Agenda

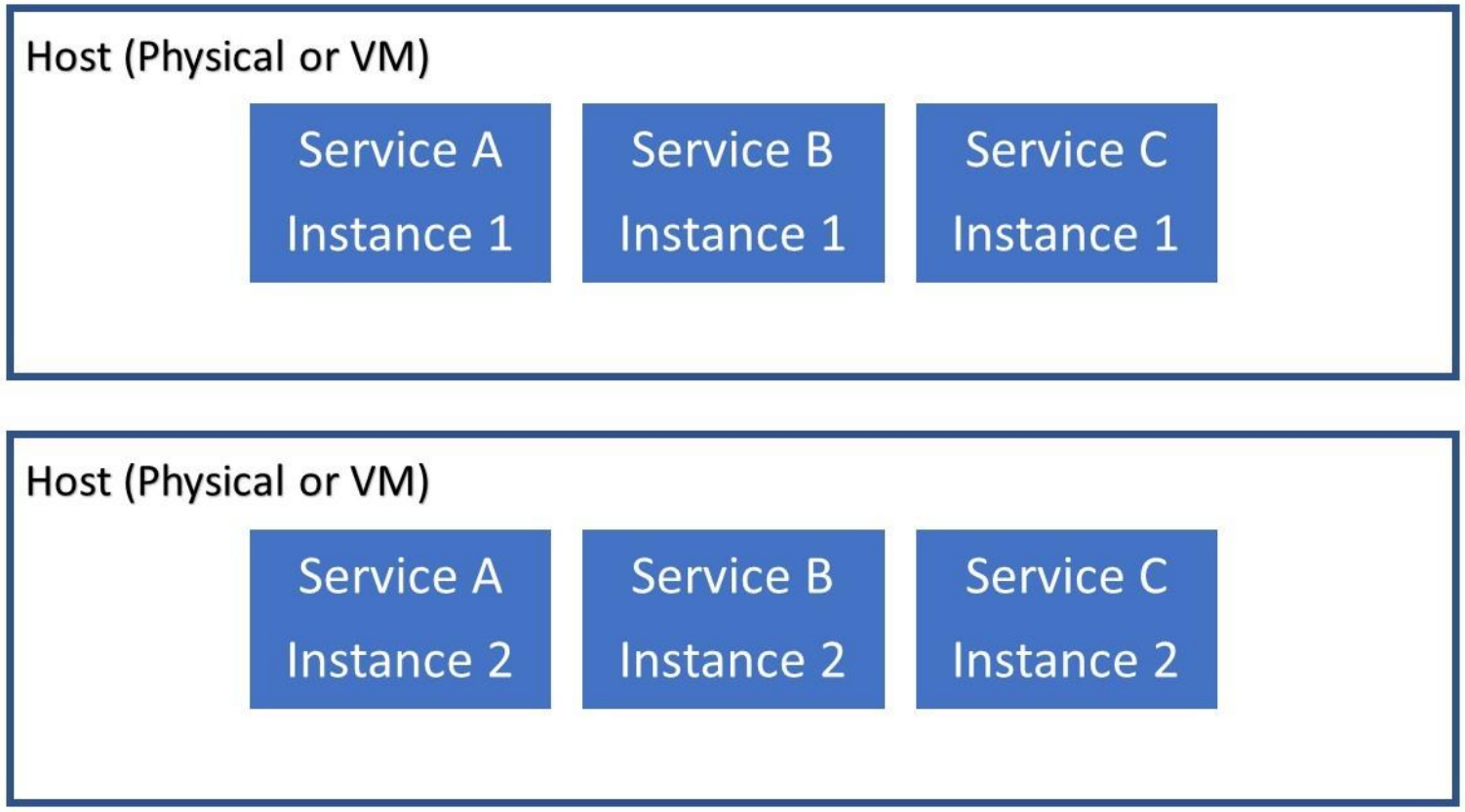
Deployment Patterns & AKS

- Deployment Patterns
- Azure Kubernetes Service (AKS)
- Accessing AKS Cluster
- AKS Deployment Using CLI
- Accessing AKS Application

