

Azure AKS Kubernetes - Masterclass | Azure DevOps, Terraform

Kalyan Reddy Daida

STACKSIMPLIFY

**Kubernetes
On
Cloud
Roadmap**

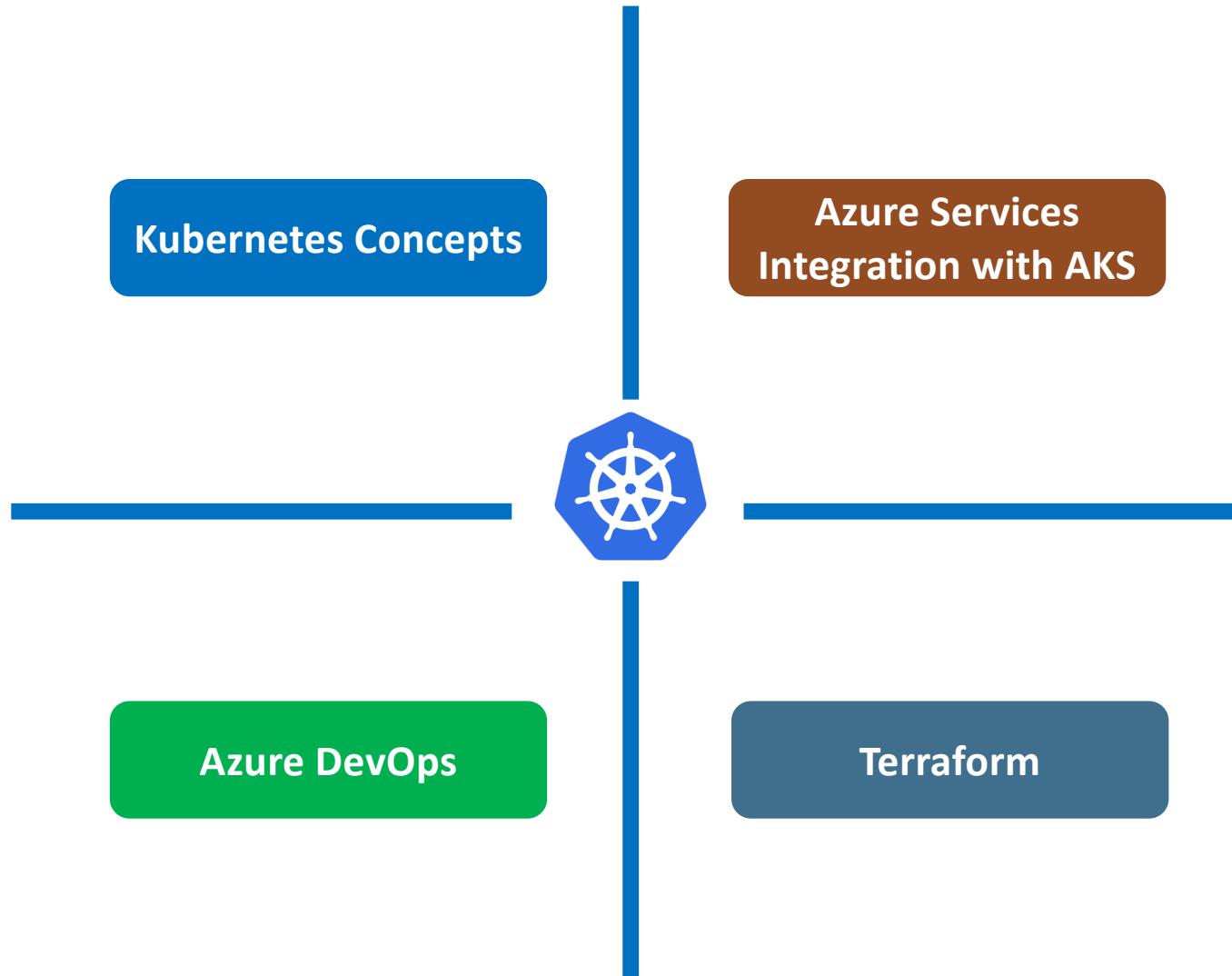
Kubernetes for Beginners On Cloud

**AWS EKS Kubernetes - Masterclass |
DevOps, Microservices**

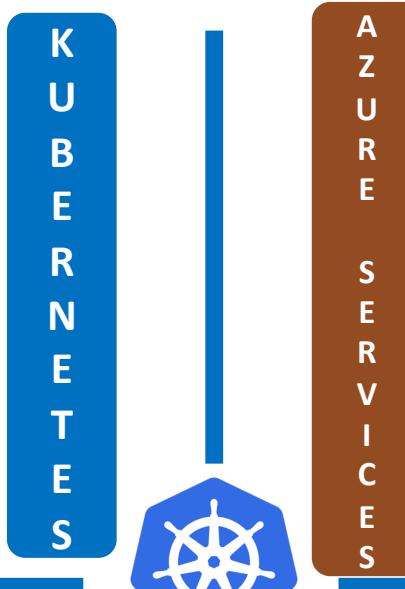
**Azure AKS Kubernetes - Masterclass |
DevOps, Terraform**

**Google GKE Kubernetes - Masterclass |
DevOps, Microservices**

**Docker & Kubernetes for Java Spring Boot
Developers on AWS**



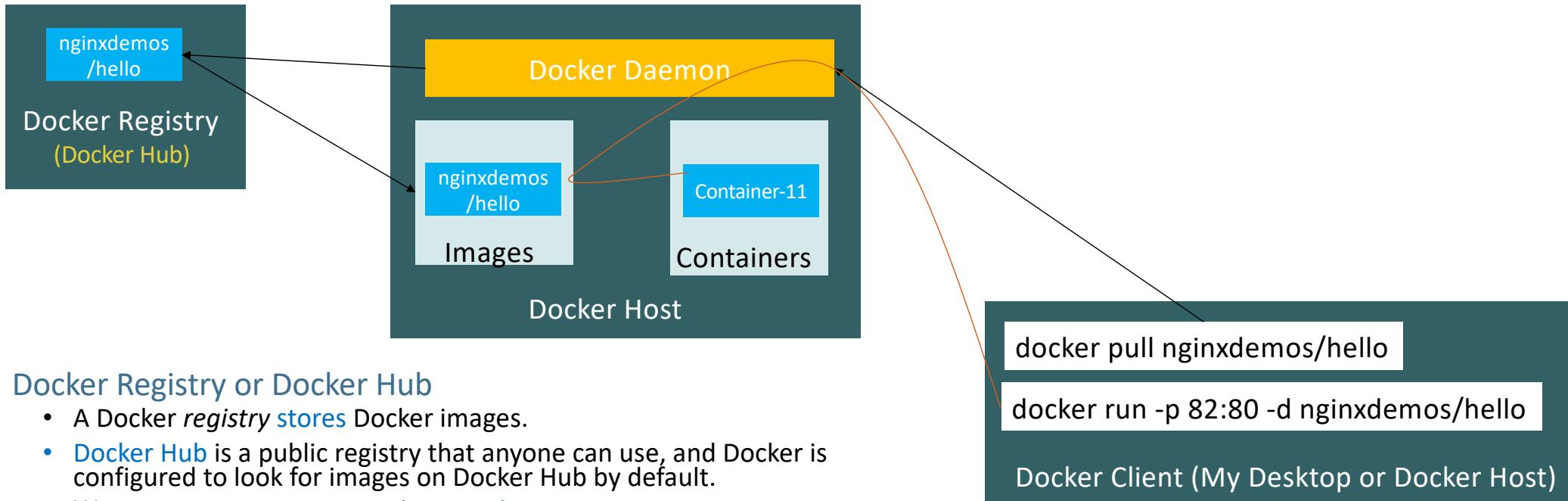
Architecture	kubectl - Imperative	Persistent Volumes
Pods	Declarative with YAML	PVC
ReplicaSets	Secrets	Load Balancers
Deployments	Init Containers	Annotations
LoadBalancer Service		
Cluster IP Service	Requests & Limits	HPA
External Name Service	Namespaces	Cluster Autoscaler
Ingress Service	Limit Range	
Ingress SSL	Resource Quota	
Ingress & External DNS	Storage Classes	ConfigMaps



Azure DevOps

Terraform

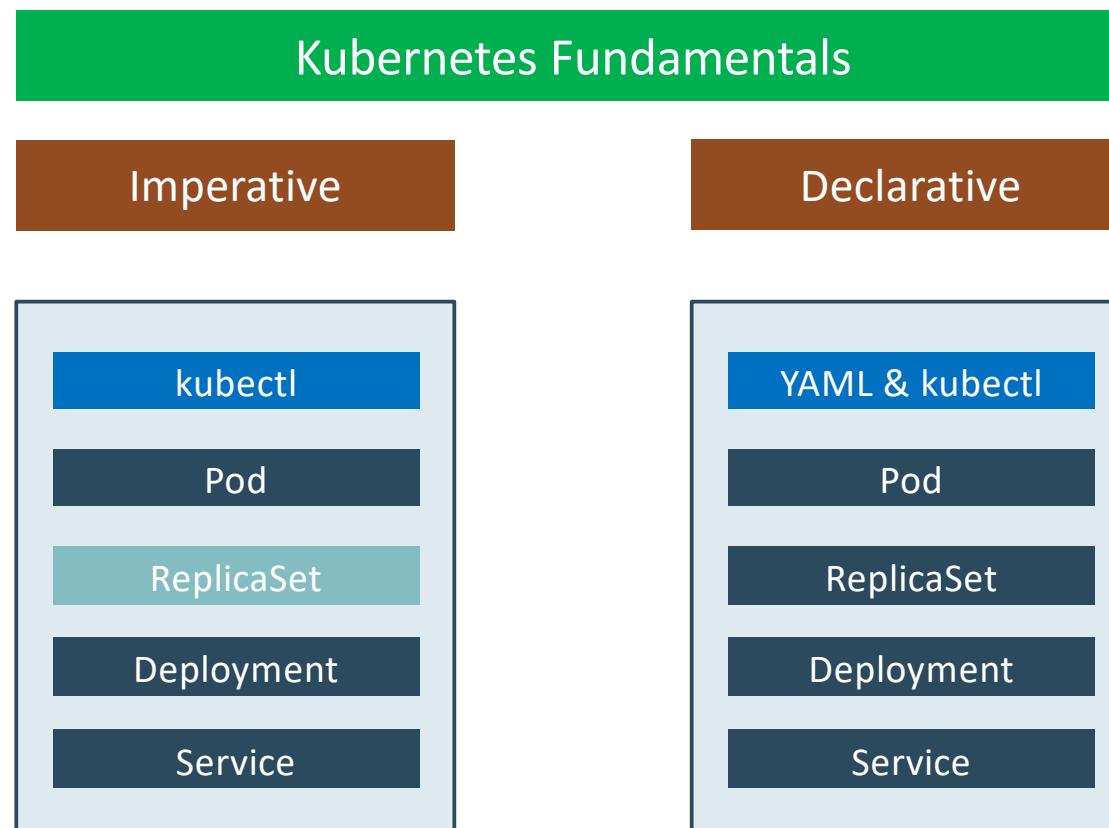
Docker - Fundamentals

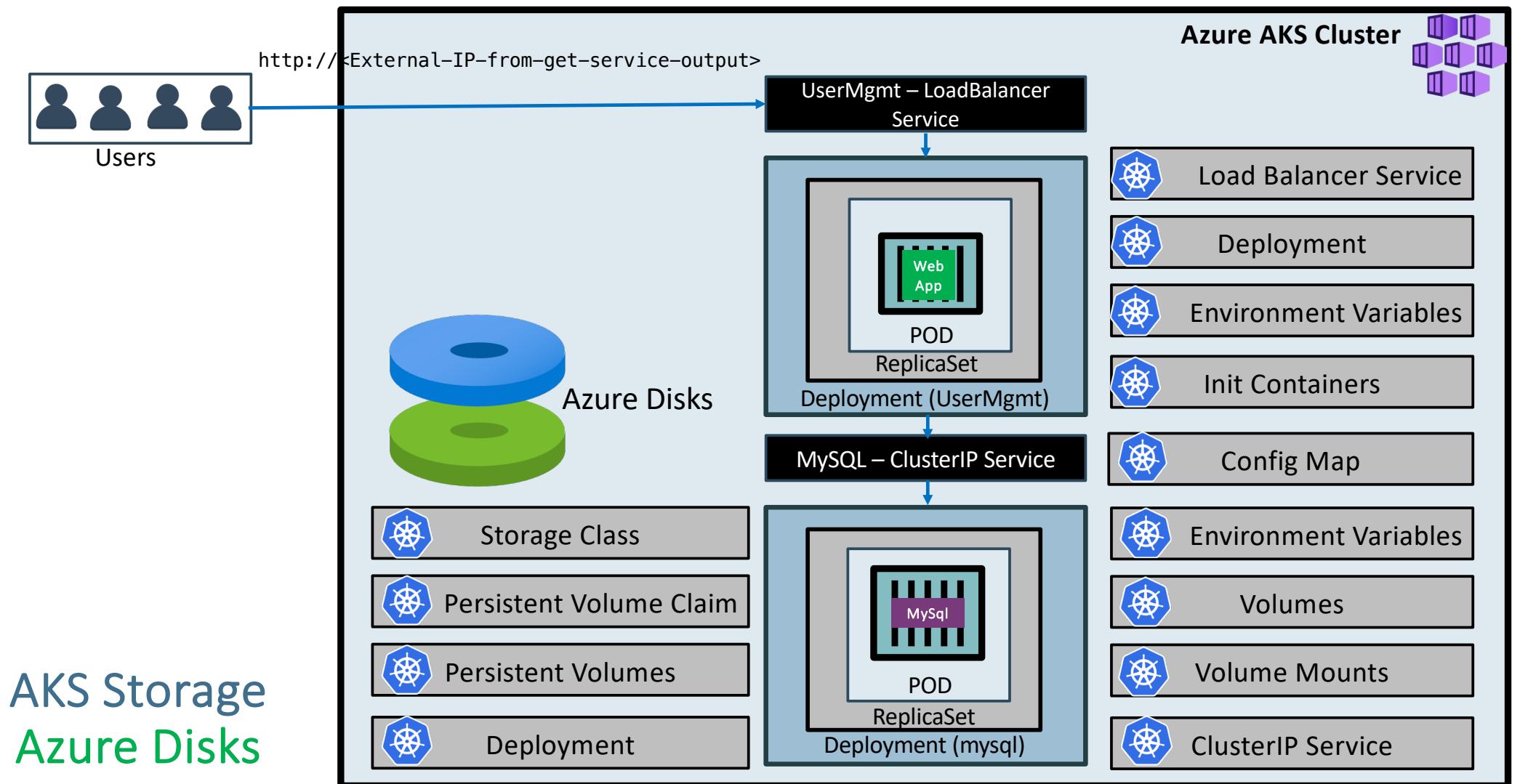


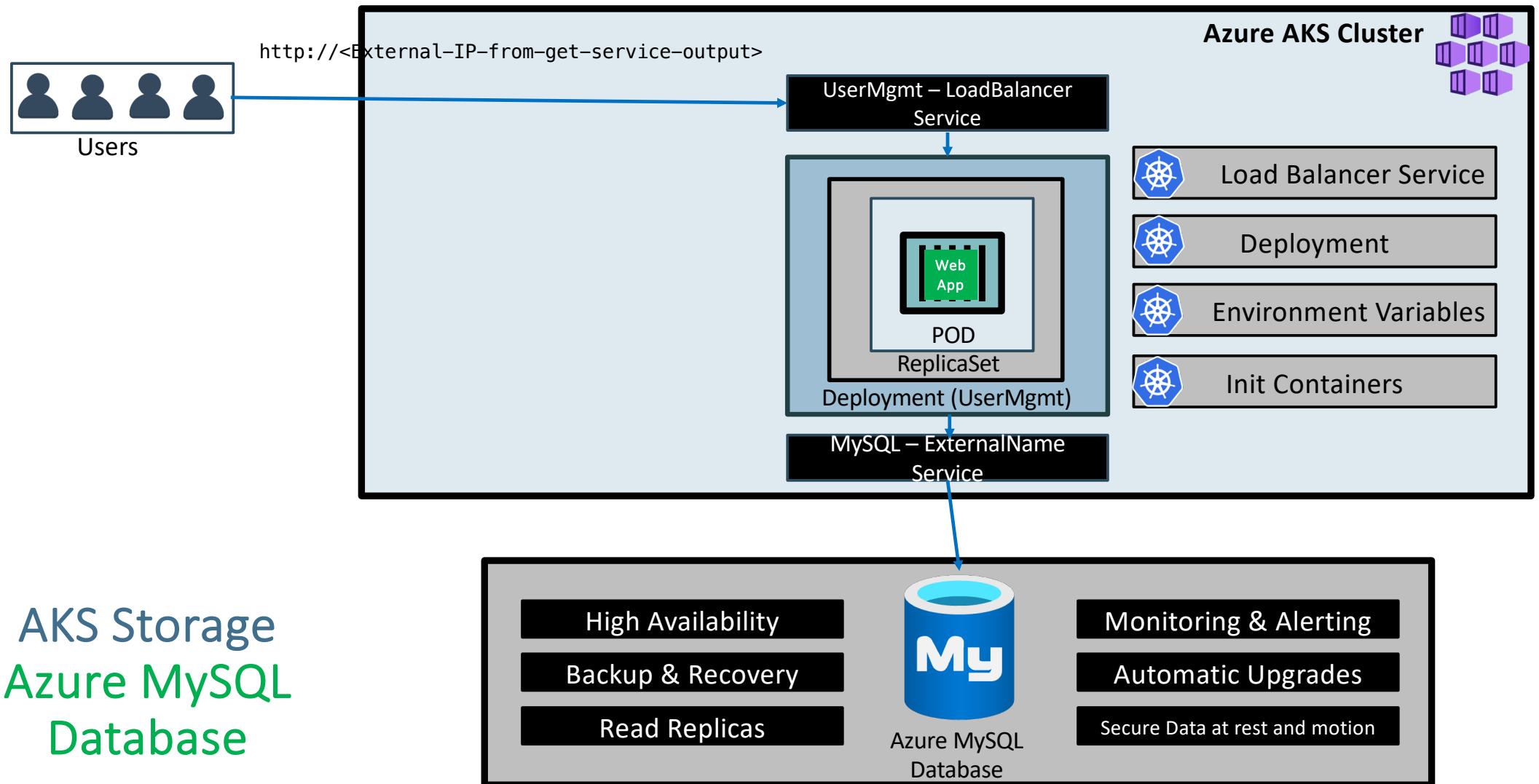
- **Docker Registry or Docker Hub**

- A Docker *registry* stores Docker images.
- **Docker Hub** is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
- We can even run our own **private registry**.
- When we use the **docker pull** or **docker run** commands, the required images are pulled from our configured registry.
- When we use the **docker push** command, our image is pushed to our configured registry.

