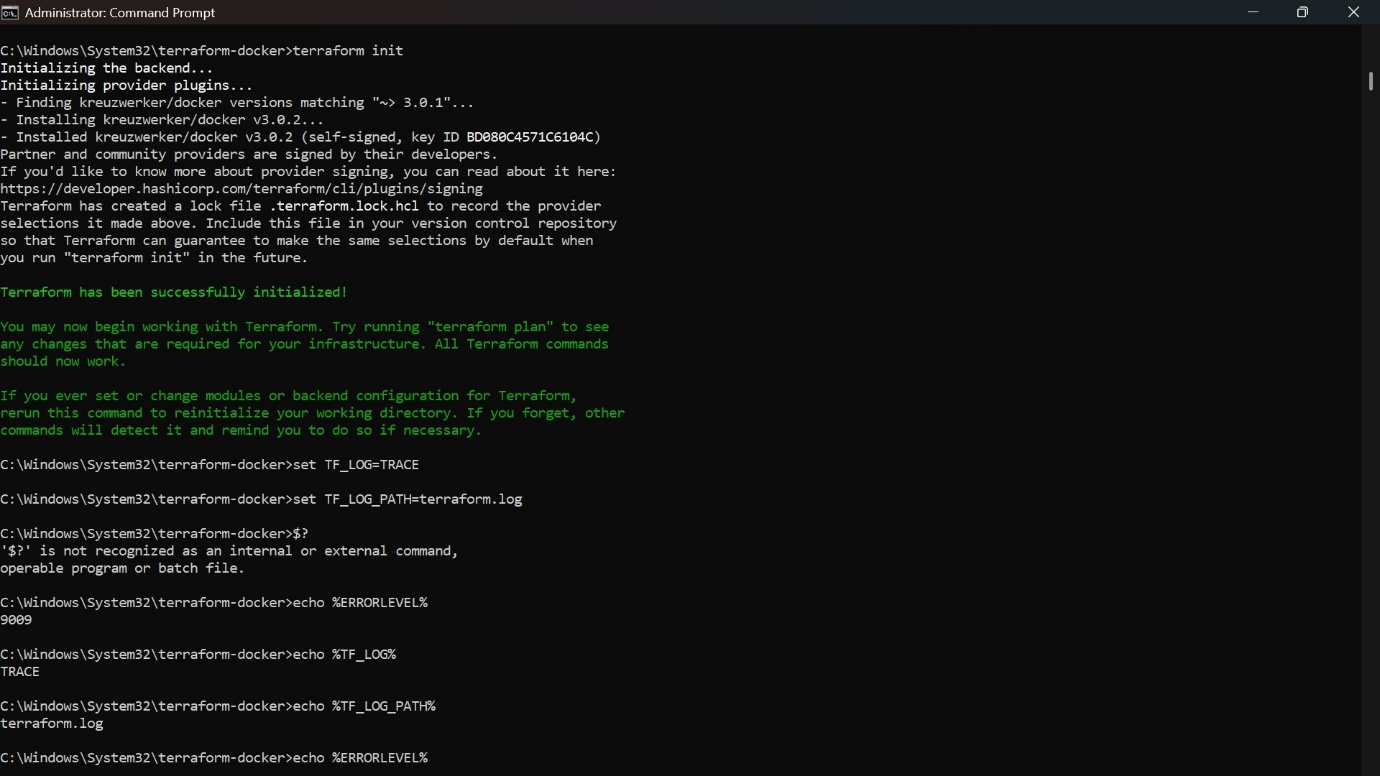
**🚀 Terraform + Docker Lifecycle Overview**

The project followed the standard **Terraform workflow** to provision and manage Docker resources on a local machine. The steps were:

1. **Write Configuration**: Define Docker image and container in main.tf.
2. **Initialize Terraform**: Download necessary providers.
3. **Plan the Changes**: Preview the resources Terraform will create.
4. **Apply the Changes**: Terraform spins up the container.
5. **Access the Application**: Check the service is running.
6. **Destroy Infrastructure**: Tear down resources when done.

