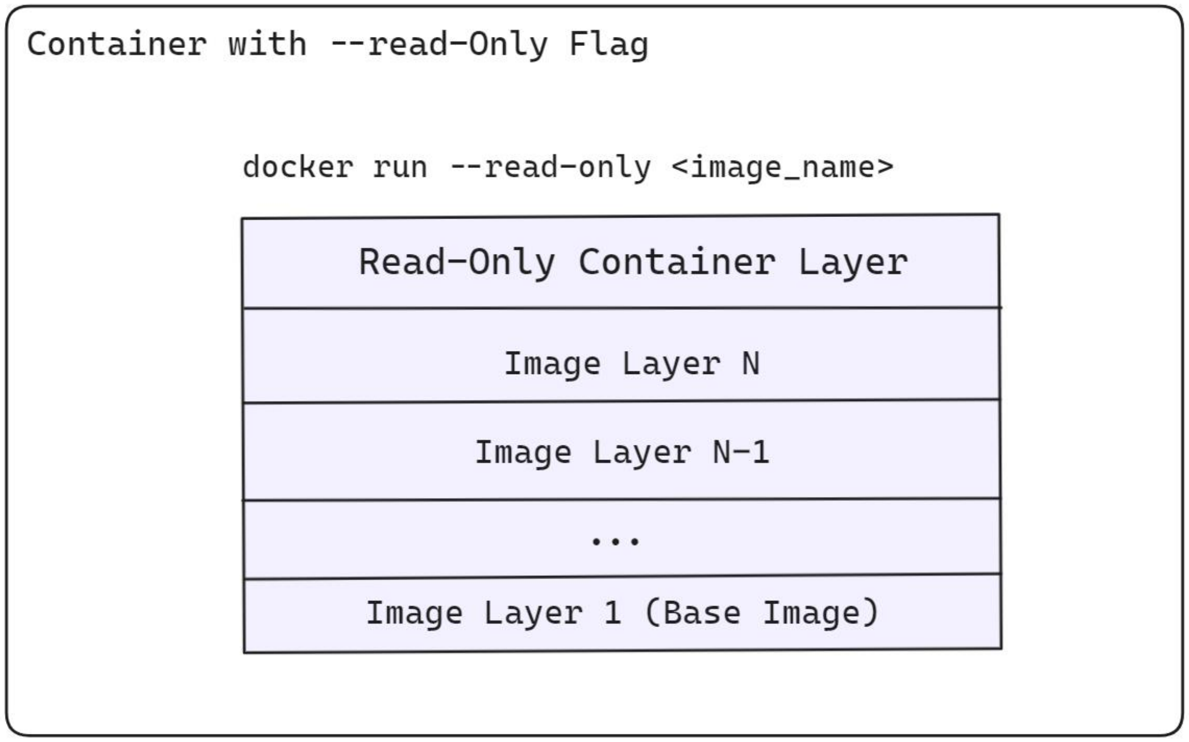
Setting Up a Docker Container with a Read-Only File System

Ensuring the security and data integrity of Docker containers is critical in any robust DevOps environment. One effective measure is to run Docker containers with a read-only file system. This prevents any modifications to the container's file system during runtime, mitigating risks associated with unauthorized changes or accidental data corruption.


Container with --read-Only Flag
docker run --read-only <image_name>
Read-Only Container Layer
Image Layer N
Image Layer N-1
...
Image Layer 1 (Base Image)

Scenario Overview

Imagine you are a DevOps engineer at a company that values security and data integrity. One of your tasks is to ensure that certain Docker containers run with a read-only file system to prevent any modifications to the container’s file system during runtime. You will set up a Docker container with a read-only file system and verify its behavior.

Objectives

- Set up a Docker container with a read-only file system.
- Verify that the container operates as expected and that the file system is indeed read-only.

Required Steps

Step 1: Create a Docker Image

Create a Dockerfile: Create a file named Dockerfile and add the following content:

```
FROM alpine:latest

# Create a directory and add a sample file
RUN mkdir /data && echo "This is a read-only test file" > /data/test.txt

# Set the working directory
WORKDIR /data

CMD ["sh"]
```

This Dockerfile creates a Docker image based on the Alpine Linux distribution, creates a directory /data and adds a sample file test.txt to it, and sets the working directory to /data.

