Verteilte Systeme und Cloud Technologien - Laborübung 03: MVP Definition und Integration

Wintersemester 2025
(c) 2025
Simon Kranzer, Gerald Lochner

1. Einführung

Diese dritte Laborübung bildet den Abschluss der Laborübungsreihe und fokussiert sich auf die Integration aller entwickelten Komponenten zu einem funktionsfähigen Minimal Viable Product (MVP).

Die Studierenden werden in zwei Kategorien eingeteilt:

- **Platform/DevOps Teams** (1-2 Gruppen): Verantwortlich für das zentrale Azure Kubernetes Service (AKS) Cluster und alle Infrastruktur-Services
- **Application Development Teams** (restliche Gruppen): Entwickeln Business-Services und betreiben lokale k3s Cluster für Energiedaten-Simulation

2. Rollenverteilung

2.1. Platform/DevOps Teams (Max. 2 Gruppen)

Verantwortlichkeiten:

- Aufsetzen und Betrieb des zentralen AKS Clusters
- Installation und Wartung aller Infrastruktur-Services
- Support für Application Development Teams
- Dokumentation der Platform-Services
- Monitoring der gesamten Platform
- Definition von SLAs und Support-Prozessen

Vorteile:

- Praktische DevOps-Erfahrung
- Azure Kubernetes Service (AKS) Administration
- Infrastructure as Code
- Platform Engineering

Herausforderung:

- Hohe Verantwortung - alle Teams sind abhängig
- Support während der Entwicklungsphase
- Koordination mit mehreren Teams

2.2. Application Development Teams (Restliche Gruppen)

Verantwortlichkeiten:

- Entwicklung von Business-Services
- Betrieb eines lokalen k3s Clusters für Smart Meter Simulation
- Integration mit Platform-Services (AKS)
- Eigene CI/CD Pipelines
- Monitoring der eigenen Services
- End-to-End Testing

Vorteile:

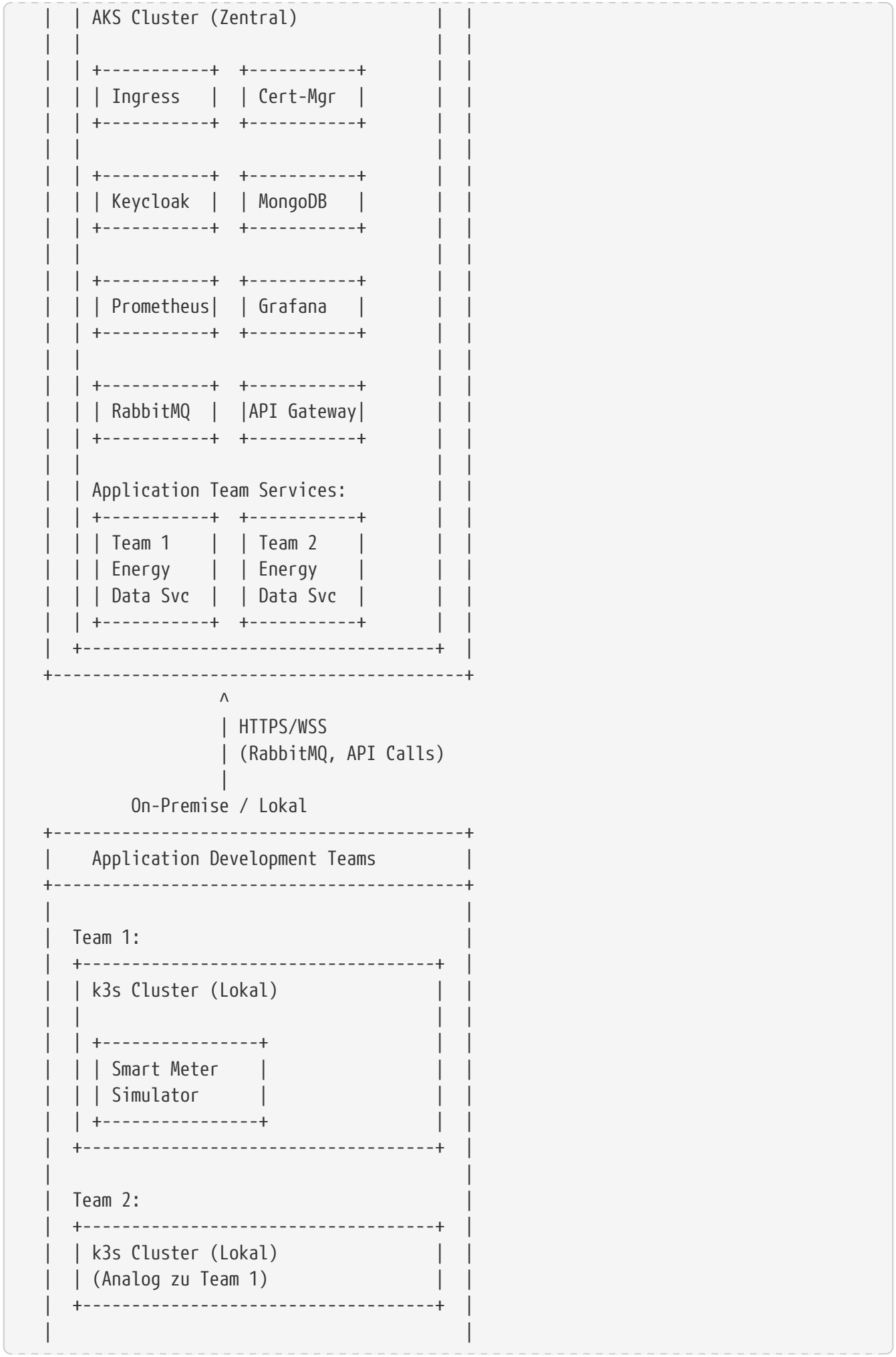
- Fokus auf Business-Logik
- Praktische Erfahrung mit lokalem Kubernetes (k3s)
- Nutzung von Cloud-Services (AKS)
- Hybrid-Setup (lokal + cloud)