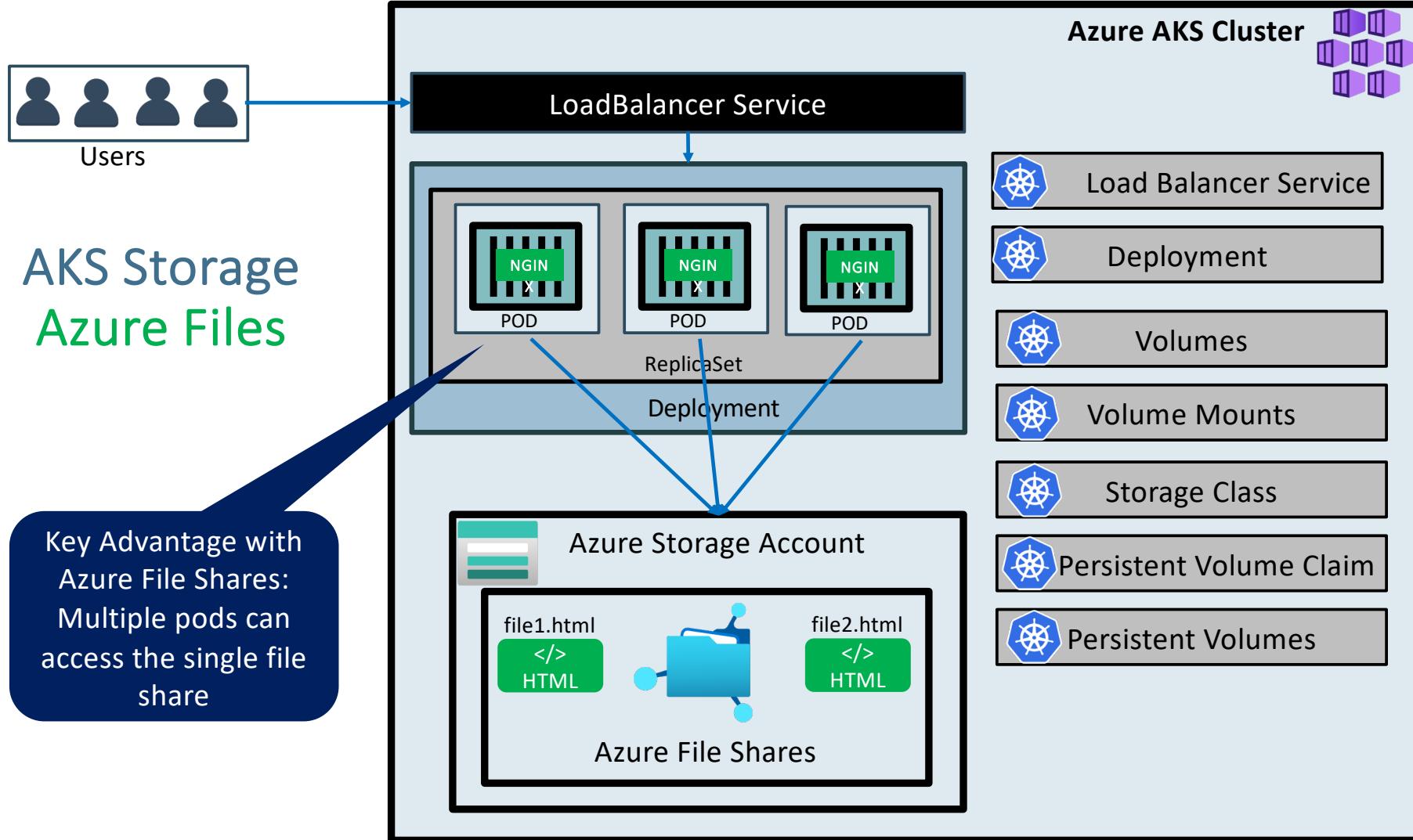
Kubernetes - Imperative & Declarative

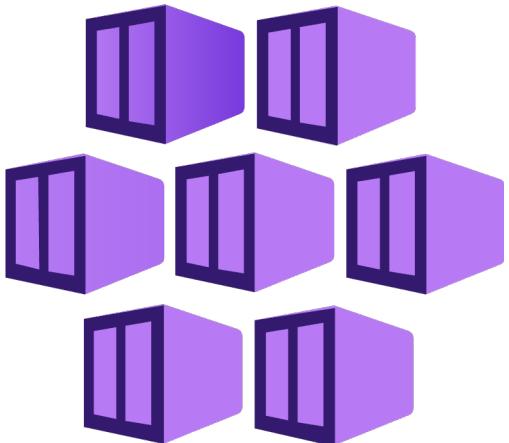






<http://<External-IP-from-get-service-output>/app1/file1.html>





Azure AKS

Create Cluster

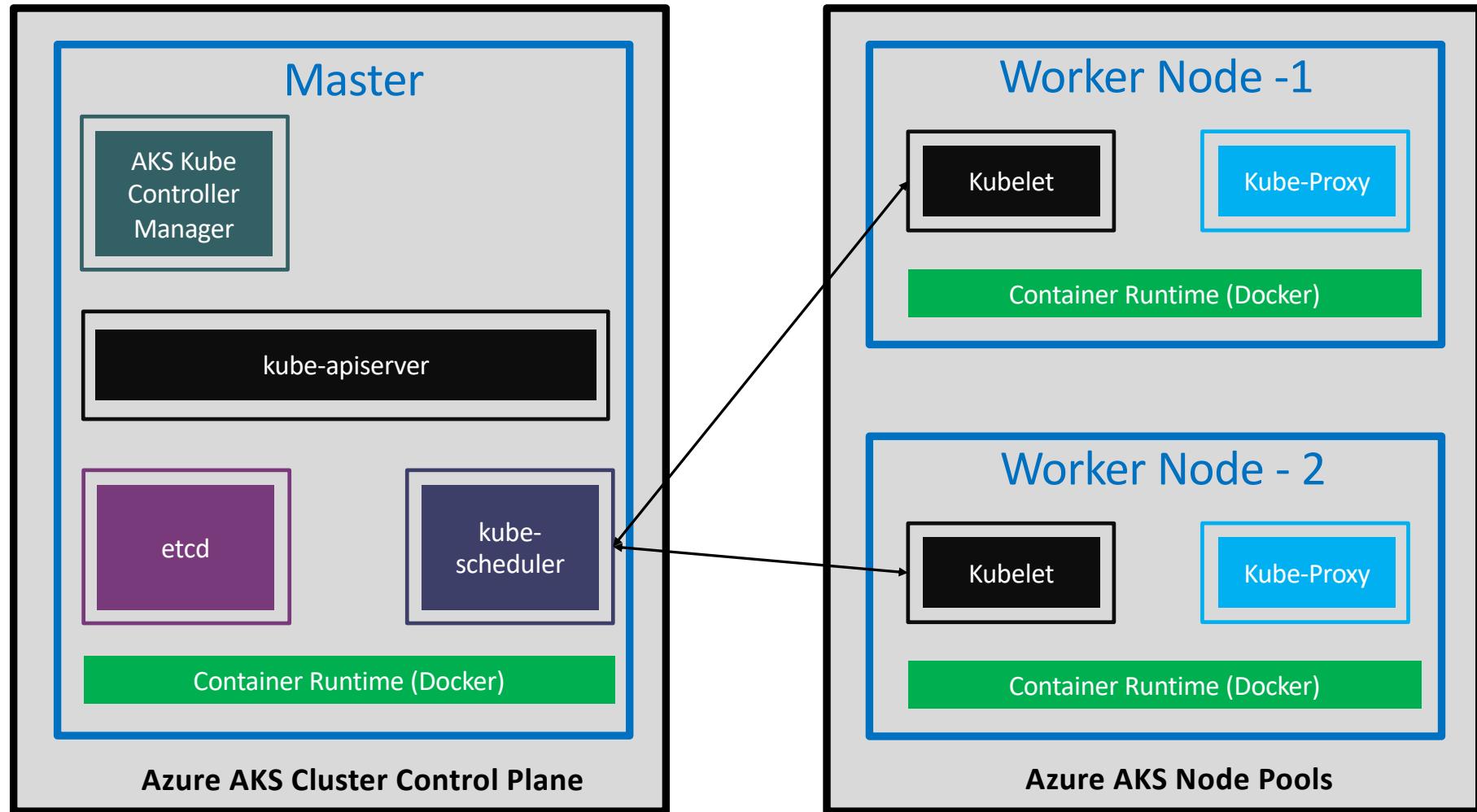
Introduction



AKS - Introduction

- AKS – Azure Kubernetes Service
- AKS is **highly available, secure and fully managed** Kubernetes Service
- As on today available in **36 regions** and growing.
- When **compared** to other cloud providers, AKS is the one which is available in highest number of regions
- Will be able to run **any type** of workloads
 - **Windows** based applications like .Net Apps
 - Linux supported applications like Java
 - **IOT device deployment** and management on demand
 - **Machine Learning** Model training with AKS

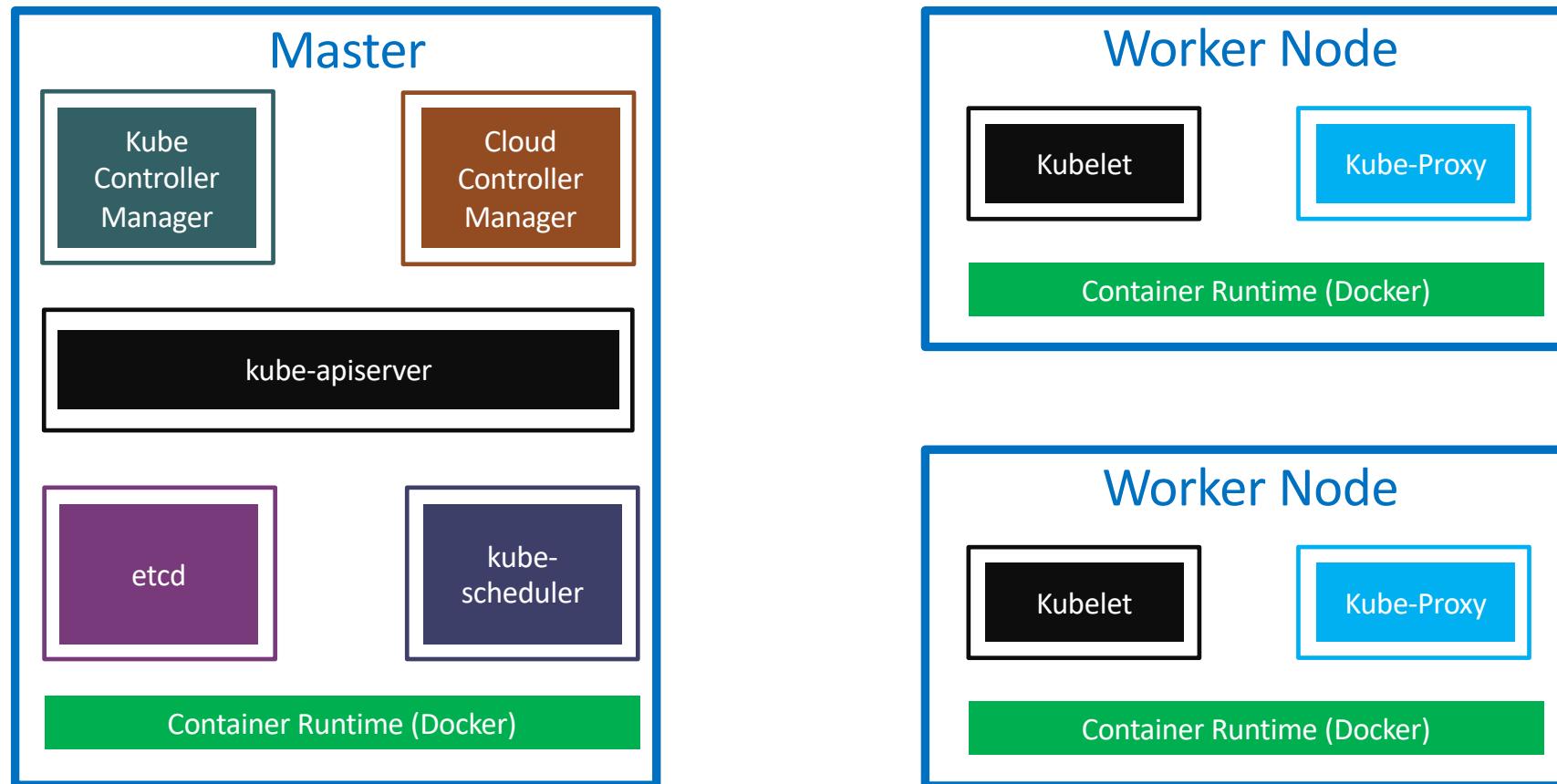
AKS Kubernetes - Architecture



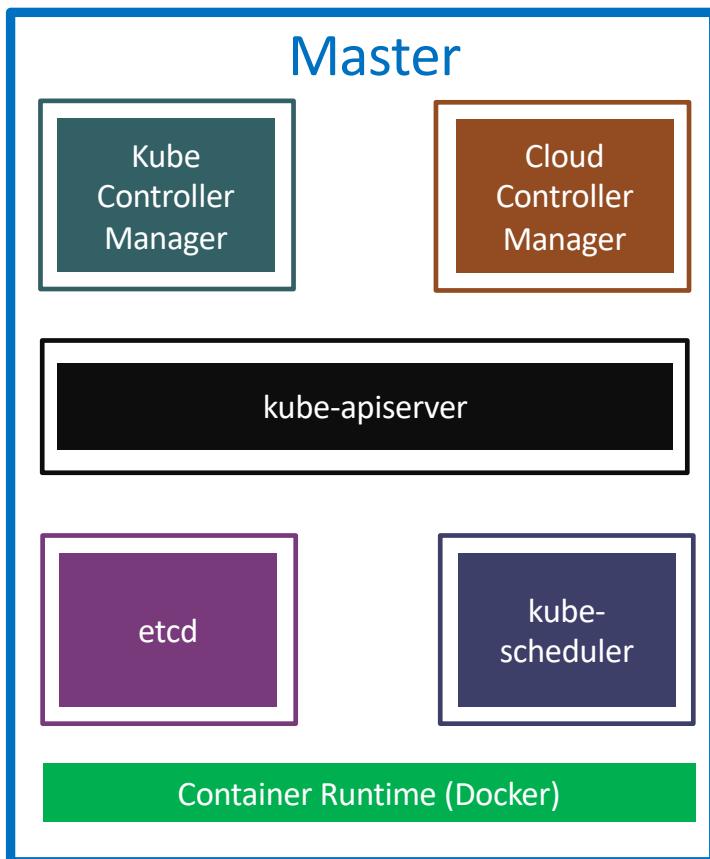
Kubernetes Architecture



Kubernetes - Architecture

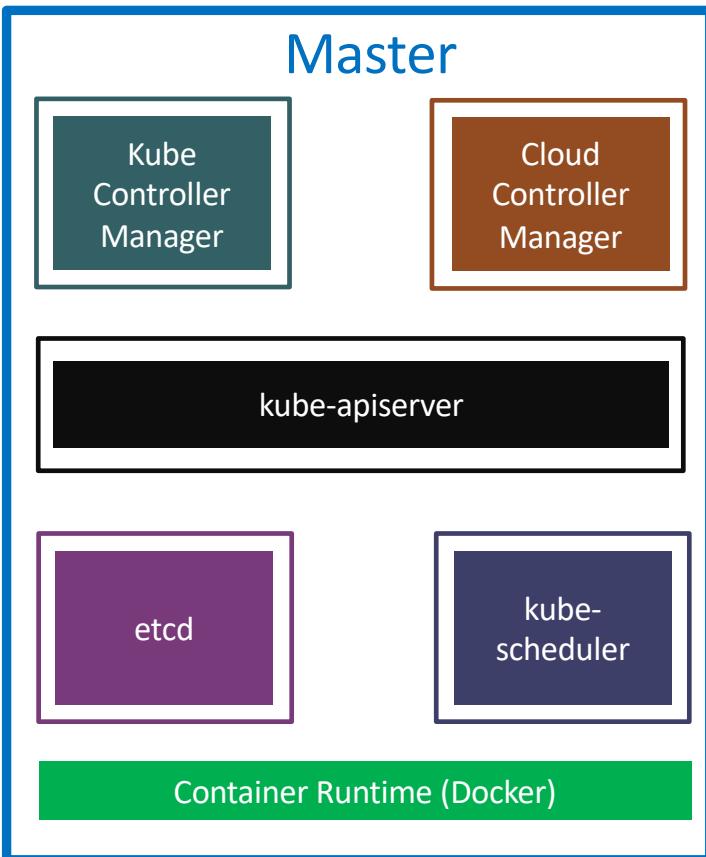


Kubernetes Architecture - Master



- **kube-apiserver**
 - It acts as **front end** for the Kubernetes control plane. It **exposes** the Kubernetes API
 - Command line tools (like kubectl), Users and even Master components (scheduler, controller manager, etcd) and Worker node components like (Kubelet) **everything talk** with API Server.
- **etcd**
 - Consistent and highly-available **key value store** used as Kubernetes' **backing store** for all cluster data.
 - It **stores** all the masters and worker node information.
- **kube-scheduler**
 - Scheduler is responsible for distributing containers across multiple nodes.
 - It **watches** for newly created Pods with no assigned node, and selects a node for them to run on.

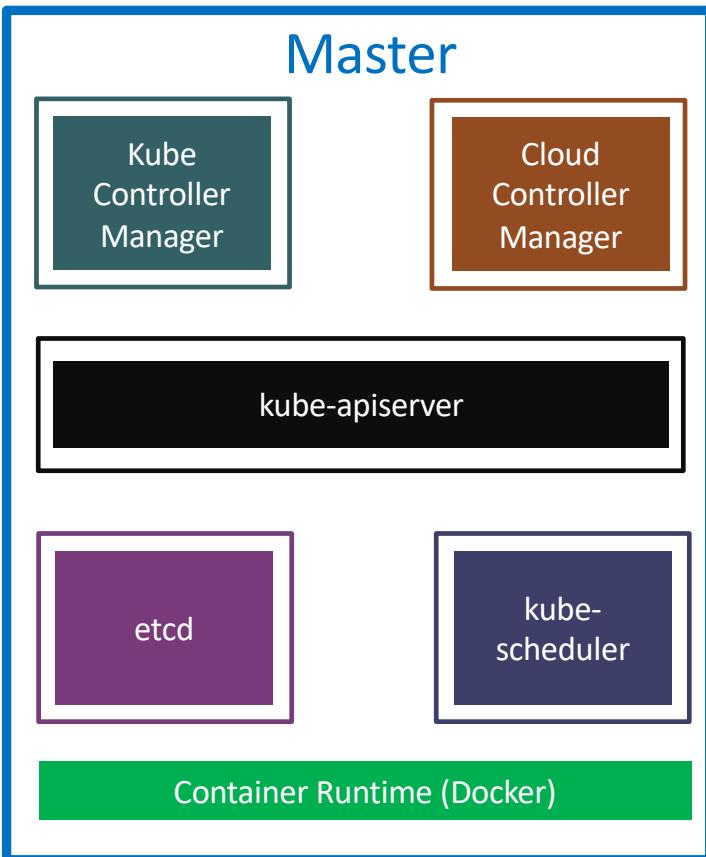
Kubernetes Architecture - Master



- **kube-controller-manager**

- Controllers are responsible for noticing and responding when nodes, containers or endpoints go down. They make decisions to bring up new containers in such cases.
- **Node Controller**: Responsible for noticing and responding when **nodes go down**.
- Replication Controller: Responsible for maintaining the **correct number of pods** for every replication controller object in the system.
- **Endpoints Controller**: **Populates** the Endpoints object (that is, joins Services & Pods)
- **Service Account & Token Controller**: Creates default accounts and API Access for **new namespaces**.

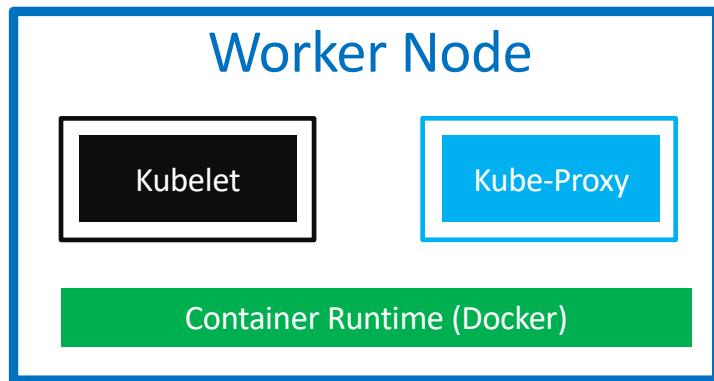
Kubernetes Architecture - Master



- **cloud-controller-manager**

- A Kubernetes control plane component that embeds **cloud-specific control logic**.
- It only runs controllers that are **specific** to your cloud provider.
- **On-Premise** Kubernetes clusters will not have this component.
- **Node controller**: For **checking** the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- **Route controller**: For setting up **routes** in the underlying cloud infrastructure
- **Service controller**: For creating, updating and deleting cloud provider **load balancer**

Kubernetes Architecture – Worker Nodes



- Container Runtime

- Container Runtime is the **underlying software** where we run all these Kubernetes components.
- We are using Docker, but we have other runtime options like rkt, container-d etc.

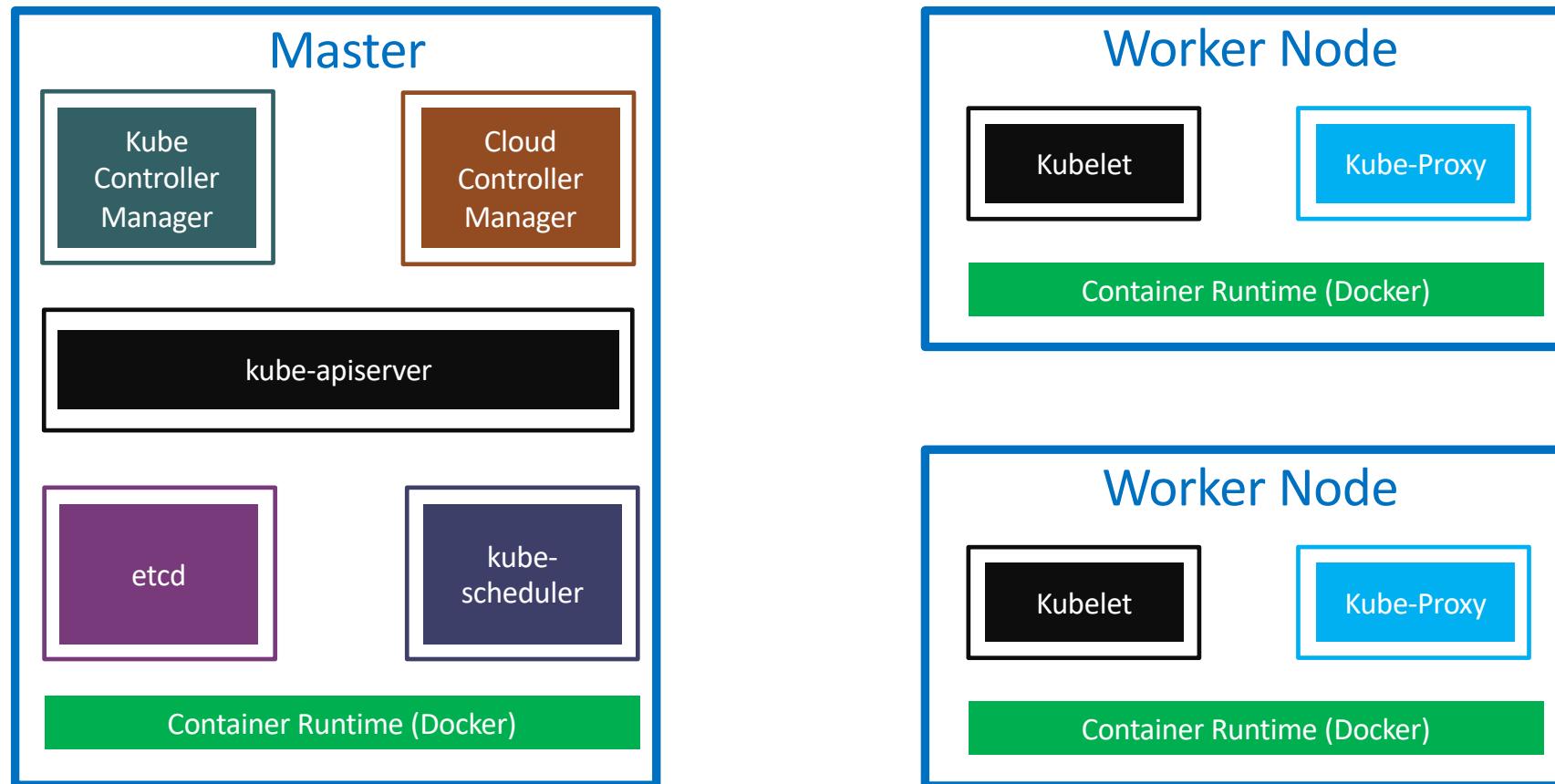
- Kubelet

- Kubelet is the **agent** that runs on every node in the cluster
- This agent is **responsible** for making sure that containers are running in a Pod on a node.

