

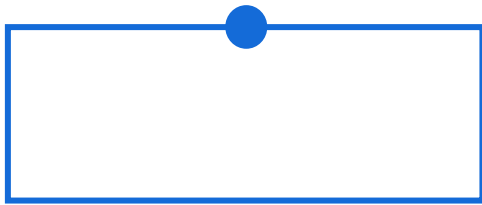
# Agenda

## 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

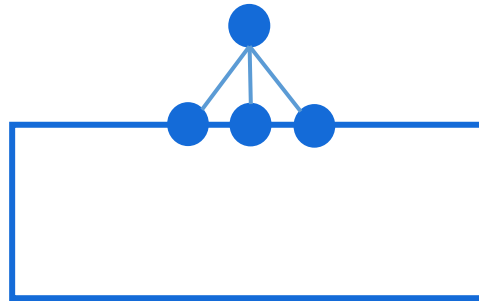
# Types of Services

---



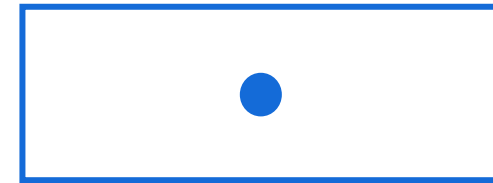
NodePort

- Expose App to external world



LoadBalancer

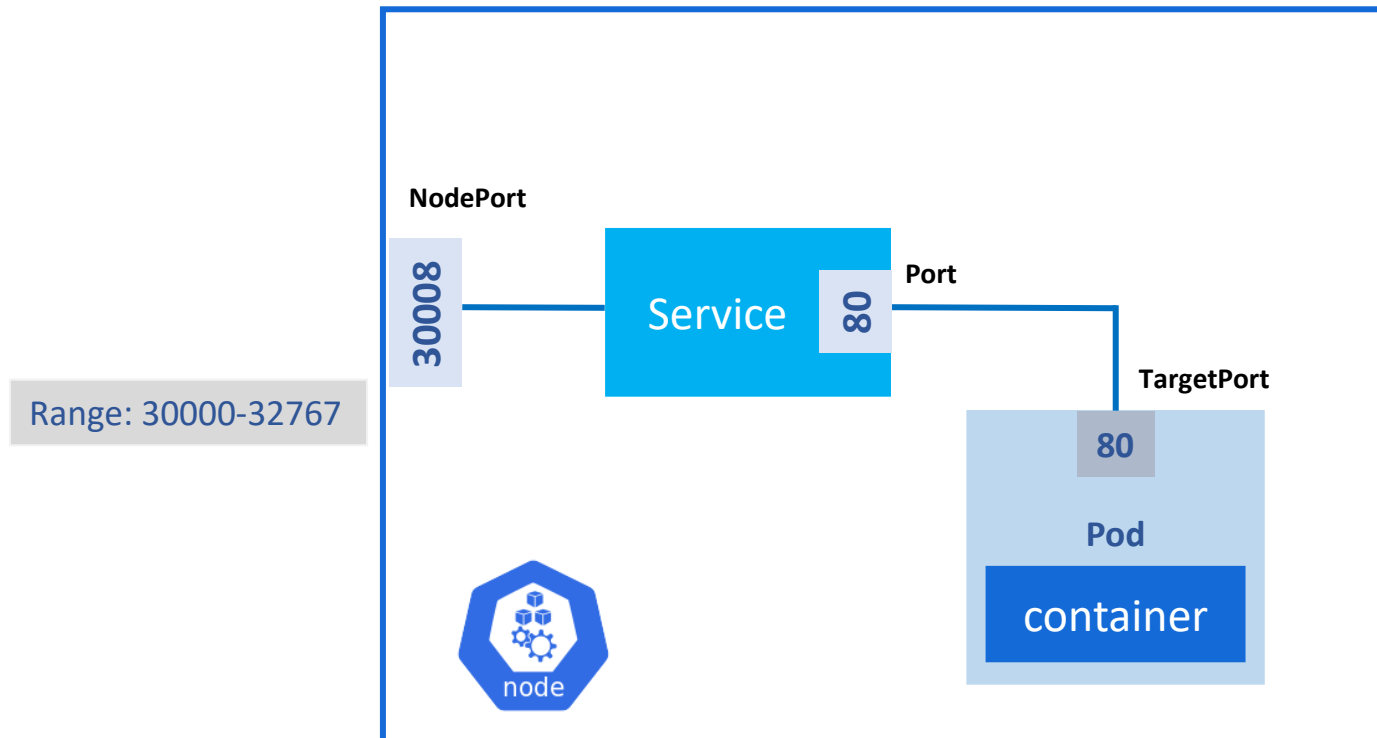
- Equally distribute the load in nodes



ClusterIP

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