Herausforderung:

- Abhängigkeit vom Platform Team
- Betrieb eigener k3s Infrastruktur
- Netzwerk-Konfiguration zwischen lokal und AKS
- Einhaltung von Platform-Standards

3. Architektur-Übersicht anhand eines Beispiels

```
                Azure Cloud
+-----------------------------------------+
|       Platform/DevOps Team(s)           |
+-----------------------------------------+
|                                         |
|  +-----------------------------------+  |
```


AKS Cluster (Zentral)
Ingress
Cert-Mgr
Keycloak
MongoDB
Prometheus
Grafana
RabbitMQ
API Gateway
Application Team Services:
Team 1 Energy Data Svc
Team 2 Energy Data Svc
HTTPS/WSS
(RabbitMQ, API Calls)
On-Premise / Lokal
Application Development Teams
Team 1:
k3s Cluster (Lokal)
Smart Meter Simulator
Team 2:
k3s Cluster (Lokal)
(Analog zu Team 1)

```
|  Team N: ...                              |
+-------------------------------------------+
```

4. MVP Definition: Platform/DevOps Teams

Das Platform Team muss folgende Infrastruktur in Azure bereitstellen:

- Installation und Konfiguration aller Services
  - NGINX Ingress Controller mit TLS (Cert-Manager) (oder Alternativen?)
  - Keycloak für Authentifizierung und Autorisierung (oder Alternativen?)
  - RabbitMQ als Message Broker
  - MongoDB als Datenbank
  - Prometheus + Grafana für Monitoring (oder Alternativen?)
  - API Gateway
- Namespace-Management und RBAC
- Monitoring und Wartung
- Support für Application Teams

Es sollte eine Priorisierung auf Services gelegt werden, die für die Application Teams essenziell sind, also z. B. Authentifizierung (Keycloak) und Messaging (RabbitMQ) sowie Zugriff auf Kubernetes für CD Pipelines.

4.1. Azure Kubernetes Service (AKS)

Das AKS Cluster wurde mit folgenden Spezifikationen erstellt: - 3 Nodes (Standard_D2s_v3) - Azure CNI Networking - Managed Identity

Zugriff zum Cluster für DevOps/Platform Teams

Platform Teams erhalten Administrator-Rechte auf dem AKS Cluster Mit diesem Befehl kann sich das DevOps/Platform Team mit dem AKS Cluster verbinden:

```
export KUBECONFIG=./k8/devops-team-kubeconfig.yaml
kubectl get nodes
```

DNS und Public IP

Die Public IP wird bei der Installation des NGINX Ingress Controllers (oder alternativen) erst vergeben. Dazu bitte bei Gerald Lochner melden damit die DNS-Einträge (A-Records) entsprechend angepasst werden können. Wir verwenden die Domain `machasing.com` für das

Projekt.

