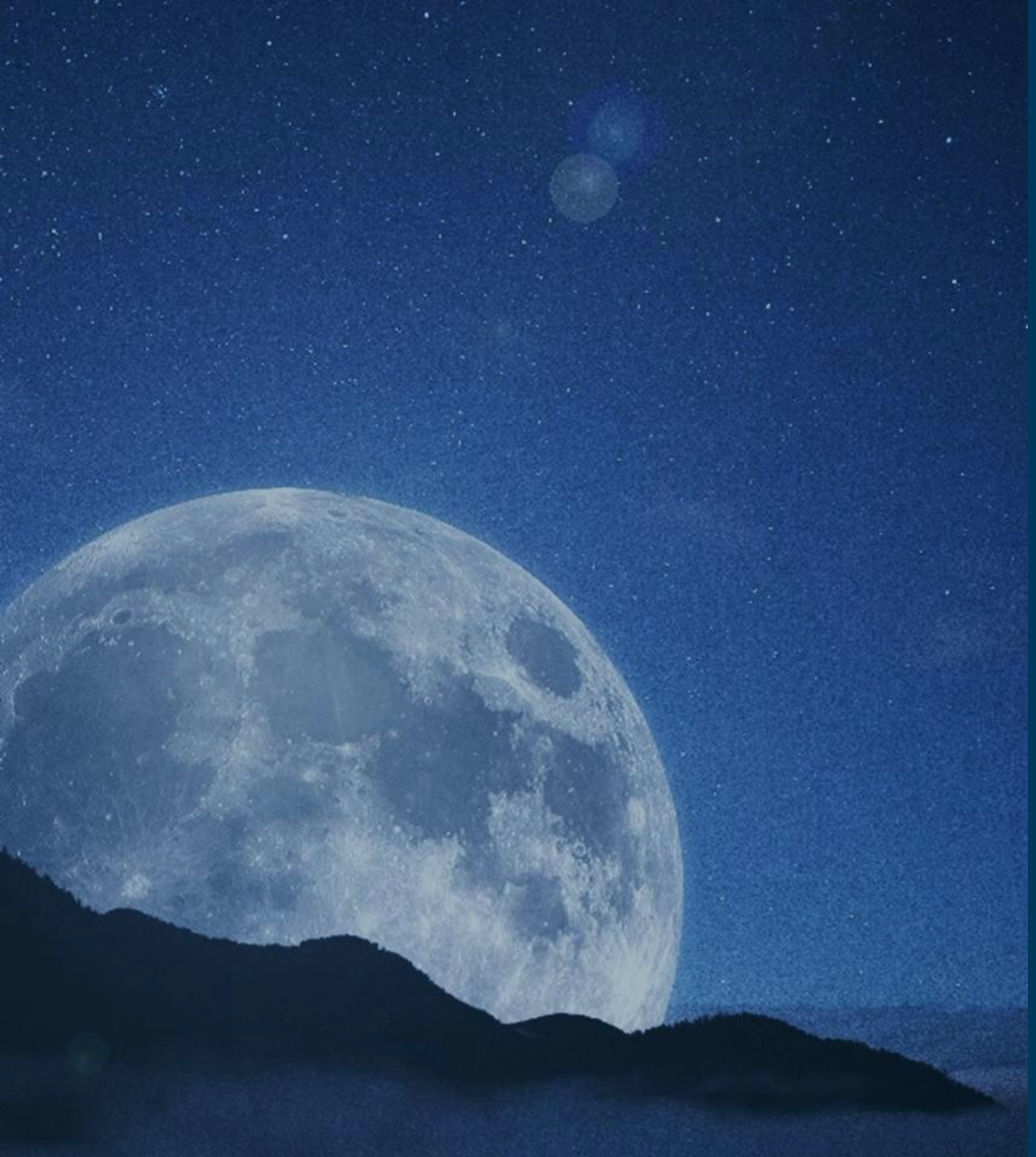
LUNARIS
Digital Solutions

CONTAINER-FIRST

Azure Container Apps und Azure Kubernetes Service im direkten Vergleich

Florian Bader



MIT WEM HABT IHR ES ZU TUN?