# NodePort Service



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

# LoadBalancer Service

---

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

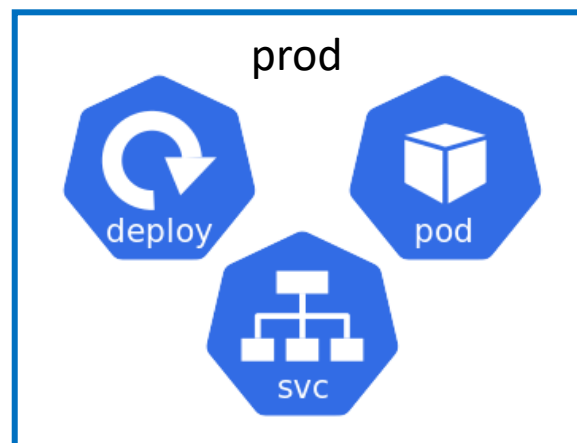
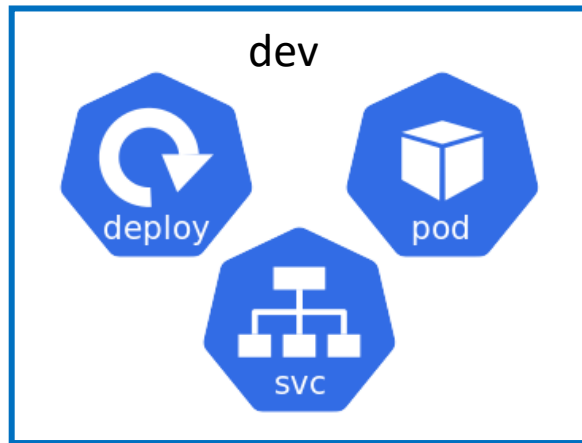
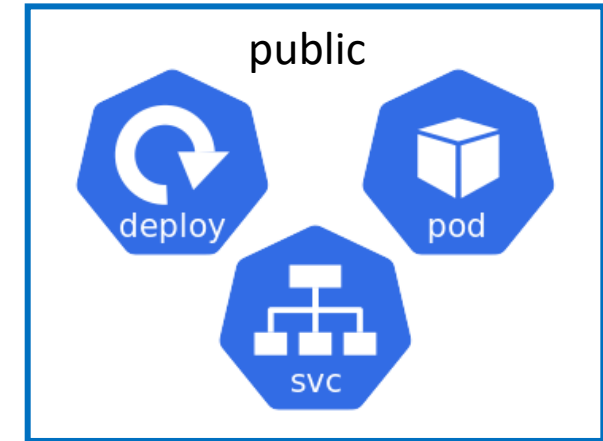
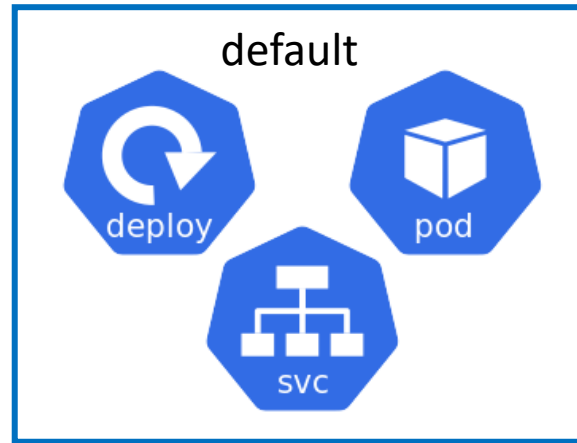
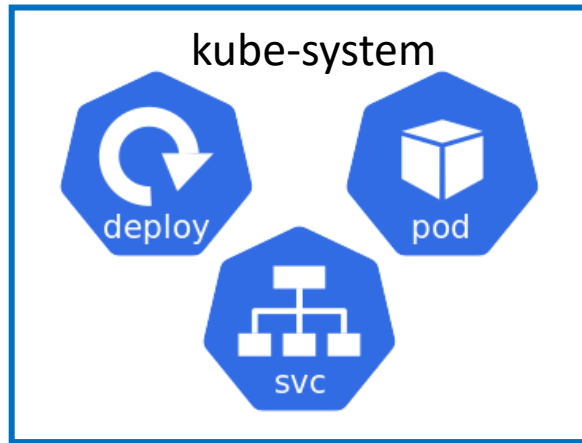
# Deployments