Folgende Azure CLI wurden für die Erstellung des AKS Clusters verwendet:

```
# Resource Group erstellen
az group create \
  --name energy-platform-rg \
  --location westeurope
```

Response:

```
{
  "id": "/subscriptions/327bb09a-492d-47ce-9601-8780ae9736cb/resourceGroups/energy-
platform-rg",
  "location": "westeurope",
  "managedBy": null,
  "name": "energy-platform-rg",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

```
# AKS Cluster erstellen
az aks create \
  --resource-group energy-platform-rg \
  --name energy-platform-aks \
  --node-count 3 \
  --node-vm-size Standard_D2s_v3 \
  --enable-managed-identity \
  --generate-ssh-keys \
  --network-plugin azure
```

Response:

```
{
  "aadProfile": null,
  "addonProfiles": null,
  "agentPoolProfiles": [
    {
      "artifactStreamingProfile": null,
      "availabilityZones": null,
      "capacityReservationGroupId": null,
      "count": 3,
      "creationData": null,
```

```
      "currentOrchestratorVersion": "1.32.7",
      "eTag": "b66eeda3-01c0-40bb-815f-464827a7df3d",
      "enableAutoScaling": false,
      "enableCustomCaTrust": false,
      "enableEncryptionAtHost": false,
      "enableFips": false,
      "enableNodePublicIp": false,
      "enableUltraSsd": false,
      "gatewayProfile": null,
      "gpuInstanceProfile": null,
      "gpuProfile": null,
      "hostGroupId": null,
      "kubeletConfig": null,
      "kubeletDiskType": "OS",
      "linuxOsConfig": null,
      "localDnsProfile": null,
      "maxCount": null,
      "maxPods": 30,
      "messageOfTheDay": null,
      "minCount": null,
      "mode": "System",
      "name": "nodepool1",
      "networkProfile": {
        "allowedHostPorts": null,
        "applicationSecurityGroups": null,
        "nodePublicIpTags": null
      },
      "nodeImageVersion": "AKSUbuntu-2204gen2containerd-202510.03.0",
      "nodeInitializationTaints": null,
      "nodeLabels": null,
      "nodePublicIpPrefixId": null,
      "nodeTaints": null,
      "orchestratorVersion": "1.32",
      "osDiskSizeGb": 128,
      "osDiskType": "Managed",
      "osSku": "Ubuntu",
      "osType": "Linux",
      "podIpAllocationMode": null,
      "podSubnetId": null,
      "powerState": {
        "code": "Running"
      },
      "provisioningState": "Succeeded",
      "proximityPlacementGroupId": null,
      "scaleDownMode": "Delete",
      "scaleSetEvictionPolicy": null,
      "scaleSetPriority": null,
      "securityProfile": {
        "enableSecureBoot": false,
        "enableVtpm": false,
        "sshAccess": "LocalUser"
```

```
      },
      "spotMaxPrice": null,
      "status": null,
      "tags": null,
      "type": "VirtualMachineScaleSets",
      "upgradeSettings": {
        "drainTimeoutInMinutes": null,
        "maxBlockedNodes": null,
        "maxSurge": "10%",
        "maxUnavailable": "0",
        "nodeSoakDurationInMinutes": null,
        "undrainableNodeBehavior": null
      },
      "virtualMachineNodesStatus": null,
      "virtualMachinesProfile": null,
      "vmSize": "Standard_D2s_v3",
      "vnetSubnetId": null,
      "windowsProfile": null,
      "workloadRuntime": "OCIContainer"
    }
  ],
  "aiToolchainOperatorProfile": null,
  "apiServerAccessProfile": null,
  "autoScalerProfile": null,
  "autoUpgradeProfile": {
    "nodeOsUpgradeChannel": "NodeImage",
    "upgradeChannel": null
  },
  "azureMonitorProfile": null,
  "azurePortalFqdn": "energy-pla-energy-platform--327bb0-
6l4a3tvl.portal.hcp.westeurope.azmk8s.io",
  "bootstrapProfile": {
    "artifactSource": "Direct",
    "containerRegistryId": null
  },
  "creationData": null,
  "currentKubernetesVersion": "1.32.7",
  "disableLocalAccounts": false,
  "diskEncryptionSetId": null,
  "dnsPrefix": "energy-pla-energy-platform--327bb0",
  "eTag": "90d7659a-dbb2-4ff1-a74f-0be3f7870031",
  "enableNamespaceResources": null,
  "enableRbac": true,
  "extendedLocation": null,
  "fqdn": "energy-pla-energy-platform--327bb0-6l4a3tvl.hcp.westeurope.azmk8s.io",
  "fqdnSubdomain": null,
  "httpProxyConfig": null,
  "id": "/subscriptions/327bb09a-492d-47ce-9601-8780ae9736cb/resourcegroups/energy-
platform-rg/providers/Microsoft.ContainerService/managedClusters/energy-platform-aks",
  "identity": {
    "delegatedResources": null,
```

