Cloud Native Architecture

- Autoscaling
- Serverless
- Community and Governance
- Roles and Personas
- Open Standards

Autoscaling

Autoscaling in Kubernetes (KCNA)

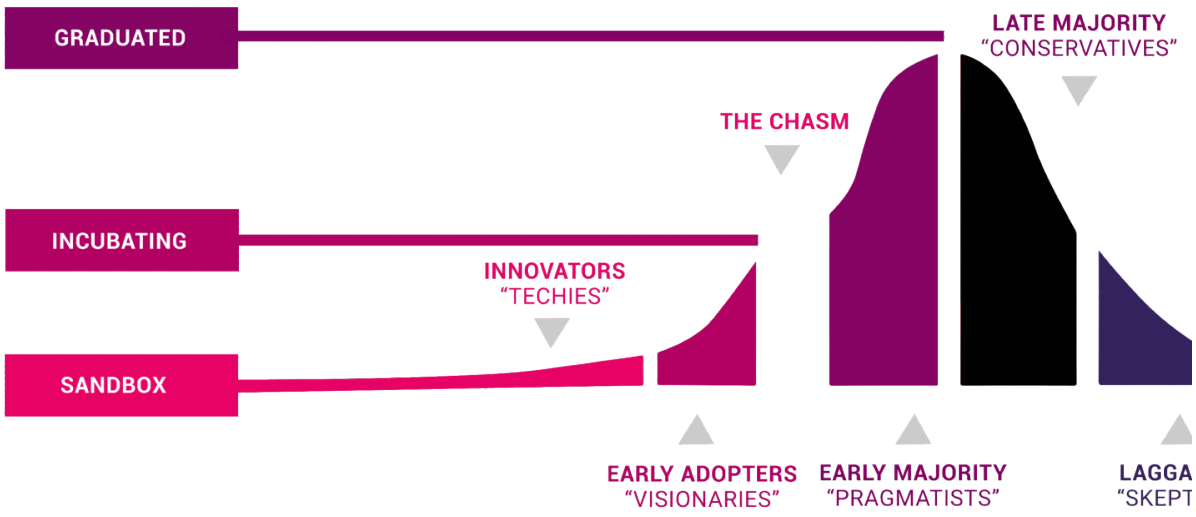
- **Horizontal Pod Autoscaler (HPA):**
  - Scales Pods based on resource utilization (CPU, memory, custom metrics).
  - Monitors and adjusts Pod replicas dynamically.
- **Vertical Pod Autoscaler (VPA):**
  - Adjusts resource requests/limits (CPU, memory) for Pods.
  - Optimizes resource allocation for individual Pods.
- **Cluster Autoscaler:**
  - Scales nodes in a cluster up or down based on resource demands.
  - Adds/removes nodes when Pods can't be scheduled due to resource constraints.
- **Custom Metrics Autoscaler:**
  - Scales based on custom metrics (e.g., application performance metrics).
- **Scaling Policies:**
  - Define minimum and maximum replica limits.
  - Set scaling thresholds (e.g., CPU, memory utilization).

Serverless

- **Serverless Overview:**
  - No infrastructure management.
  - Auto-scaling, event-driven execution.
- **Serverless on Kubernetes:**
  - Use tools like **Knative**, **Fission**.
  - Event-driven, auto-scaling on Kubernetes.
- **Knative:**
  - Extends Kubernetes for serverless workloads.
  - Auto-scaling, routing, event-driven.
- **Fission:**
  - Serverless framework for Kubernetes.
  - Functions-as-a-service (FaaS).
- **Benefits:**
  - Focus on code, not infrastructure.
  - Dynamic scaling based on demand.

Community and Governance

- **CNCF Mission:**
  - Promote cloud-native computing ubiquity.
  - Support open-source projects for scalable, resilient, and manageable applications.
- **Project Maturity Levels (CNCF):**
  - **Sandbox:** Experimentation and community support.
  - **Incubation:** Maturity, traction, and growing community.
  - **Graduated:** Stability, security, and widespread adoption.
- **Crossing the Chasm:**
  - Transition from **Incubated** to **Graduated**.
  - Indicates maturity, stability, and enterprise readiness.
- **Technical Oversight Committee (TOC):**
  - Evaluates and oversees CNCF project maturity.
- **Special Interest Group (SIG) / Technical Advisory Group (TAG):**
  - SIG (formerly TAG) represents community-driven initiatives.
  - TAG differentiates from Kubernetes SIGs.
- **Project Maturity Factors:**
  - Adoption, change rate, external committers, and adherence to the **CNCF Code of Conduct**.
- **Conflict Resolution:**
  - Open discussion, followed by voting and consensus.



Roles and Personas

- **DevOps Engineer:**
  - Bridges development and operations.
  - Optimizes processes, automates workflows, ensures smooth releases.
- **Site Reliability Engineer (SRE):**
  - Ensures reliability, scalability, and performance of systems.
  - Combines development and operations expertise.
- **CloudOps Engineer:**
  - Manages and optimizes workloads on cloud infrastructure.
  - Ensures secure, reliable, and efficient cloud services.
- **DevSecOps Engineer:**
  - Integrates security into DevOps processes.
  - Ensures security is part of the entire software lifecycle.
- **Full Stack Developer:**
  - Works on both frontend and backend development.
  - Handles user interfaces and server-side logic.
- **Cloud Architect:**
  - Designs cloud applications and infrastructure.
  - Focuses on scalability, reliability, and cost-effectiveness.
- **Data Engineer:**
  - Designs and builds data systems and pipelines.
  - Focuses on scalability, efficiency, and reliability.
- **Security Engineer:**
  - Implements and enforces security best practices.
  - Focuses on secure architectures, risk assessments, and compliance.
- **FinOps:**
  - Manages and optimizes financial aspects of cloud operations.
  - Ensures cost-effective use of cloud resources.

Open Standards

- **Definition:**
  - Public, open specifications for interoperability.
  - Avoids vendor lock-in, promotes flexibility and integration.
- **CNCF & Open Standards:**
  - Promotes open standards for cloud-native technologies (e.g., Kubernetes, Helm).
  - Ensures cross-platform compatibility.
- **OCI (Open Container Initiative):**
  - Defines standards for container runtimes and images.
  - **runC:** First open standard for container runtimes.
  - **Runtime Specification:** Standardizes container execution.
  - **Image Specification:** Defines format for container images (layers, metadata).
  - **Distribution Specification:** API for managing container images across systems.
- **Container Storage Interface (CSI):**
  - Open standard for storage integration in container orchestration.
  - Allows compatibility with multiple storage solutions through a single plugin.
- **Container Runtime Interface (CRI):**
  - API for Kubernetes to interact with different container runtimes.
  - Provides flexibility in choosing containerization solutions.
- **Benefits of Open Standards:**
  - Promotes interoperability between vendors and technologies.
  - Ensures flexibility, avoiding reliance on a single vendor.