Docker and Kubernetes for Beginners

This course will help you to learn and understand the basics of Docker and Kubernetes. This course includes Hands-On session of each topics, So that students can not only learn but they can apply the Hands-On practice and gain more knowledge of the subject.

Module 1: Introduction to Containerization

- Overview of Virtualization vs. Containerization
- Evolution of Containers: From Chroots to Docker
- Benefits of Containerization

Module 2: Getting Started with Docker

- Installing Docker on Different Platforms
- Docker Architecture: Images, Containers, and Registries
- Running Your First Docker Container

Hands-On Practice:

Task 1: Installing Docker on Different Platforms

Objective: Install Docker on various operating systems.

Tasks:

- Install Docker on Windows
- Install Docker on macOS
- Install Docker on Linux

Choose a Linux distribution and follow the official Docker documentation to install Docker Engine.

Verify the installation by running the docker --version command in the terminal.

Task 2: Docker Architecture - Images, Containers, and Registries
Objective: Understand the basic concepts of Docker architecture.

Tasks:

- Explore Docker Images
- Create a Dockerfile
- Build and Tag an Image
- Run a Container
- Explore Docker Registries

Task 3: Running Your First Docker Container
Objective: Learn how to run a basic Docker container.

Tasks:

- Run a Hello World Container
- Interactive Container Session
- Run a Detached Container
- Container Cleanup

Module 3: Docker Images and Containers

- Creating Docker Images
- Dockerfile Syntax and Best Practices
- Managing Docker Containers - Start, Stop, and Remove

Hands-On Practice:

Task 1: Creating Docker Images

Objective: Understand the process of creating Docker images.

Tasks:

- Image Creation with Docker Commit
- Introduction to Docker Build
- Hands-On Image Creation Using Dockerfile

Task 2: Dockerfile Syntax and Best Practices

Objective: Learn the syntax and best practices for writing Dockerfiles.

Tasks:

- Dockerfile Syntax Overview
- Multi-Stage Builds

Task 3: Managing Docker Containers - Start, Stop, and Remove

Objective: Learn basic container management commands.

Tasks:

- Container Creation and Start
- Container Stop and Restart
- Container Removal
- Docker Image Layers
- Dockerfile Caching
- Docker Hub and Registries
- Container Lifecycle

Module 4: Docker Networking and Volumes

- Understanding Docker Networking
- Linking Containers
- Persistent Data with Docker Volumes

Hands-On Practice:

Task 1: Understanding Docker Networking

Objective: Explore the basics of Docker networking.

Tasks:

- Learn & Practice more about Default Bridge Network
- Learn & Practice more about Host Network
- Learn & Practice more about Custom Bridge Networks

Task 2: Linking Containers

Objective: Understand container linking for communication.

Tasks:

- Container Linking Basics
- Environment Variable Passing
- Container Linking Challenges

Task 3: Persistent Data with Docker Volumes

Objective: Learn how to use Docker volumes for persistent data storage.

Tasks:

- Introduction to Docker Volumes
- Volume Creation and Mounting
- Data Sharing between Containers

Module 5: Docker Compose

- Introduction to Docker Compose
- Defining Multi-Container Applications with Docker Compose
- Running and Managing Multi-Container Applications

Hands-On Practice:

Task 1: Introduction to Docker Compose

Objective: Understand the basics of Docker Compose.

Tasks:

- Install Docker Compose
- Docker Compose Basics

Task 2: Defining Multi-Container Applications with Docker Compose

Objective: Learn how to define multi-container applications using Docker Compose.

Tasks:

- Simple Docker Compose File
- Multi-Service Compose File
- Environment Variables and Volumes

Task 3: Running and Managing Multi-Container Applications

Objective: Learn how to run and manage multi-container applications using Docker Compose.

Tasks:

- Docker Compose Up
- Scaling Services
- Container Interactions

Module 6: Introduction to Kubernetes

- Overview of Kubernetes
- Key Kubernetes Concepts: Pods, Services, and Deployments
- Kubernetes Architecture

Hands-On Practice:

Task 1: Key Kubernetes Concepts: Pods, Services, and Deployments

Objective: Learn about fundamental Kubernetes concepts.

Tasks:

- Understanding Pods
- Introduction to Services
- Deployments in Kubernetes

Task 2: Kubernetes Architecture

Objective: Understand the high-level architecture of Kubernetes.

Tasks:

- Understanding Control Plane
- Cloud-controller-manager
- Etcd
- Kube-api-server
- Scheduler
- Controller Manager
- Kubelet
- Kube-Proxy

Module 7: Introduction to Kubernetes

- Creating and Managing Kubernetes Deployments
- Scaling Applications in Kubernetes
- Rolling Updates and Rollbacks

Hands-On Practice:

Task 1: Creating and Managing Kubernetes Deployments

Objective: Deploy and manage applications using Kubernetes Deployments.