```
    "principalId": "55c258fe-57ea-4a05-9bbe-e87afe5e6a28",
    "tenantId": "bba4121b-4d76-467a-9646-509616515867",
    "type": "SystemAssigned",
    "userAssignedIdentities": null
  },
  "identityProfile": {
    "kubeletidentity": {
      "clientId": "96a5aab3-efa0-4a73-9ef0-3463b79afe37",
      "objectId": "001ee600-fd3e-4de2-a647-3bc2996761e4",
      "resourceId": "/subscriptions/327bb09a-492d-47ce-9601-
8780ae9736cb/resourcegroups/MC_energy-platform-rg_energy-platform-
aks_westeurope/providers/Microsoft.ManagedIdentity/userAssignedIdentities/energy-
platform-aks-agentpool"
    }
  },
  "ingressProfile": null,
  "kind": "Base",
  "kubernetesVersion": "1.32",
  "linuxProfile": {
    "adminUsername": "azureuser",
    "ssh": {
      "publicKeys": [
        {
          "keyData": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCk89ElsClzr76eZHNEPkZ7pbBZtHo0xT9a4HU3IZkxChkec+0G+rTK8N
nQJ6gftathZOx9wxGLVdH4MmEjW0qNOvl1YTtbMNI1ISaN5xUJoSRro4JBvnIqGM3Nx8gWlpioe7uqZX1BzOSg
psw7SvWQvf/P2LrFIUzsp6+/MqbtcHuzaM5ESXP/rmsGJIDsbwmn/+Z1HLZ4nscAq8GnHBI874VJZipgqEBcUW
FkTzoHQPeJPsgBg+Ctb1tCic070eYIqokhUmEluuCq+LHBlysJyT6n3gIKfrpeVh1YtDWL284O3+aJ8sx/2YKi
e+armFHDc5VE5RmbtrbHYHFjiiSl"
        }
      ]
    }
  },
  "location": "westeurope",
  "maxAgentPools": 100,
  "metricsProfile": {
    "costAnalysis": {
      "enabled": false
    }
  },
  "name": "energy-platform-aks",
  "networkProfile": {
    "advancedNetworking": null,
    "dnsServiceIp": "10.0.0.10",
    "ipFamilies": [
      "IPv4"
    ],
    "kubeProxyConfig": null,
    "loadBalancerProfile": {
      "allocatedOutboundPorts": null,
      "backendPoolType": "nodeIPConfiguration",
```

```
      "clusterServiceLoadBalancerHealthProbeMode": null,
      "effectiveOutboundIPs": [
        {
          "id": "/subscriptions/327bb09a-492d-47ce-9601-
8780ae9736cb/resourceGroups/MC_energy-platform-rg_energy-platform-
aks_westeurope/providers/Microsoft.Network/publicIPAddresses/c27b3136-e09b-49a9-95e6-
2c1019a88341",
          "resourceGroup": "MC_energy-platform-rg_energy-platform-aks_westeurope"
        }
      ],
      "enableMultipleStandardLoadBalancers": null,
      "idleTimeoutInMinutes": null,
      "managedOutboundIPs": {
        "count": 1,
        "countIpv6": null
      },
      "outboundIPs": null,
      "outboundIpPrefixes": null
    },
    "loadBalancerSku": "standard",
    "natGatewayProfile": null,
    "networkDataplane": "azure",
    "networkMode": null,
    "networkPlugin": "azure",
    "networkPluginMode": null,
    "networkPolicy": "none",
    "outboundType": "loadBalancer",
    "podCidr": null,
    "podCidrs": null,
    "podLinkLocalAccess": "IMDS",
    "serviceCidr": "10.0.0.0/16",
    "serviceCidrs": [
      "10.0.0.0/16"
    ],
    "staticEgressGatewayProfile": null
  },
  "nodeProvisioningProfile": {
    "defaultNodePools": "Auto",
    "mode": "Manual"
  },
  "nodeResourceGroup": "MC_energy-platform-rg_energy-platform-aks_westeurope",
  "nodeResourceGroupProfile": null,
  "oidcIssuerProfile": {
    "enabled": false,
    "issuerUrl": null
  },
  "podIdentityProfile": null,
  "powerState": {
    "code": "Running"
  },
  "privateFqdn": null,
```

