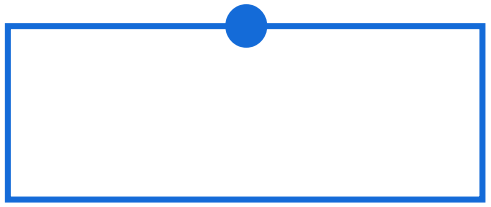
Agenda

Kubernetes Advanced

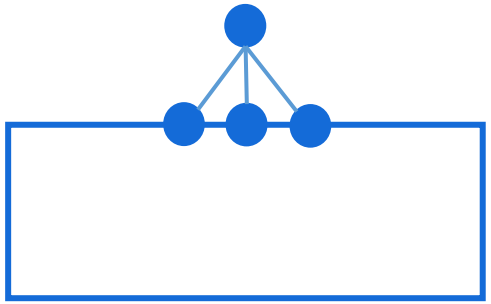
- Types of Service
- Deployment
- Replica set
- Namespace
- What is Azure Kubernetes Service (AKS)
- AKS Benefits and Use Cases
- Accessing AKS Cluster
- AKS Deployment Using CLI
- Accessing AKS Application
- Azure DevOps CI/CD Pipeline with AKS

Types of Services



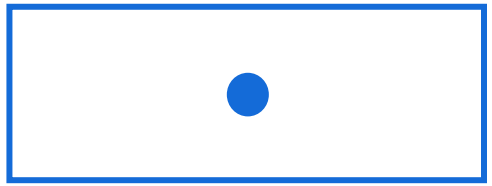
NodePort

- Expose App to external world



LoadBalancer

- Equally distribute the load in nodes



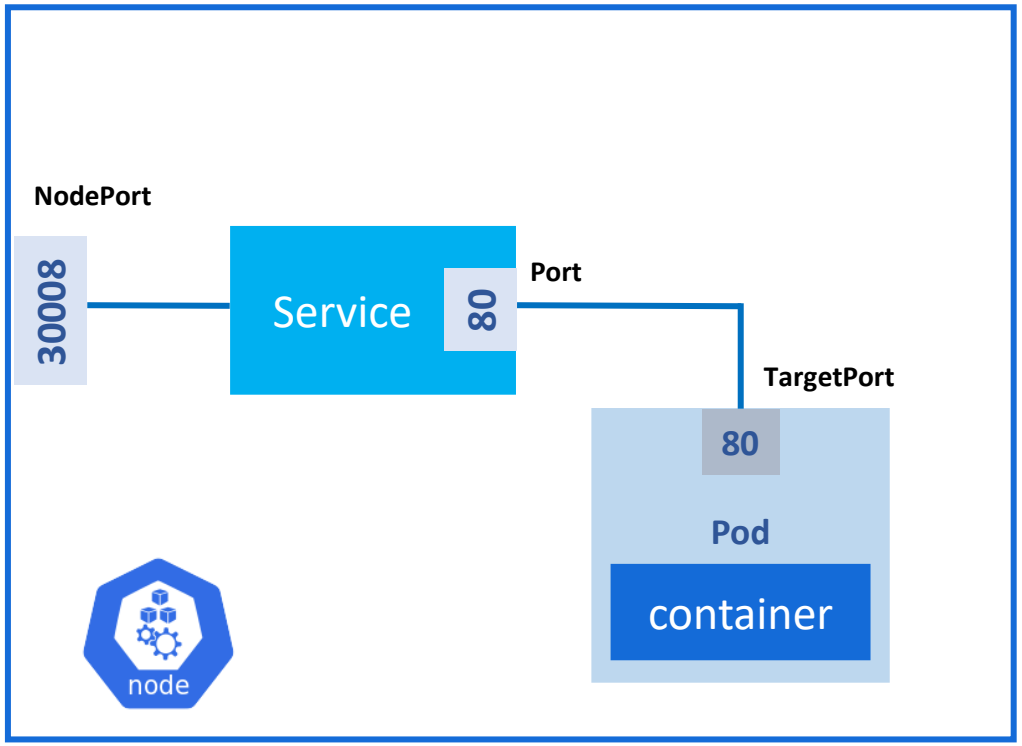
ClusterIP

- Reachable within the cluster
- Connect front-end pod to backend pod

ScholarHat

NodePort Service

Range: 30000-32767




```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    name: my-app
  ports:
  - port: 80
    nodePort: 30008
  type: NodePort
```