- Kube-Proxy

- It is a **network proxy** that runs on each node in your cluster.
- It maintains **network rules** on nodes
- In short, these network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

Kubernetes - Architecture

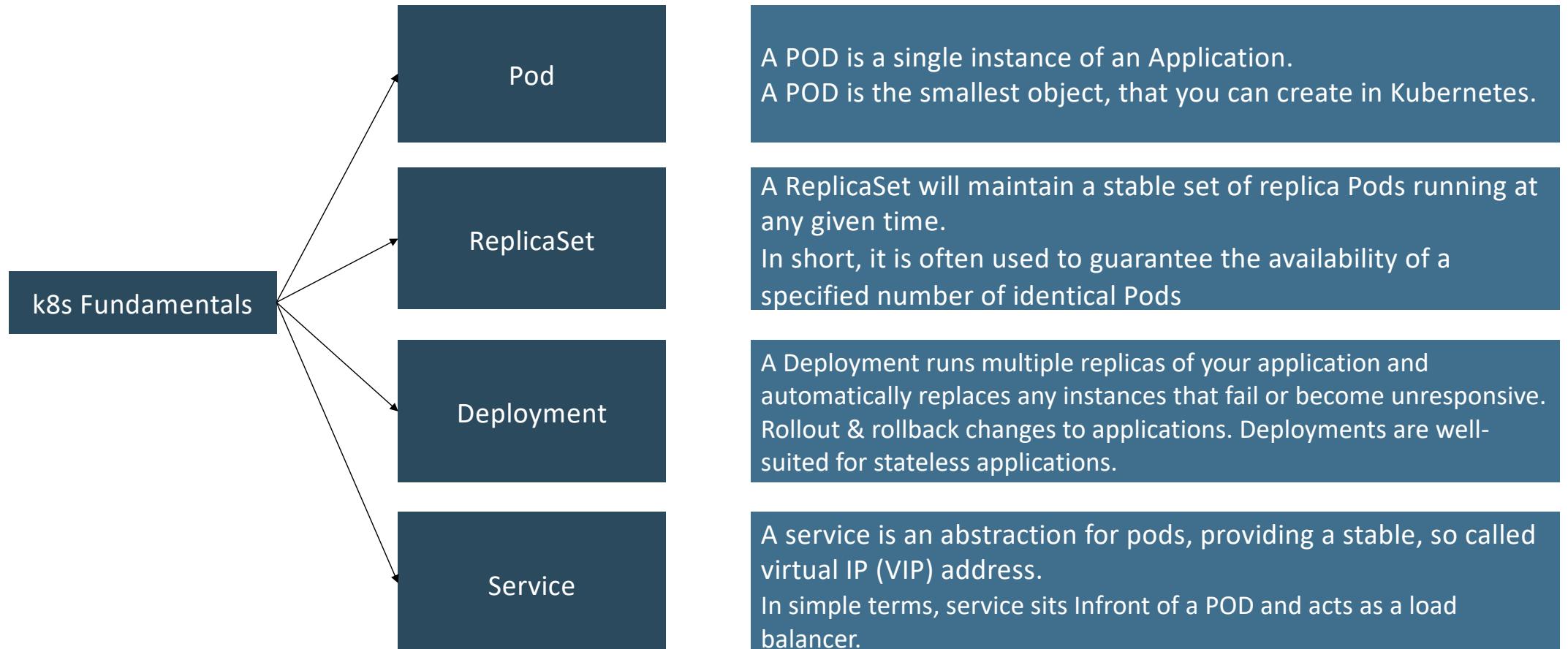


Kubernetes Fundamentals

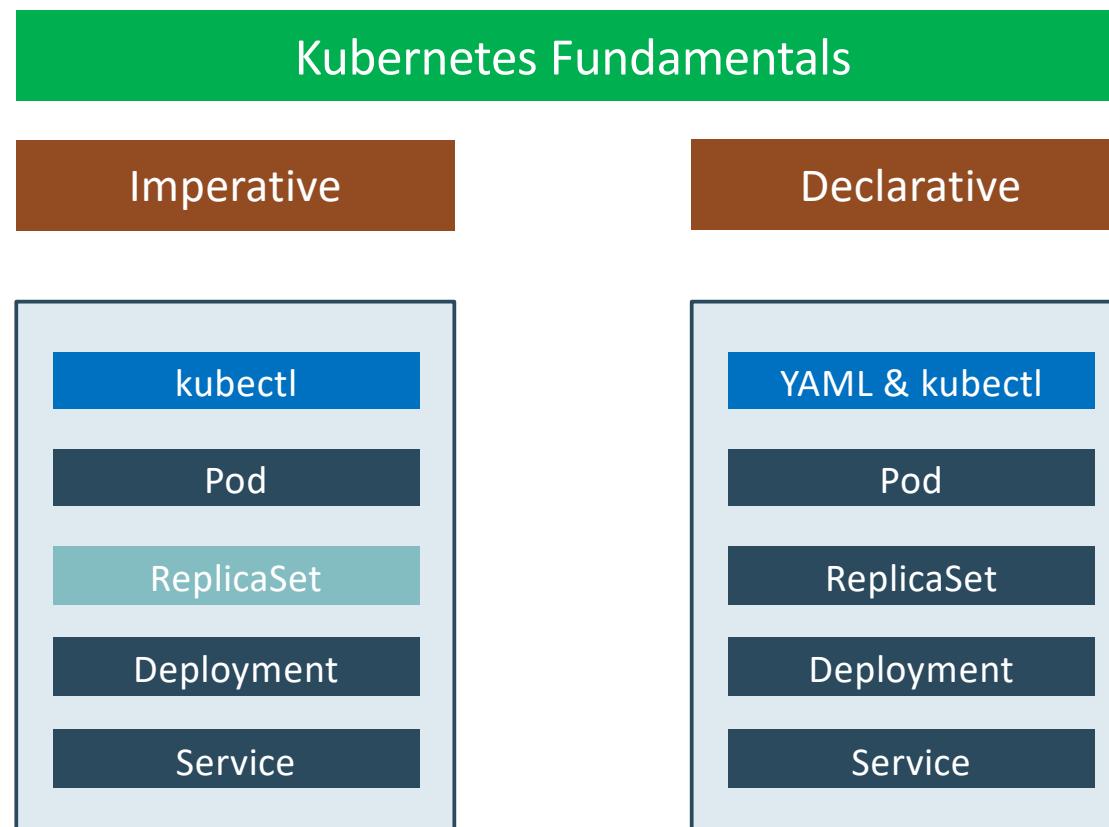
Pod, ReplicaSet, Deployment & Service



Kubernetes - Fundamentals



Kubernetes - Imperative & Declarative



Kubernetes POD

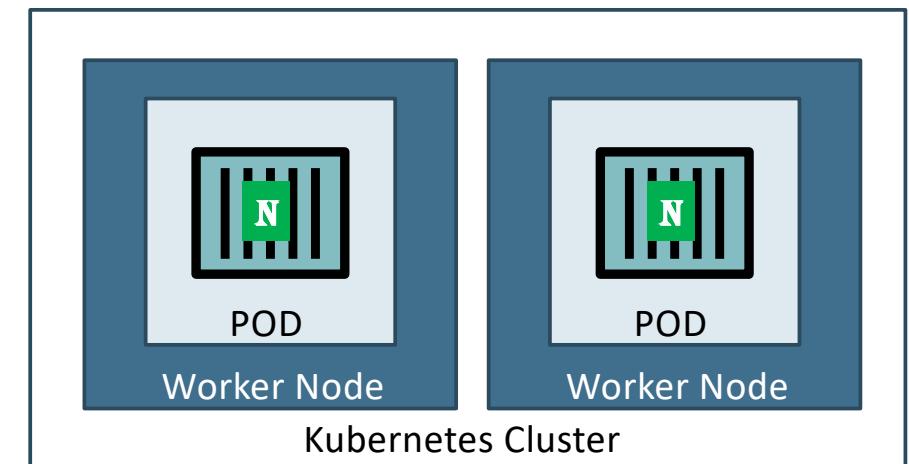


Kubernetes - POD

- With Kubernetes our core goal will be to **deploy our applications** in the form of **containers** on **worker nodes** in a k8s cluster.
- Kubernetes **does not** deploy containers directly on the worker nodes.
- Container is **encapsulated** in to a Kubernetes Object named **POD**.
- A POD is a **single instance** of an application.
- A POD is the **smallest object** that we can create in Kubernetes.

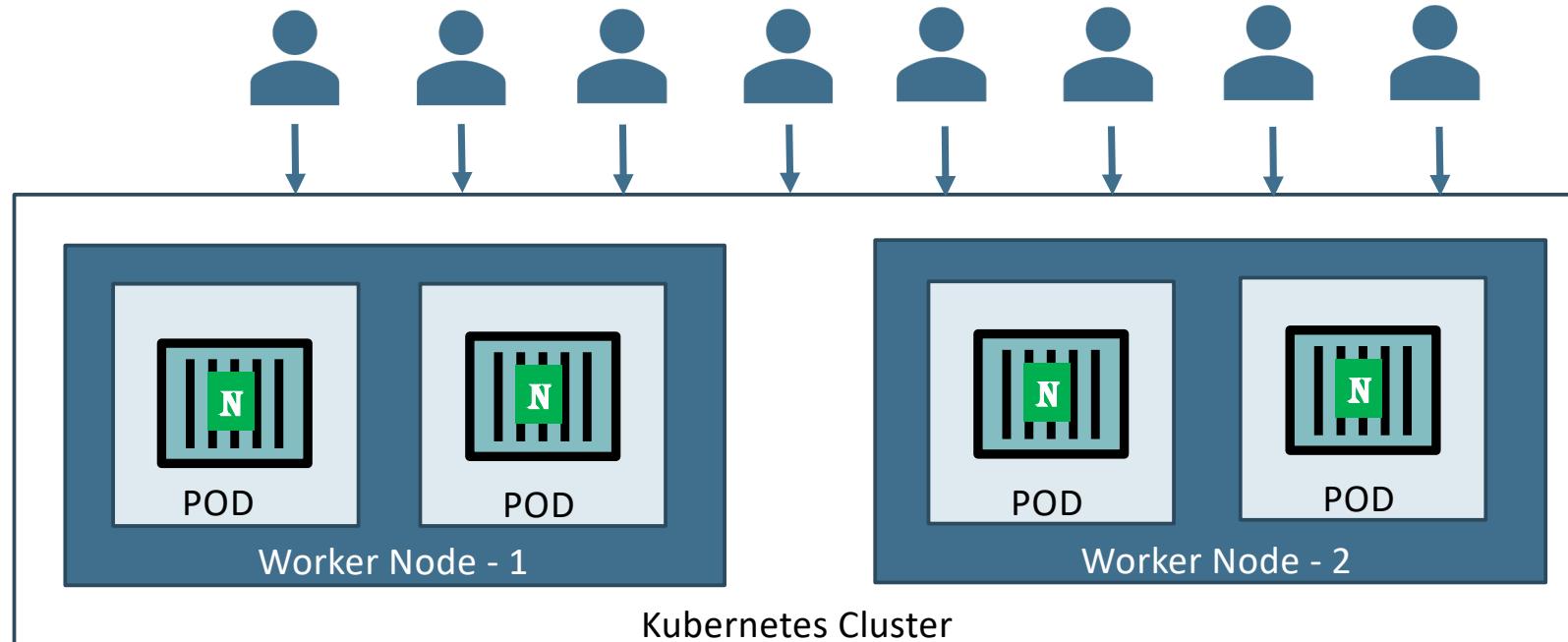


Nginx Container Image



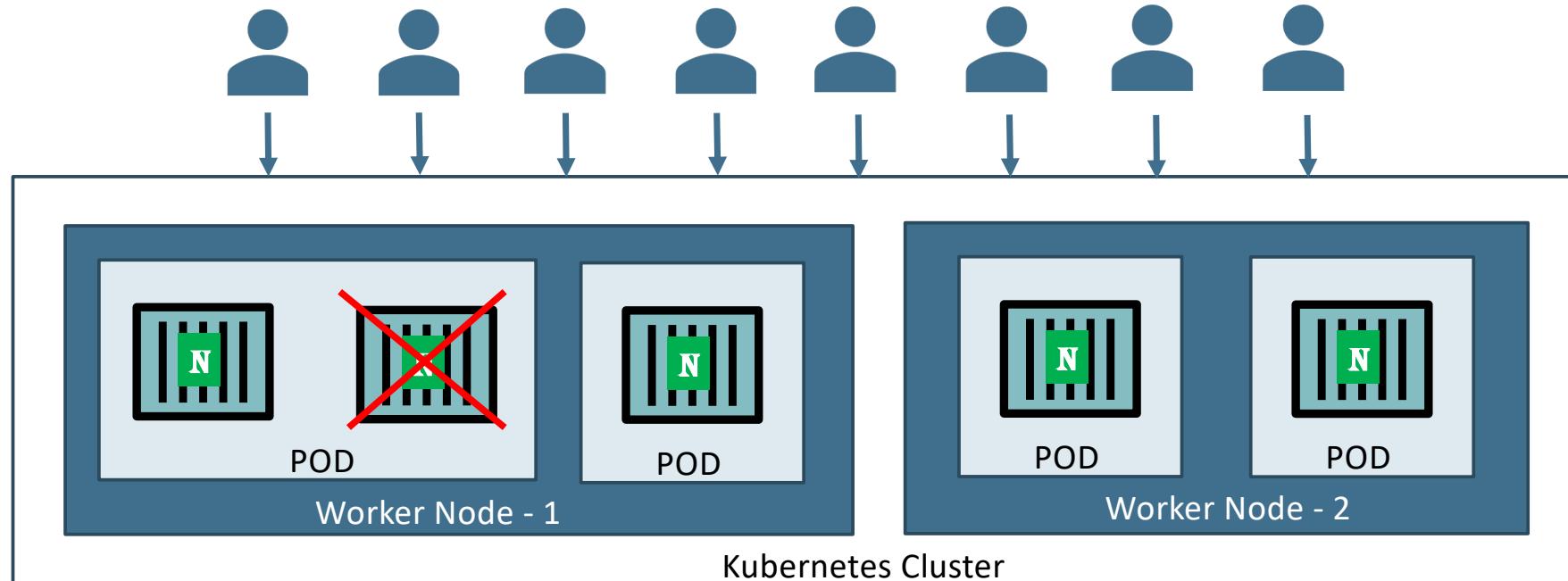
Kubernetes - POD

- PODs generally have **one to one** relationship with containers.
- To scale up we **create** new POD and to scale down we **delete** the POD.



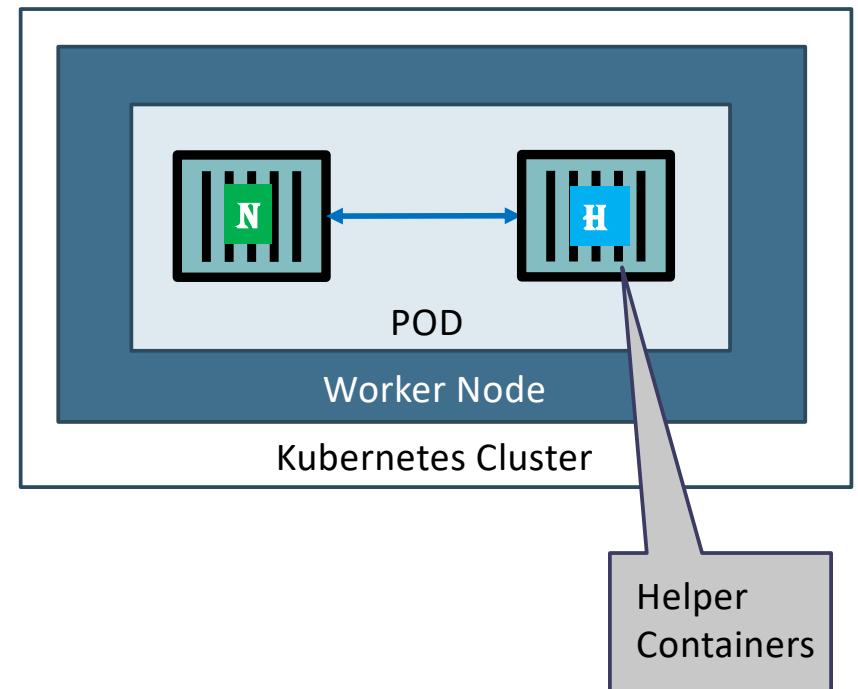
Kubernetes – PODs

- We **cannot have** multiple containers of **same kind** in a single POD.
- Example: Two NGINX containers in single POD serving same purpose is **not recommended**.



Kubernetes – Multi-Container Pods

- We can have multiple containers in a single POD, provided **they are not of same kind**.
- **Helper Containers (Side-car)**
 - Data Pullers: Pull data required by Main Container
 - Data pushers: Push data by collecting from main container (logs)
 - Proxies: Writes static data to html files using Helper container and Reads using Main Container.
- **Communication**
 - The two containers can easily communicate with each other easily as they share same **network space**.
 - They can also easily share **same storage space**.
- Multi-Container Pods is a **rare use-case** and we will try to focus on core fundamentals.

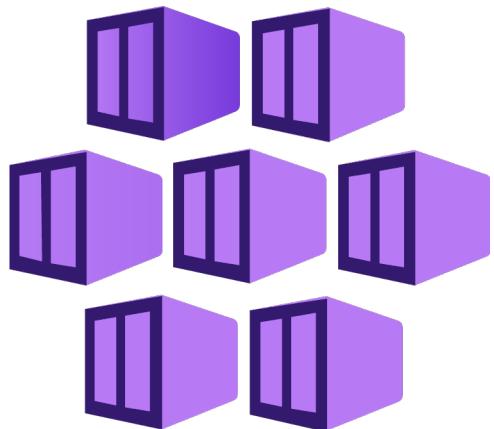


Kubernetes PODs Demo





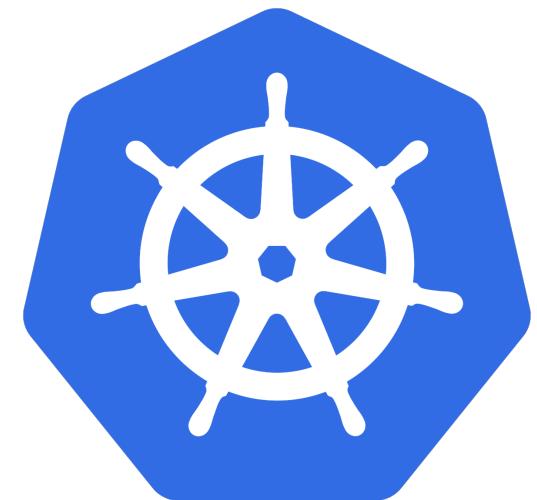
Azure Public
IP Address



Kubernetes Services Load Balancer

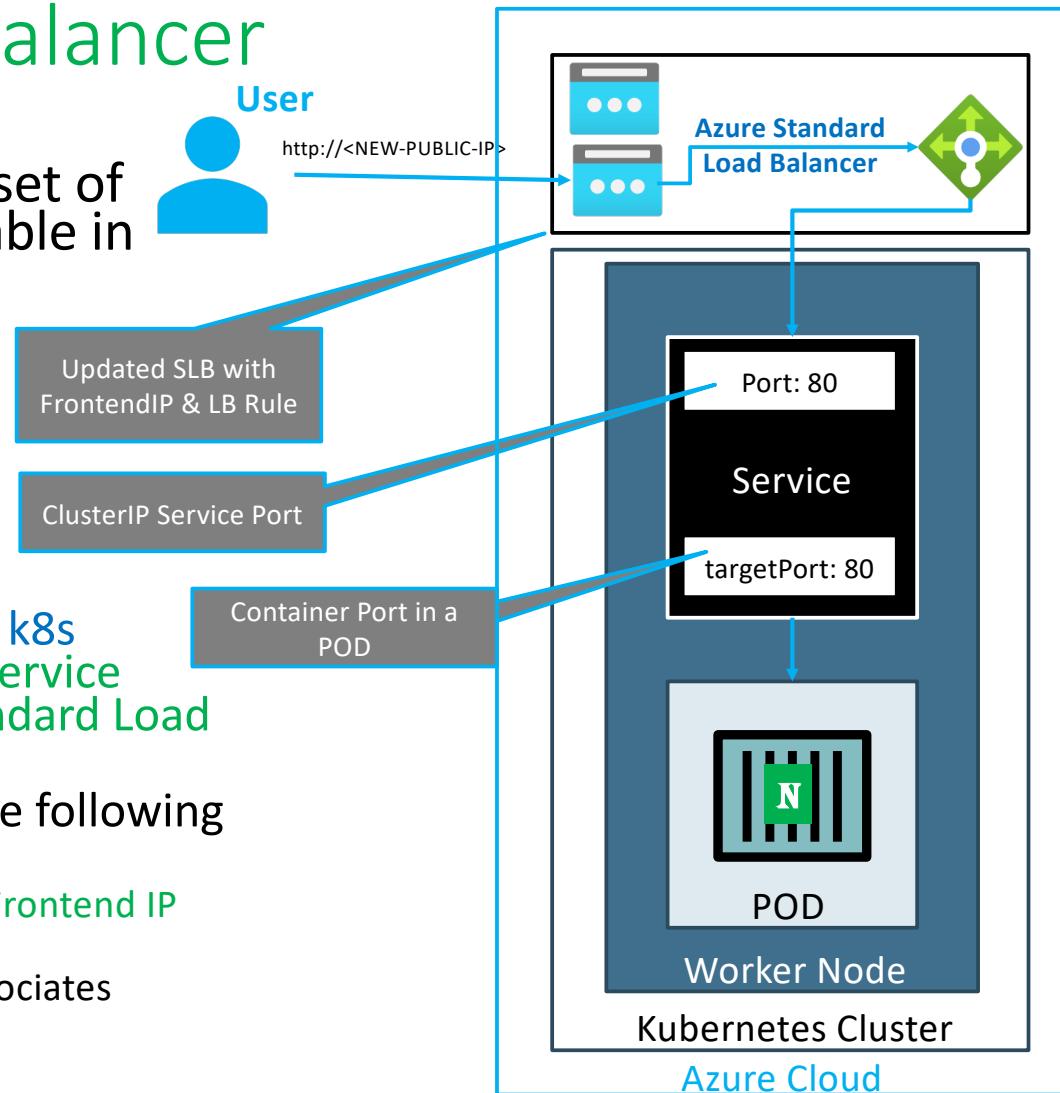


Azure Standard
Load Balancer



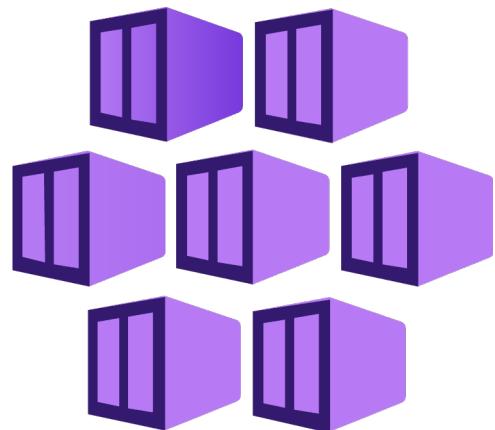
Kubernetes – Service - LoadBalancer

- We can expose an application running on a set of PODs using different types of Services available in k8s.
 - ClusterIP Service(Internal to k8s cluster)
 - NodePort Service (To internet)
 - LoadBalancer Service (To internet)
 - Ingress Service (To internet)
- LoadBalancer Service
 - To access our application outside of Azure AKS k8s cluster, we can use Kubernetes LoadBalancer service which will be eventually mapped to Azure Standard Load Balancer.
 - When we deploy k8s load balancer service, the following will happen in Azure Standard Load Balancer
 - A new Public IP gets created and associates that to Frontend IP Configuration
 - A new Load Balancing rule will be created which associates frontend ip and backend pool