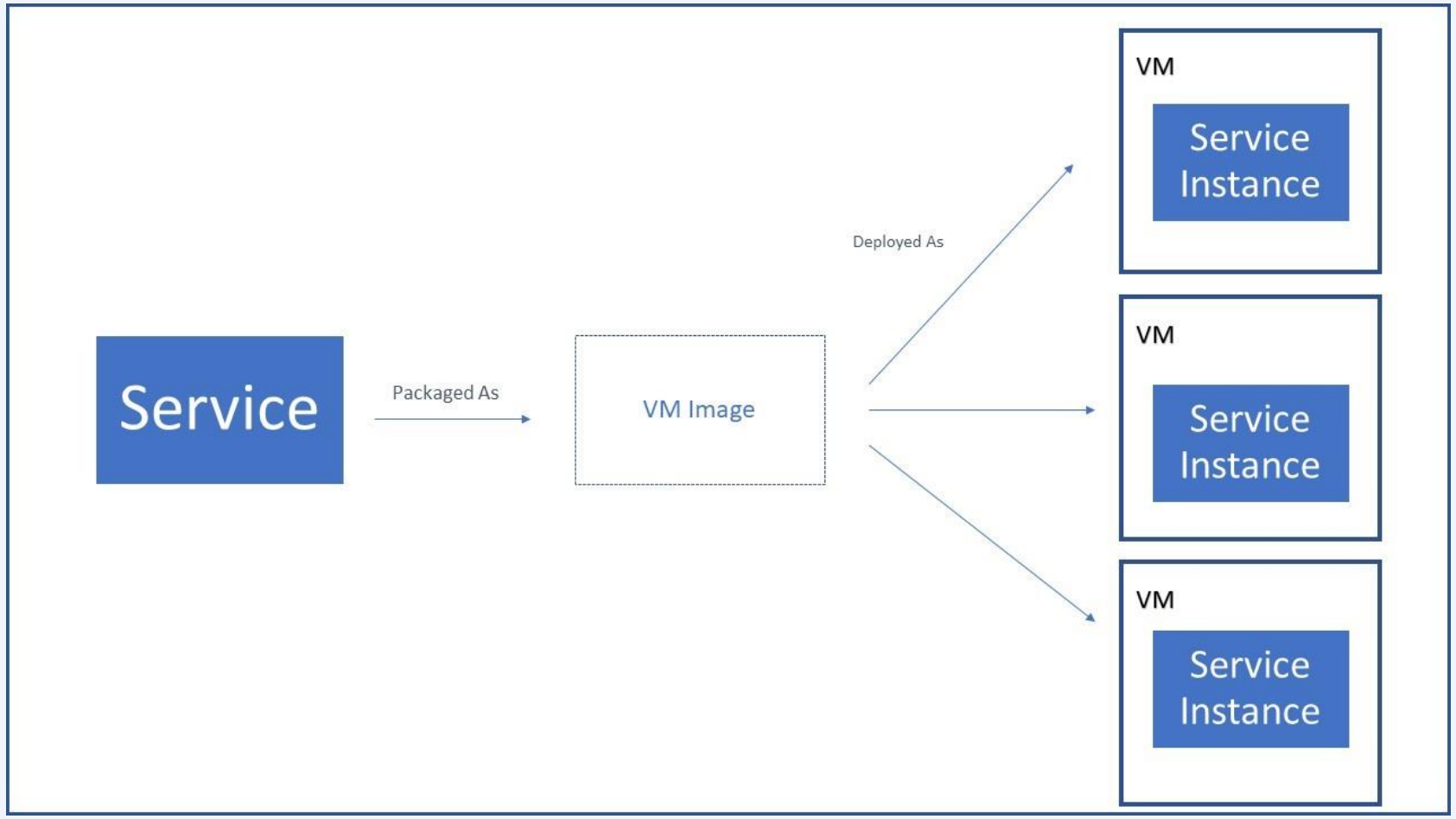
Deployment Patterns

- **Problem**: How to deploy an application's microservices?
- **Multiple Service Instances per Host** - deploy multiple service instances on a single host/VM
- **Service Instance per Host** - deploy each service instance in its own Host/VM/Container
- **Serverless Deployment** - deploy a service using serverless deployment platform
- **Service Deployment Platform** - deploy services using a highly automated deployment platform that provides a service abstraction