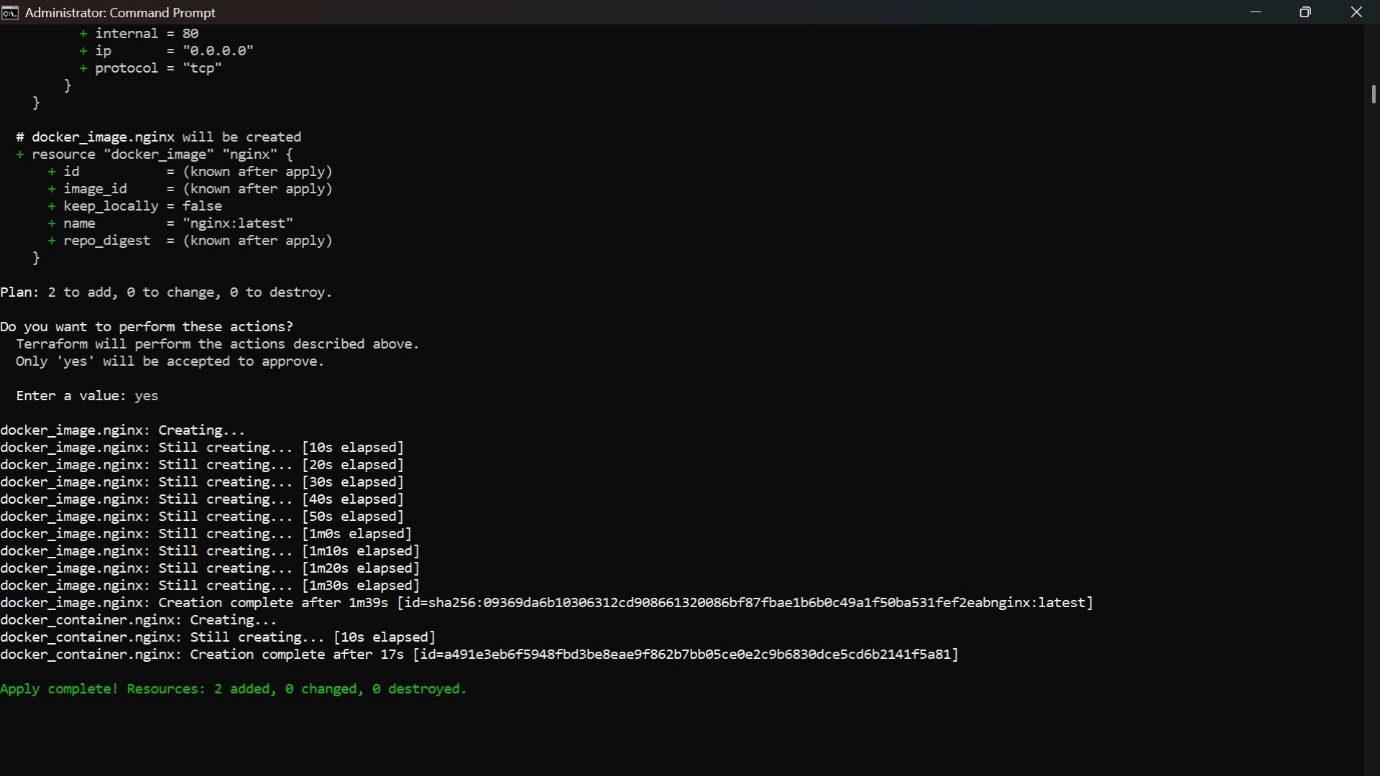
**🖼 Screenshot: init.jpg**

****

📌 **Explanation**:

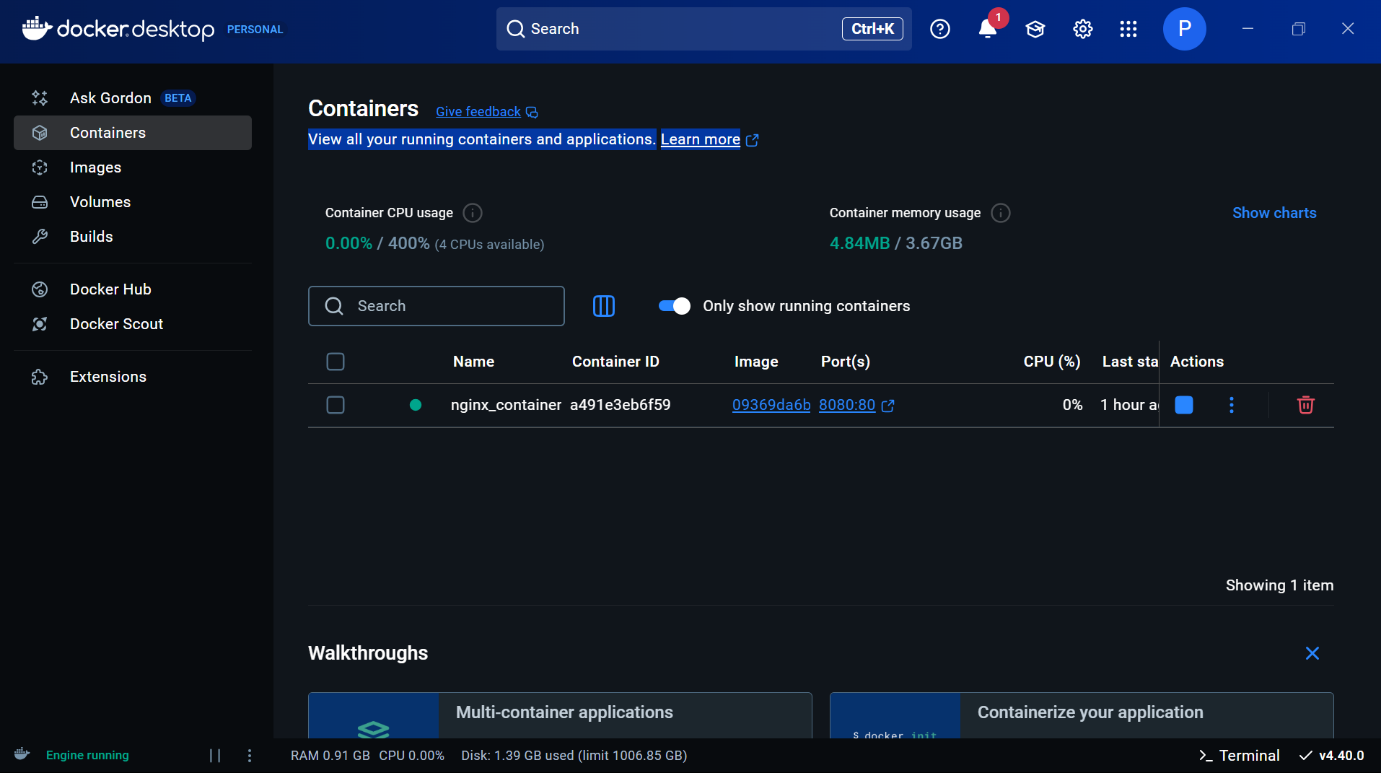
This screenshot shows the terraform init command being executed. It initializes the Terraform working directory and downloads the Docker provider plugin. This is the first step after writing the Terraform configuration to ensure Terraform is ready to manage Docker resources.