Tasks:

- Setup Kubernetes Cluster [minikube or kind]
- Creating a Basic Deployment [ex: nginx]
- Scaling Deployments

Task 2: Scaling Applications in Kubernetes

Objective: Explore how Kubernetes handles application scaling.

Tasks:

- Horizontal Pod Auto scaling
- Manual Scaling

Task 3: Rolling Updates and Rollbacks

Objective: Understand the process of updating and rolling back applications in Kubernetes.

Tasks:

- **Rolling Updates:**

Introduce a simple web application (e.g., a static HTML page) and deploy it using a Kubernetes Deployment.

Update the application by modifying the Docker image version in the Deployment manifest.

Use kubectl apply to trigger a rolling update and observe the gradual replacement of pods.

- **Rollbacks:**

Intentionally introduce an issue (e.g., an error in the new version) in the updated application.

Roll back to the previous version using kubectl rollout undo deployment.

- **Additional Challenge Task:**

Objective: Combine deployment, scaling, and updates in a more complex scenario.

- **Multi-Container Application Deployment:**

Instruct students to create a Kubernetes Deployment for an application with multiple containers (e.g., frontend and backend).

Experiment with scaling individual components.

Perform a rolling update with minimal downtime.

Module 8: Kubernetes Networking

- Kubernetes Service Discovery
- Ingress Controllers and Ingress Resources
- Network Policies in Kubernetes

Hands-On Practice:

Task 1: Kubernetes Service Discovery

Objective: Explore Kubernetes service discovery capabilities.

Tasks:

- Create a Basic Service

Deploy a simple web application (e.g., Nginx) with a Kubernetes Deployment. Create a Kubernetes Service to expose the application internally within the cluster.

- External Service Access

Extend the previous example to expose the service externally using a Load Balancer or Node Port service type. Access the application using the external IP or node port.

Task 2: Ingress Controllers and Ingress Resources

Objective: Learn about Ingress controllers and how to use Ingress resources.

Tasks:

- Install and Configure Ingress Controller [Ex: Nginx Ingress Controller]
- Define Ingress Resource

Task 3: Network Policies in Kubernetes

Objective: Implement network policies for controlling traffic between pods.

Tasks:

- Deploy Multi-Tier Application

Deploy a multi-tier application with frontend, backend, and database components using Kubernetes Deployments.

- Define and Apply Network Policies

Introduce Network Policies to control communication between different tiers. Create and apply Network Policies to allow or deny traffic based on pod labels.

Additional Challenge Task

Objective: Combine service discovery, Ingress, and network policies in a more complex scenario.

Task:

Secure Multi-Service Application:

Deploy a multi-service application with frontend, backend, and caching components. Configure Ingress for external access, and implement Network Policies to control internal communication.

Module 9: Kubernetes Storage

- Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)
- Storage Classes in Kubernetes
- Dynamic Provisioning of Storage

Hands-On Practice:

Task 1: Persistent Volumes (PVs) and Persistent Volume Claims (PVCs)
Objective: Understand the concepts of Persistent Volumes and Persistent Volume Claims in Kubernetes.

Tasks:

- Create a Persistent Volume
- Create a Persistent Volume Claim
- Mount PVC in a Pod

Task 2: Storage Classes in Kubernetes

Objective: Learn about Storage Classes and their role in dynamic provisioning.

Tasks:

- Define a Storage Class
- Create PVC with Storage Class

Task 3: Dynamic Provisioning of Storage

Objective: Explore dynamic provisioning of storage in Kubernetes.

Tasks:

- Configure Dynamic Provisioning
- Create PVC with Dynamic Provisioning

Additional Challenge Task:

Objective: Implement a more complex storage scenario involving multiple PVCs and dynamic provisioning.

Task:

Multi-PVC Scenario:

Design a scenario where multiple Pods, each with its PVC, require storage. Use dynamic provisioning to fulfill the storage requirements of each PVC.

Module 10: Monitoring and Logging in Kubernetes

- Overview of Monitoring in Kubernetes
- Logging Best Practices with Kubernetes
- Tools for Monitoring and Logging

Hands-On Practice:

Task 1: Overview of Monitoring in Kubernetes

Objective: Gain hands-on experience with basic monitoring in Kubernetes.

Tasks:

- Deploy a Sample Application
- Kubernetes Dashboard Installation
- Use kubectl Commands for Monitoring

Utilize kubectl commands to check the status of nodes, pods, and services in the cluster. Explore commands like kubectl top for resource usage metrics

---

Task 2: Logging Best Practices with Kubernetes

Objective: Implement logging best practices for Kubernetes applications.

Tasks:

- Configure Container Logging
- Centralized Logging with Fluentd
- Logging Aggregation

Set up Fluentd as a centralized log collector in the Kubernetes cluster. Configure Pods to forward logs to Fluentd.

Explore tools for log aggregation (e.g., Elastic search, Log stash, Kibana - ELK stack). Set up a simple ELK stack for centralized log storage and visualization.