ScholarHat

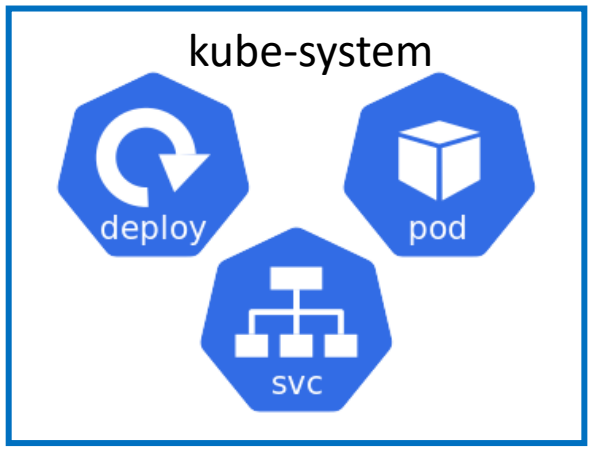
LoadBalancer Service

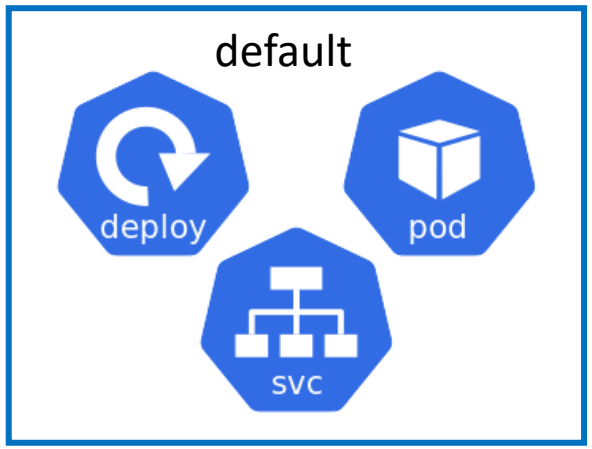
```yaml
apiVersion: v1
kind: Service
metadata:
  name: aspnet-service
spec:
  selector:
    app: aspnet-pod
  ports:
  - port: 3080
    targetPort: 80
  type: LoadBalancer
```

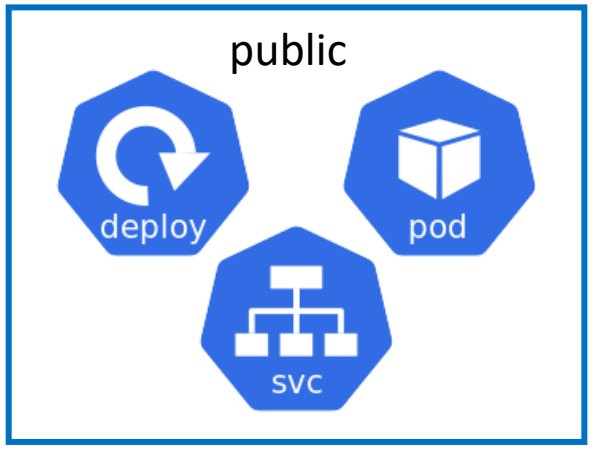
Deployments

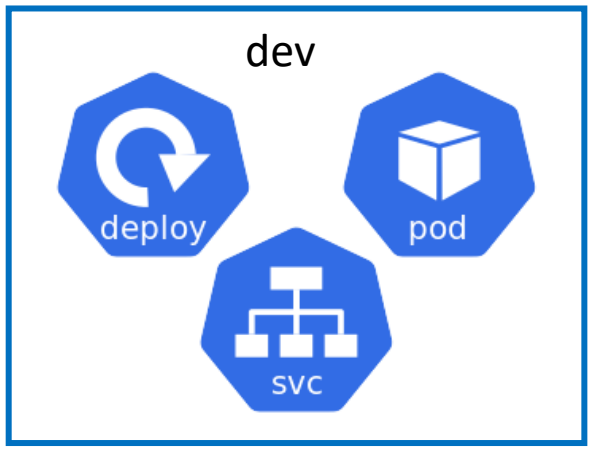
- Represent a set of multiple and identical Pods .
- A deployment is responsible for keeping a set of pods running.
- A deployment can be used without a service to keep a set of identical pods running in the Kubernetes cluster.
- Without service, Each pod could be accessed individually via direct network requests (rather than abstracting them behind a service).
- Services and Deployments can work together.

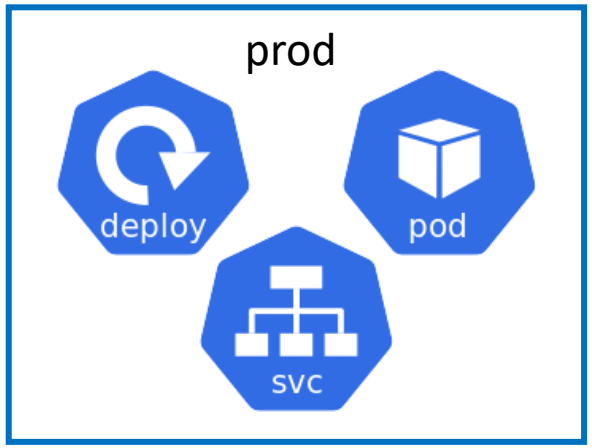
Namespace - Isolation











ScholarHat

Namespace Commands

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name:  dev
```

```
> kubectl apply –f dev-namespace.yaml
> Kubectl get namespaces
> kubectl apply –f my-pod.yaml --namespace=dev
> kubectl get pods --namespace=dev
```