```
  "privateLinkResources": null,
  "provisioningState": "Succeeded",
  "publicNetworkAccess": null,
  "resourceGroup": "energy-platform-rg",
  "resourceUid": "68fa2e1d77ed8100010a872b",
  "schedulerProfile": null,
  "securityProfile": {
    "azureKeyVaultKms": null,
    "customCaTrustCertificates": null,
    "defender": null,
    "imageCleaner": null,
    "imageIntegrity": null,
    "nodeRestriction": null,
    "workloadIdentity": null
  },
  "serviceMeshProfile": null,
  "servicePrincipalProfile": {
    "clientId": "msi",
    "secret": null
  },
  "sku": {
    "name": "Base",
    "tier": "Free"
  },
  "status": null,
  "storageProfile": {
    "blobCsiDriver": null,
    "diskCsiDriver": {
      "enabled": true,
      "version": "v1"
    },
    "fileCsiDriver": {
      "enabled": true
    },
    "snapshotController": {
      "enabled": true
    }
  },
  "supportPlan": "KubernetesOfficial",
  "systemData": null,
  "tags": null,
  "type": "Microsoft.ContainerService/ManagedClusters",
  "upgradeSettings": null,
  "windowsProfile": {
    "adminPassword": null,
    "adminUsername": "azureuser",
    "enableCsiProxy": true,
    "gmsaProfile": null,
    "licenseType": null
  },
  "workloadAutoScalerProfile": {
```

```
    "keda": null,
    "verticalPodAutoscaler": null
  }
}
```

4.2. Beispiele

4.2.1. Namespaces und RBAC

Auf diese Weise kann die Isolation und Sicherheit zwischen den Teams gewährleistet werden.

- **Namespaces pro Team erstellen**

```
kubectl create namespace team-1
kubectl create namespace team-2
kubectl create namespace team-3
kubectl create namespace monitoring
kubectl create namespace auth
kubectl create namespace data
kubectl create namespace ingress-nginx
```

- **Service Accounts und RBAC pro Team**

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: team-1-deployer
  namespace: team-1
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: team-1-full-access
  namespace: team-1
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: team-1-full-access
  namespace: team-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: team-1-full-access
```

```
subjects:
- kind: ServiceAccount
  name: team-1-deployer
  namespace: team-1
---
# Read-Only Zugriff auf monitoring und auth
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: team-1-monitoring-readonly
  namespace: monitoring
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- kind: ServiceAccount
  name: team-1-deployer
  namespace: team-1
```

- **Kubeconfig für Teams generieren**

```
# Script um Team-spezifische Kubeconfig zu erstellen
./scripts/create-team-kubeconfig.sh team-1 team-1-deployer team-1
```

4.2.2. Network Policies

- **Isolation zwischen Teams**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-from-other-namespaces
  namespace: team-1
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector: {}
  - from:
    - namespaceSelector:
        matchLabels:
          name: monitoring
  - from:
    - namespaceSelector:
        matchLabels:
```

```
          name: ingress-nginx
```

4.2.3. NGINX Ingress Controller

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

helm upgrade --install ingress-nginx ingress-nginx/ingress-nginx \
   --namespace ingress-nginx \
   --create-namespace \
   --set controller.config.http2=true \
   --set controller.config.http2-push="on" \
   --set controller.config.http2-push-preload="on" \
   --set controller.ingressClassByName=true \
   --set controller.ingressClassResource.controllerValue=k8s.io/ingress-nginx \
   --set controller.ingressClassResource.enabled=true \
   --set controller.ingressClassResource.name=public \
   --set controller.service.externalTrafficPolicy=Local \
   --set controller.setAsDefaultIngress=true \
```

- **Cert-Manager für TLS**

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --set installCRDs=true
```

Im Anschluss ClusterIssuer für Let's Encrypt anlegen (siehe k8/cluster-issuer.yaml). Üblicherweise könnnen so kostenlose TLS-Zertifikate automatisch verwaltet werden.

Nach der Installation müssen die DNS-Einträge (A-Records) für die gewünschten Domains (z. B. keycloak.machasing.com, grafana.machasing.com) auf die Public IP des AKS Clusters angepasst werden.

Zum Testen kann auch k8/hello-world.yaml verwendet werden. Dies erstellt einen einfachen NGINX Service mit Ingress und TLS. Das beigelegte YAML file verwendet die URI `https://test.machasing.com`.

Keycloak

- https://keycloak.machasing.com (wurde DNS seitig angelegt)