Task 3: Tools for Monitoring and Logging

Objective: Familiarize yourself with popular tools for monitoring and logging in Kubernetes.

Tasks:

- Prometheus Installation
- Grafana Integration
- Explore Additional Tools

Explore other monitoring and logging tools such as:

- cAdvisor for container-specific metrics.
- Jaeger for distributed tracing.
- Loki for log aggregation.
- Install and configure one additional tool based on the chosen scenario.

Additional Challenge Task:

Objective: Integrate monitoring and logging for a multi-service application.

Task:

Deploy Multi-Service Application:

- Deploy a multi-service application with micro-services architecture in Kubernetes.
- Configure logging and monitoring for each service.
- Create dashboards and alerts for critical metrics.

Module 11: Kubernetes Security

- Kubernetes RBAC (Role-Based Access Control)
- Securing Container Images
- Network Policies and Pod Security Policies

Hands-On Practice:

Task 1: Kubernetes RBAC (Role-Based Access Control)

Objective: Implement RBAC to control access to Kubernetes resources.

Tasks:

- Create RBAC Roles and Role Bindings
- Use kubectl with RBAC
- Define a custom RBAC Role granting specific permissions (e.g., list pods, get services).
- Create a Role Binding to associate the Role with a user or a group.
- Configure a Kubernetes cluster with RBAC enabled.
- Create a Kubernetes user with a client certificate.
- Use kubectl with the user's client certificate to perform actions based on RBAC permissions.

Task 2: Securing Container Images

Objective: Implement security practices for container images.

Tasks:

- Scan Container Images for Vulnerabilities
- Implement Image Signing

Ex:

- Use a container image vulnerability scanner (e.g., Trivy, Clair) to scan a sample container image for security vulnerabilities.
- Analyze the scan results and prioritize remediation.
- Sign a container image using container image signing tools (e.g., Notary).
- Configure a Kubernetes deployment to only run signed images.

Task 3: Network Policies and Pod Security Policies

Objective: Enforce network segmentation and pod security practices.

Tasks:

- Define and Apply Network Policies
- Implement Pod Security Policies

Ex:

- Create a Network Policy YAML manifest to restrict communication between pods.
- Apply the Network Policy to the relevant namespace and observe the network segmentation.
- Enable Pod Security Policies (PSP) in your Kubernetes cluster.
- Define a Pod Security Policy that enforces specific security constraints (e.g., non-root user, read-only file system).
- Apply the Pod Security Policy to a namespace and observe its impact on pod deployments.

Additional Challenge Task:

Objective: Combine RBAC, image security, network policies, and pod security policies for a comprehensive security setup.

Task:

Secure Multi-Tier Application:

Deploy a multi-tier application in Kubernetes with frontend, backend, and database components.

Implement RBAC roles for different teams (e.g., dev, ops).

Enforce image signing and vulnerability scanning for all containers.

Apply Network Policies and Pod Security Policies to enhance security.

Module 12: CI/CD with Docker and Kubernetes

- Building and Pushing Docker Images
- Continuous Integration and Deployment Pipelines
- Deploying Applications with GitOps

Hands-On Practice:

Task 1: Building and Pushing Docker Images

Objective: Set up a CI/CD pipeline to build and push Docker images.

Tasks:

- Dockerfile Creation
- Configure CI/CD Pipeline [Ex: Jenkins or Github Action]
- Integration with Container Registry

Task 2: Continuous Integration and Deployment Pipelines

Objective: Implement CI/CD pipelines for deploying applications to Kubernetes.

Tasks:

- Deploy to Staging Environment
- Implement Testing in Pipeline
- Promote to Production

Module 13: Advanced Topics and Best Practices

- Kubernetes Operators
- Helm: Package Manager for Kubernetes
- Best Practices for Docker and Kubernetes

Hands-On Practice:

Task 1: Introduction of Kubernetes Operators

Objective: Get a better understanding of Kubernetes Operators.

Tasks:

- Explore more about Kubernetes Operators
- Deploy some Kubernetes Operators on your cluster

Task 2: Introduction of Helm Package Manger

Objective: Get a better understanding of Helm Package Manger.

Tasks:

- Install Helm
- Create Helm Chart
- Deploy application with Helm Chart
- Upgrade and Rollback with Helm

Task 3: Best Practices for Docker and Kubernetes

Objective: Implement best practices for Docker and Kubernetes deployments.

Tasks:

- Docker Image Security Scanning
- Optimize Kubernetes Resource Requests and Limits
- Implement Horizontal Pod Autoscaling (HPA)

Ex:

- Integrate a Docker image security scanning tool (e.g., Trivy, Clair) into your CI/CD pipeline.
- Review the resource requests and limits in your Kubernetes Deployment manifest.
- Adjust them based on application requirements and cluster capacity.
- Configure Horizontal Pod Auto scaling based on metrics such as CPU utilization.
- Demonstrate how the number of pods scales based on demand.