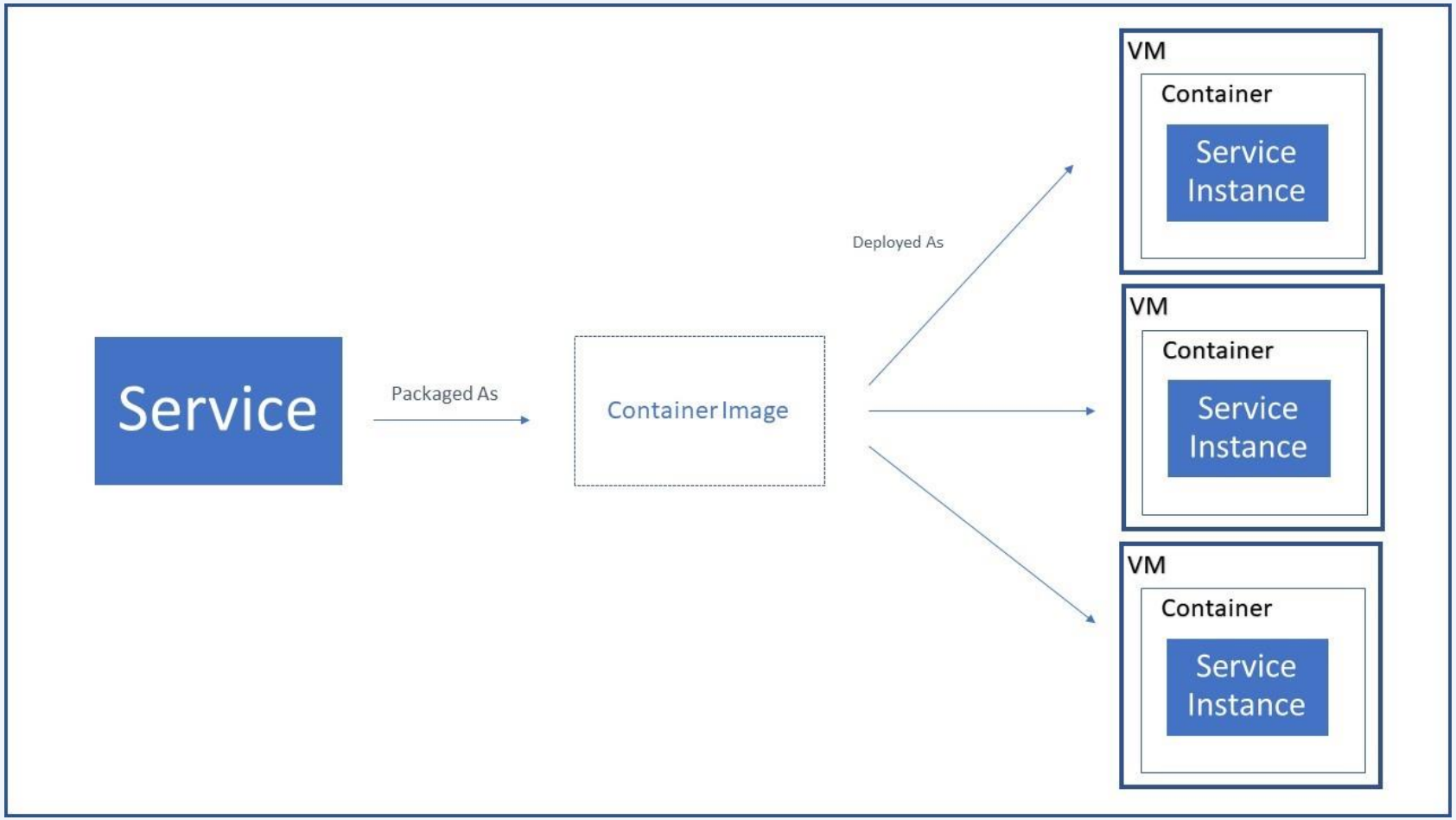
1. Multiple Service Instances Per Host



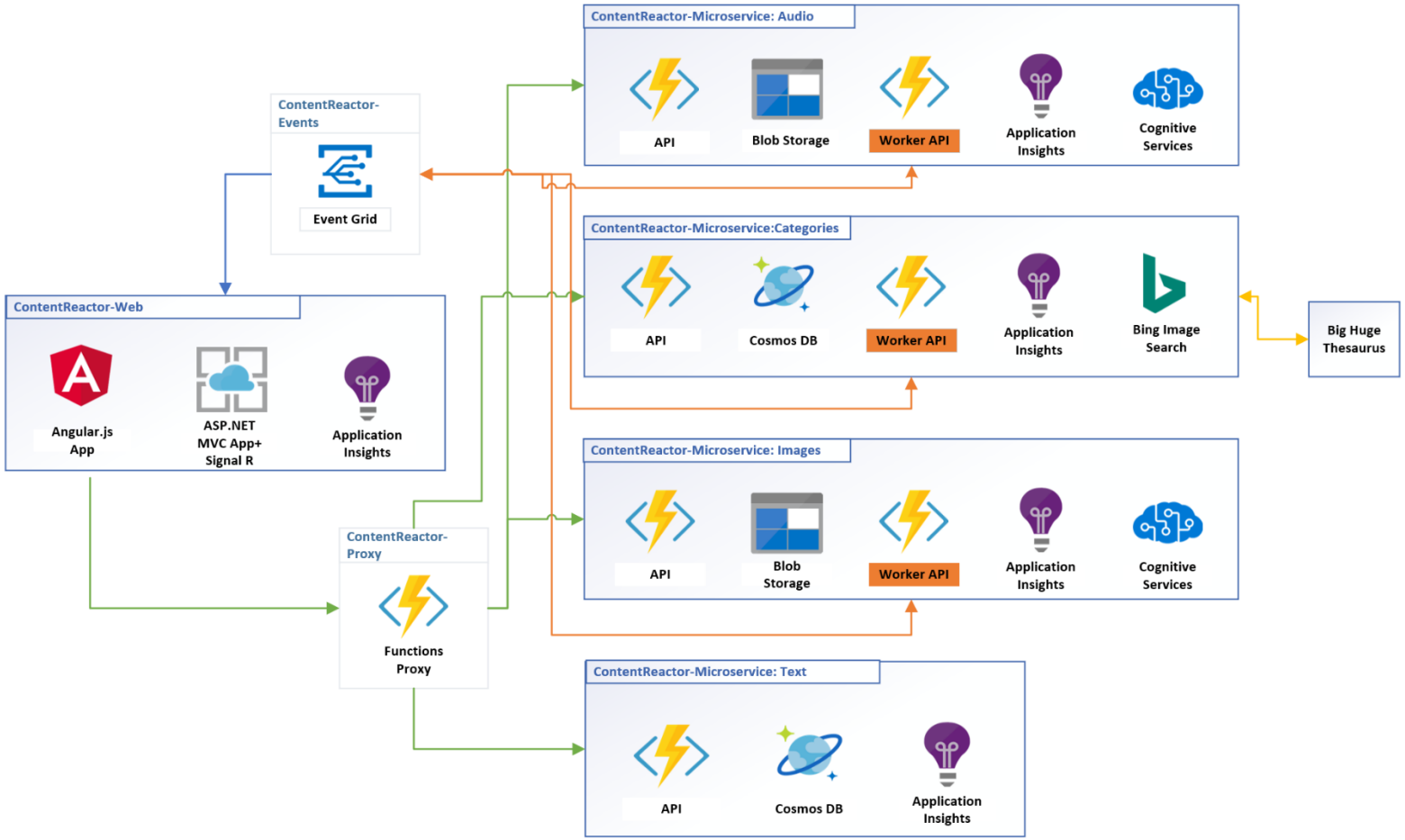
2. Service Instance Per Host/VM



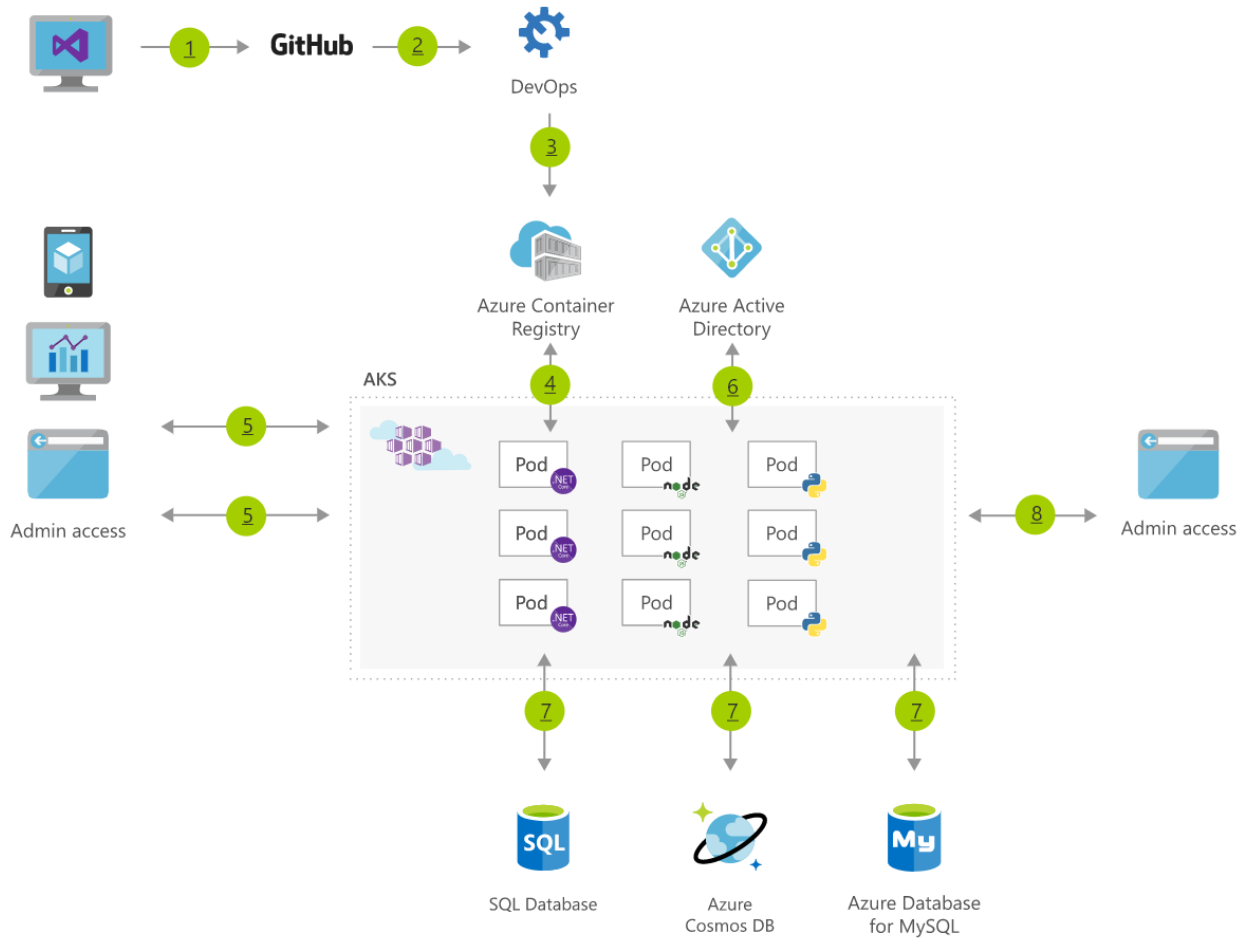
2. Service Instance Per Container



3. Serverless Deployment



4. Service Deployment Platform: AKS