---

- 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

---



# Namespace Commands

---

```
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)

---



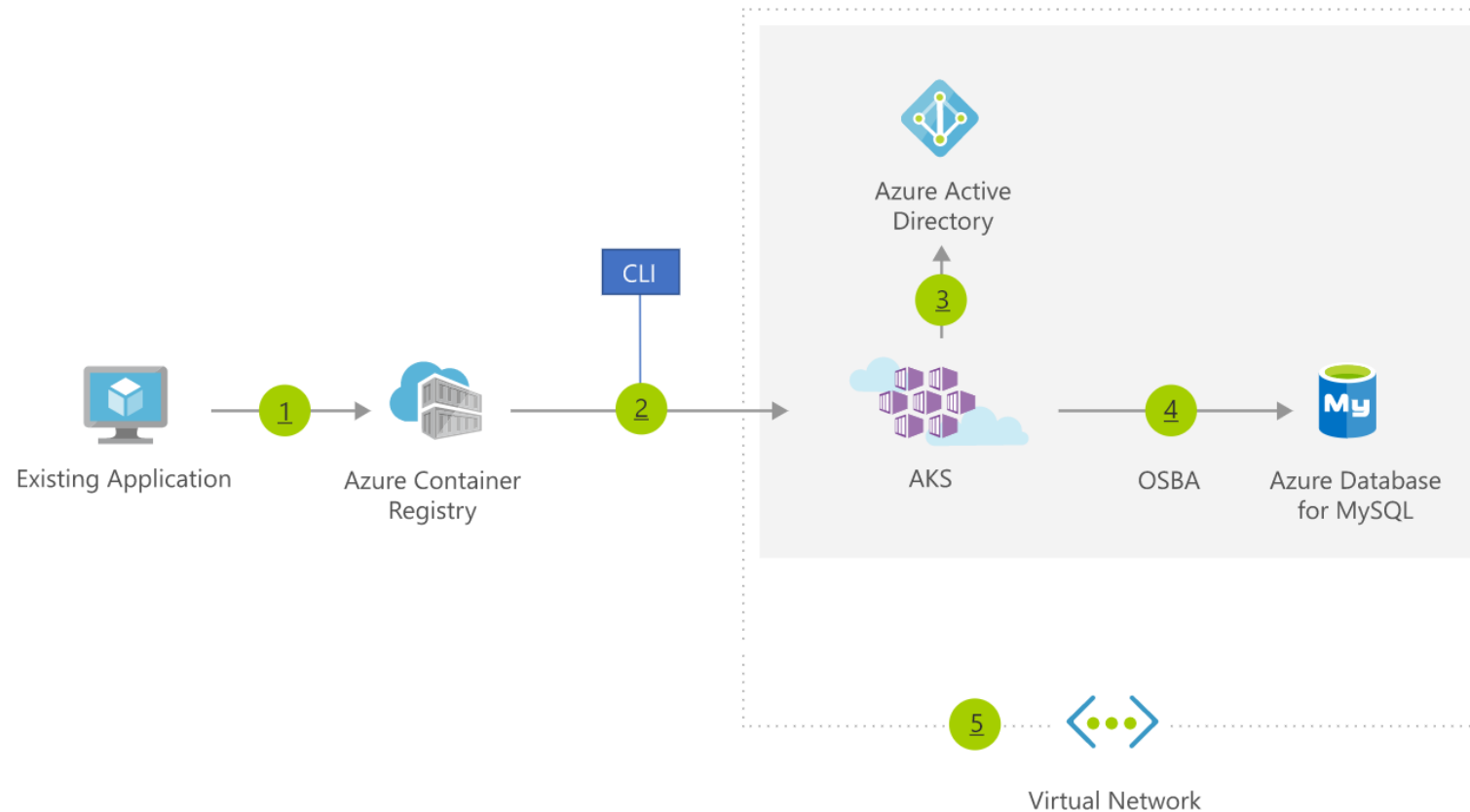


# Azure Kubernetes Service Benefits

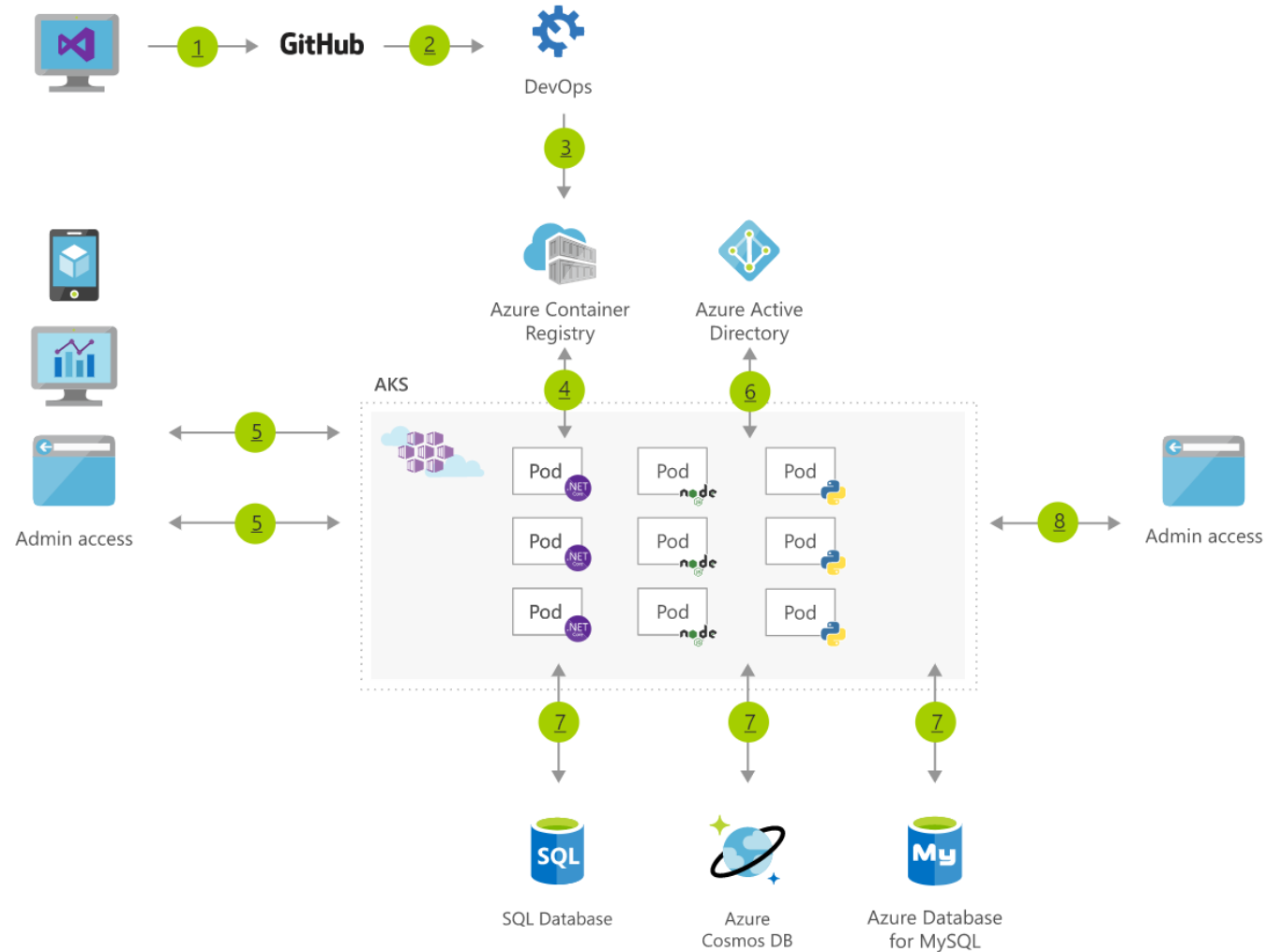
---

- 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



# 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

Dashboard > DNTRg > DNTAKS

## DNTAKS | Services and ingresses (preview)

Kubernetes service

Search (Ctrl+ /) << + Add Delete Refresh Show labels

Services Ingresses

Filter by service name: Enter the full service name Filter by namespace: All namespaces

<input type="checkbox"/>	Name	Namespace	Status
<input type="checkbox"/>	kubernetes	default	Ok
<input type="checkbox"/>	healthmodel-replicaset-...	kube-system	Ok
<input type="checkbox"/>	kube-dns	kube-system	Ok
<input type="checkbox"/>	metrics-server	kube-system	Ok
<input type="checkbox"/>	dashboard-metrics-scr...	kube-system	Ok

**Kubernetes resources**

- Namespaces (preview)
- Workloads (preview)
- Services and ingresses (previe...)
- Storage (preview)

# AKS Deployment Using CLI