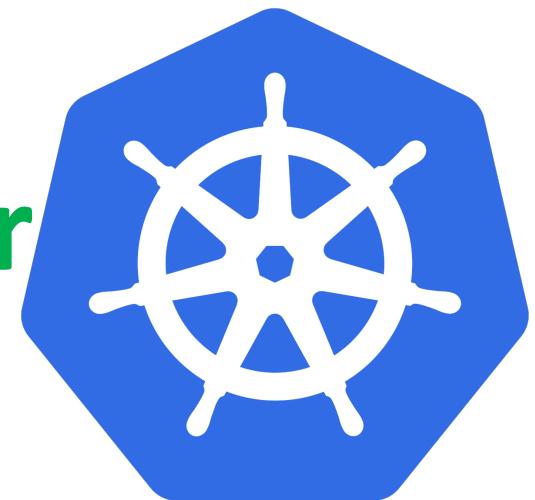
Azure Public
IP Address



Kubernetes POD & Load Balancer Service Demo



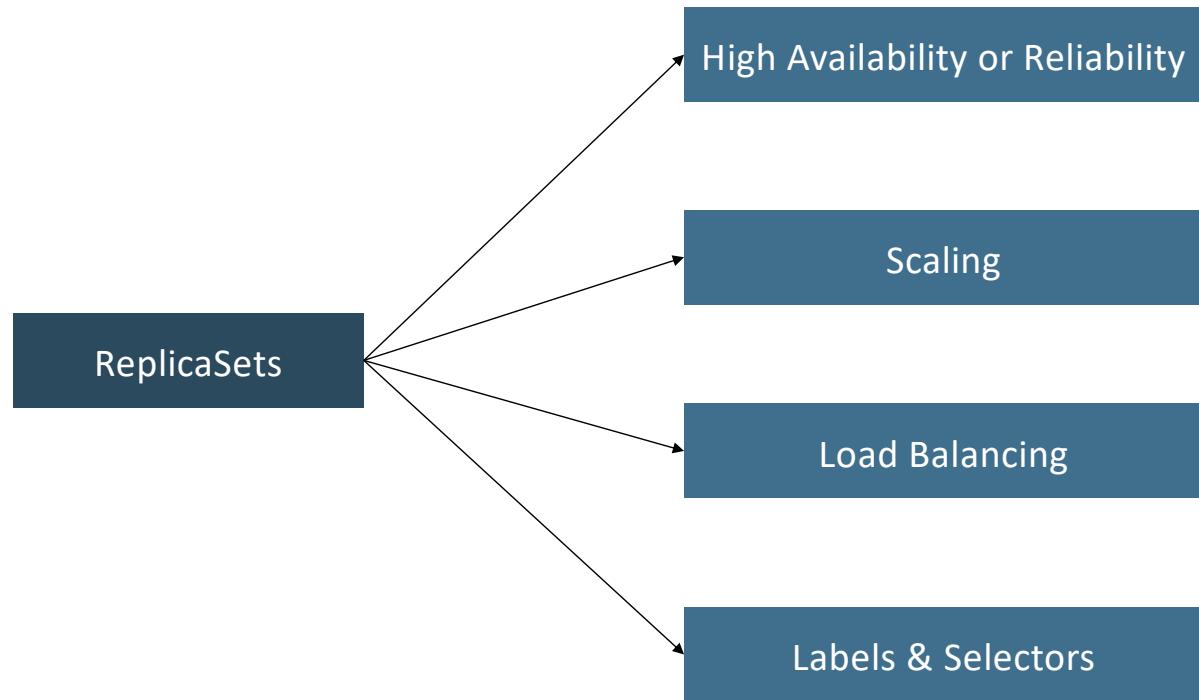
Azure Standard
Load Balancer



Kubernetes ReplicaSets



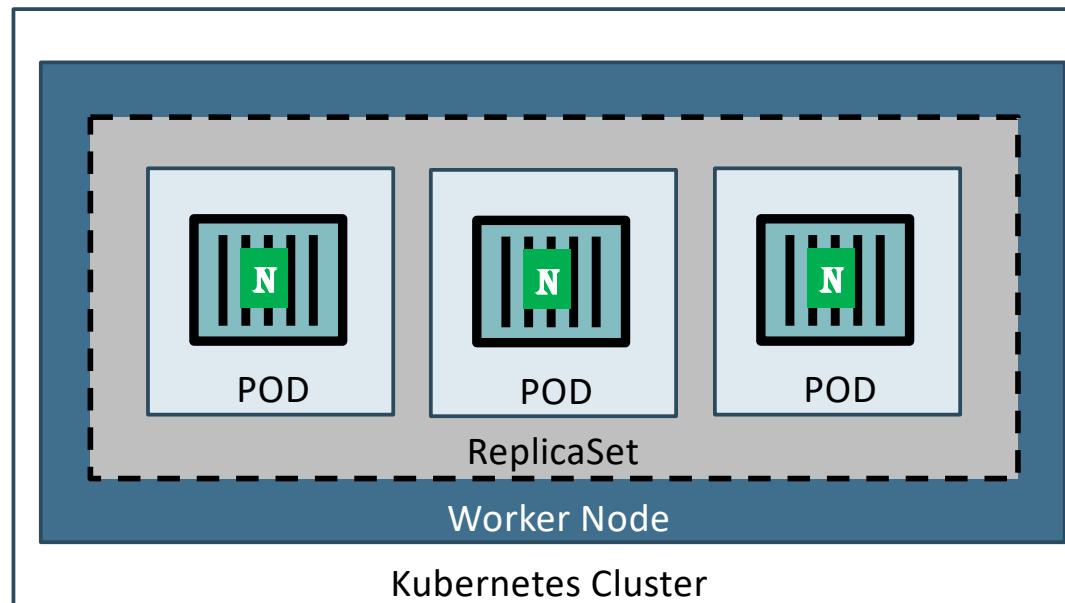
Kubernetes - ReplicaSets



Kubernetes – ReplicaSet

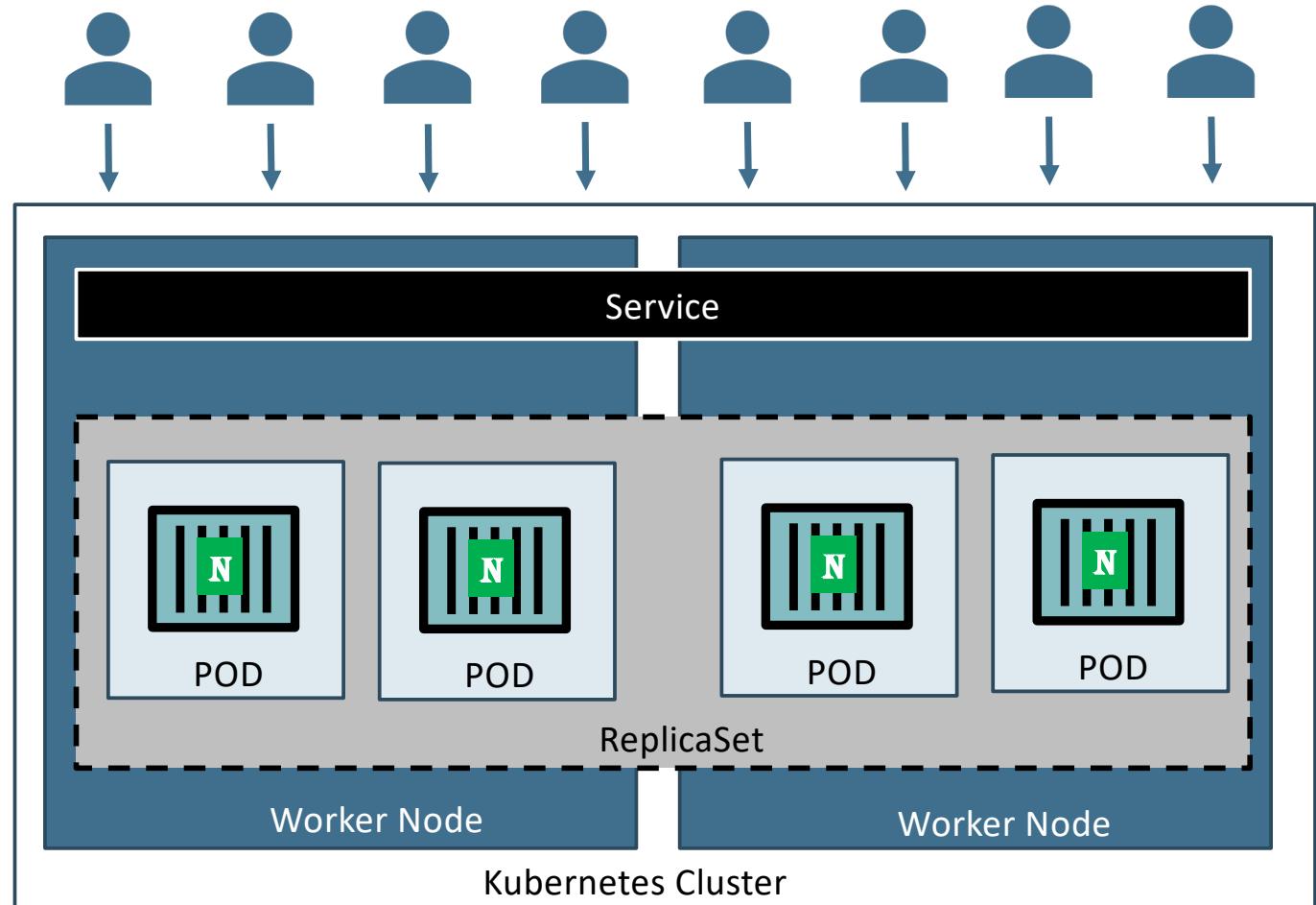
- A ReplicaSet's purpose is to maintain a **stable set of replica Pods** running at any given time.
- If our **application crashes (any pod dies)**, replicaset will **recreate** the pod immediately to ensure the configured number of pods running at any given time.

Reliability
Or
High Availability



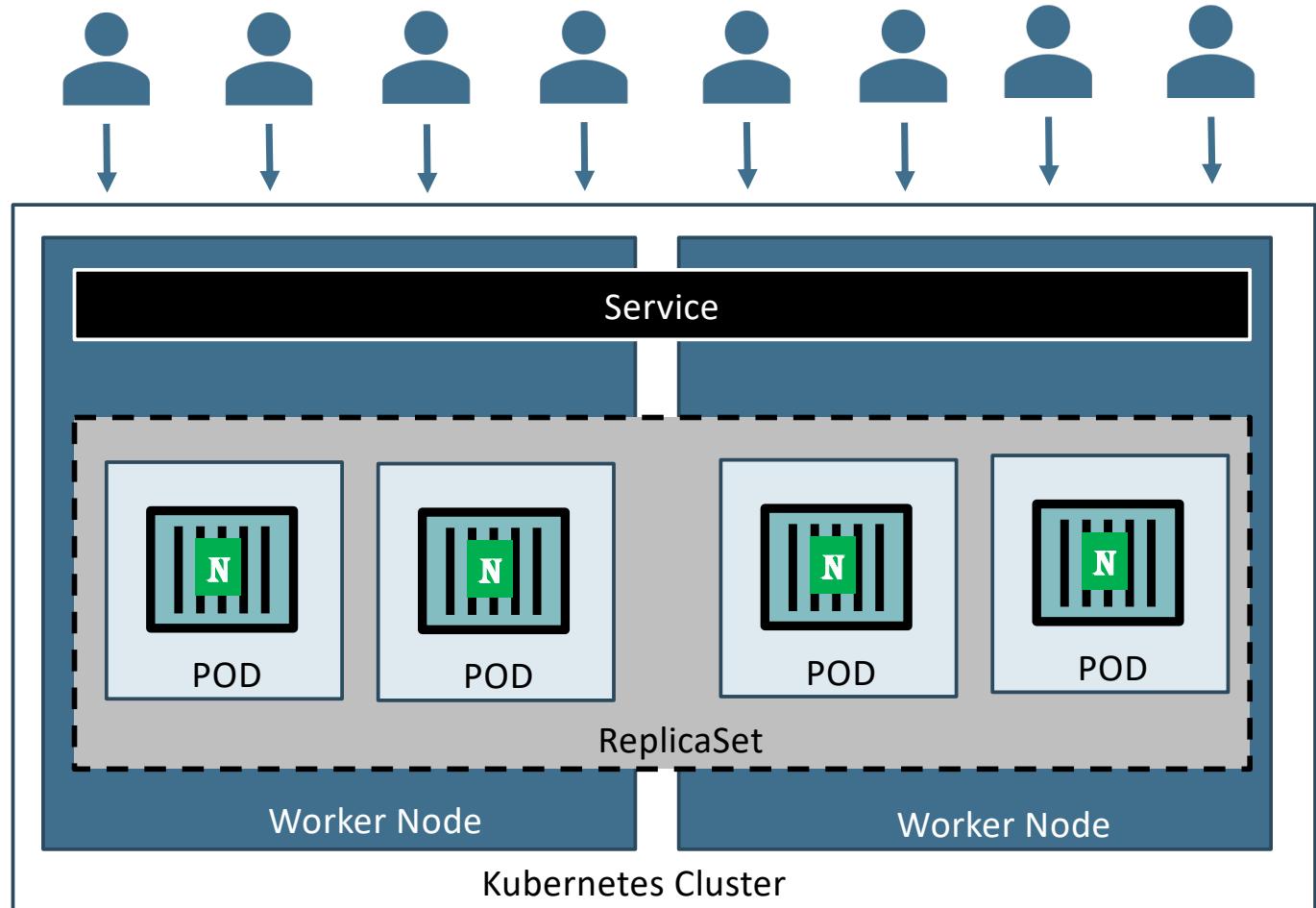
Kubernetes – ReplicaSet

- Load Balancing
- To avoid overloading of traffic to single pod we can use **load balancing**.
- Kubernetes provides pod load balancing **out of the box** using **Services** for the pods which are part of a ReplicaSet
- **Labels & Selectors** are the **key items** which **ties** all 3 together (Pod, ReplicaSet & Service), we will know in detail when we are writing YAML manifests for these objects



Kubernetes – ReplicaSet

- Scaling
- When load become too much for the number of existing pods, Kubernetes enables us to easily **scale** up our application, adding additional pods as needed.
- This is going to be **seamless and super quick**.