```
> kubectl get pods --namespace=dev
> kubectl get pods --all-namespaces
```

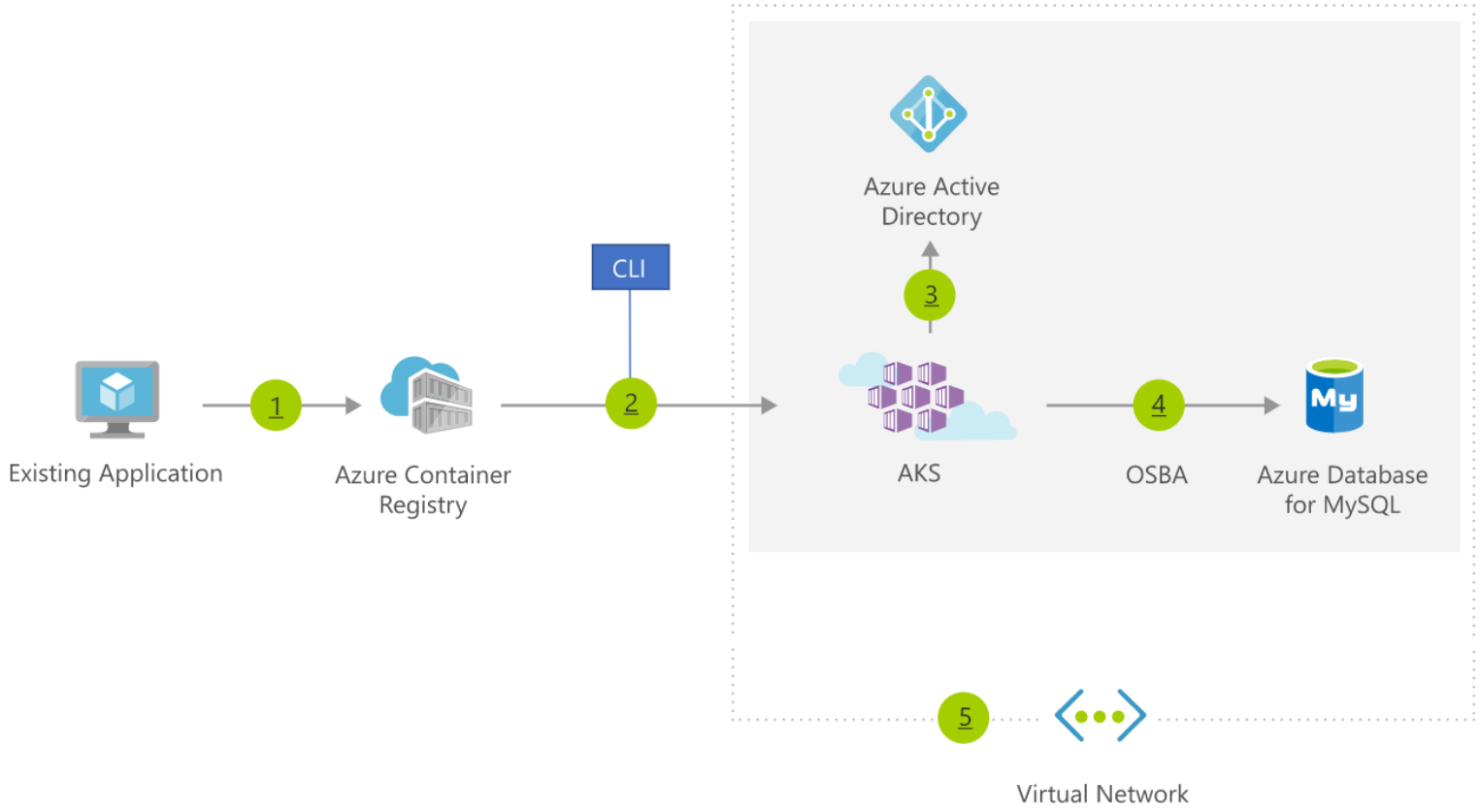
```
> kubectl config set-context $(kubectl config current-context) --namespace=dev
```

Azure Kubernetes Service (AKS)