```
> az acr login --name <registry-name>
```

```
> az account set -s <subscription_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
```

<input type="checkbox"/>	Name	Namespace	Status	Type	Cluster IP	External IP	Ports	keys
<input type="checkbox"/>	kubernetes	default	✔ Ok	ClusterIP	10.0.0.1		443/TCP	ion
<input type="checkbox"/>	healthmodel-replicaset-...	kube-system	✔ Ok	ClusterIP	10.0.3.61		25227/TCP	
<input type="checkbox"/>	kube-dns	kube-system	✔ Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP	king
<input type="checkbox"/>	metrics-server	kube-system	✔ Ok	ClusterIP	10.0.81.42		443/TCP	r
<input type="checkbox"/>	dashboard-metrics-scr...	kube-system	✔ Ok	ClusterIP	10.0.30.33		8000/TCP	
<input type="checkbox"/>	kubernetes-dashboard	kube-system	✔ Ok	ClusterIP	10.0.141.160		443/TCP	
<input type="checkbox"/>	catalog-service	default	✔ Ok	LoadBalancer	10.0.98.49	40.89.244.8 <a href="#">🌐</a>	80:30340/TCP	
<input type="checkbox"/>	authentication-service	default	✔ Ok	LoadBalancer	10.0.174.28	52.143.247.97 <a href="#">🌐</a>	80:30082/TCP	

Microsoft Azure

Search resources, services, and docs (G+/)

Dashboard > mydntaks

mydntaks | Access keys

Container registry

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Quick start

Settings

Registry name

mydntaks

Login server

mydntaks.azurecr.io

Admin user ⓘ

Enable Disable

Username

mydntaks

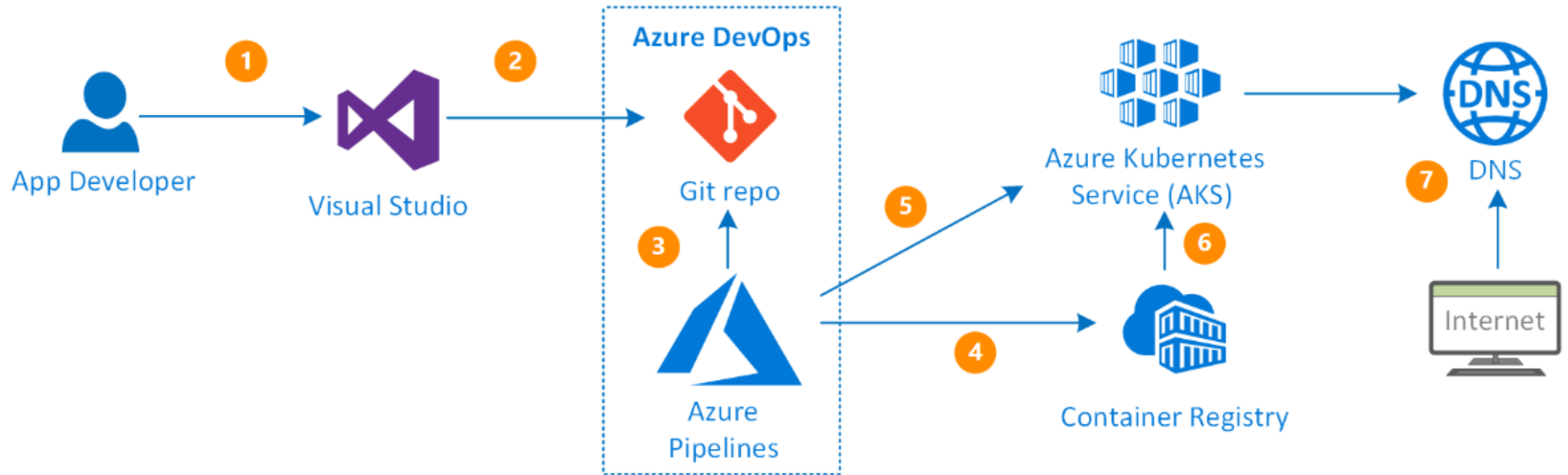
Name	Password
password	THcUU/yUAyKNgUjQQqK2tXLNFmJF
password2	g1FM5/n4MsfMNqaw+WTKpk/ygZRn

# 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



# Building Image and Publish to ACR

The screenshot displays the Azure DevOps web interface for a pipeline named "School-Azure Kubernetes Serv...". The left sidebar shows the project structure for "aspnetappk8s", including folders like ".vscode", "Controllers", "Models", "Properties", "Views", and "wwwroot", along with files such as ".dockerignore", "appsettings.Development.j...", "appsettings.json", "ASPNetAppK8S.csproj", "Dockerfile", "C# Program.cs", "aks-deployment.yaml", "ASPNetAppK8S.sln", and "azure-pipelines.yml".