Florian Bader

Solution Architect | CTO

Florian.Bader@lunaris.digital

https://lunaris.digital
https://github.com/florianbader

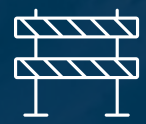
LUNARIS
Digital Solutions

KEY TAKE AWAYS

Warum braucht es Container überhaupt?

Was können die einzelnen Services?

Was sind die grundlegenden Unterschiede?



Was sind die typischen Einsatzszenarien?

WARUM CONTAINER?

Basis für moderne Cloud-Architektur

Konsistenz & Portabilität

Skalierung

Versionierung

Isolation & Sicherheit

AZURE ❤️ CONTAINERS

Welche Plattform passt zu welchem Szenario?

Core

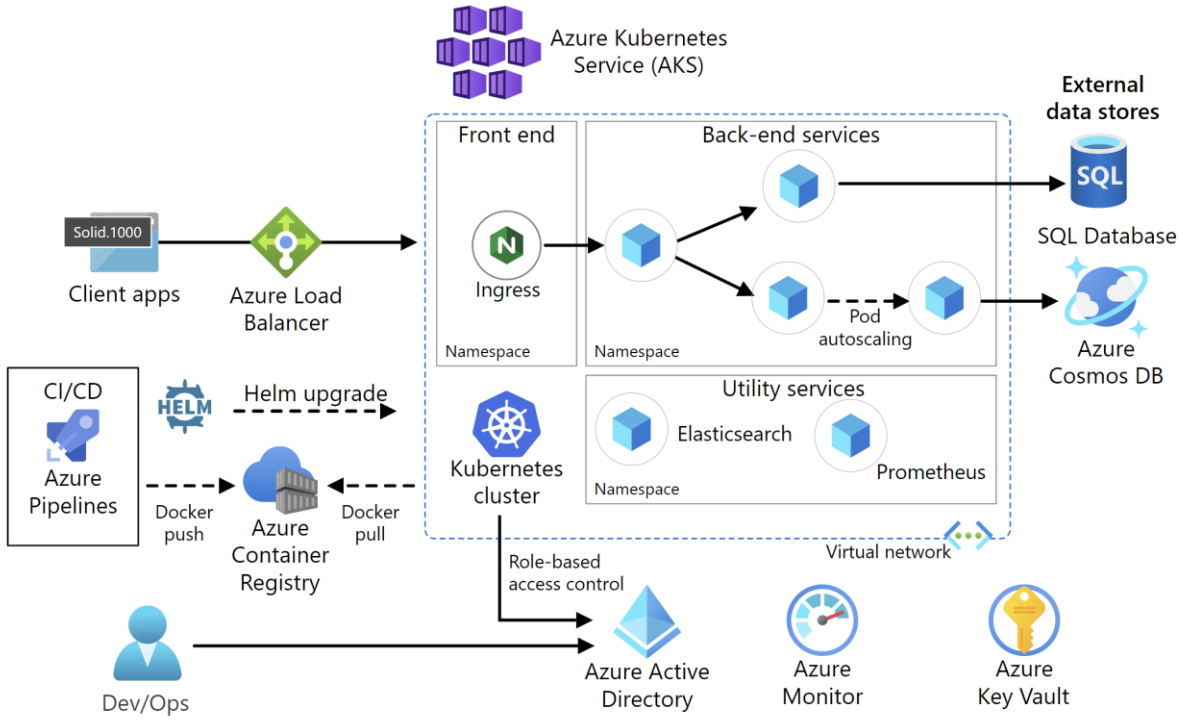
- Azure Container Registry

Standard

- Azure App Service
- Azure Kubernetes Service (AKS)
  - Azure Container Storage
- Azure Container Apps (ACA)

Extended

- Azure Container Instances
- Azure Batch
- Azure Service Fabric



Microservices architecture on AKS - Azure Architecture Center | Microsoft Learn

WARUM KUBERNETES?

Was Azure Kubernetes Service (AKS) für dich übernimmt

Warum Kubernetes?