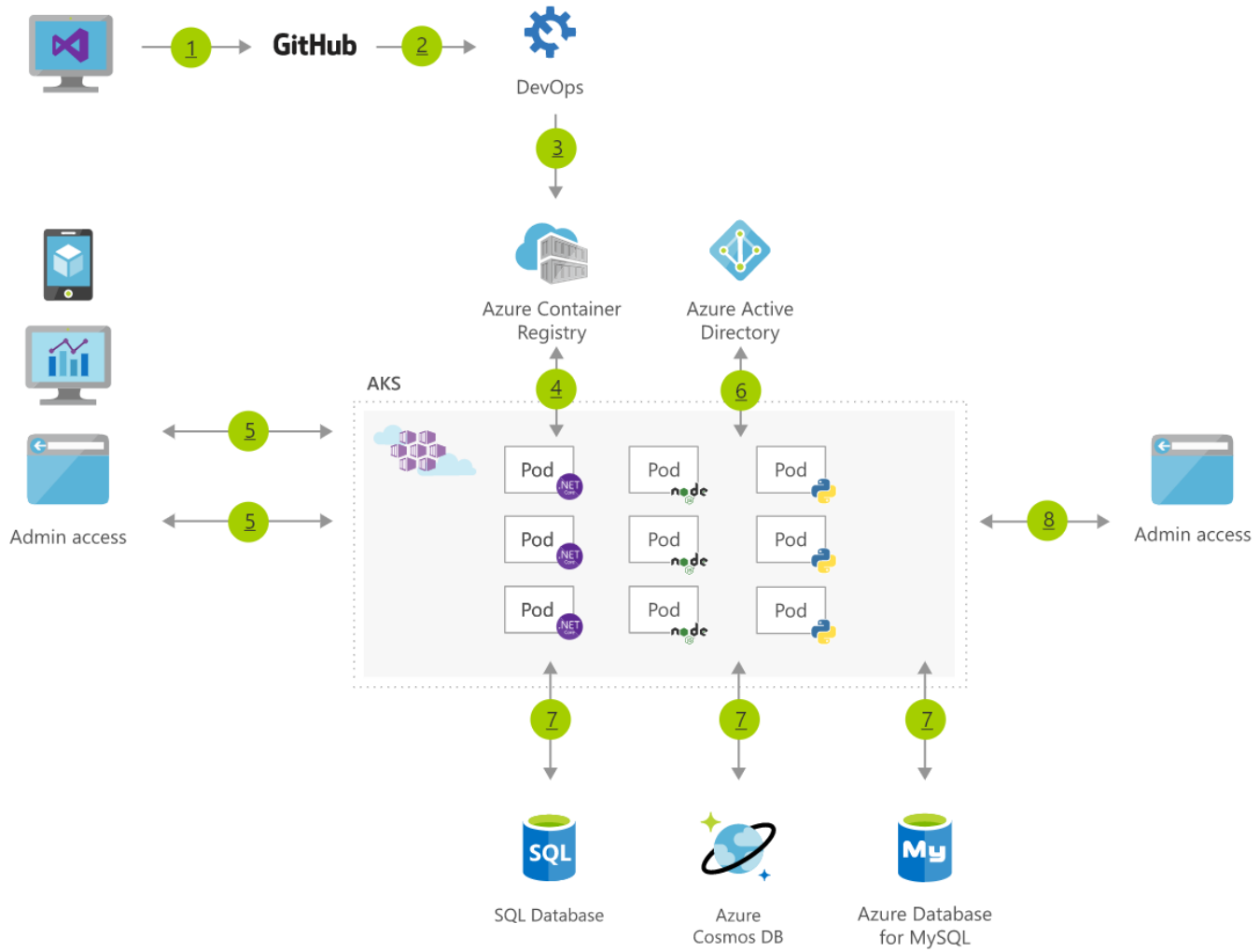
Azure Kubernetes Service Benefits

- 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.

Use Case : Migrate an existing Application



Use Case: Microservices Deployment

