10/29/2024

CLOUD MANAGEMENT

LAB #5: CONTAINERS

JOVITA ANDREWS

Part 1:

What is Docker, and how is it related to containers?

- Docker is a platform that allows developers to create, deploy, and manage applications in containers. Containers are lightweight, standalone packages of software that include everything needed to run an application: code, runtime, system tools, and libraries. Docker provides an easy way to package and distribute applications across various environments consistently, as it abstracts away the specifics of the host operating system.

What is a container image?

- A container image is a standalone, executable package that includes all the necessary components (such as code, runtime, libraries, and environment variables) to run an application. Container images serve as templates for containers; when an image is instantiated, it becomes a running container.

What is a Dockerfile?

- A Dockerfile is a text file that contains a set of instructions for assembling a Docker image. It describes the steps needed to create an image, including specifying the base image, installing necessary packages, copying files, and defining environment variables and commands to run within the container.

What does the docker build command do?

- The docker build command reads the instructions from a Dockerfile and assembles a Docker image based on those instructions. When the command is run in the directory containing the Dockerfile, it generates an image that can then be used to create containers.

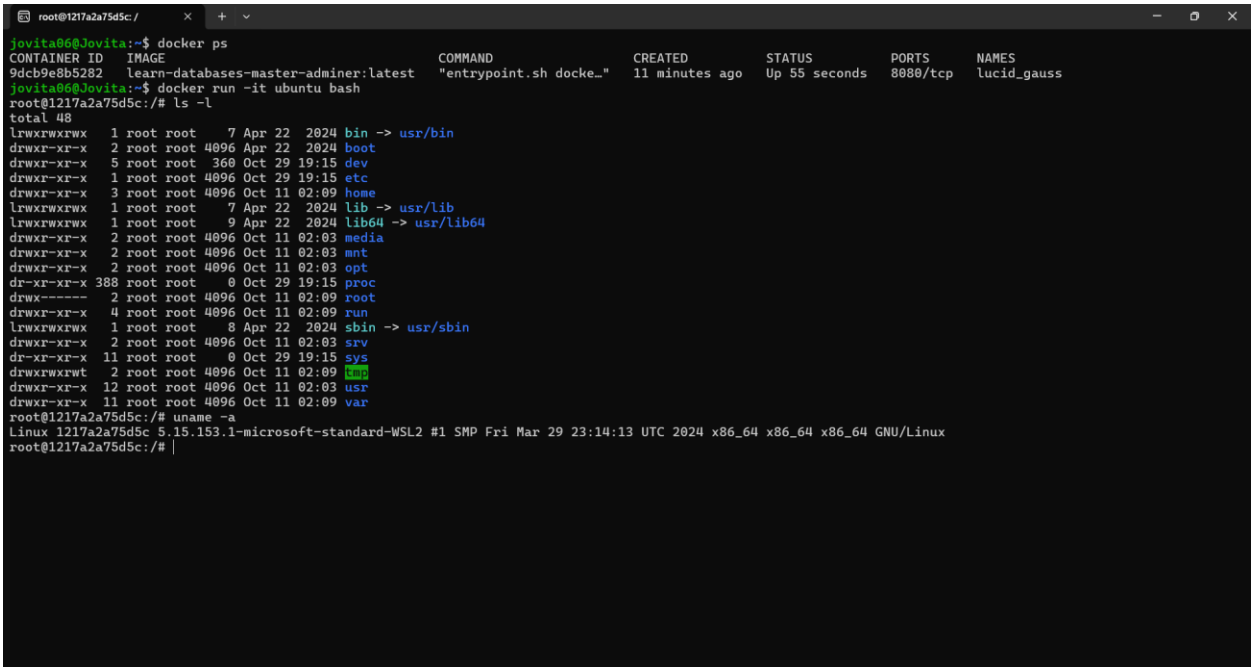
What is a web app, and why are they useful?

- A web app is an application that users interact with through a web browser. It operates on a web server and processes data over the internet. Web apps are useful because they are accessible from anywhere with an internet connection, are platform-independent, and require no installation on the user's device, making them highly accessible and easy to maintain.