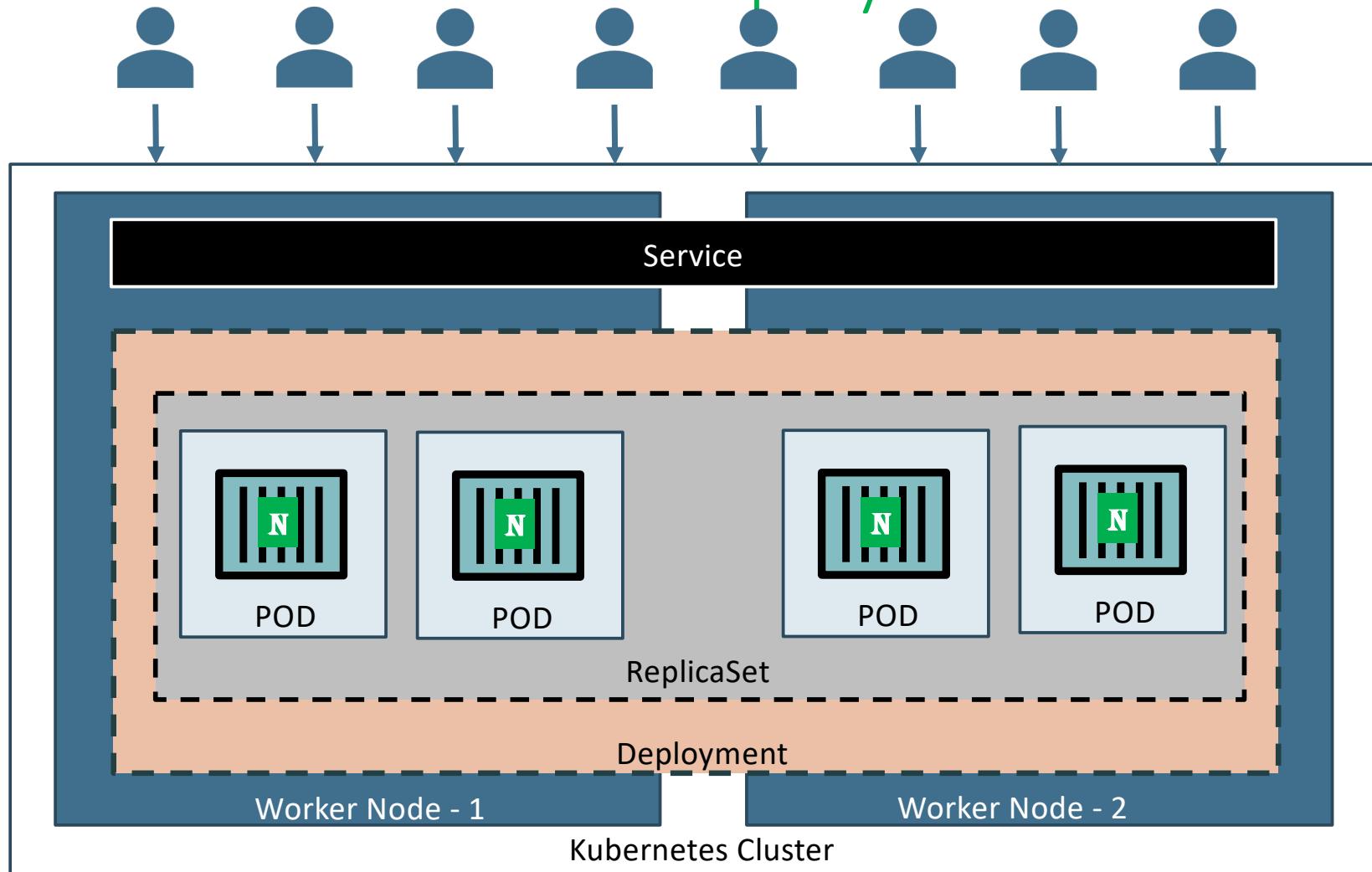
Kubernetes ReplicaSets Demo



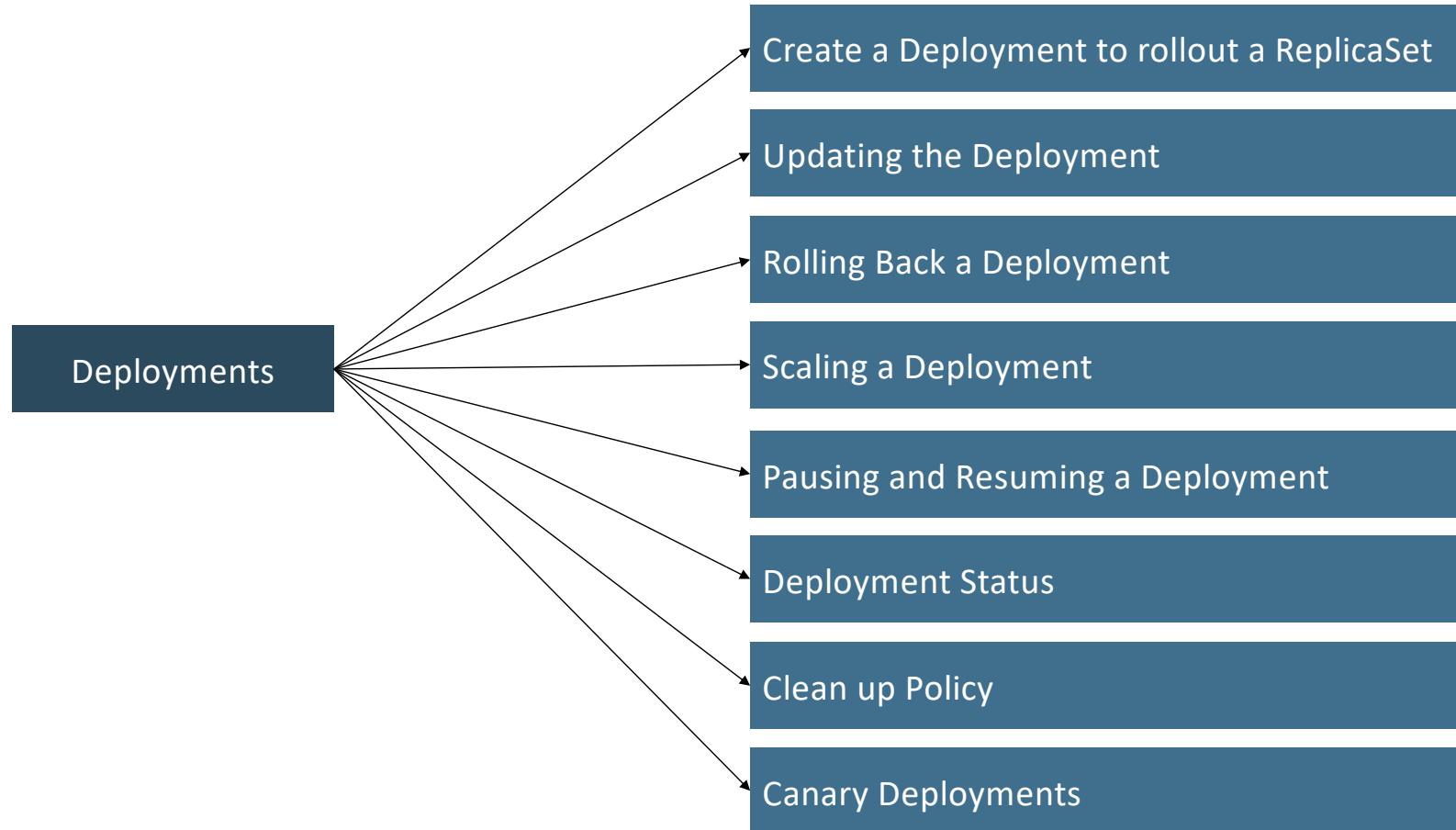
Kubernetes Deployments



Kubernetes – Deployments



Kubernetes - Deployment



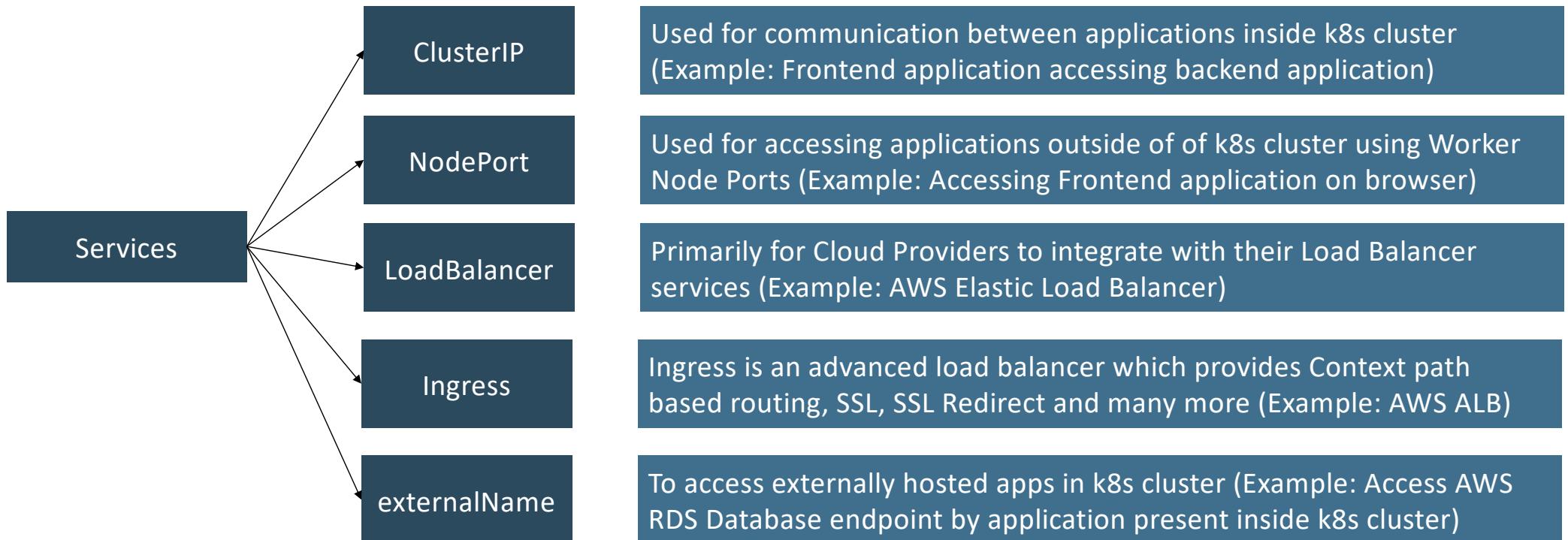
Kubernetes Deployments Demo



Kubernetes Services



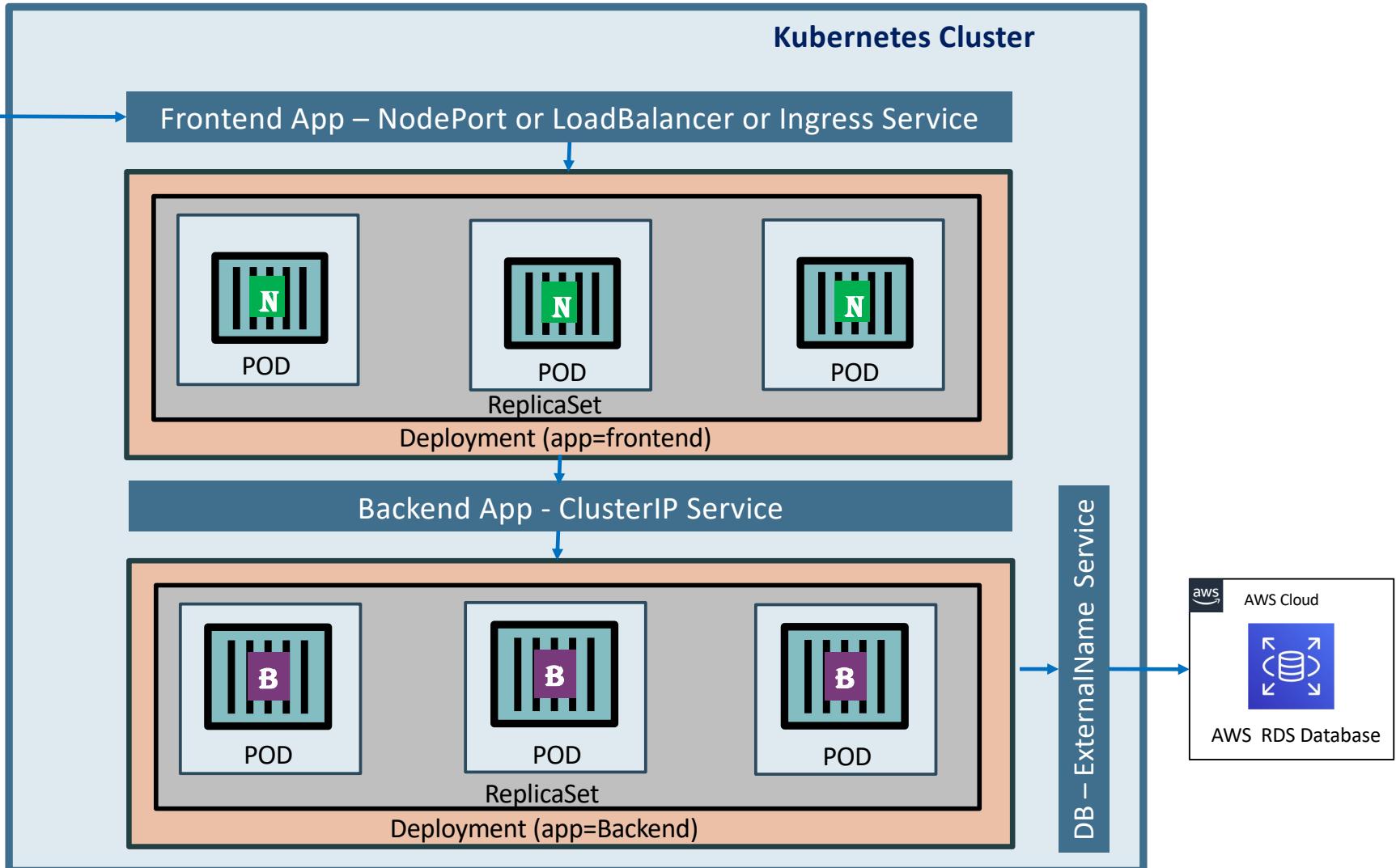
Kubernetes - Services



Services



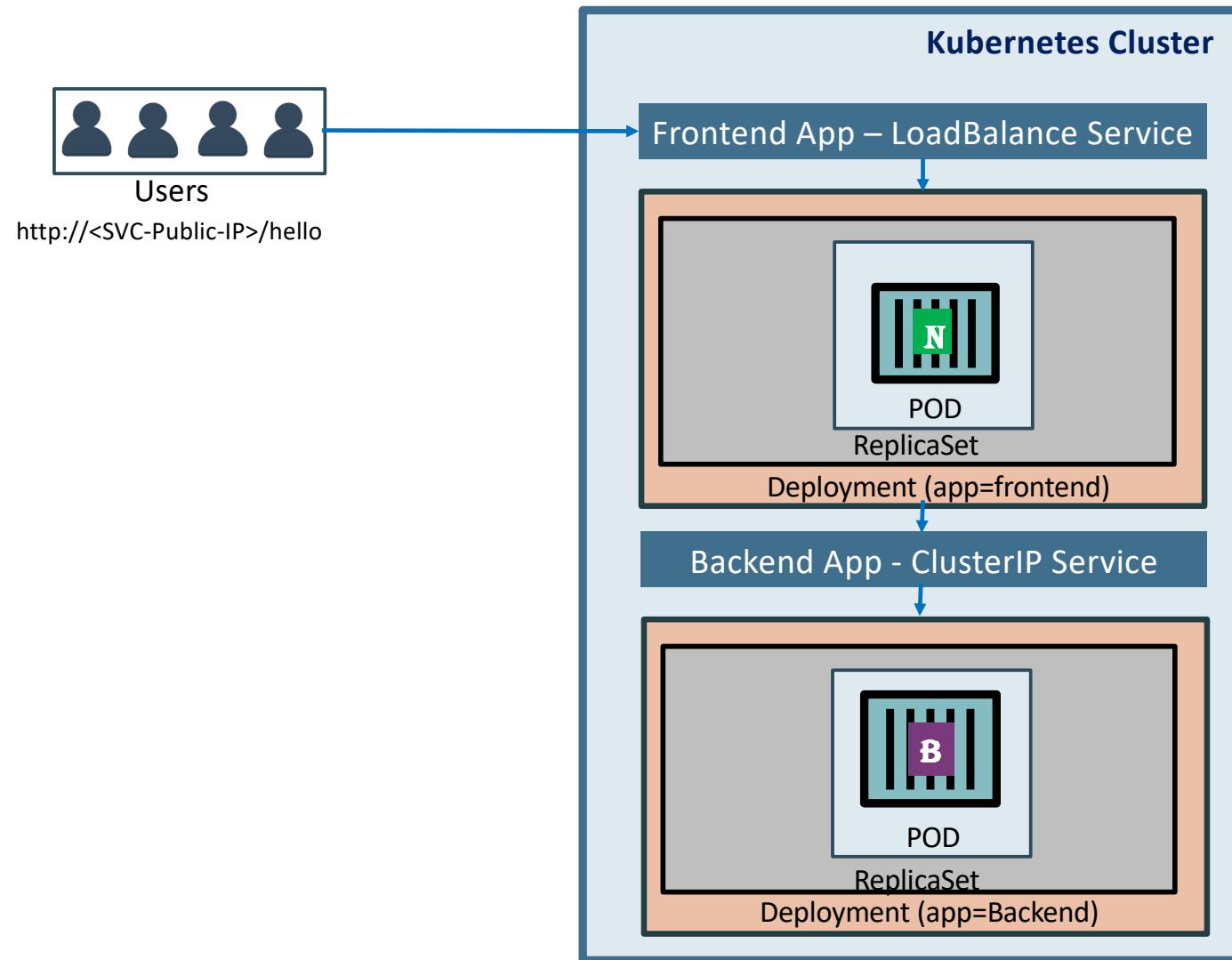
Users



Kubernetes Services Demo



Services Demo



Kubernetes YAML Basics

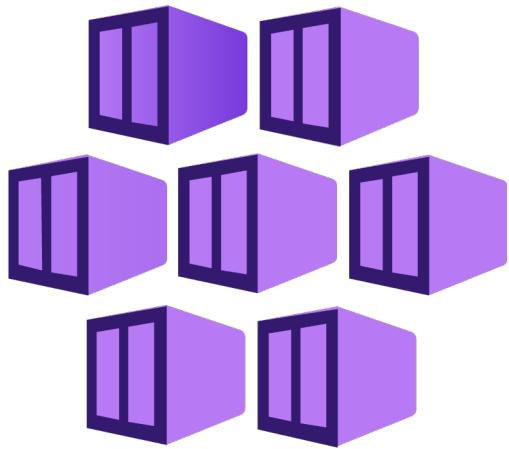


YAML Basics

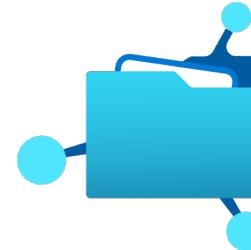
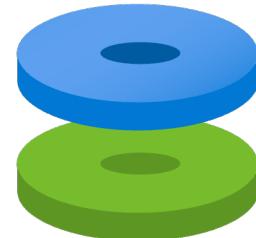
- YAML is **not a** Markup Language
- YAML is used to **store information** about different things
- We can use YAML to define **key, Value pairs** like variables, lists and objects
- YAML is very similar to **JSON** (Javascript Object Notation)
- YAML primarily focuses on **readability** and **user friendliness**
- YAML is designed to be **clean and easy to read**
- We can define YAML files with two different extensions
 - abc.**yml**
 - abc.**yaml**

YAML Basics

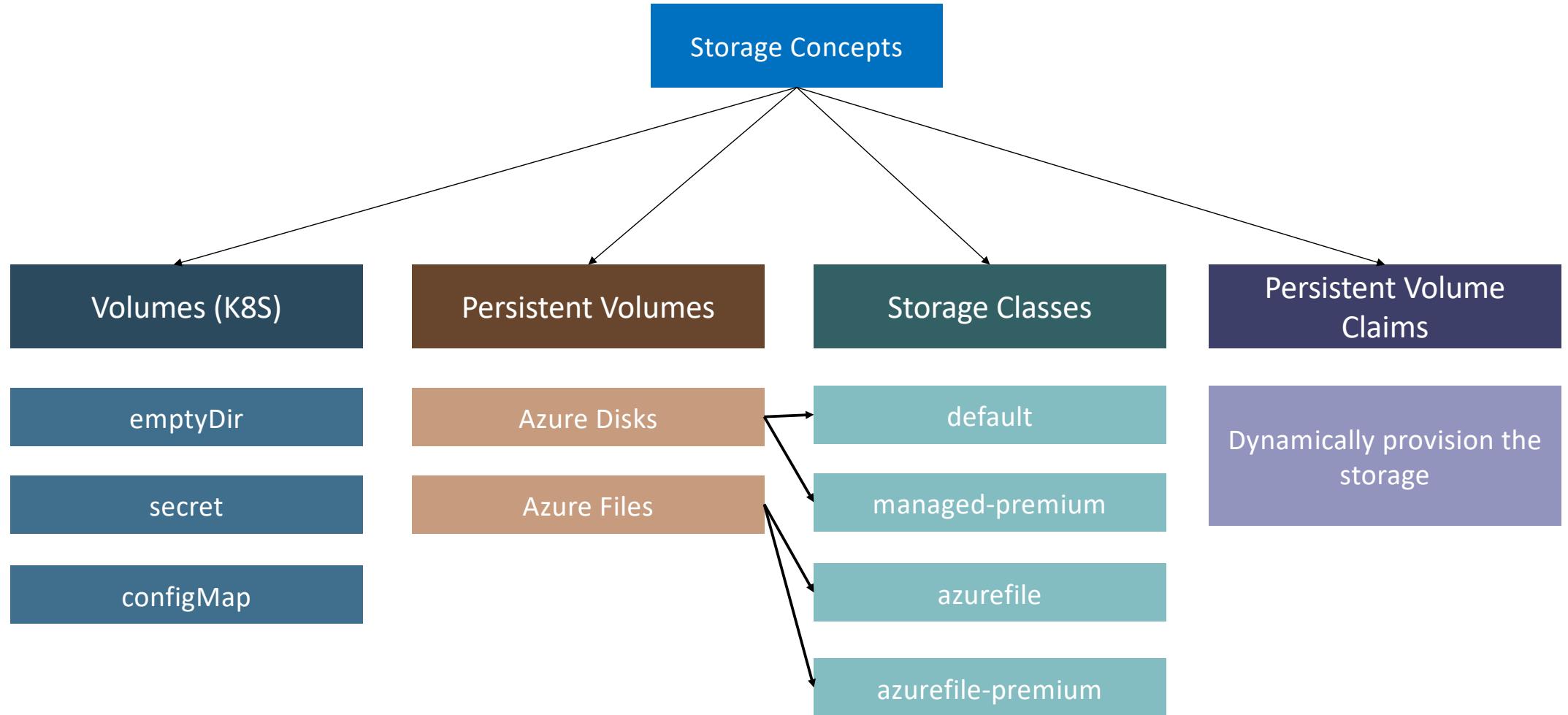
- YAML Comments
- YAML Key Value Pairs
- YAML Dictionary or Map
- YAML Array / Lists
- YAML Spaces
- YAML Document Separator



Azure AKS Storage with Azure Disks & Files

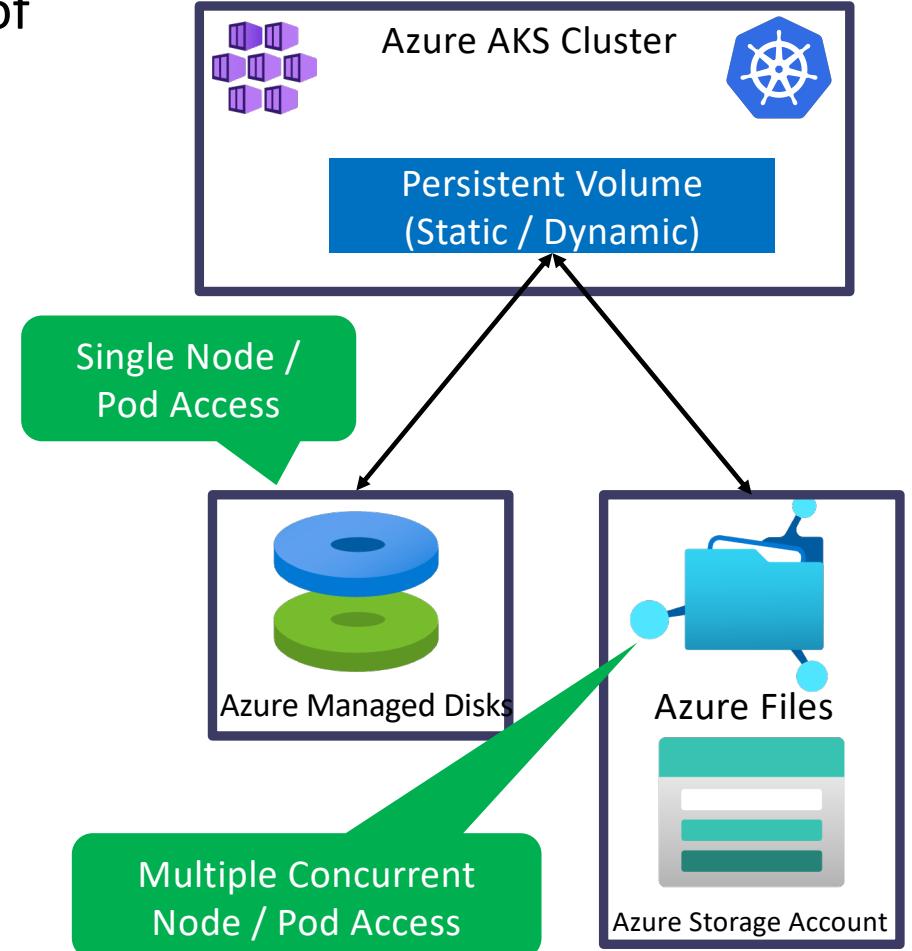


AKS Storage Concepts

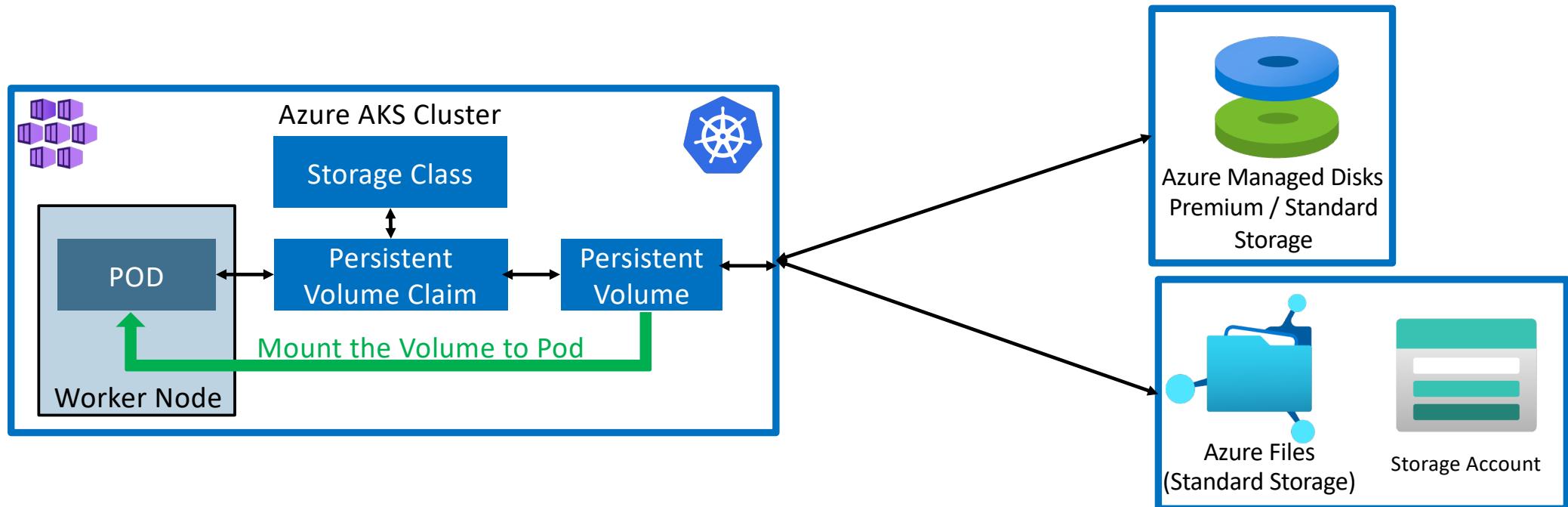


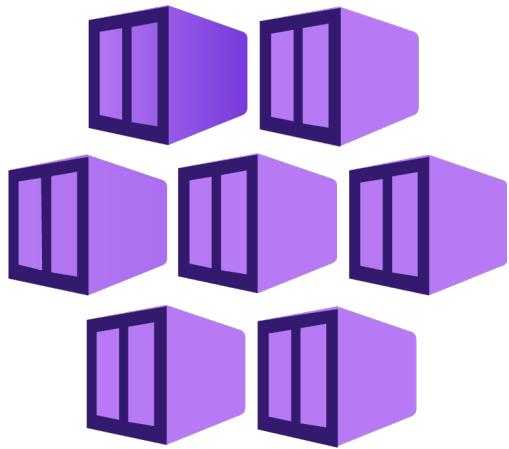
Why do we need Persistent Storage in Kubernetes?

- Volumes that are defined and created as part of the **pod lifecycle** only exist until the pod is **deleted**.
- Pods often expect their storage to remain if a pod is **rescheduled** on a different host during a maintenance event, especially in StatefulSets.
- A ***persistent volume (PV)*** is a storage resource created and managed by the Kubernetes API that can exist beyond the lifetime of an individual pod.
- For AKS, **Azure Disks or Azure Files** are used to provide the PersistentVolume.

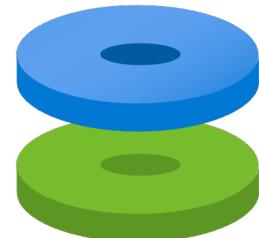


Persistent Volume Claims – How it works?