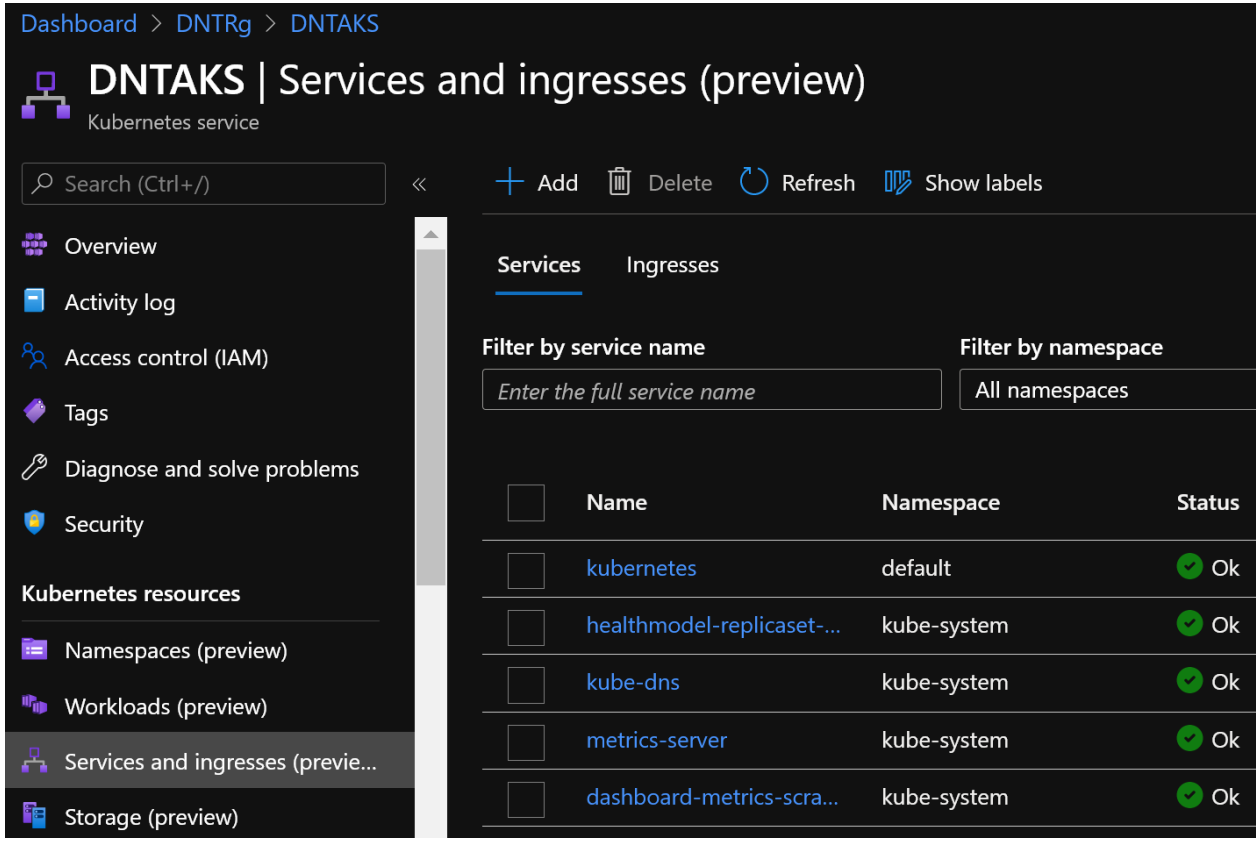


ScholarHat

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

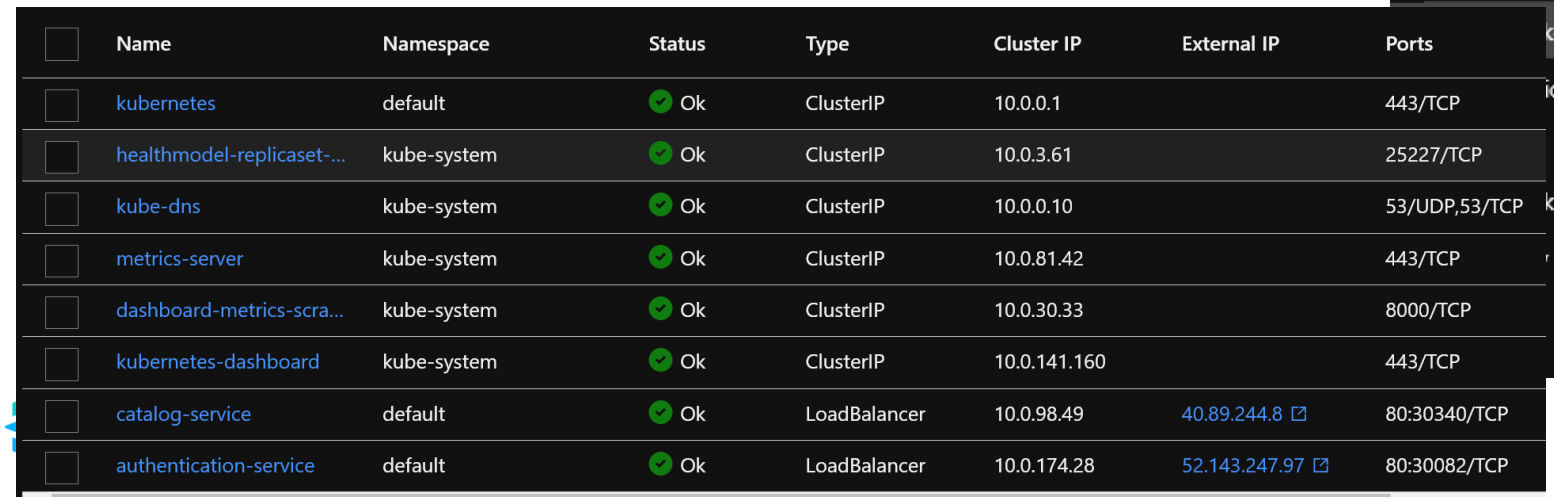
AKS Deployment Using CLI

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

