**Module 8 Activity 1: Application Deployment**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

November 26th, 2023

Step-by-Step Guide:

1. **Restore Dependencies for project**

A computer screen shot of a black screen

Description automatically generated

**Code running sucessfully-**

A computer screen shot of a black screen

Description automatically generated

1. **Create a Docker file-**

A screenshot of a computer screen

Description automatically generated

1. **Add .dockerignore file-**

A computer screen shot of a black screen

Description automatically generated

1. **Build docker image using vscode-**

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Run the docker container-**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

**Successfully running on port 8080**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

1. **Accessing nodejs application**

Project working on 8080-

A screenshot of a computer

Description automatically generated

Docker ps-

A computer screen shot of a program

Description automatically generated

1. **Push image to docker hub**

A computer screen shot of a program

Description automatically generated

1. **Clean up-**

A screenshot of a computer

Description automatically generated

Removed Container-

A computer screen shot of a program

Description automatically generated

1. **Running the docker container in Terminal**

A computer screen shot of a program

Description automatically generated

Port mapping in Docker, as specified by the **-p** flag in the **docker run** command, establishes a communication bridge between the host machine and the running Docker container. In the above code **-p 8080:4000**, it signifies that port 4000 inside the Docker container is mapped to port 8080 on the host machine. This means that any traffic directed to port 8080 on the host machine is forwarded to port 4000 inside the container. Port mapping is crucial for exposing services and applications running inside containers to the external world, enabling external access to the containerized applications. It allows multiple containers to run on the same host machine without port conflicts, providing a way to isolate and manage network communication for each container.

**Accessing the application-**

A screenshot of a computer

Description automatically generated

Port mapping in Docker, as exemplified by specifying a mapping like **-p 8080:4000** in the **docker run** command, serves to establish a connection between the host machine and the running container. This mapping directs traffic from port 8080 on the host machine to port 4000 within the Docker container, allowing external access to the application hosted inside the container through a web browser or any other client. Essentially, it provides a way for the host machine to communicate with the containerized application by redirecting requests to the designated port, facilitating seamless accessibility to the application running within the isolated environment of the Docker container.

**Interacting with the Container using Terminals CLI:**

A computer screen shot of a program

Description automatically generated

Running the **docker exec -it ift554 /bin/bash** command enables interactive access to a running Docker container named "ift554" by launching a Bash shell inside it. This capability is valuable for tasks such as troubleshooting, debugging, and executing commands within the container. It allows developers and system administrators to inspect the container's internal state, run diagnostic commands, check logs, and perform maintenance tasks. Interacting with a running container facilitates real-time exploration of its environment, aiding in the verification of configurations, installation of software, and validation of data, contributing to efficient container management and development workflows.

**Stopping the container-**

A computer screen shot of a program

Description automatically generated

Executing the **docker stop** command, as in **docker stop ift554**, is crucial for the proper and controlled cessation of a Docker container. This command initiates a graceful shutdown, allowing the containerized application to complete ongoing processes and perform necessary cleanup tasks before termination. Properly stopping a container is fundamental for preserving data integrity, releasing allocated resources, preventing network conflicts, and contributing to a well-managed container lifecycle. It ensures a controlled and orderly process, reducing the risk of data corruption, resource leaks, and other issues associated with abrupt termination. Additionally, it enables subsequent container restarts, offering flexibility in managing the lifecycle of containerized applications within a Docker environment.

Top of Form

**Removing the Container-**

A computer screen shot of a program

Description automatically generated

Using the **docker rm** command, as in **docker rm ift554**, allows for the removal of a Docker container, and there are several reasons why one might want to do this. First and foremost, it helps in efficient resource management by reclaiming the disk space occupied by the container's filesystem, particularly valuable when dealing with containers that generate temporary or large amounts of data during runtime. Removing containers also ensures a clean environment, minimizing the clutter of unused or obsolete containers, and streamlining the Docker host for better performance. Furthermore, removing containers is often part of an iterative development and deployment process where containers are frequently stopped, removed, and recreated to incorporate changes or updates, contributing to a dynamic and agile containerized workflow. Additionally, removing containers is essential for maintaining security hygiene by reducing the potential attack surface and limiting the exposure of unnecessary containers, reinforcing a best practice in container orchestration and management.