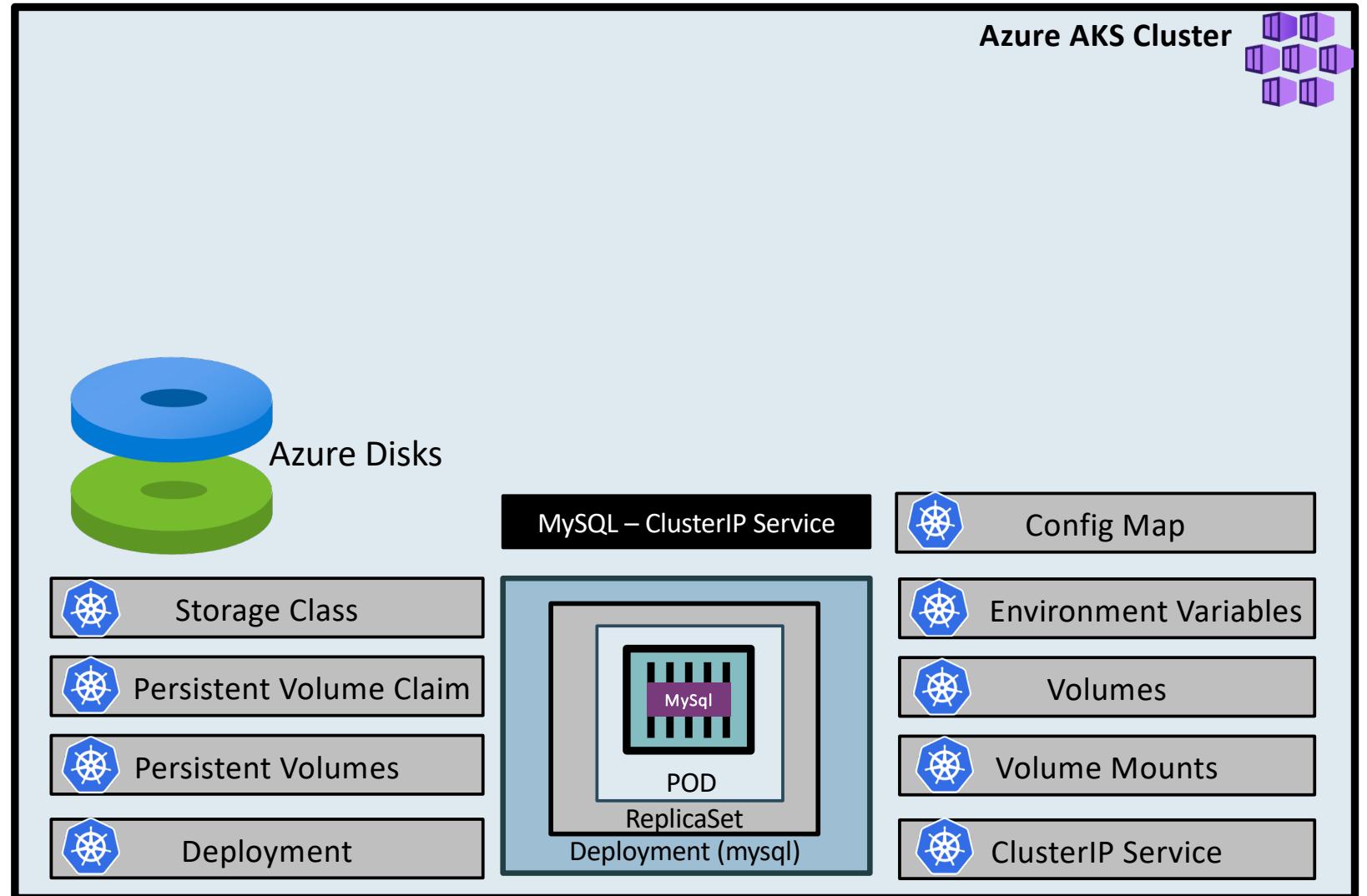
Azure AKS Storage with Azure Disks

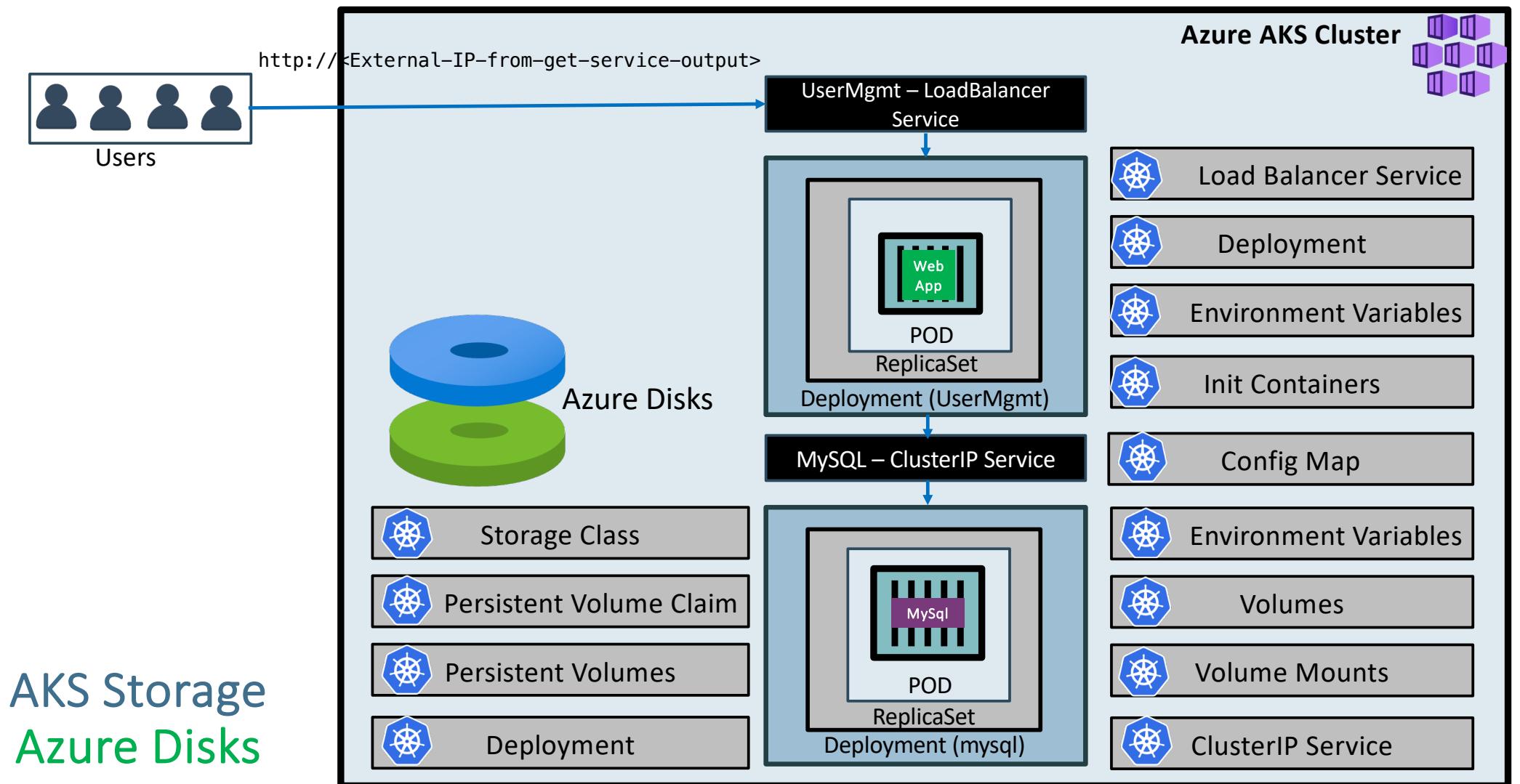


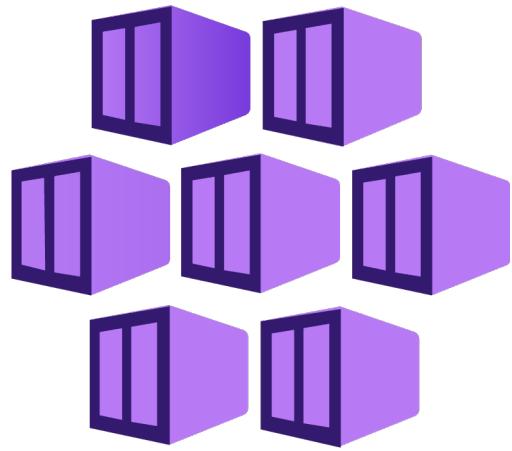
Azure Disks - Introduction

- Azure Disk Storage offers **high-performance, highly durable** block storage for our mission- and business-critical workloads
- We can mount these **volumes as devices** on our Virtual Machines & Container instances.
- **Cost-effective storage**
 - Built-in bursting capabilities to handle unexpected traffic and process batch jobs cost-effectively
- **Unmatched resiliency**
 - 0 percent annual failure rate for consistent enterprise-grade durability
- **Seamless scalability**
 - Dynamic scaling of disk performance on Ultra Disk Storage without disruption
- **Built-in security**
 - Automatic encryption to help protect your data using Microsoft-managed keys or your own

AKS Storage Azure Disks







Azure AKS Storage with Azure MySQL Database





Users

AKS Storage Azure MySQL Database

Drawbacks of Azure Disks

Complex setup to achieve HA,
Statefulsets

Complex Master-Master MySQL setup

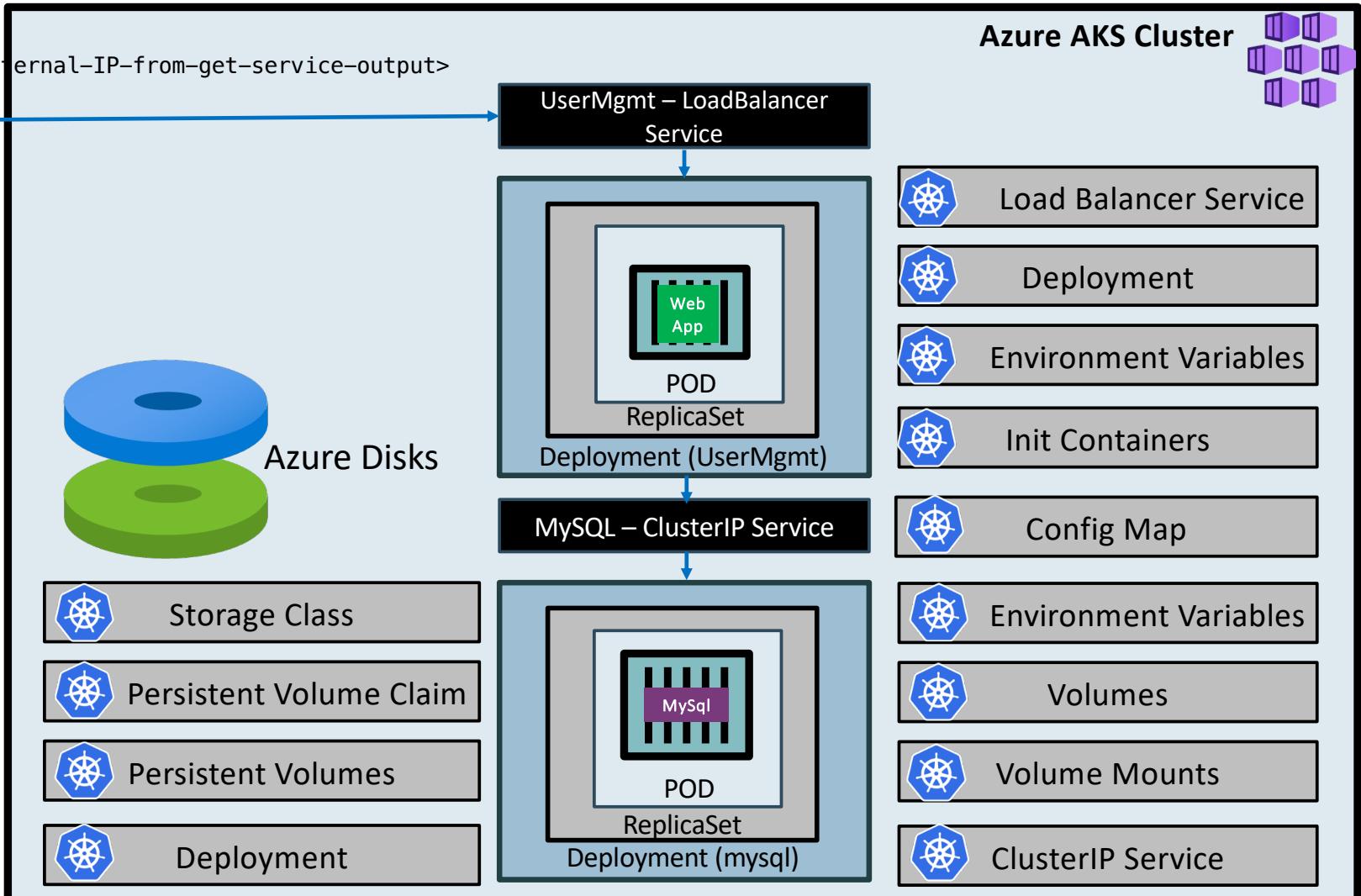
Complex Master-Slave MySQL setup

No Automatic Backup & Recovery

No Auto-Upgrade MySQL

Logging, Monitoring needs custom
scripts

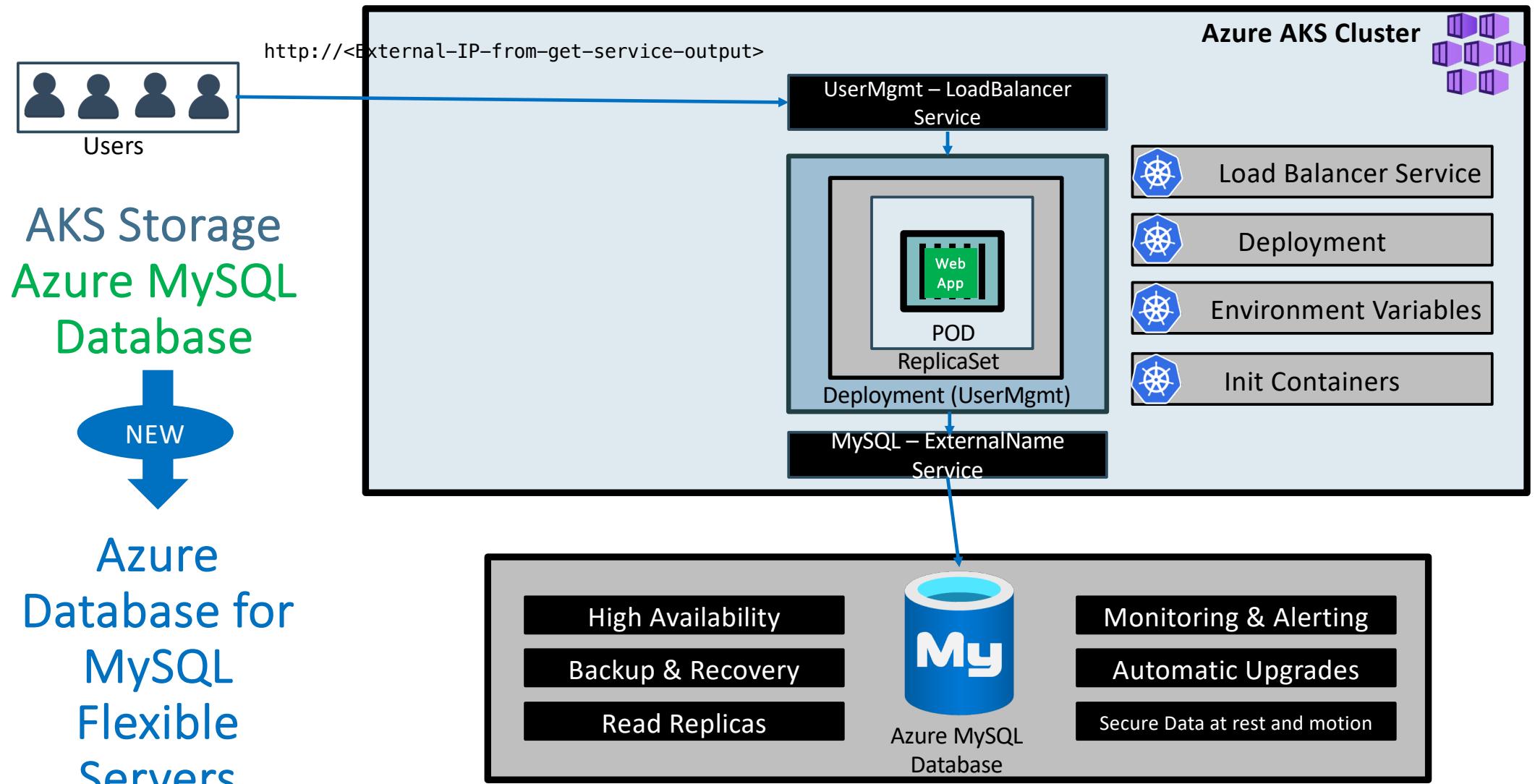
http://<External-IP-from-service-output>

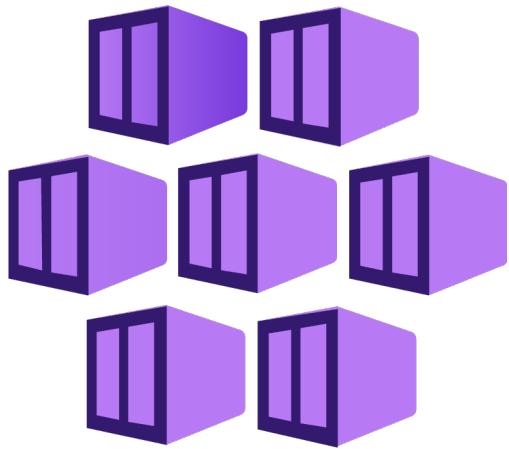


Azure MySQL Database

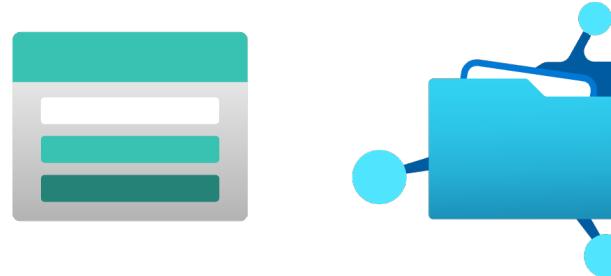


- Features
- Built-in **high availability** with no additional cost.
- Predictable **performance**, using inclusive pay-as-you-go pricing.
- **Scale** as needed within seconds.
- Secured to protect sensitive **data at-rest and in-motion**.
- Automatic **backups** and point-in-time-restore for up to 35 days.
- Enterprise-grade **security** and compliance.





Azure AKS Storage with Azure Files

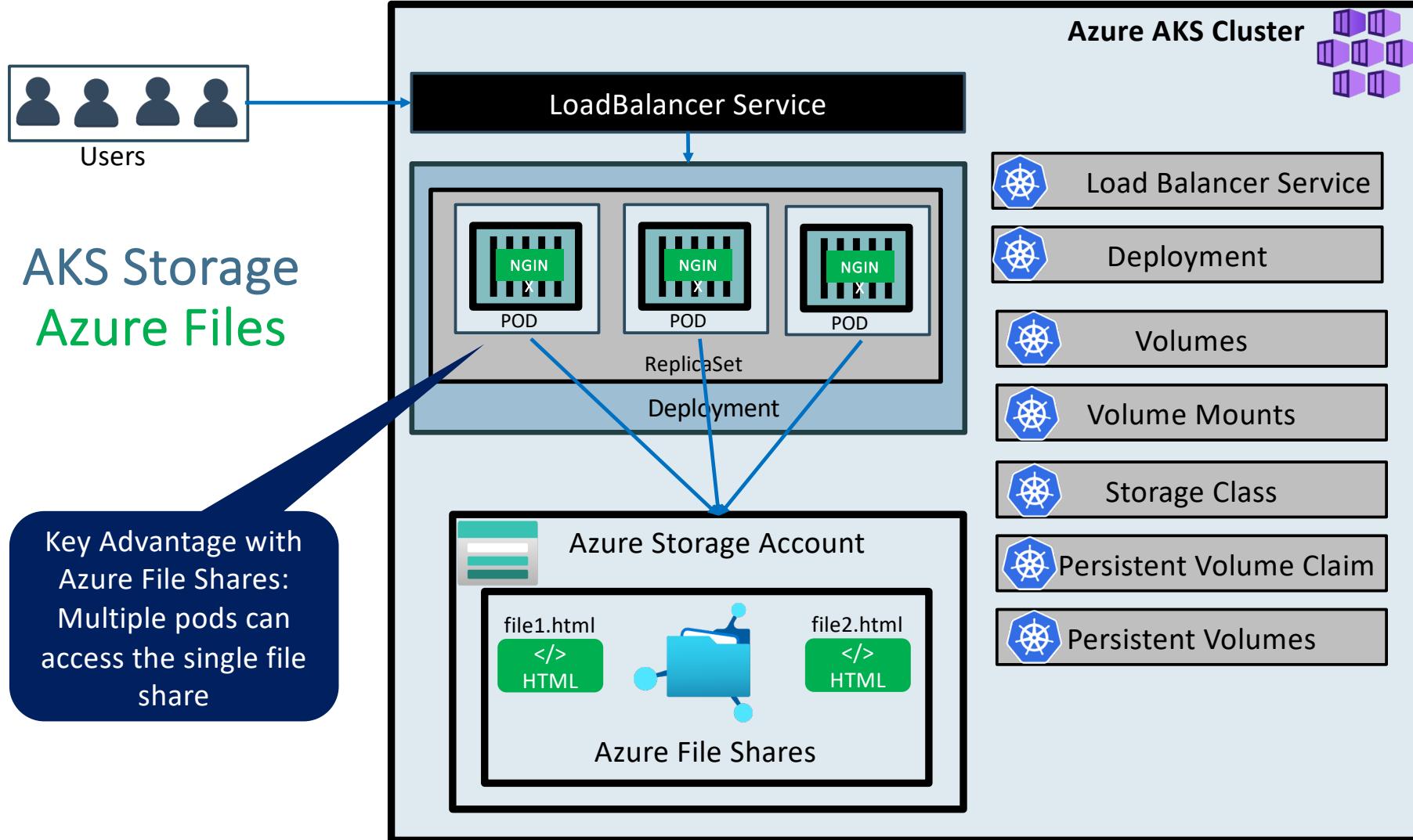


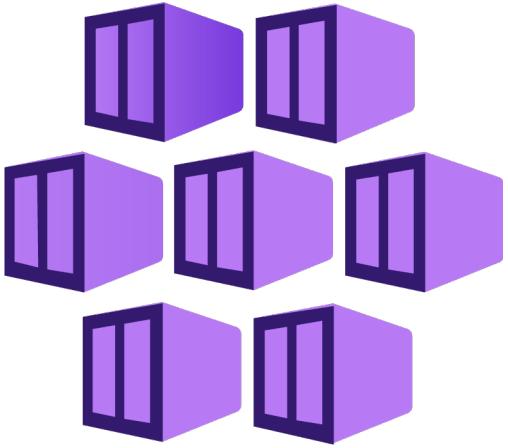
Azure Files



- These are simple, secure and **fully managed** cloud file shares
- We can secure **data at rest** and **in-transit** using SMB 3.0 and HTTPS
- We can create **high-performance file shares** using the Premium Files storage tier
- We can replace or supplement **on-premises file servers**
- **Scripting and tooling:** **PowerShell** cmdlets and **Azure CLI** can be used to create, mount, and manage Azure file shares as part of the administration of Azure applications.
- We can create and manage Azure file shares using **Azure portal** and **Azure Storage Explorer**.
- For Application workloads we can use for use cases like **Static Content storage**, shared **configuration access** to multiple JVMs etc.