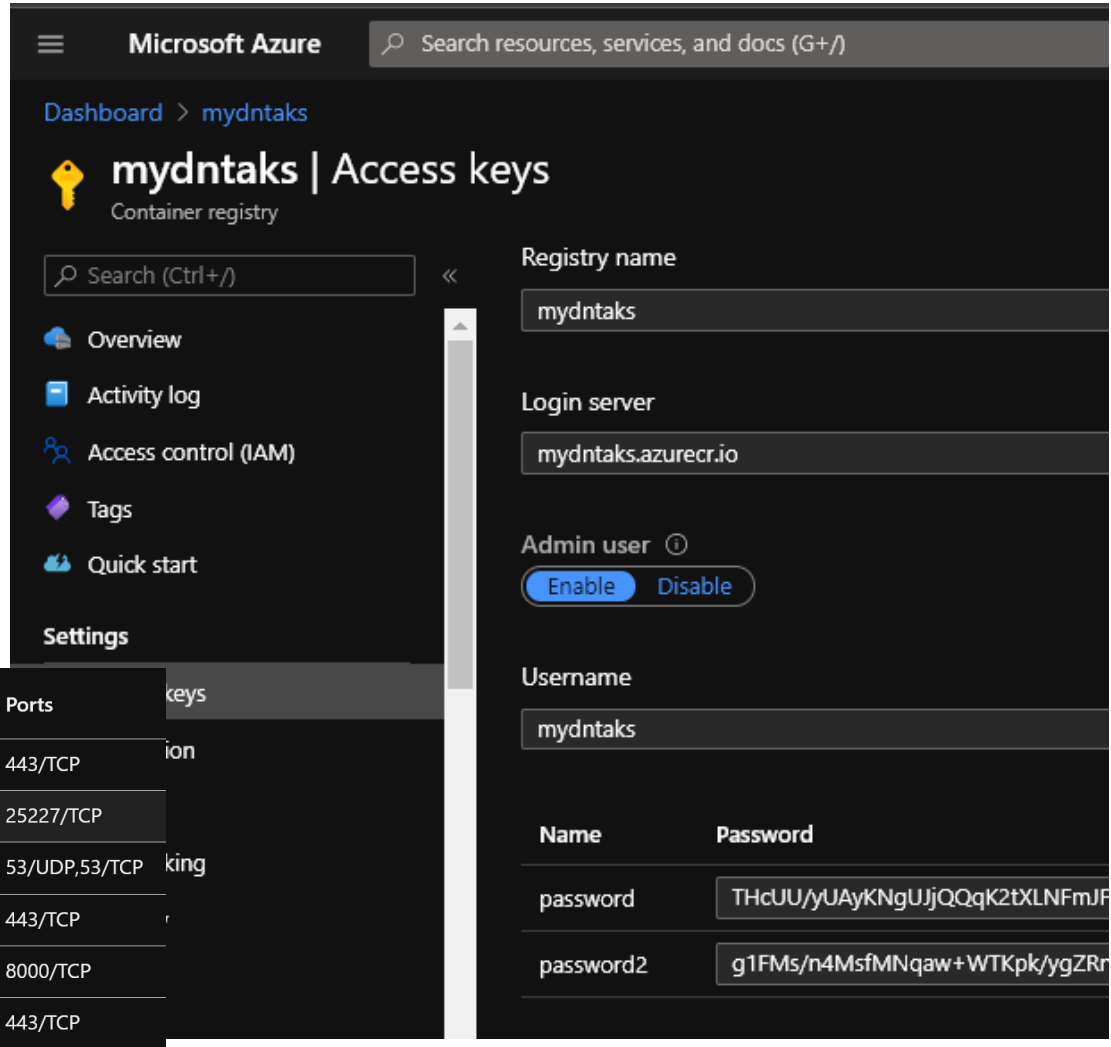
```
>docker build -t catalog:v1 . -f ./CatalogService/Dockerfile
>docker tag catalog:v1 dntrepo.azurecr.io/catalogservice:v1
>docker push dntrepo.azurecr.io/catalogservice:v1
```