Part 2:

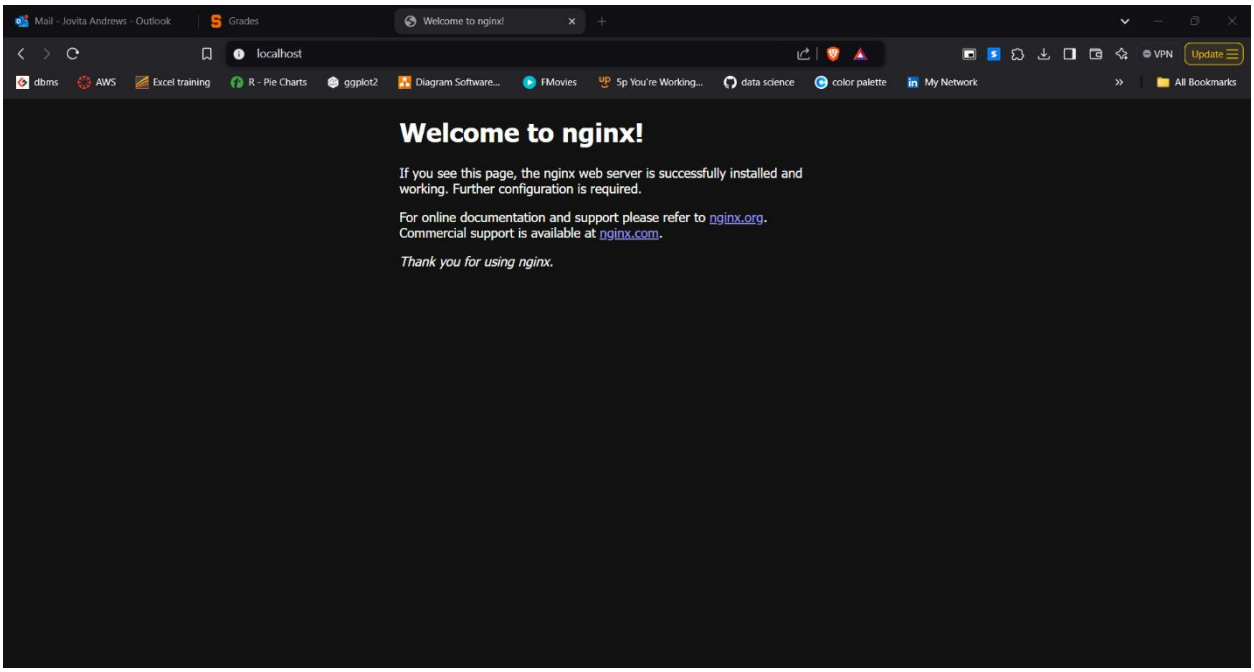
SS01: Screenshot of ls -l and uname -a commands

Description: This screenshot shows the output of the ls -l command, listing files and directories within the root directory of the container, and uname -a, which displays system information about the container, confirming it's an Ubuntu environment.



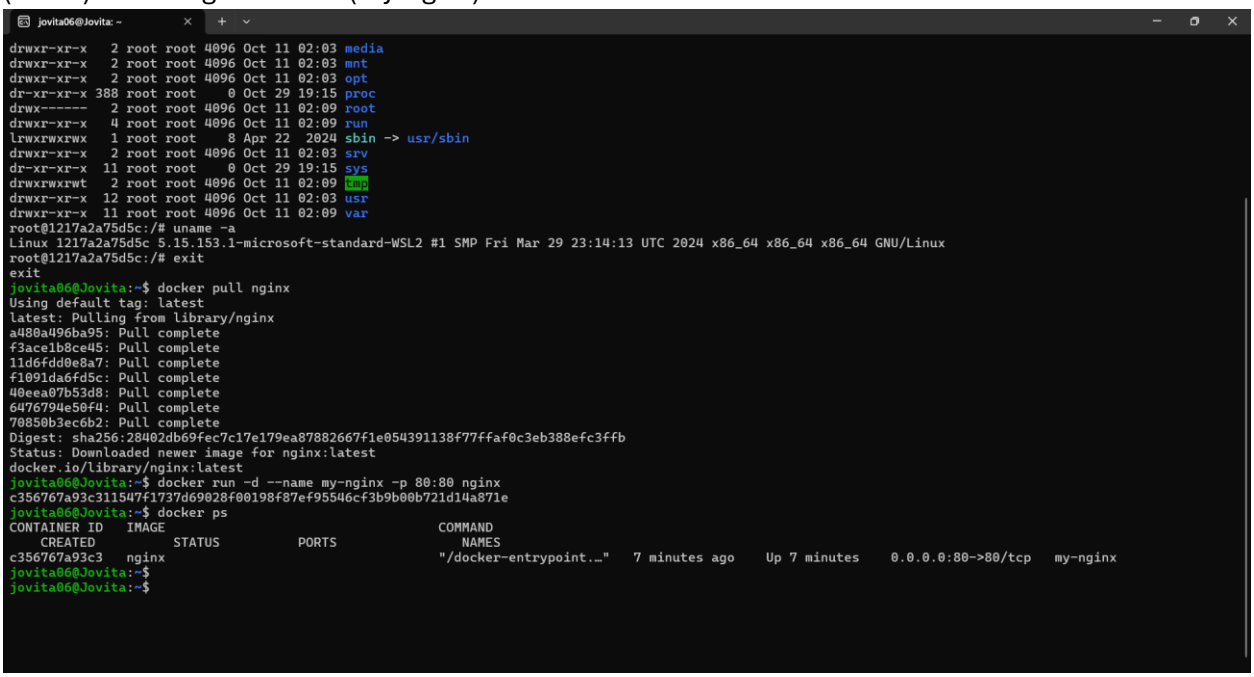
SS02 Screenshot of Nginx Welcome Page:

Description: After running the Nginx container, this screenshot captures the default Nginx welcome page accessed via a web browser. It confirms that the Nginx server is running on port 80 of your VM and is accessible through your host's browser.



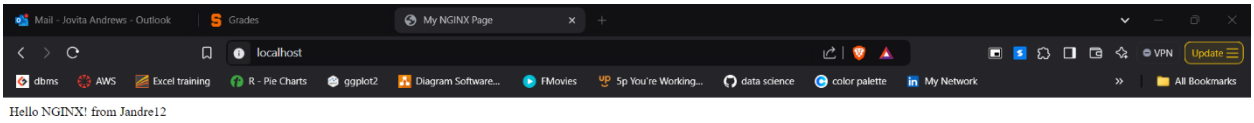
SS03: Screenshot of docker ps Showing Running Nginx Container

Description: This screenshot shows the output of docker ps, listing active Docker containers. It verifies that the Nginx container is running in detached mode with the specified port mapping (80:80) and assigned name (my-nginx).



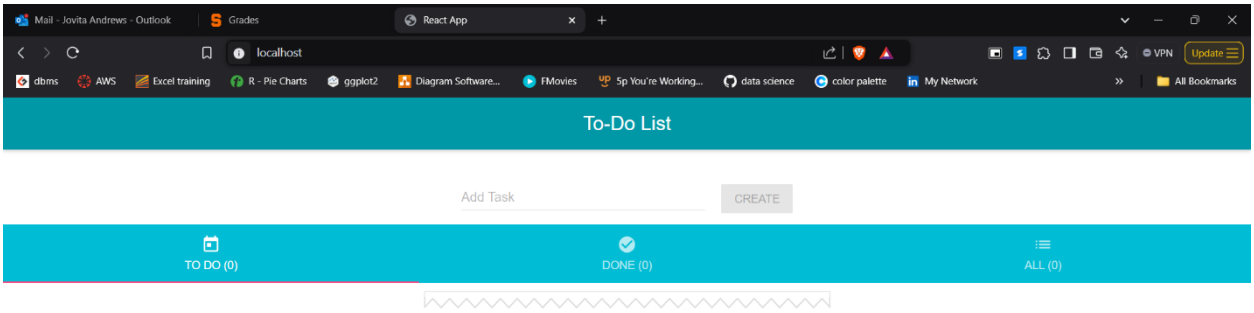
SS04: Screenshot of Customized Nginx Web Page

Description: After creating a custom Dockerfile and index.html, building the Docker image, and running the container, this screenshot captures the customized web page served by Nginx, displaying the personalized message you added in index.html.



SS05: Screenshot of To-Do List Web Application

Description: This screenshot shows the deployed React-based To-Do list application. After building and running the React app inside a Docker container, this page confirms the app is functioning as expected and accessible through your VM's IP.



Part 3:

The main challenge I encountered was downloading Ubuntu. Ultimately, I opted to use the Ubuntu Subsystem, conveniently available in the Windows Store, which allowed me to continue with Lab 5. Aside from this initial hurdle, everything else went smoothly.