Containers - Learning Path

Containers and Orchestration

Containers

- Lightweight virtualization technology.
- Packages applications and dependencies in a consistent environment.
- Ensures consistent application execution across different systems.
- Ideal for development, testing, and deployment.
- Offers portability and efficient resource utilization.

Kubernetes (K8s)

- Open-source container orchestration platform.
- Automates deployment, scaling, and management of containerized applications.
- Provides features like load balancing, auto-scaling, self-healing, and rolling updates.
- Abstracts infrastructure complexities, focusing on application management.
- Enables efficient deployment and scaling of container workloads across clusters.

container diagram

```
[ Host Server (with OS)]
[Container0]
[Container1]
[Container2]
[Container3]
```

Orchestration

- Container orchestration automates and manages containers.
- Orchestration tools like Kubernetes automate container deployment and scaling.
- Orchestration ensures containers are distributed across hosts for redundancy.
- It simplifies complex requirements for managing containers.
- Zero-downtime deployments ensure uninterrupted service during updates.
- Containers running old and new code coexist during a deployment.
- Orchestration tools coordinate container deployment steps efficiently.
- Orchestration automates tasks like spinning up containers, traffic switching, and cleanup.
- Container orchestration enables quick, reliable, and efficient management of containers.

Zero-Downtime deployment

1. Spin up conatiners running new Code
2. When they are fully up, direct user traffic to the new containers
3. Remove the old containers running the old code (no downtime for users)

what containers can be used for ...

1. Software Portability
2. Isolation
3. Scaling
4. Automation
5. Efficient Resource Usage

Advantages of Containers

- same isolation and portability of VMs
- lightweight - less resources
- Faster than VMs
- smaller than VMs
- faster and simpler automation

Limitations of Containers

- less flex than VMs (No windows containers on linux machines . yet)
- new challenges with orchestrationa and automation

Docker

create manage and run containers

- This section of the course covers specific container and orchestration technologies.
- The focus of this lesson is on Docker, one of the most popular container runtimes.
- Docker is a container runtime, which enables the implementation and running of containers.
- Containers are a concept, and Docker is a tool that implements this concept.
- Docker provides tools for running, building, and managing containers and container images.
- The course does not provide technical details about using Docker but recommends exploring official Docker documentation for more information.
- There are alternative container runtimes available besides Docker, which will be briefly mentioned in the next lesson.

other container runtimes

- RKT "rocket" - security and composability
- containerd "container d" - simple

Kubernetes

- This lesson introduces Kubernetes as a container orchestration tool.
- Kubernetes simplifies building and managing container infrastructure and automation.
- Orchestration tools like Kubernetes automate tasks such as deploying, scaling, and managing containers.
- Kubernetes enables self-healing applications, automated scaling, and easy automated deployments.
- It is the industry-leading container orchestration tool, known for its extensive features and power.

- You can explore more about Kubernetes in the official Kubernetes documentation at kubernetes.io.
- The next lesson will discuss alternative orchestration tools to consider.

other orchestration tools

- Docker Swarm - native to docker
- Marathon - tons of APIs
- nomad - simple and lightweight
- Amazon Elastic Container Service
- Amazon ECS for Kubernetes
- Azure Kubernetes Service
- IBM Cloud Kubernetes
- RedHat OpenShift
- Google's Kubernetes Engine

microservices

- This section discusses use cases of containers and orchestration, starting with microservices.
- Microservices involve splitting applications into small, independent services.
- Containers make it easier to implement and manage microservices.
- Microservices offer benefits like rapid development, reduced risk, and technology optimization.
- Containers are suited for managing a large number of small, independent microservices.
- Orchestration simplifies the deployment, scaling, and connection of microservice instances.
- Containers enable easy resource scaling for microservices, with orchestration automating the process.
- Orchestration can automatically detect increased usage and scale microservices without manual intervention, improving end-user experience.
- Containers and microservices provide business value by enhancing end-user experiences and reducing manual administrative work.

```
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│   Microservice 1│─────│   Microservice 2│─────│   Microservice 3│
└─────────────────┘     └─────────────────┘     └─────────────────┘
       │                       │                       │
       ▼                       ▼                       ▼
┌───────────┐           ┌───────────┐           ┌───────────┐
│  Database │           │  Database │           │  Database │
└───────────┘           └───────────┘           └───────────┘
```

or

```
[Microservices]
[    App      ]
[product data]
[customer data]
```

```
[authentication]
[order history]
```

Cloud Transformation

- This lesson explores the use case of "cloud transformation" for containers and orchestration.
- Cloud transformation involves migrating existing IT infrastructure to the cloud.
- Containers can simplify the process of wrapping existing software for cloud migration.
- Containers offer flexibility and automation benefits when running software in the cloud.
- Containers use fewer resources than virtual machines, resulting in cost savings in cloud environments.
- By using containers in cloud transformation, organizations can modernize and optimize their infrastructure for the cloud while reducing operational costs.