```
> 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  ./catalogservice catalog-deployment.yaml
```



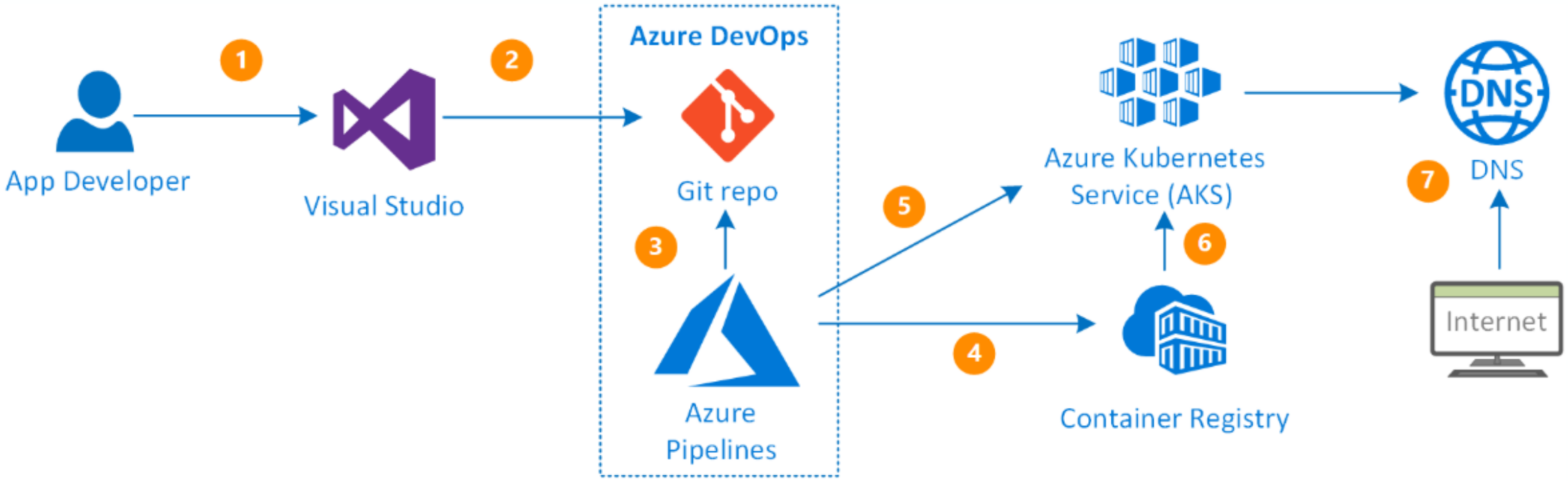
Name	Namespace	Status	Type	Cluster IP	External IP	Ports
kubernetes	default	Ok	ClusterIP	10.0.0.1		443/TCP
healthmodel-replicaset-...	kube-system	Ok	ClusterIP	10.0.3.61		25227/TCP
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP
metrics-server	kube-system	Ok	ClusterIP	10.0.81.42		443/TCP
dashboard-metrics-scra...	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 For Deployment

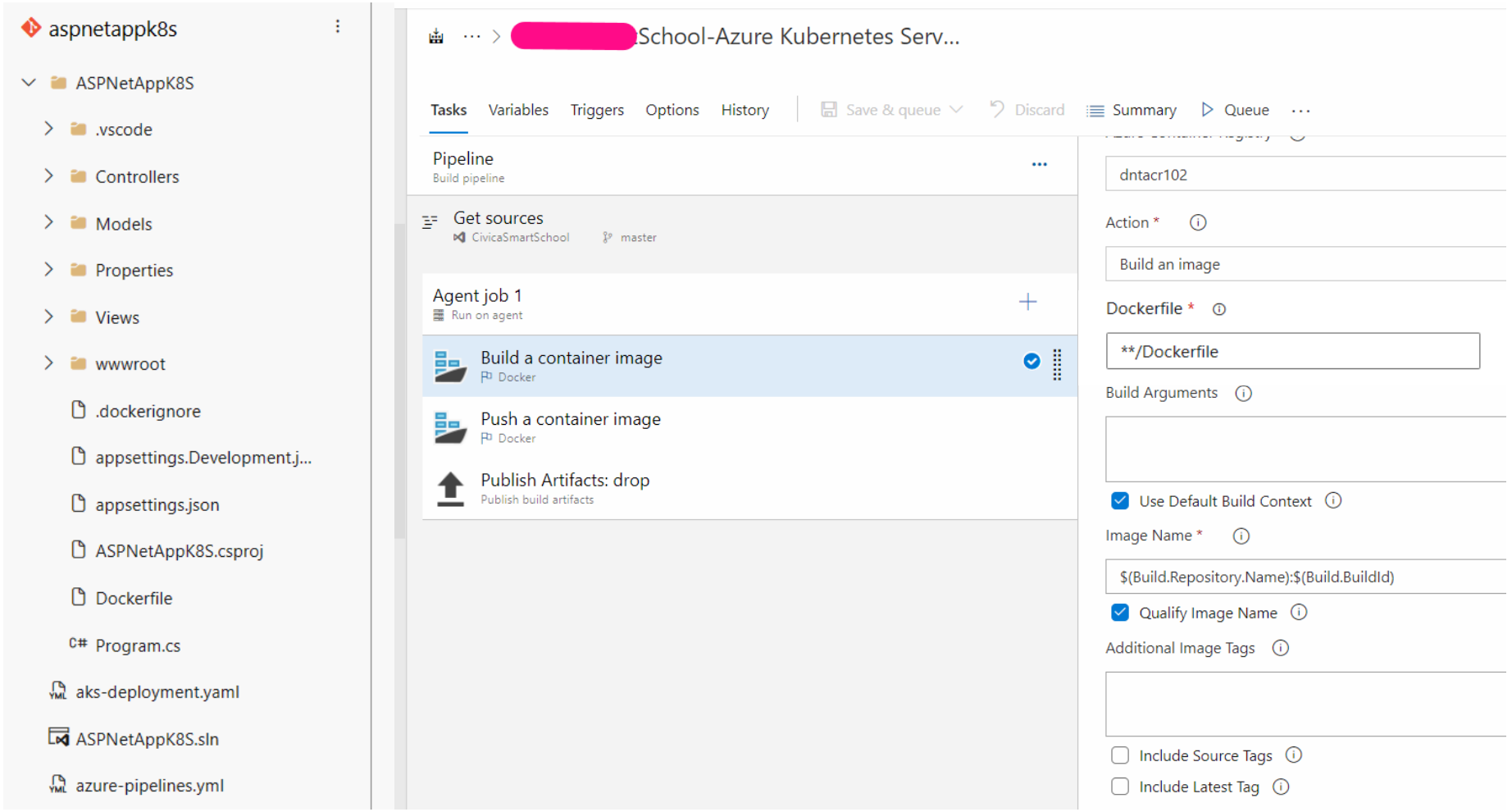
- **Step1:** Create Application Using ASP.NET Core.
- **Step2:** Add Docker Support using Visual Studio.
- **Step3:** Create ACR and Push Docker Images to ACR using VS.
- **Step4:** Create AKS Cluster and Configure it to Access Locally.
- **Step5:** Create Kubernetes Deployment Files.
- **Step6:** Deploy Application to AKS using CLI.
- **Step7:** Verify and Test Your Deployments.

CI/CD Pipeline Using Azure DevOps and AKS



ScholarHat

Building Image and Publish to ACR

