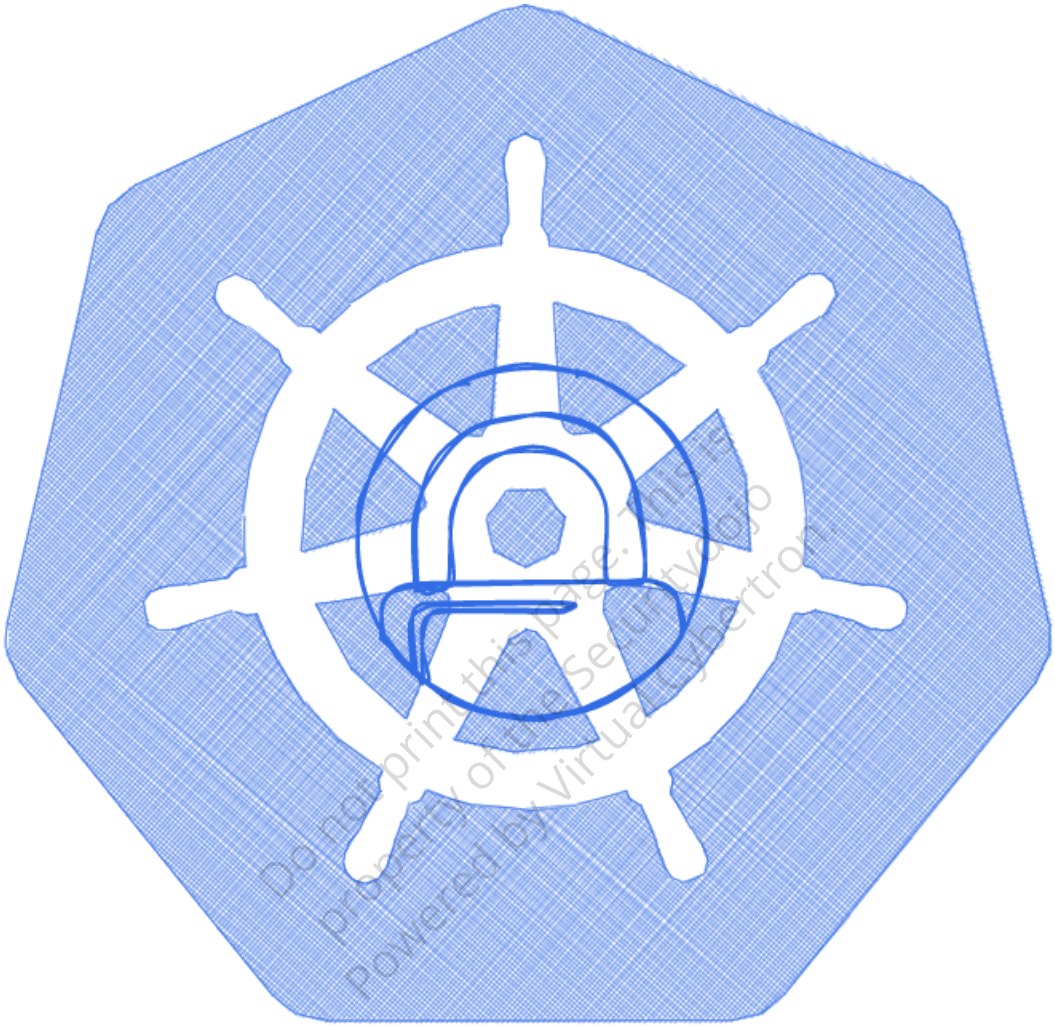
Introduction to Kubernetes



Kubernetes Orchestration

- Kubernetes orchestration refers to the process of managing and automating the deployment, scaling, and management of containerized applications using Kubernetes.
- It provides a framework for managing the entire lifecycle of containerized workloads across a distributed network of hosts.

Benefits of Kubernetes Orchestration

- **Scalability**: Kubernetes allows users to easily scale their containerized applications up or down, based on their needs, without any downtime.
- **Automation**: Kubernetes automates many of the routine tasks associated with deploying and managing containerized applications, freeing up developers to focus on other important tasks.
- **Portability**: Kubernetes provides a consistent platform for deploying and managing containerized applications, regardless of the underlying infrastructure.
- **Resiliency**: Kubernetes offers self-healing capabilities, allowing it to automatically recover from failures and ensure the availability of containerized applications.

Use Cases for Kubernetes Orchestration

- Cloud-native application development: Kubernetes provides a platform for developing cloud-native applications that can be easily deployed and managed in the cloud.
- Microservices architecture: Kubernetes allows users to easily manage microservices-based applications, which are composed of small, independently deployable services.
- Hybrid cloud deployments: Kubernetes provides a consistent platform for deploying and managing containerized applications across different cloud providers and on-premises infrastructure.

Do not print this page. This is property of the securitydojo Powered by Virtual Cyber Dojo

Kubernetes Alternatives

- Docker Swarm
- Apache Mesos
- Nomad
- OpenShift
- Rancher

Kubernetes Security Best Practices

- **Image Scanning**: Scan container images for vulnerabilities and malware before deploying them to a Kubernetes cluster to prevent security breaches.
- **Host Operating System Hardening**: Secure the host operating system to prevent

attackers from accessing or manipulating containers running on the host. This can include measures such as applying security patches, disabling unnecessary services, and using secure boot.

- **Hardening Base Image**: Ensure that the base images used to build containers are hardened and do not contain any known vulnerabilities or malicious code.
- **Harden Your Kubernetes Clusters**: Implement strong access controls, including role-based access control (RBAC), and limit access to sensitive resources such as the Kubernetes API server and etcd data store.
- **Network Security**: Implement network policies to control network traffic between pods and services, and use secure transport protocols such as TLS to protect data in transit. Use mutual authentication between pods and services to prevent spoofing attacks.

Do not print this page. This is property of the Securitydojo Powered by Virtual Cybertron.