- Automatisierte Orchestrierung (Deployment, Skalierung, Self-Healing)
- Hohe Verfügbarkeit
- Effiziente Resourcennutzung
- Schnelle Rollouts & Rollbacks

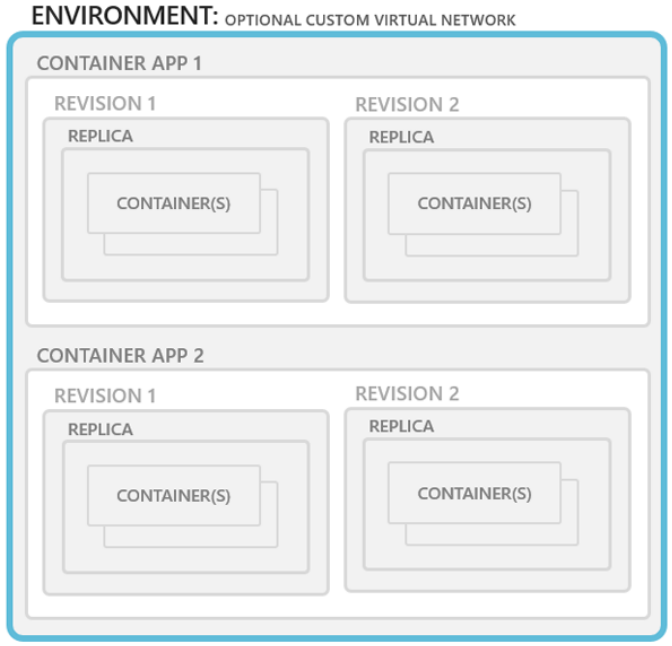
Was übernimmt AKS?

- Betrieb & Wartung der Control Plane
- Automatische Updates & Patches
- Einfache Skalierung & Monitoring
- Integrierte Sicherheit
- Integration mit anderen Azure Services

KUBERNETES OHNE CLUSTER-STRESS

Abstraktion statt Administration

Environments are an isolation boundary around a collection of container apps.



Containers in Azure Container Apps | Microsoft Learn

Warum Azure Container Apps?

- Serverless Container-Plattform
- Einfache Bereitstellung & Skalierung
- Schneller Einstieg

Was übernimmt ACA?

- Vollständig verwaltete Umgebung für Container-Workloads
- Automatische Skalierung (Event-Driven Autoscaling)
- Integriertes Dapr (Pub/Sub, State Management)
- Secret Management
- Consumption-Based Pricing
- Integriertes Monitoring

UNTERSCHIEDE

Basis für moderne Cloud-Architektur

Architektur

Betrieb

Kostenmodell

Deployment

ARCHITEKTUR UND SKALIERUNG

Azure Kubernetes Service

Instanzmodell

- Anwendungen laufen in Pods, mit einem oder mehreren Containern
- Sidecar-Pattern ist gängige Praxis (z.B. für Logging, Monitoring)

Stateful vs. Stateless

- Unterstützt sowohl Stateless als auch Stateful Workloads

Ingress

- Ingress über eigene Controller
- TLS-Zertifikate via Cert-Manager oder manuelle Konfiguration