ScholarHat

Azure Kubernetes Service

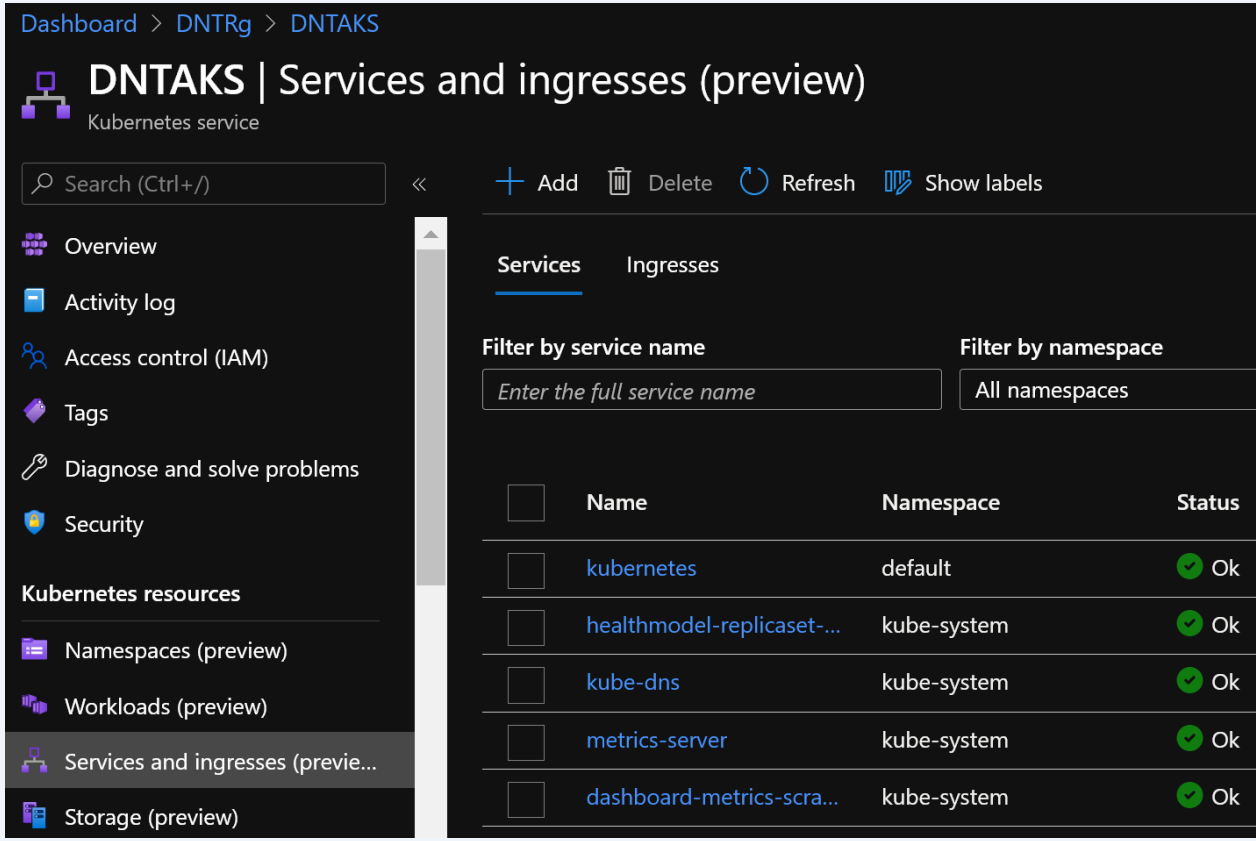
- A fully-managed Kubernetes service
- Offers serverless Kubernetes, an integrated continuous integration and continuous delivery (CI/CD) experience
- Offers enterprise-grade security and governance.
- Paying for only the virtual machines and associated storage and networking resources.
- There is no charge for cluster management.

Azure Kubernetes Service (AKS)

Accessing AKS Using CLI

```
> az aks install-cli
> az aks get-credentials -g <DNTRg> -n <DNTAKS>
> kubectl get nodes
```

```
NAME                                STATUS   ROLES   AGE     VERSION
aks-agentpool-28305802-vmss000000   Ready    agent   3h11m   v1.17.11
aks-agentpool-28305802-vmss000001   Ready    agent   3h11m   v1.17.11
aks-agentpool-28305802-vmss000002   Ready    agent   3h11m   v1.17.11
```



