CLOUD COMPUTING – 20MCA265

ASSIGNMENT

Submitted To:
Ms. Navyamol K T
Assistant Professor
AJCE

Submitted By:
Benz Baby
S3, RMCA-A
Roll No:32

Containers in Computing

Containers in computing are standalone, lightweight, and executable software packages that encompass all the necessary components to run an application, including code, runtime, system tools, system libraries, and settings. They offer a consistent and isolated environment for software development and deployment, making it simpler to create, test, and deploy software across different environments.

Key Concepts Related to Containers

Containerization: The process of packaging an application and its dependencies into a container image, providing a consistent and portable means of distribution.

Container Image: A standalone, executable package containing an application and its dependencies. These images are used as the basis for creating container instances.

Docker: A prominent containerization platform that offers tools and a runtime environment for managing containers using a client-server architecture.

Kubernetes: A container orchestration platform automating the deployment, scaling, and management of containerized applications.

Container Registry: A repository for storing and distributing container images, including popular options like Docker Hub, Google Container Registry, and Amazon Elastic Container Registry (ECR).

Container Orchestration: The automated management of containerized applications, including deployment, scaling, load balancing, and self-healing, with Kubernetes as a prominent tool.

Microservices: Containers are often employed in a microservices architecture, breaking down applications into smaller, loosely coupled services running in separate containers.

Isolation: Containers offer process and file system isolation, enhancing security by preventing access to other containers' file systems or processes.

Portability: Containerized applications can be developed locally and run in any environment that supports containers without compatibility concerns.

Resource Efficiency: Containers are lightweight, sharing the host operating system's kernel, leading to enhanced resource efficiency compared to traditional virtual machines.

Lifecycle Management: Containers are easily started, stopped, and removed, simplifying application lifecycle management.

Orchestration Tools: In addition to Kubernetes, there are other container orchestration tools like Docker Swarm, Apache Mesos, and Amazon ECS.

Security: Ensuring container security is critical, requiring regular updates, proper isolation, and the use of trusted images.

Anatomy of Containers

Containers encapsulate everything needed to run an application, including:

Container Image: Application code, runtime, system libraries.

Filesystem: Isolated filesystem containing code, runtime, and libraries.

Container Configuration: Environment variables, network settings, exposed ports, and entry point.

Operating System Kernel: Shared with the host system, enhancing lightweight and efficient operation.

Container Runtime: Software responsible for creating and managing containers.

Isolation: Using technologies like namespaces and groups for process and resource isolation.

Network Namespace: Each container has its network stack and interfaces.

Process Isolation: Ensuring processes within one container cannot directly interact with others.

Container Orchestration: Tools like Kubernetes manage multiple containers for deploying, scaling, and managing distributed applications.

Container Registry: Storage and distribution of container images, e.g., Docker Hub.

Building, Running, and Testing a Container Image (with Docker)

Create Your Application: Prepare your application code and dependencies.

Write a Docker file: Create a Docker file specifying how your container image should be built.

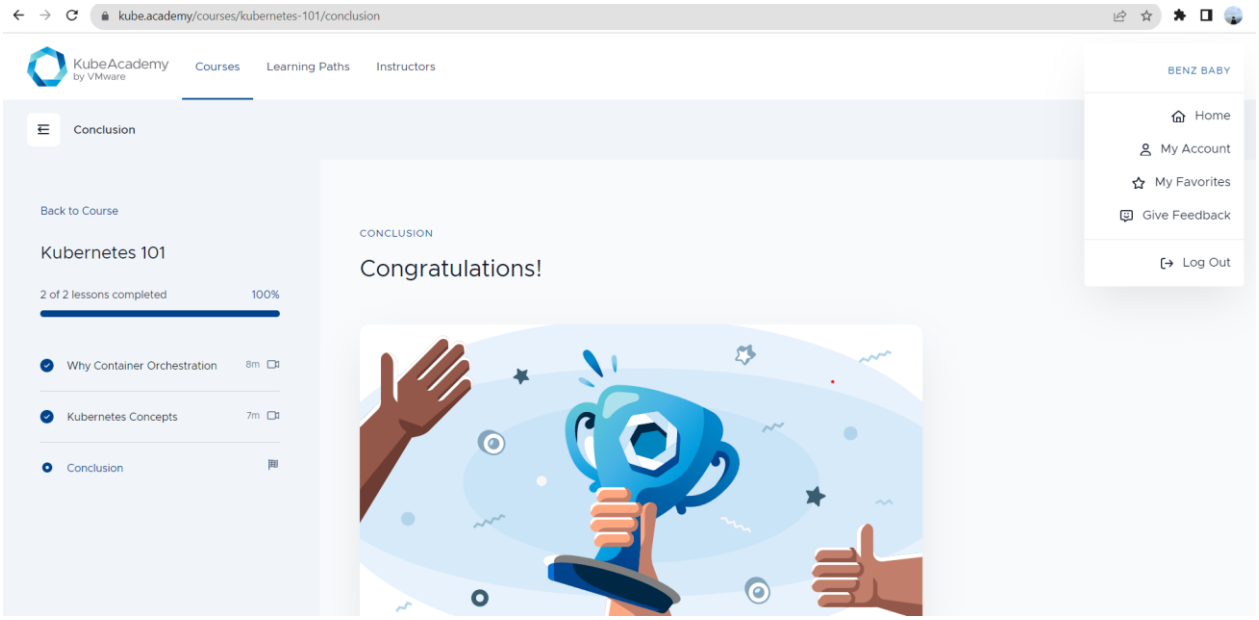
Build the Container Image: Use docker build to construct the image.

Run the Container: Launch a container using docker run.

Test the Container: Access your application running in the container.

Cleanup: Stop and remove the container using docker stop and dock

Screenshot:


kube.academy/courses/kubernetes-101/conclusion
KubeAcademy
by VMware
Courses
Learning Paths
Instructors
BENZ BABY
Home
My Account
My Favorites
Give Feedback
Log Out
Conclusion
Back to Course
Kubernetes 101
2 of 2 lessons completed
100%
Why Container Orchestration
8m
Kubernetes Concepts
7m
Conclusion
CONCLUSION
Congratulations!