**🖼 Screenshot: apply.jpg**

****

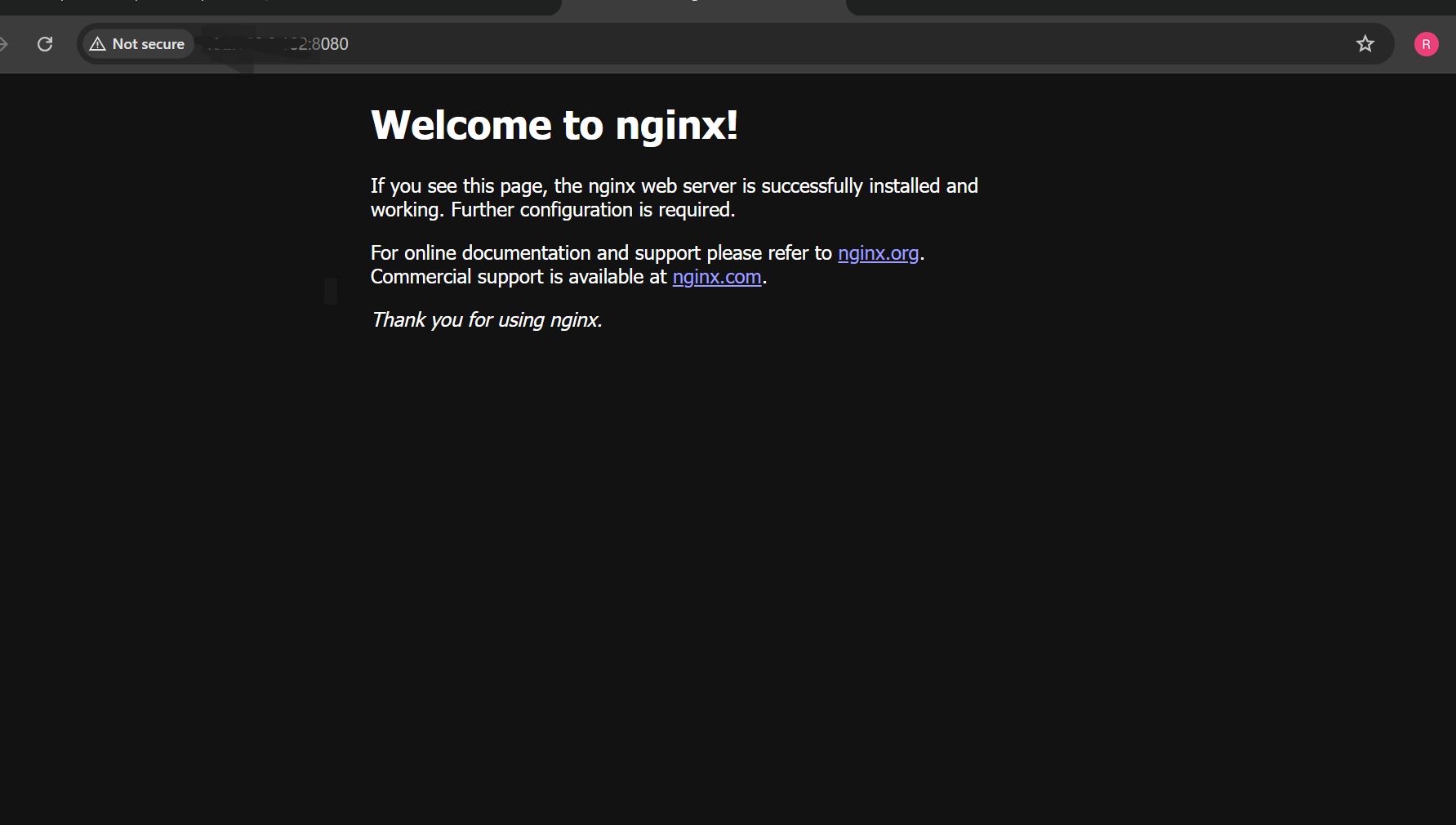
The terraform apply command is shown here. This step provisions the infrastructure defined in main.tf, including pulling the Docker image and creating a container. The command prompts for confirmation before proceeding, ensuring you can review planned actions beforehand.