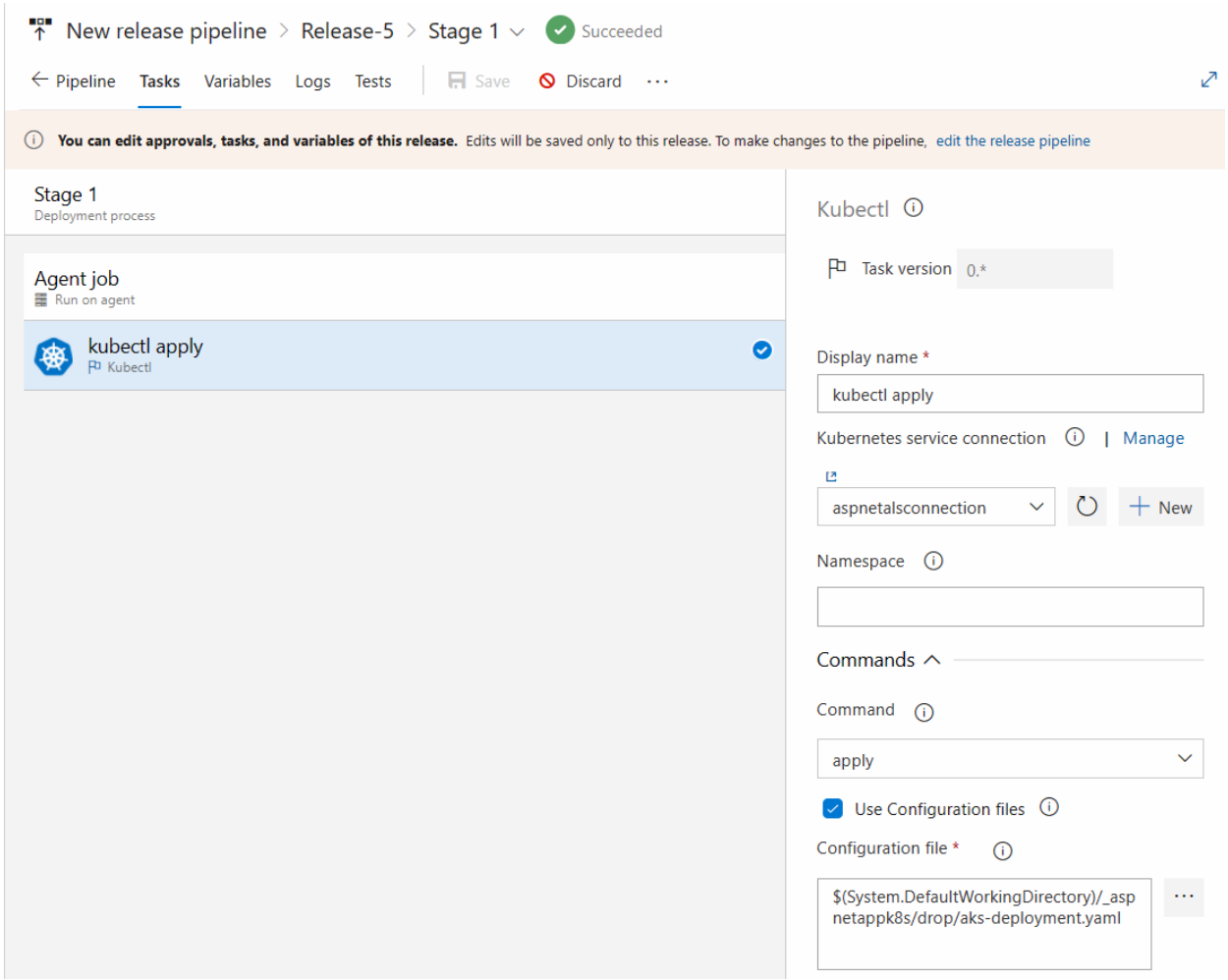


ScholarHat

```yaml
trigger:
- master
resources:
- repo: self
variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: '253899a3-f9ae-4b6f-91ab-7cefbf4cd834'
  imageRepository: 'aspnetappks'
  containerRegistry: 'shtk8s.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/ASPNetAppK8S/Dockerfile'
  tag: '$(Build.BuildId)'

  # Agent VM image name
  vmImageName: 'ubuntu-latest'
stages:
- stage: Build
  displayName: Build and push stage
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: $(vmImageName)
    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        command: buildAndPush
        repository: $(imageRepository)
        dockerfile: $(dockerfilePath)
        containerRegistry: $(dockerRegistryServiceConnection)
        tags: |
          $(tag)
    - task: PublishBuildArtifacts@1
      inputs:
        PathtoPublish: 'aks-deployment.yaml'
        ArtifactName: 'drop'
        publishLocation: 'Container'
```

Release Pipeline: Publish to AKS



ScholarHat

CI/CD Pipeline for Microservices