```
helm install my-keycloak oci://registry-1.docker.io/cloudpirates/keycloak \
  --namespace auth \
```

```
  --create-namespace \
  -f ./keycloak-values.yaml
```

RabbitMQ

Hier ein Beispieldeployment mit einem values File (rabbitmq-values.yaml):

```
helm install my-rabbitmq oci://registry-1.docker.io/cloudpirates/rabbitmq \
    --namespace rabbitmq \
    --create-namespace \
    -f ./rabbitmq-values.yaml
```

MongoDB

Chart: https://artifacthub.io/packages/helm/cloudpirates-mongodb/mongodb

- **Azure Disk für Persistence**
  - Storage Class: `managed-premium` (SSD)
  - Backup-Strategie definieren
- **Datenbanken pro Team erstellen**

```
// MongoDB Shell
use admin
db.auth("root", "<secure-password>")

// Pro Team
db.createUser({
  user: "team1",
  pwd: "<team1-password>",
  roles: [{ role: "readWrite", db: "team1_energy_data" }]
})

db.createUser({
  user: "team2",
  pwd: "<team2-password>",
  roles: [{ role: "readWrite", db: "team2_energy_data" }]
})
```

- **Connection Strings für Teams als Secrets**

```
kubectl create secret generic mongodb-team1 \
  --namespace team-1 \
  --from-literal=connection
-string="mongodb://team1:<password>@mongodb.data.svc.cluster.local:27017/team1_energy_
data"
```

4.2.4. RabbitMQ

- RabbitMQ benötigt **LoadBalancer Service** damit lokale k3s Cluster sich verbinden können!
- **VHosts und User pro Team** (Beispiel)

```
# RabbitMQ Management API oder UI
rabbitmqctl add_vhost /team1
rabbitmqctl add_user team1 <team1-password>
rabbitmqctl set_permissions -p /team1 team1 ".*" ".*" ".*"

rabbitmqctl add_vhost /team2
rabbitmqctl add_user team2 <team2-password>
rabbitmqctl set_permissions -p /team2 team2 ".*" ".*" ".*"
```

4.2.5. Monitoring Stack (Prometheus + Grafana)

Hier ein weiteres Beispiel wie der Monitoring Stack mit Helm installiert und konfiguriert werden kann. Der Stack besteht aus * Prometheus * Grafana * Alertmanager

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm install prometheus prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false \
  --set prometheus.prometheusSpec.retention=30d \
  --set
prometheus.prometheusSpec.storageSpec.volumeClaimTemplate.spec.storageClassName=manage
d-premium \
  --set
prometheus.prometheusSpec.storageSpec.volumeClaimTemplate.spec.resources.requests.stor
age=100Gi \
  --set grafana.ingress.enabled=true \
  --set grafana.ingress.ingressClassName=public \
  --set grafana.ingress.hosts[0]=grafana.machasing.com
```

DNS grafana.machasing.com wurde bereits angelegt.

ServiceMonitor für Team-Services:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: team-services
  namespace: monitoring
  labels:
    prometheus: kube-prometheus
spec:
```

```
  namespaceSelector:
    matchNames:
      - team-1
      - team-2
      - team-3
  selector:
    matchLabels:
      prometheus: "true"
  endpoints:
  - port: metrics
    path: /metrics
    interval: 30s
```

5. MVP Definition: Application Development Teams

Jedes Application Development Team muss bis zum **23. Januar 2026** folgendes implementieren:

- **Smart Meter Simulator**: Simuliert Verbrauchsdaten und sendet diese an RabbitMQ.
- **CI/CD Pipeline**: Automatisiert den Build, Test und Deployment Prozess.
- **Energie-Daten Service**: Empfängt und speichert die Daten in MongoDB.
- **Integration mit API Gateway**: Service ist über das API Gateway erreichbar.
- **Dashboard in Grafana**: Produzierte/Verbrauchte Energie in einem definierbaren Zeitraum gesamt und pro Zählpunkt.
- **Ein** weiterer Service ihrer Wahl als MVP im Rahmen der definierten Anforderungen.

6. Hilfreiche Ressourcen

6.1. Azure & AKS

- AKS Documentation

6.2. RabbitMQ

- RabbitMQ Virtual Hosts

6.3. Allgemein

- Helm Documentation
- Kubernetes Documentation
- Prometheus Documentation
- Grafana Documentation

Viel Erfolg beim Projekt!