The main area shows the pipeline configuration for "Agent job 1". The tasks are:

- Get sources (CivicaSmartSchool, master)
- Build a container image (Docker) - This task is selected and expanded.
- Push a container image (Docker)
- Publish Artifacts: drop (Publish build artifacts)

The configuration for the "Build a container image" task is shown on the right:

- Repository: dntacr102
- Action: Build an image
- Dockerfile: \*\*/Dockerfile
- Build Arguments: (empty)
- Use Default Build Context: ☒
- Image Name: \$(Build.Repository.Name):\$(Build.BuildId)
- Qualify Image Name: ☒
- Additional Image Tags: (empty)
- Include Source Tags: ☐
- Include Latest Tag: ☐



```

trigger:
- master
resources:
- repo: self
variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: '253899a3-f9ae-4b6f-91ab-7cefbf4cd834'
  imageRepository: 'aspnetappks'
  containerRegistry: 'shstk8s.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

New release pipeline > Release-5 > Stage 1 ✓ Succeeded

← Pipeline **Tasks** Variables Logs Tests | Save Discard ...

**You can edit approvals, tasks, and variables of this release.** Edits will be saved only to this release. To make changes to the pipeline, [edit the release pipeline](#)

### Stage 1

Deployment process

#### Agent job

Run on agent

**kubectl apply** Kubectl ✓

#### Kubectl

Task version 0.\*

Display name \*  
kubectl apply

Kubernetes service connection ⓘ | [Manage](#)

aspnetalsconnection ↻ + New

Namespace ⓘ

Commands ^

Command ⓘ  
apply ▼

☒ Use Configuration files ⓘ

Configuration file \* ⓘ  
\$(System.DefaultWorkingDirectory)/\_aspnetappk8s/drop/aks-deployment.yaml ...

# CI/CD Pipeline for Microservices