Secrets

- Secrets über Kubernetes-native Resourcen oder Azure Key Vault CSI

Skalierung

- Horizontal Pod Autoscaler (HPA)
- Cluster Autoscaler

Azure Container App

Instanzmodell

- Anwendungen in mehreren Container Apps
- Mehrere Container pro App möglich

Stateful vs. Stateless

- Fokus auf Stateless Workloads

Ingress

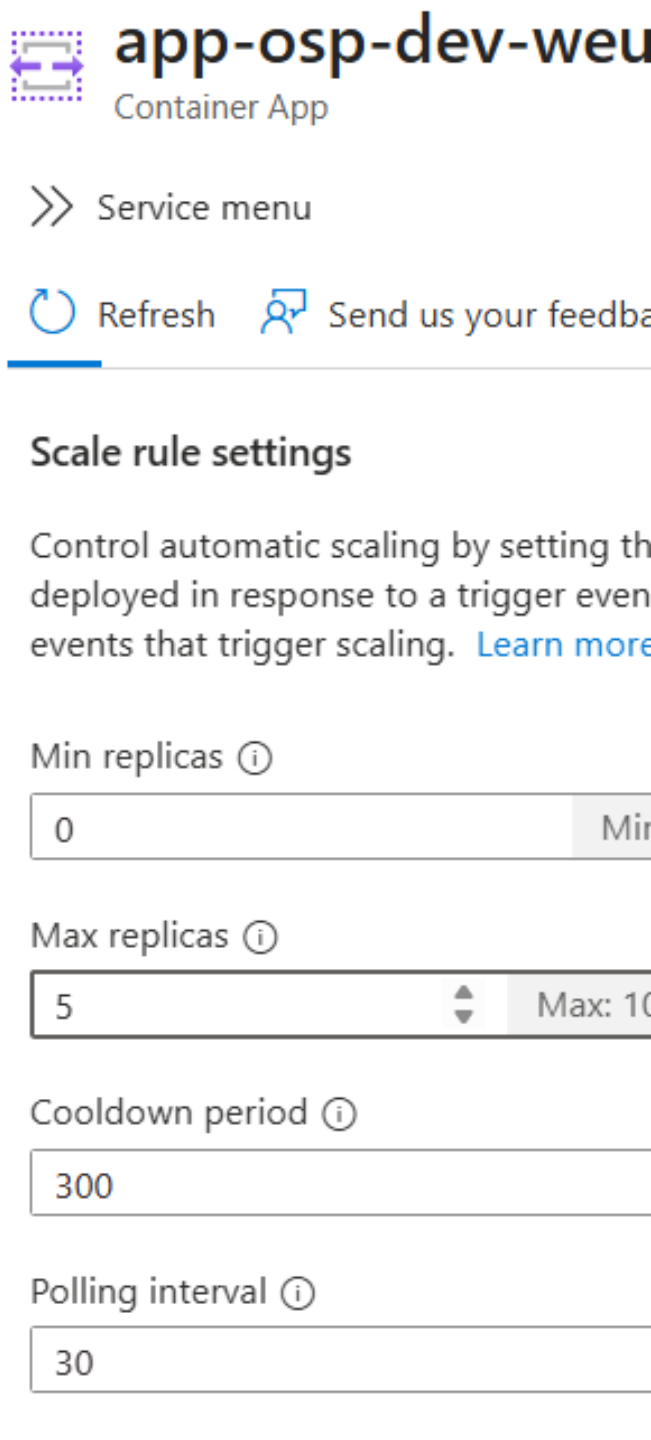
- Ingress ist nativ integriert, inkl. TLS

Secrets

- Secrets direkt in ACA oder über Azure Key Vault

Skalierung

- KEDA-basiertes Autoscaling
- Scale-to-Zero möglich



BETRIEB UND WARTUNG

Azure Kubernetes Service

Support und Verantwortung

- Nutzer verwaltet Cluster, Nodes, Netzwerk, Storage
- Microsoft betreibt nur die Control Plane

Updates und Wartung

- Kubernetes-Versionen müssen aktiv gepflegt werden
- Updates für Nodes, Add-ons & Tools (z.B. Ingress, Cert Manager)