<http://<External-IP-from-get-service-output>/app1/file1.html>

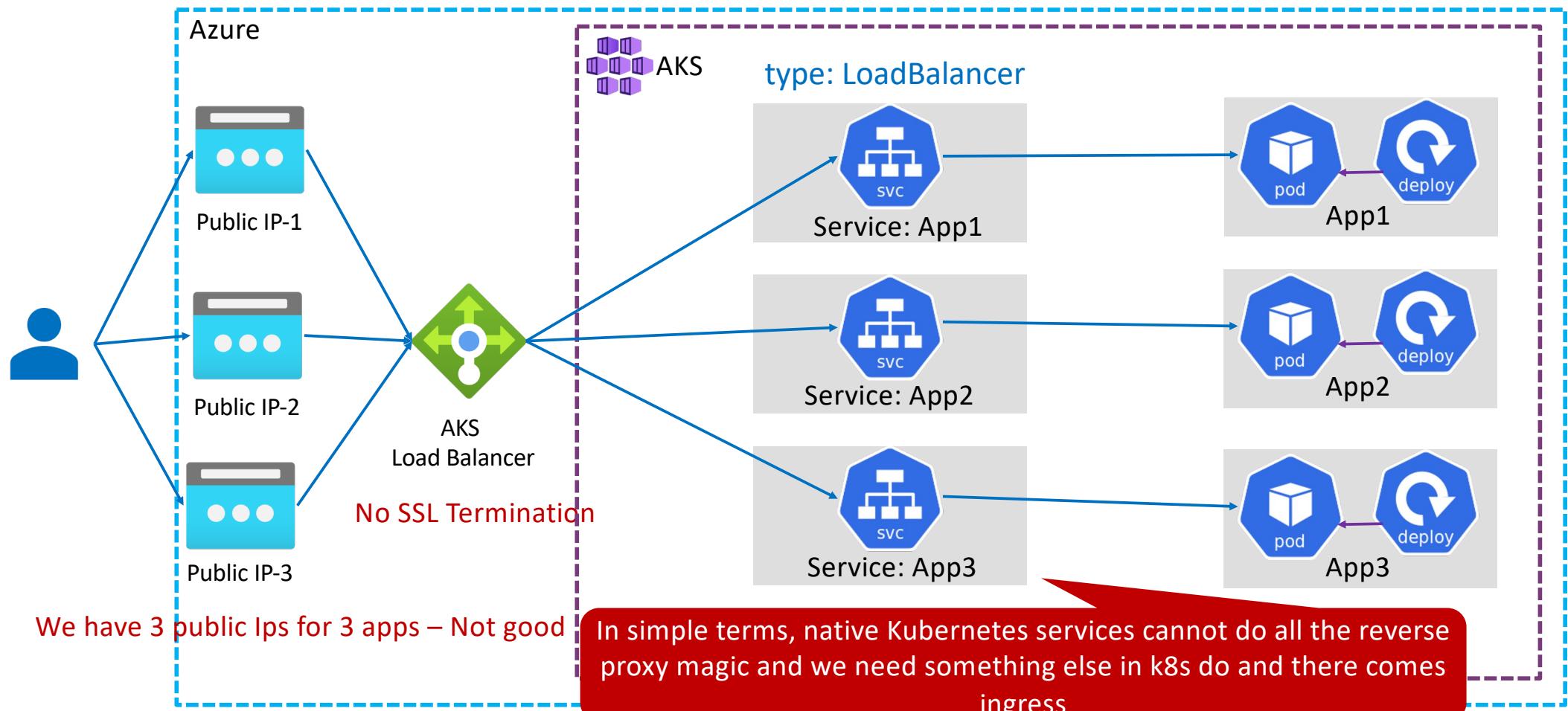




Azure AKS Ingress

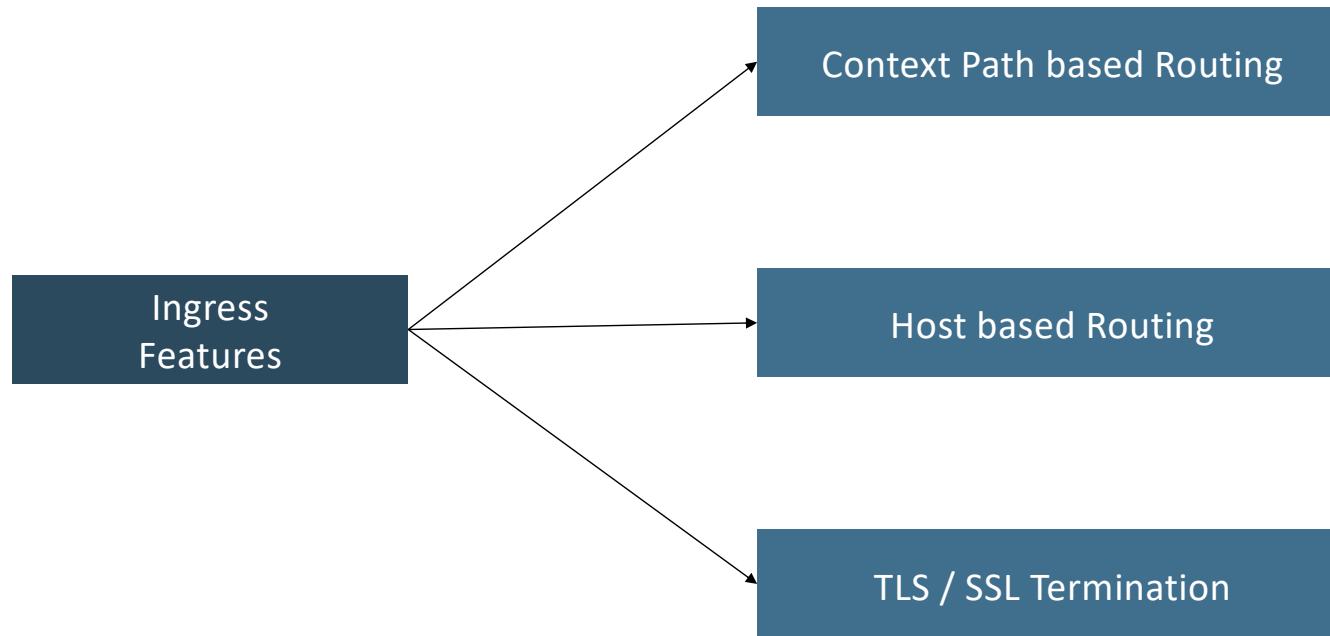


Why do we need Ingress?



What is Ingress?

Ingress is a Kubernetes resource that lets us configure an **HTTP load balancer** for applications running on Kubernetes, with advanced capabilities at HTTP layer.



Ingress - Terminology

Ingress Controller

Ingress controller is an application that runs in a cluster and configures an HTTP load balancer according to Ingress resources.

The load balancer can be a software load balancer running in the cluster or a hardware or cloud load balancer running externally.

In the case of NGINX, the Ingress controller is deployed in a pod along with the load balancer.

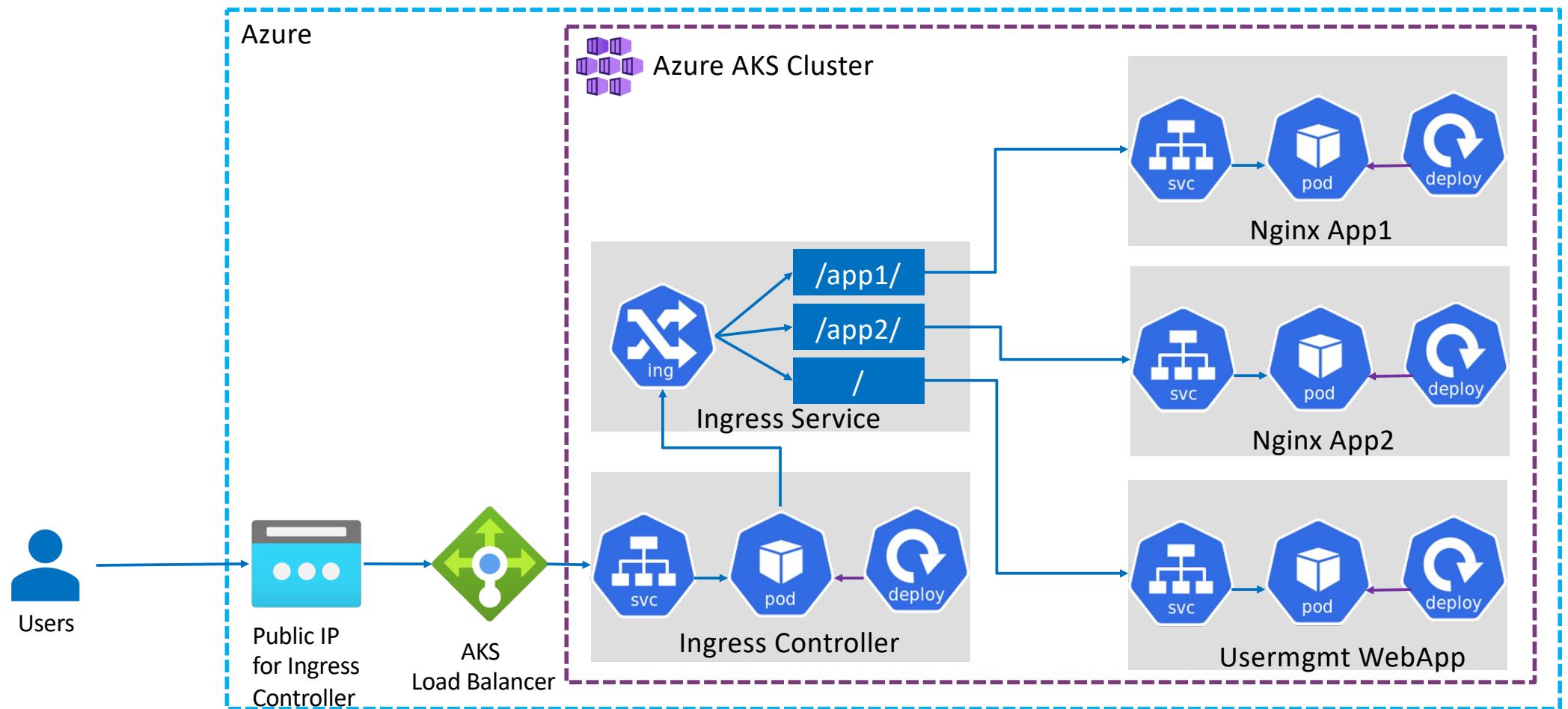
Ingress Resource or Service

Equivalent to Kubernetes Services but with lot of additional features

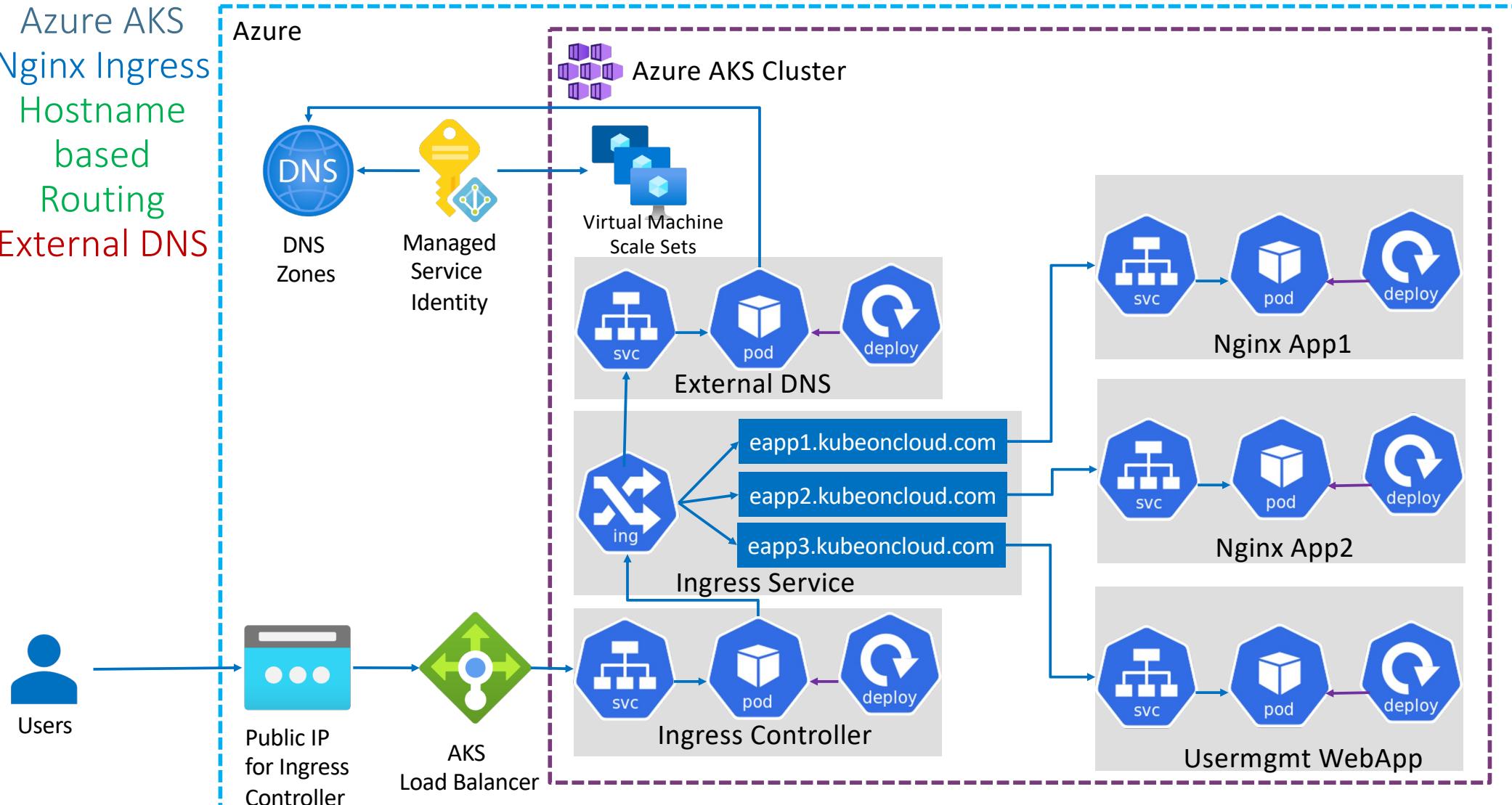
We can define routing rules based on URI or Hostname in Ingress Resource Manifest

We can even define SSL / TLS related information in Ingress Resource Manifest

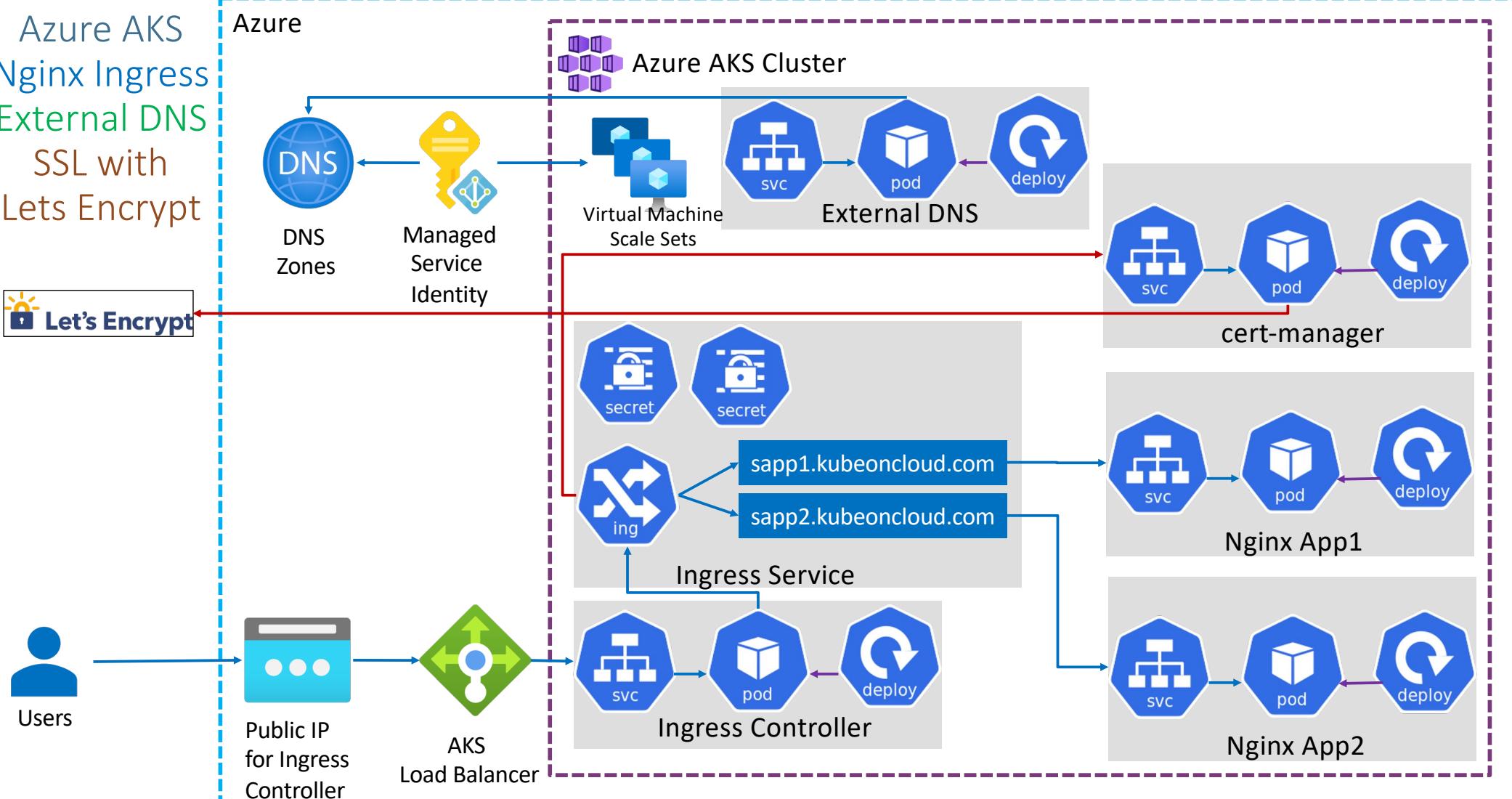
Azure AKS & Nginx Ingress – Context Path Based Routing



Azure AKS
Nginx Ingress
Hostname
based
Routing
External DNS

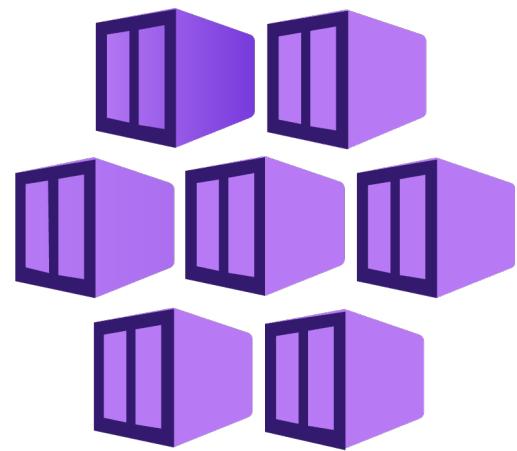


Azure AKS
Nginx Ingress
External DNS
SSL with
Lets Encrypt



Nginx Ingress Controller - References

- List of Ingress Controllers
 - <https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/>
- Azure AGIC (Application Gateway Ingress Controller)
 - <https://github.com/Azure/application-gateway-kubernetes-ingress>
 - <https://azure.microsoft.com/en-in/pricing/details/application-gateway/>



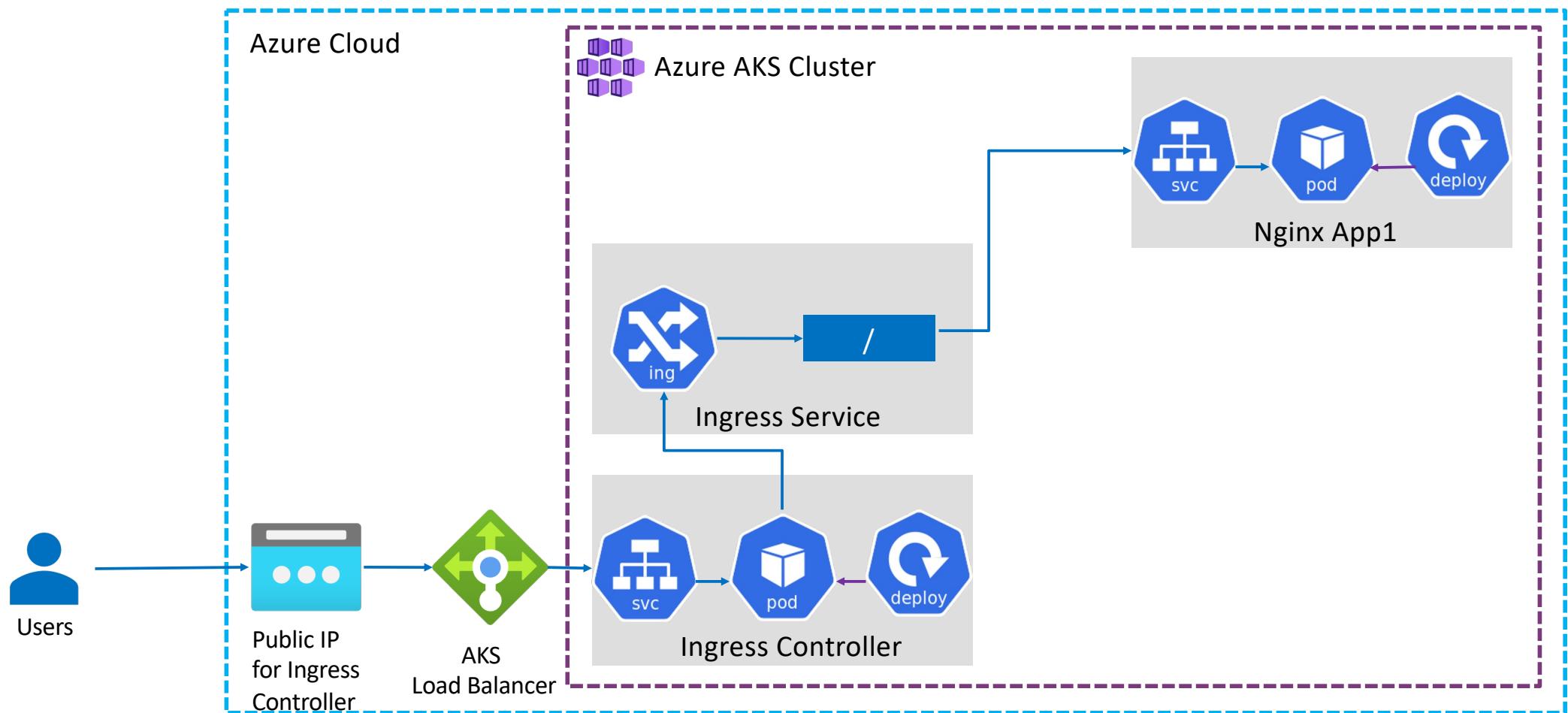
Azure AKS

Nginx Ingress Service

Install & Implement



Azure AKS & Nginx Ingress – Basic Architecture



Basic Ingress Manifest

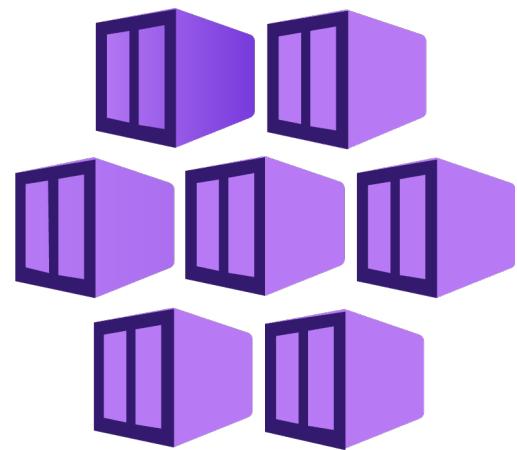
Deprecated

```
! 03-Ingress-Basic.yml ×
azure-aks-kubernetes-masterclass > 09-Ingress-Basic > kube-manifests > ! 03-Ingress-Basic.yml > apiVersion
1  apiVersion: networking.k8s.io/v1beta1
2  kind: Ingress
3  metadata:
4    name: nginxapp1-ingress-service
5    annotations:
6      | kubernetes.io/ingress.class: "nginx"
7  spec:
8    rules:
9      - http:
10        | paths:
11        |   - path: /
12        |     | backend:
13        |       | serviceName: app1-nginx-clusterip-service
14        |       | servicePort: 80
15
```