Automated Scaling

- This lesson discusses automated scaling and how containers contribute to delivering business value through automation.
- Automated scaling involves provisioning or de-provisioning resources based on real-time data metrics.
- Without automated scaling, organizations must provision enough resources for peak usage at all times, leading to resource wastage.
- Containers enable efficient automated scaling because they can start and stop quickly.
- Containers allow organizations to respond rapidly to changes in usage by creating or removing instances as needed.
- Automated scaling with containers increases stability during high usage periods and reduces costs during low usage times.
- Containers provide business value by optimizing resource usage and ensuring service reliability in dynamic usage scenarios.

Continuous Deployments

- This lesson covers continuous deployment pipelines and how containers facilitate continuous deployment.
- Continuous deployment involves automatically and frequently deploying new code.
- Traditional deployments are infrequent, while continuous deployment breaks changes into small, frequent deployments.
- Small, automated deployments reduce risk and get new functionality to customers faster.
- Automation is crucial for maintaining stability and consistency in continuous deployments.
- Containers are well-suited for continuous deployment as they facilitate testing in environments identical or similar to production.
- Containers allow automated pipelines to build and test the same container image that will be used in production.
- Testing in the actual production environment (the container image) ensures code behaves consistently during deployment.

- Containers enhance continuous deployment by integrating image building into the automation pipeline.

self-healthing Systems

- This lesson discusses how containers can facilitate the implementation of self-healing applications.
- Self-healing applications automatically detect and resolve problems without human intervention.
- Self-healing systems are more resilient and reliable as they can correct issues on their own.
- Containers start up quickly and are easily automated, making them suitable for self-healing scenarios.
- Containers can replace broken containers with brand new, clean, fully working containers, enhancing self-healing capabilities.
- Container orchestration tools like Kubernetes offer features for automating self-healing processes.
- Containers and orchestration tools add business value by creating more reliable and resilient systems through self-healing applications.

Developer Visibility

- This lesson discusses the use case of "developer visibility" enabled by containers.
- In traditional environments, developers often lack access to production systems, hindering troubleshooting when issues arise.
- Developers may only have access to their local development environments, leading to the "It works on my machine" problem.
- Lack of developer visibility can slow down problem diagnosis and resolution.
- Containers enhance developer visibility by making the container environment similar to production.
- Developers can run containers locally, allowing them to see how their code behaves in a production-like environment.
- Containers reduce environmental differences, increasing developer visibility and helping troubleshoot code efficiently.
- Containers also enable testing code before production, providing better performance insights.
- Enhanced visibility offered by containers contributes to more efficient code development and troubleshooting.
- Containers offer real business value by improving developer visibility within organizations.

Containers in the Cloud

- This lesson discusses the relationship between containers and the cloud.
- The cloud is essentially running software on remote servers provided by a cloud provider over the internet.
- Cloud providers like Amazon Web Services, Microsoft Azure, and Google Cloud Platform offer server resources in their data centers for users to run their software.
- Private cloud refers to using cloud practices and technologies within one's own data center.
- Containers can be run in the cloud, meaning that containers run on servers provided by a cloud provider.
- Cloud platforms often offer specialized services to support containerized applications.
- Using containers in the cloud leverages cloud infrastructure and container-specific services for more efficient deployment and management.

- In the next lesson, the benefits of containers in the context of the cloud and specific cloud services supporting containers will be discussed.

continued

- Containers offer the regular benefits of portability, speed, and ease of automation in the context of the cloud.
- Containers can easily autoscale, self-heal, and automate various tasks in the cloud environment.
- Containers save money in the cloud, especially when compared to virtual machines (VMs), as they have minimal overhead.
- In a cloud platform, you only pay for the resources you use, making containers a cost-effective choice.
- Many cloud platforms provide built-in support for containers, simplifying setup and management.
- Amazon Web Services (AWS) offers Amazon Elastic Container Service (ECS) and Amazon ECS for Kubernetes.
- Microsoft Azure provides Azure Kubernetes Service (AKS).
- Google Cloud Platform (GCP) offers the Google Kubernetes Engine (GKE).
- These cloud services offer ready-made Kubernetes clusters for easy container management.
- Containers in the cloud offer scalability, cost-efficiency, and simplicity in deploying and managing applications.

next Steps

- Containers
  - LXC,LXD
  - CoreOS Essentials
- Docker
  - Docker Quickstart
  - Docker DeepDive
  - Docker Cert Associate Prep Course 📃
- Kubernetes
  - Cert Kubernettes Admin (CKA) 📃
  - Kubernetes the hard way
- DevOps
  - DevOps Essentials
  - implementing a full CI/CD pipeline