Monitoring und Logging

- Eigene Wahl (Prometheus, Grafana, Azure Monitor)
- **Logging über Sidecars**

Fehlerbehandlung & Rollbacks

- Rollbacks über Helm, GitOps oder CI/CD

Azure Container App

Support und Verantwortung

- Vollständig verwalteter Dienst
- Microsoft übernimmt Betrieb & Skalierung

Updates und Wartung

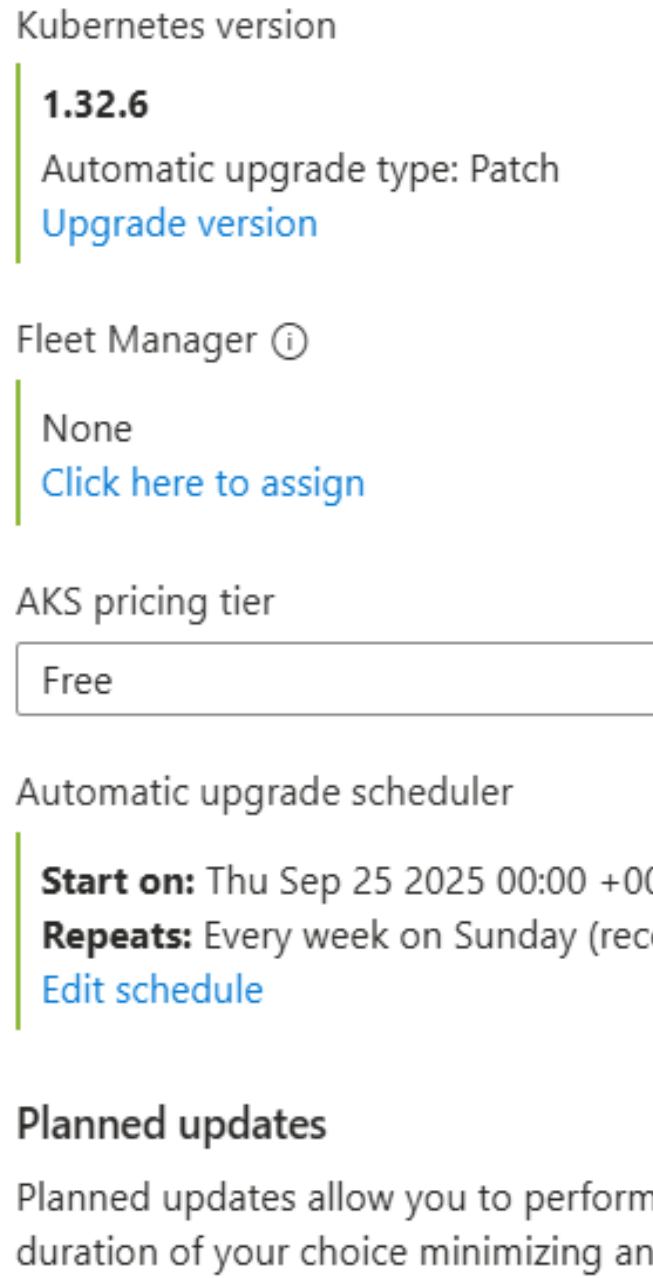
- Plattform wird automatisch aktualisiert (Maintenance Window)

Monitoring und Logging

- Integriertes Monitoring via Azure Monitor & Log Analytics

Fehlerbehandlung & Rollbacks

- Revisions ermöglichen einfachen Rollback



KOSTENMODELL

Azure Kubernetes Service

- Abrechnung nach Nodes (VM-Größe, Anzahl und Laufzeit)
- Clusterkosten (Free, Standard, Premium)
- Zusatzkosten für Storage
- Kostenreduktion durch Saving Plans, Reservations und Spot Instances möglich

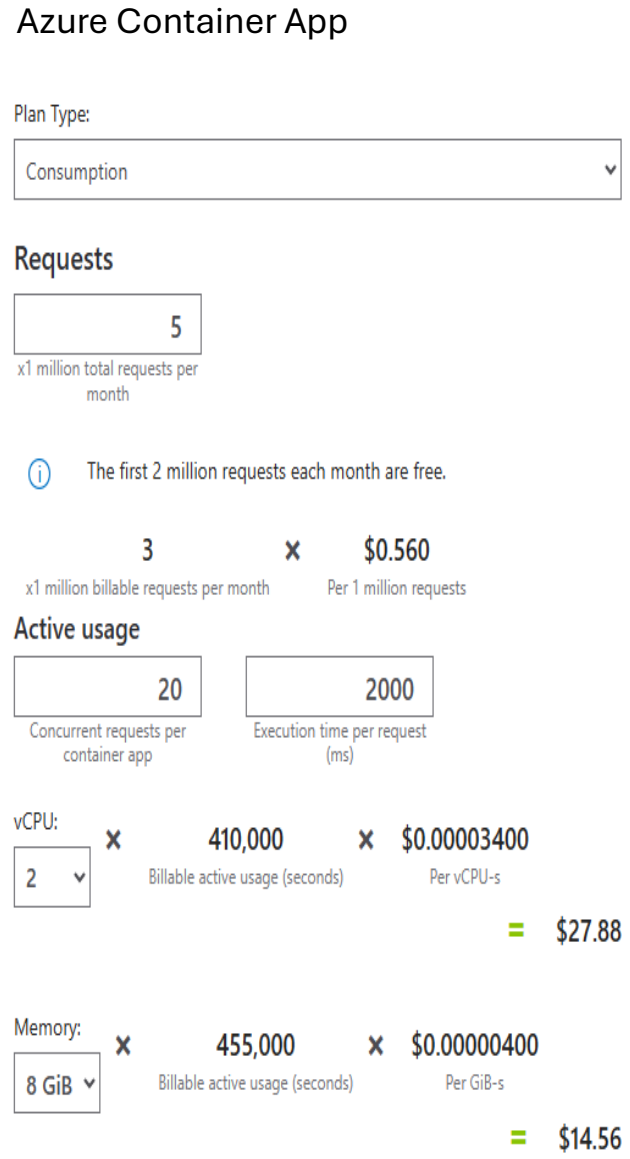
Azure Container App

Consumption Plan

- Abrechnung nach vCPU, Memory und Requests
- Kostenreduktion durch Idle Time möglich
- Kostenreduktion durch Saving Plans möglich