Build the Docker Image: Open your terminal, navigate to the directory containing the Dockerfile, and run the following command:

```
docker build -t readonly-test .
```

Step 2: Run the Docker Container with a Read-Only File System

Run the Docker Container: Use the --read-only flag to start the container with a read-only file system:

```
docker run --rm -it --read-only readonly-test
```

Verify the Read-Only File System: Once inside the container, attempt to modify the file or create a new file:

```
# Try to modify the existing file

echo "Attempting to write to a read-only file system" >> /data/test.txt


# Try to create a new file

touch /data/newfile.txt
```

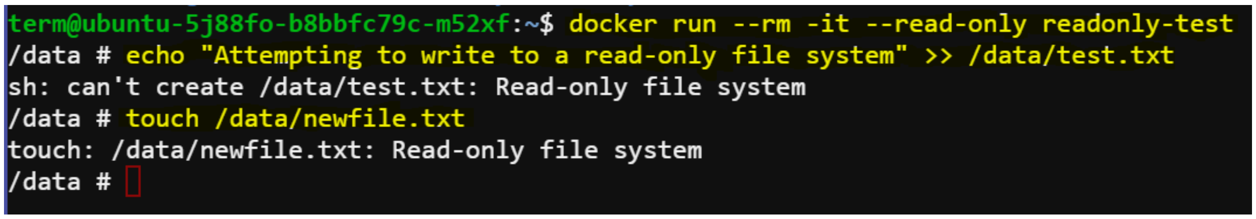
Both commands should fail, indicating that the file system is indeed read-only.

Step 3: Verify Read-Only Behavior

Check for Errors: The commands above should result in error messages similar to:

sh: can't create /data/test.txt: Read-only file system

touch: /data/newfile.txt: Read-only file system



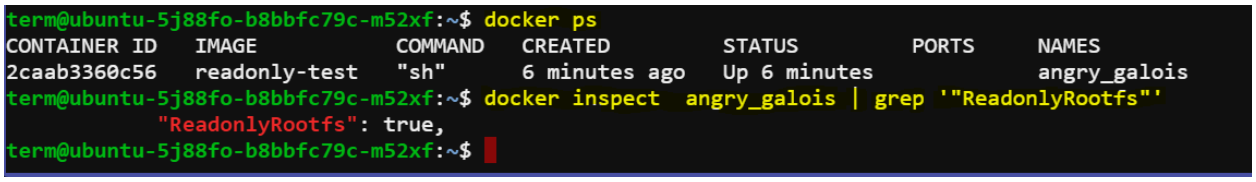
```
term@ubuntu-5j88fo-b8bbfc79c-m52xf:~$ docker run --rm -it --read-only readonly-test
/data # echo "Attempting to write to a read-only file system" >> /data/test.txt
sh: can't create /data/test.txt: Read-only file system
/data # touch /data/newfile.txt
touch: /data/newfile.txt: Read-only file system
/data # 
```

Confirm File System Status: You can further confirm the read-only status by inspecting the file system options:

```
docker inspect  container_name | grep '"ReadonlyRootfs"'
```

Run this command in a new terminal so that the container state remains running. Replace container_name with the actual container name or ID.

This command should output true, indicating that the root file system is indeed read-only.



```
term@ubuntu-5j88fo-b8bbfc79c-m52xf:~$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED         STATUS         PORTS     NAMES
2caab3360c56   readonly-test   "sh"      6 minutes ago   Up 6 minutes             angry_galois
term@ubuntu-5j88fo-b8bbfc79c-m52xf:~$ docker inspect  angry_galois | grep '"ReadonlyRootfs"'
            "ReadonlyRootfs": true,
term@ubuntu-5j88fo-b8bbfc79c-m52xf:~$
```

Conclusion

By following these steps, you have successfully set up and verified a Docker container with a read-only file system. This configuration helps enhance security and data integrity by preventing any modifications to the container's file system during runtime.