**🖼 Screenshot: sucessfully spin up container using terraform**

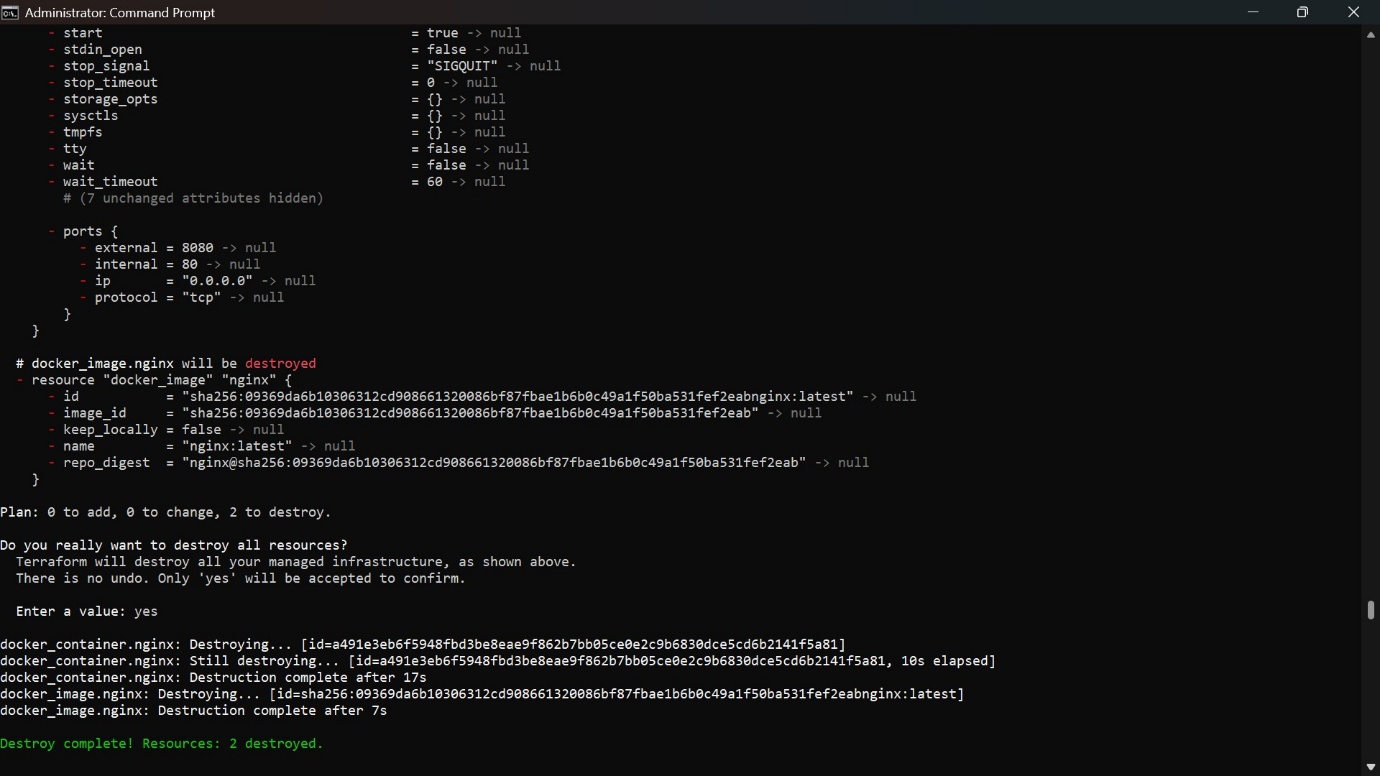
****

Here, we see a successful Terraform run where the Docker container has been spun up. It shows the output after applying the configuration, confirming that both the Docker image and container were created without errors.

**🖼 Screenshot: accessing docker container using ip.png**

****

This screenshot demonstrates accessing the Docker container using the external mapped port (localhost:8080). It confirms that the container is up and running and the web service (like nginx) is accessible via browser or curl.

**🖼 Screenshot: destroy.jpg**

This shows the terraform destroy command in action. It is used to remove all infrastructure created by Terraform, in this case, stopping and removing the Docker container and image. It’s part of the lifecycle for cleanup and testing purposes.