Dedicated Plan

- Abrechnung nach Nodes (VM-Größe, Anzahl und Laufzeit)
- Kostenreduktion durch Saving Plans möglich



DEPLOYMENT-STRATEGIEN

Azure Kubernetes Service

Deployment-Prozess

- Deployment über YAML-Manifest, Helm, Kustomize oder GitOps
- Volle Kontrolle über Ressourcen, Lifecycle und Rollout-Strategien
- CI/CD meist über AzD, GitHub, ArgoCD oder Flux

Rollout & Versionierung

- Rollout und Rollback manuell oder automatisiert via Helm/GitOps
- Canary, Blue/Green über eigene Logik oder Tools

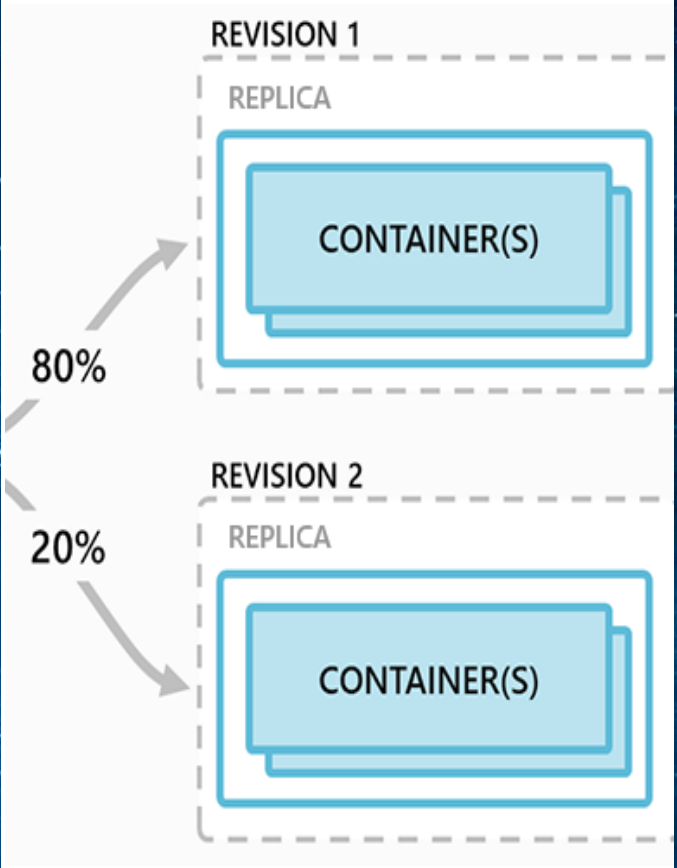
Azure Container App

Deployment-Prozess

- Deployment über Azure CLI, Bicep, oder manuell
- CI/CD meist über AzD oder GitHub

Rollout & Versionierung

- Revisions als integriertes Versionsmodell
- Traffic-Split für Canary Releases
- Rollbacks einfach durch Umschalten auf vorherige Revision



PRAKTISCHE EINSATZSZENARIEN

Azure Kubernetes Service

- Komplexe Plattform mit Multi-Tenancy
- Sicherheitskritische Umgebung
- GPU-Training oder Spezialtreiber
- Windows Container
- Stateful Workloads (Kafka, Postgres, ...)

Azure Container App

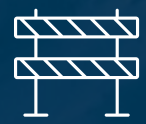
- Event-getriebene Verarbeitung
- Scale-to-Zero
- Kurzlebige Jobs / Batch-Verarbeitung
- Background Services als Sidecar zur eigentlichen Anwendung
- Einfache Nutzung von Dapr

RECAP

AKS bietet maximale Kontrolle, ACA maximale Einfachheit.

AKS erfordert mehr Infrastrukturverständnis

ACA abstrahiert Infrastruktur für mehr Fokus auf Serverless App Logik



Entscheidung muss bewusst und kontextbezogen getroffen werden


LUNARIS
Digital Solutions