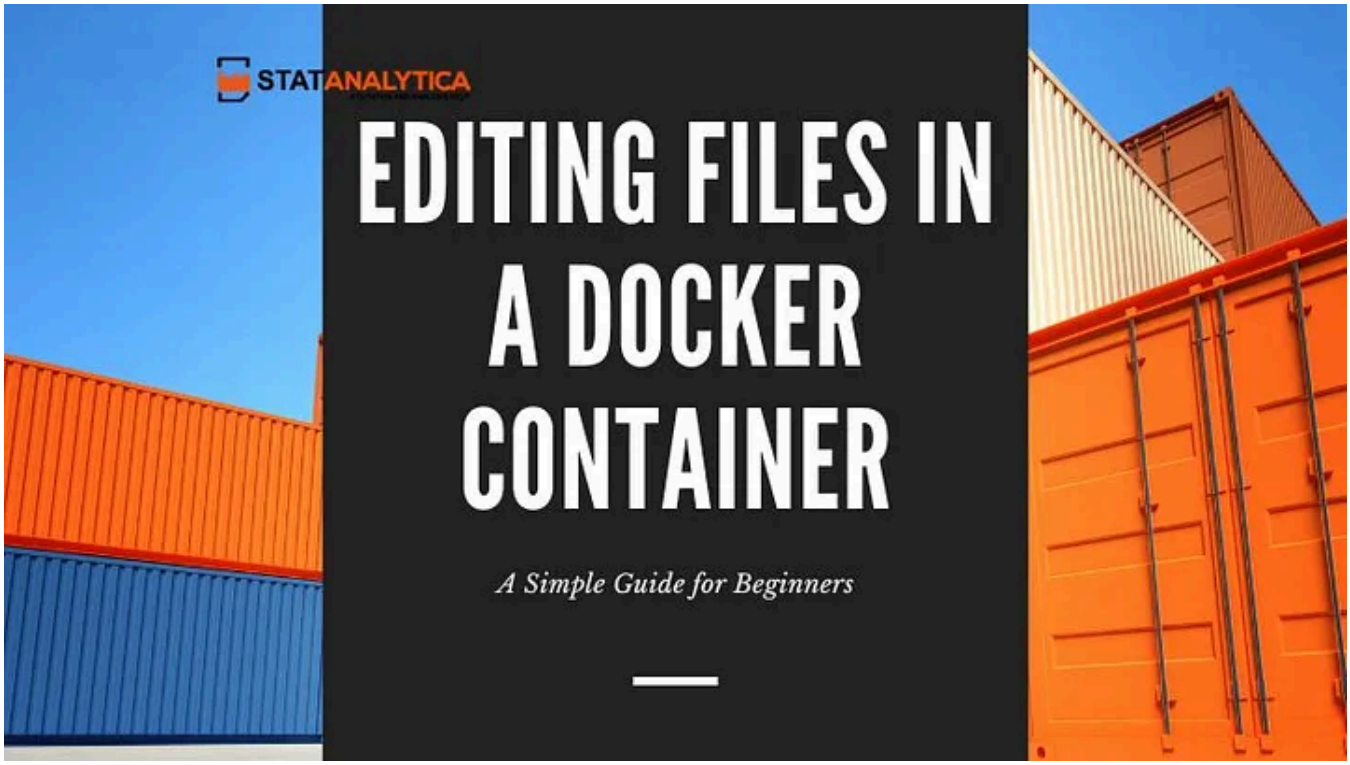
Editing Files in a Docker Container: A Simple Guide for Beginners

Akshay · Follow

4 min read · Feb 3, 2024

Listen Share

Docker containers have become integral to modern software development, allowing developers to create and deploy applications consistently across different environments. One common task developers often encounter is editing files within a Docker container. In this guide, we will walk you through the process step by step, using straightforward language and avoiding complex technical terms.



Understanding Docker

Before diving into the details of editing files in a Docker container, let's briefly understand what Docker is. Docker is a platform that helps developers create,

deploy, and run applications in containers. Containers are lightweight, portable, and isolated environments that package everything an application requires to run, including code, runtime, system tools, and libraries.

Getting Started

1. Install Docker

To begin, make sure you have Docker installed on your system. Download the version for your operating system from the official Docker website. (https://www.docker.com/get-started).

2. Run a Docker Container

After installing Docker, open a terminal or command prompt and run a simple container. Use the following command:

```
docker run hello-world
```

This command downloads a small test image, runs a container from it, and verifies that your Docker installation works correctly.

Accessing the Terminal Inside a Container

1. Run an Interactive Container

You must access its terminal to edit files within a Docker container. Run an interactive container with a command like this:

```
docker run -it ubuntu
```

This example uses the Ubuntu image, a popular choice for containers. The -it flag allows you to interact with the container's terminal.

2. Explore the Container's Terminal

Once inside the container, you'll see a different command prompt, indicating that you are now within the container's environment. You can explore the file system and execute commands like a regular one.

1. Install a Text Editor

Most Docker images don't come with a text editor pre-installed. You can install one using the container's package manager. For example, on ubuntu, you can install nano with the following:

apt-get update

apt-get install nano

2. Open a File

Now that you have a text editor installed open a file for editing. You can use the text editor's command, followed by the file name. For example:

nano myfile.txt

3. Make Edits

Like many text editors, Nano provides a simple interface for making edits. Use the arrow keys to navigate, make changes, and save your edits.

4. Save and Exit

After making changes, save the file by pressing **Ctrl + O** (for Write Out) and press **Enter**. To exit, press **Ctrl + X.** These commands are displayed at the bottom of the nano editor for reference.

Copying Files In and Out of a Container

Copying Files Into a Container

Sometimes, you may need to copy files from your host machine into the Docker container. Use the **docker cp** command to achieve this. For example:

docker cp myfile.txt container_id:/path/to/destination

Replace **container_id** with the actual ID or name of your running container.

2. Copying Files Out of a Container

Copy the files from a container to your host machine and use the reverse syntax:

docker cp container_id:/path/to/file.txt /path/on/host

Persisting Changes

1. Commit Changes to a New Image

If you want to persist the changes you made in the container, you can create a new Docker image. First, exit the container, and then use the **docker commit** command:

```
docker commit container_id my-custom-image
```

It creates a new image named **my-custom-image** based on your changes in the container.

2. Run a Container from the New Image

Now, you can run a new container based on the image you just created:

```
docker run -it my-custom-image
```

This container will have all the changes you made in the previous container.

Using Docker Volumes

1. Understanding Docker Volumes

Docker volumes allow you to store data outside of containers. They allow you to share files between your host machine and containers.

2. Create a Docker Volume

To use a volume, create one using the **docker volume create** command:

```
docker volume create my-volume
```

3. Mount a Volume in a Container

When running a container, you can mount a volume to a specific route inside the container:

```
docker run -it -v my-volume:/path/in/container ubuntu
```

4. Edit Files in the volume

Any changes to files inside the mounted volume are reflected on both the host machine and the container. Edit files in the mounted volume using the steps mentioned earlier.

Conclusion

Modifying files within a Docker container can be a feasible undertaking. Following these straightforward procedures, you can confidently modify files, transfer them into and out of containers, preserve changes across images, and utilize Docker volumes to facilitate smooth collaboration between your host system and containers. As you continue to explore Docker and containerization, remember that practice is vital, and the more comfortable you become with these essential tasks, the more efficiently you'll navigate the world of containerized applications.

Docker Programming Programming Languages Programming Tips Coding

Follow

Written by Akshay

60 Followers

More from Akshay