ScholarHat

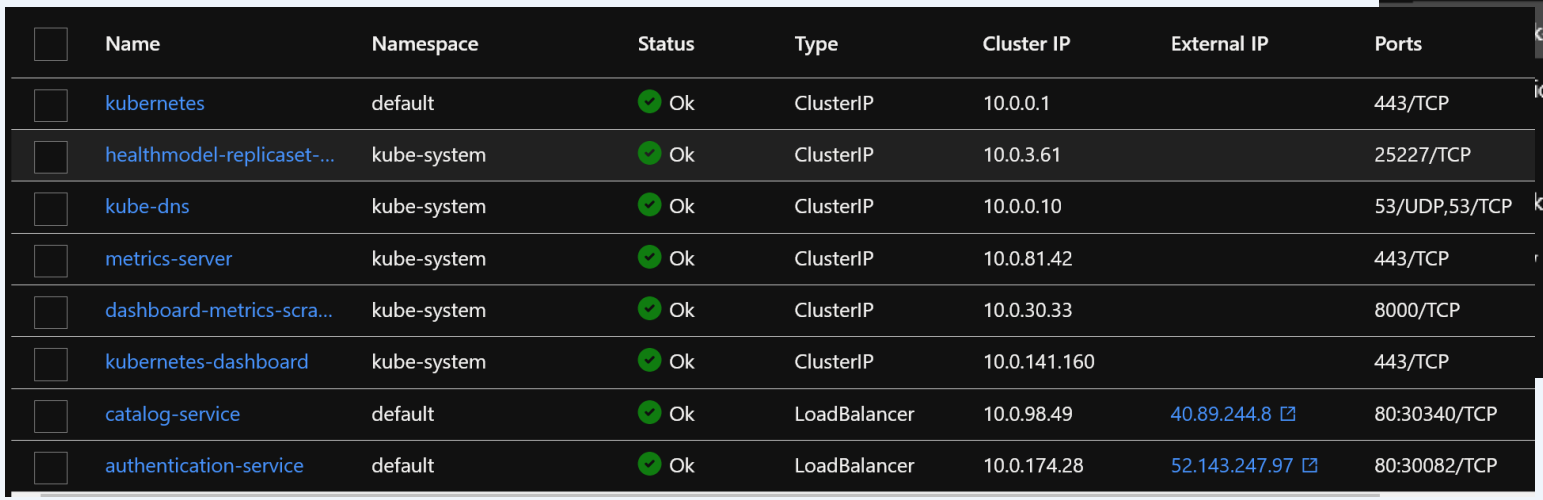
AKS Deployment Using CLI

```
> az acr login --name <registry-name>
> az account set -s <subsription_name_or_id>
```

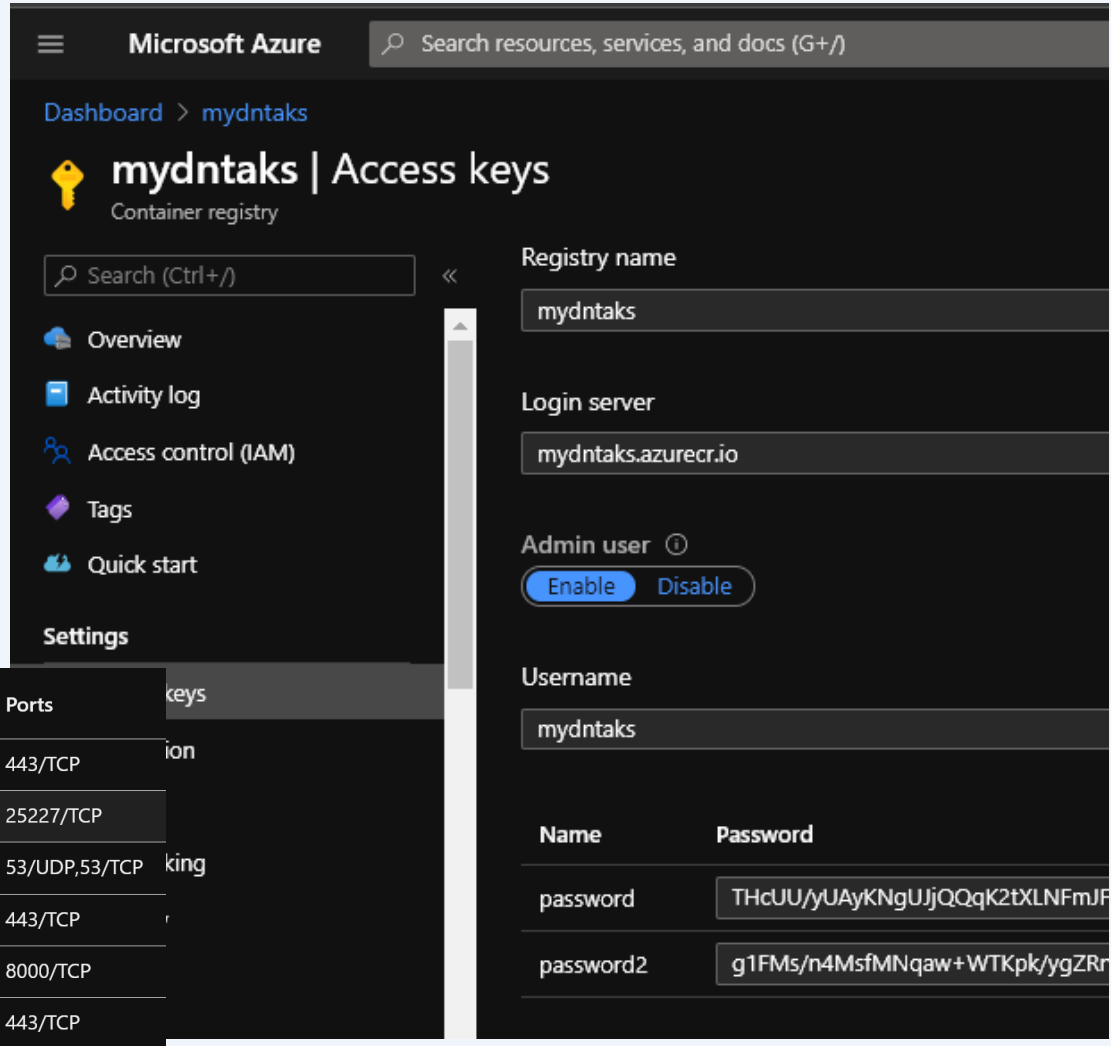
```
>docker build -t catalog:v1 . -f ./CatalogMicroservice/Dockerfile
>docker tag catalog:v1 dntrepo.azurecr.io/catalogservice:v1
>docker push dntrepo.azurecr.io/catalogservice:v1
```

```
> kubectl create secret docker-registry <secret-name> --docker-server=<container-registry-name>.azurecr.io --docker-username=<acr-username> --docker-password=<acr-password>
```

```
> kubectl apply -f catalog-deployment.yaml
```



Name	Namespace	Status	Type	Cluster IP	External IP	Ports
kubernetes	default	Ok	ClusterIP	10.0.0.1		443/TCP
healthmodel-replicaset-...	kube-system	Ok	ClusterIP	10.0.3.61		25227/TCP
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP
metrics-server	kube-system	Ok	ClusterIP	10.0.81.42		443/TCP
dashboard-metrics-scra...	kube-system	Ok	ClusterIP	10.0.30.33		8000/TCP
kubernetes-dashboard	kube-system	Ok	ClusterIP	10.0.141.160		443/TCP
catalog-service	default	Ok	LoadBalancer	10.0.98.49	40.89.244.8	80:30340/TCP
authentication-service	default	Ok	LoadBalancer	10.0.174.28	52.143.247.97	80:30082/TCP



Steps to Build and Deploy Microservices

- **Step1:** Create Microservices Using ASP.NET Core.
- **Step2:** Add Docker Support using Visual Studio.
- **Step3:** Create ACR and Push Docker Images to ACR using VS.
- **Step4:** Create AKS Cluster and Configure it to Access Locally.
- **Step5:** Create Kubernetes Deployment Files.
- **Step6:** Deploy Microservices to AKS using CLI.
- **Step7:** Verify and Test Your Deployments.
- **Step8:** Create Azure API Management Instance and Add Your APIs.
- **Step9:** Deploy Your Frontend and Access backend Services using APIM.