Basic Ingress Manifest

NEW

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginxapp1-ingress-service
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: app1-nginx-clusterip-service
                port:
                  number: 80
```



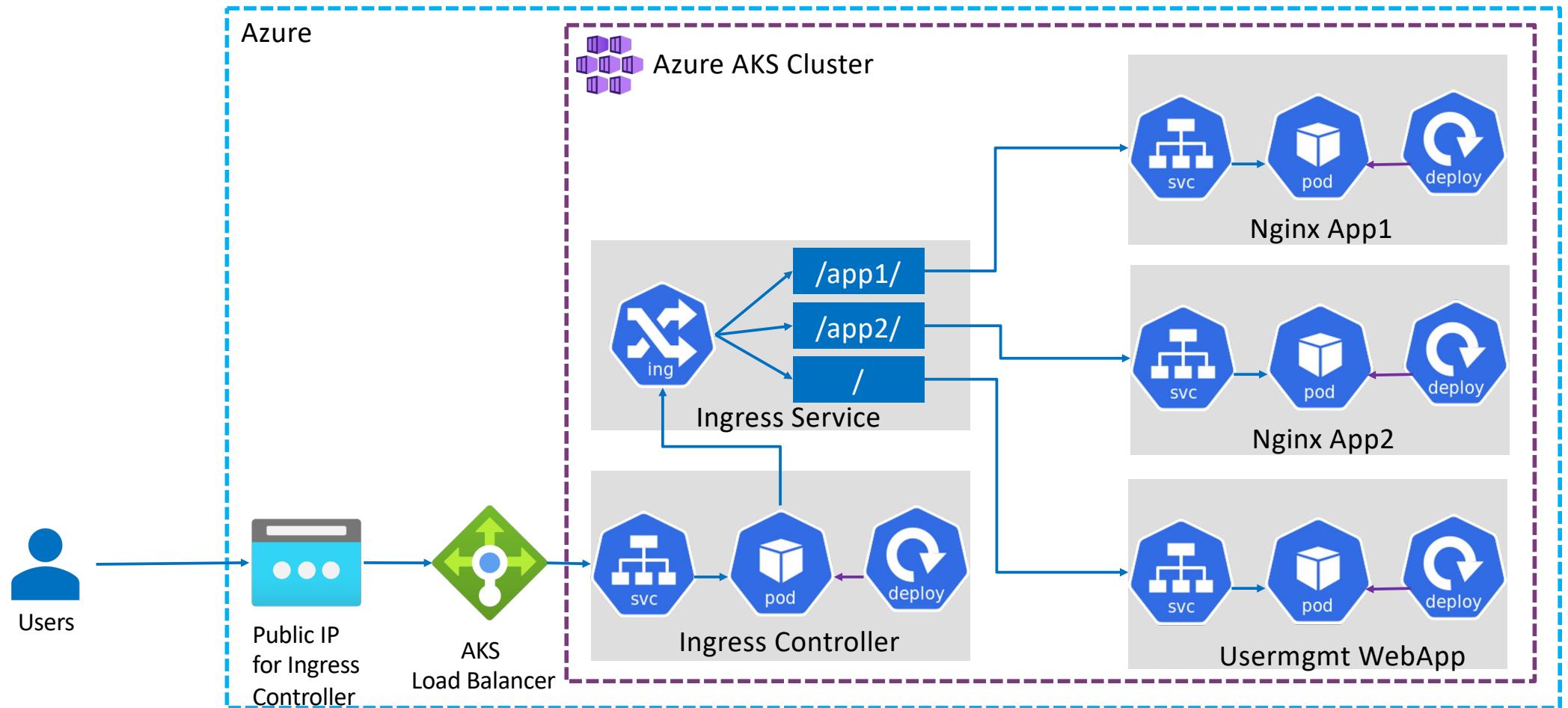
Azure AKS

Nginx Ingress Service

Context Path based Routing

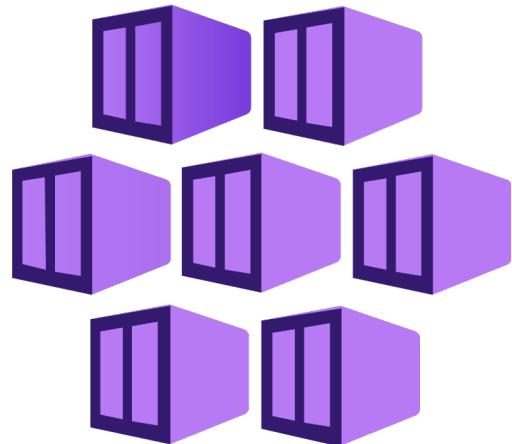


Azure AKS & Nginx Ingress – Context Path Based Routing

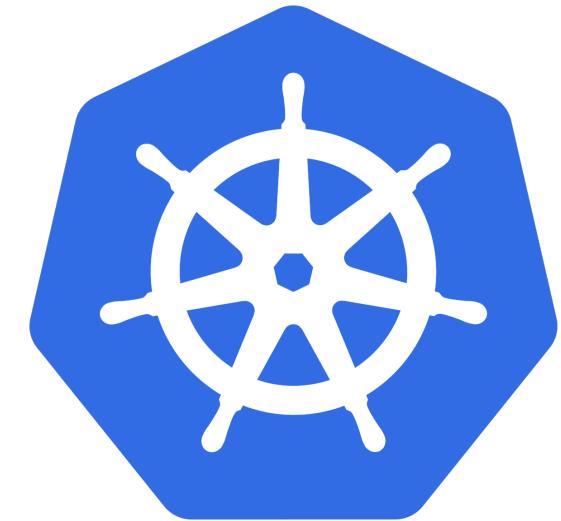


Ingress CPR Manifest

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-cpr
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - http:
      paths:
      - path: /app1/
        backend:
          serviceName: app1-nginx-clusterip-service
          servicePort: 80
      - path: /app2/
        backend:
          serviceName: app2-nginx-clusterip-service
          servicePort: 80
      - path: /
        backend:
          serviceName: usermgmt-webapp-clusterip-service
          servicePort: 80
```



Azure AKS & Azure DNS Zones



Domain Registrar

- A **domain registrar** is a company who can provide **Internet domain names**.
 - Example: GoDaddy, Wix, Namcheap, AWS Route53
- They **verify** if the Internet domain you want to use is available and allow you to **purchase it**.
- Once the domain name is **registered**, you are the **legal owner** for the domain name.
- If you already have an Internet domain, you will use the current **domain registrar to delegate to Azure DNS**.

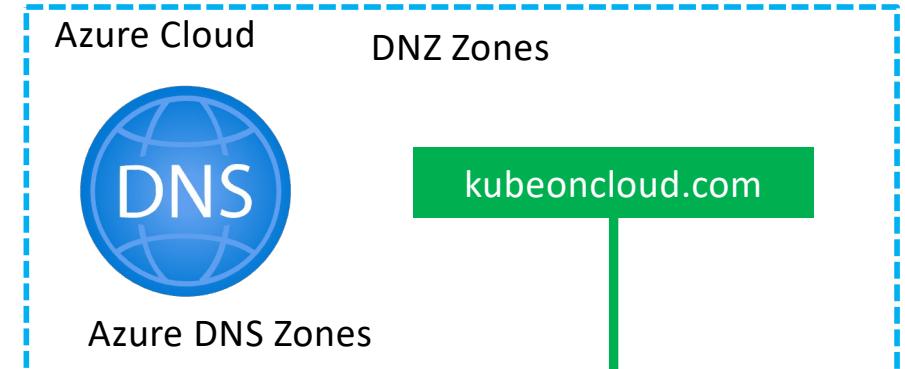
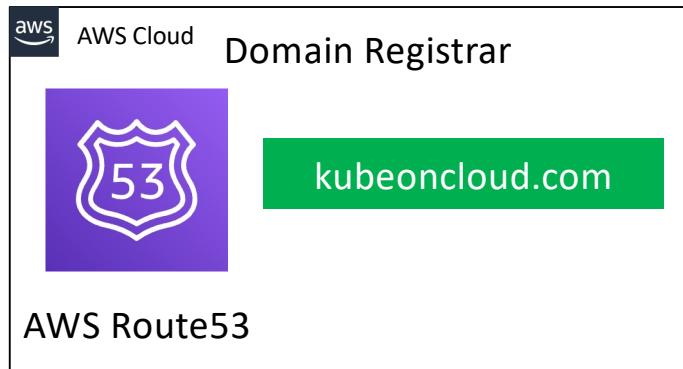
DNS Zone

- A domain is a **unique name** in the Domain Name System, for example stacksimplify.com.
- A **DNS zone** is used to host the **DNS records** for a particular domain.
- For example, the domain stacksimplify.com may contain several DNS records such as
 - mail.stack simplify.com (for a mail server)
 - www.stack simplify.com (for a website).
 - courses.stack simplify.com (Sub domains to host other applications)

Azure DNS Zones

- Azure DNS is **not the domain registrar**.
- Azure DNS allows you to host a **DNS zone** and manage the DNS records for a domain in Azure.
- For **DNS queries** for a domain to reach **Azure DNS**, the domain must be **delegated to Azure DNS** from the parent domain

Delegate Domain to Azure DNS



```
Kalyans-MacBook-Pro:/ kdaida$ nslookup -type=NS kubeoncloud.com
Server:          192.168.0.1
Address:        192.168.0.1#53

Non-authoritative answer:
kubeoncloud.com nameserver = ns3-04.azure-dns.org.
kubeoncloud.com nameserver = ns1-04.azure-dns.com.
kubeoncloud.com nameserver = ns4-04.azure-dns.info.
kubeoncloud.com nameserver = ns2-04.azure-dns.net.

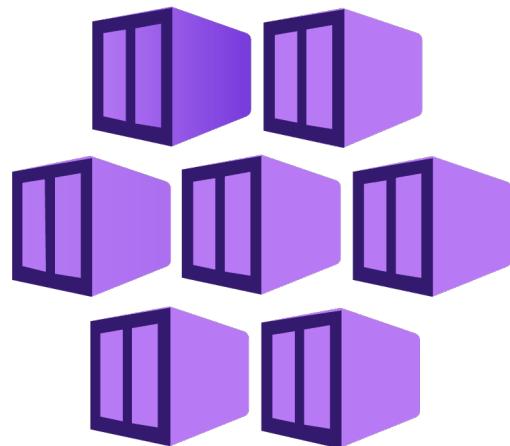
Authoritative answers can be found from:
```



DNS
Zones



Managed
Service
Identity



Azure AKS

Nginx Ingress Service

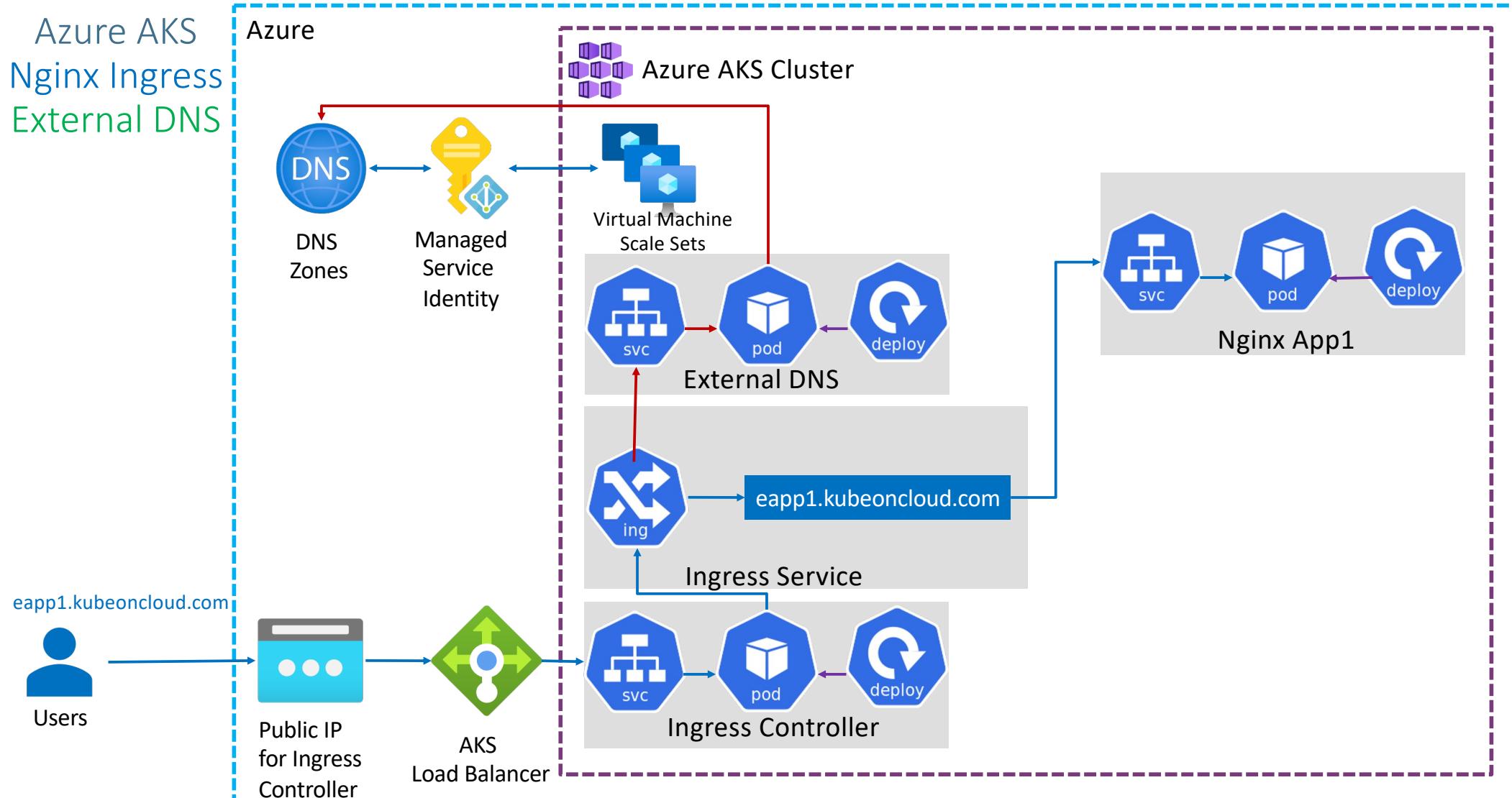
External DNS



Kubernetes External DNS

- ExternalDNS [synchronizes](#) exposed Kubernetes Services and Ingresses with [DNS providers](#).
- In simple terms, ExternalDNS allows you to [control DNS records dynamically via Kubernetes resources](#) in a DNS provider-agnostic way.
- Reference: <https://github.com/kubernetes-sigs/external-dns>

Azure AKS Nginx Ingress External DNS



Ingress with External DNS

This will help us to add DNS Record Set in Azure DNS Zones using k8s External DNS

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: nginxapp1-ingress-service
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
  - host: eapp1.kubeoncloud.com
    http:
      paths:
      - path: /
        backend:
          serviceName: app1-nginx-clusterip-service
          servicePort: 80
```

External DNS Updates

66 lines - 4 Removals

```
1 apiVersion: v1
2 kind: ServiceAccount
3 metadata:
4   name: external-dns
5 ---
6 apiVersion: rbac.authorization.k8s.io/v1beta1
7 kind: ClusterRole
8 metadata:
9   name: external-dns
10 rules:
11 - apiGroups: [""]
12 resources: ["services","endpoints","pods"]
13 verbs: ["get","watch","list"]
14 - apiGroups: ["extensions","networking.k8s.io"]
15   resources: ["ingresses"]
16   verbs: ["get","watch","list"]
17 - apiGroups: [""]
18   resources: ["nodes"]
19   verbs: ["list"]
20 ---
21 apiVersion: rbac.authorization.k8s.io/v1beta1
22 kind: ClusterRoleBinding
23 metadata:
24   name: external-dns-viewer
25 roleRef:
26   apiGroup: rbac.authorization.k8s.io
27   kind: ClusterRole
28   name: external-dns
29 subjects:
30 - kind: ServiceAccount
31   name: external-dns
32   namespace: default
33 ---
```

Copy all

66 lines + 4 Additions

```
1 apiVersion: v1
2 kind: ServiceAccount
3 metadata:
4   name: external-dns
5 ---
6 apiVersion: rbac.authorization.k8s.io/v1
7 kind: ClusterRole
8 metadata:
9   name: external-dns
10 rules:
11 - apiGroups: [""]
12 resources: ["services","endpoints","pods", "nodes"]
13 verbs: ["get","watch","list"]
14 - apiGroups: ["extensions","networking.k8s.io"]
15   resources: ["ingresses"]
16   verbs: ["get","watch","list"]
17 - apiGroups: [""]
18   resources: ["nodes"]
19   verbs: ["list"]
20 ---
21 apiVersion: rbac.authorization.k8s.io/v1
22 kind: ClusterRoleBinding
23 metadata:
24   name: external-dns-viewer
25 roleRef:
26   apiGroup: rbac.authorization.k8s.io
27   kind: ClusterRole
28   name: external-dns
29 subjects:
30 - kind: ServiceAccount
31   name: external-dns
32   namespace: default
33 ---
```

Copy all

External DNS Updates

```
33 ---  
34 apiVersion: apps/v1  
35 kind: Deployment  
36 metadata:  
37   name: external-dns  
38 spec:  
39   strategy:  
40     type: Recreate  
41   selector:  
42     matchLabels:  
43       app: external-dns  
44   template:  
45     metadata:  
46       labels:  
47         app: external-dns  
48   spec:  
49     serviceAccountName: external-dns  
50   containers:  
51     - name: external-dns  
52       image: registry.opensource.zalan.do/teapot/external-dns:latest  
53     args:  
54       - --source=service  
55       - --source=ingress  
56       #- --domain-filter=example.com # (optional) limit to only example.com domains; change to match the zone created above.  
57       - --provider=azure  
58       #- --azure-resource-group=externaldns # (optional) use the DNS zones from the specific resource group  
59       volumeMounts:  
60         - name: azure-config-file  
61           mountPath: /etc/kubernetes  
62           readOnly: true  
63       volumes:  
64         - name: azure-config-file  
65           secret:  
66             secretName: azure-config-file  
  
33 ---  
34 apiVersion: apps/v1  
35 kind: Deployment  
36 metadata:  
37   name: external-dns  
38 spec:  
39   strategy:  
40     type: Recreate  
41   selector:  
42     matchLabels:  
43       app: external-dns  
44   template:  
45     metadata:  
46       labels:  
47         app: external-dns  
48   spec:  
49     serviceAccountName: external-dns  
50   containers:  
51     - name: external-dns  
52       image: k8s.gcr.io/external-dns/external-dns:v0.11.0  
53     args:  
54       - --source=service  
55       - --source=ingress  
56       #- --domain-filter=example.com # (optional) limit to only example.com domains; change to match the zone created above.  
57       - --provider=azure  
58       #- --azure-resource-group=externaldns # (optional) use the DNS zones from the specific resource group  
59       volumeMounts:  
60         - name: azure-config-file  
61           mountPath: /etc/kubernetes  
62           readOnly: true  
63       volumes:  
64         - name: azure-config-file  
65           secret:  
66             secretName: azure-config-file
```

Ingress Service Updates

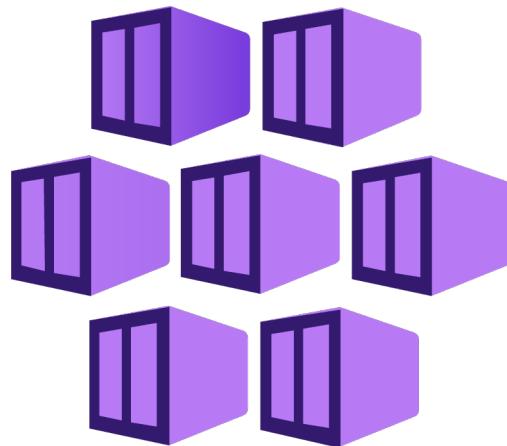
```
azure-aks-kubernetes-masterclass > 12-ExternalDNS-for-AzureDNS-on-AKS > kube-manifests > 02-NginxApp1 > ! 03-Ingress-with-ExternalDNS.yaml
1 apiVersion: networking.k8s.io/v1
2 kind: Ingress
3 metadata:
4   name: nginxapp1-ingress-service
5   annotations:
6     kubernetes.io/ingress.class: "nginx"
7 spec:
8   rules:
9     - host: eapp1.kubeoncloud.com
10       http:
11         paths:
12           - path: /
13             pathType: Prefix
14             backend:
15               service:
16                 name: app1-nginx-clusterip-service
17                 port:
18                   number: 80
```



DNS
Zones



Managed
Service
Identity



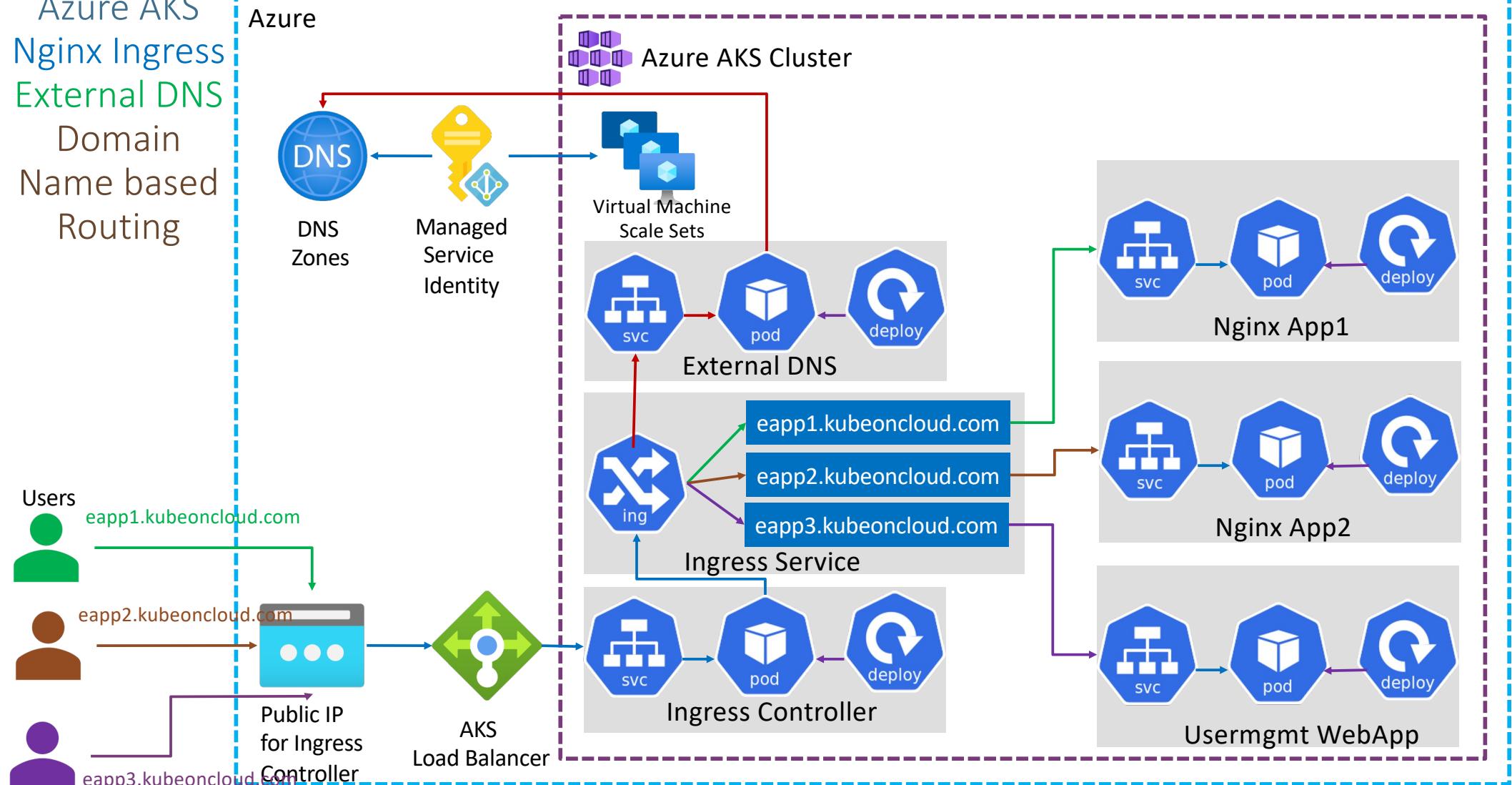
Azure AKS

Nginx Ingress Service

Domain Name based Routing

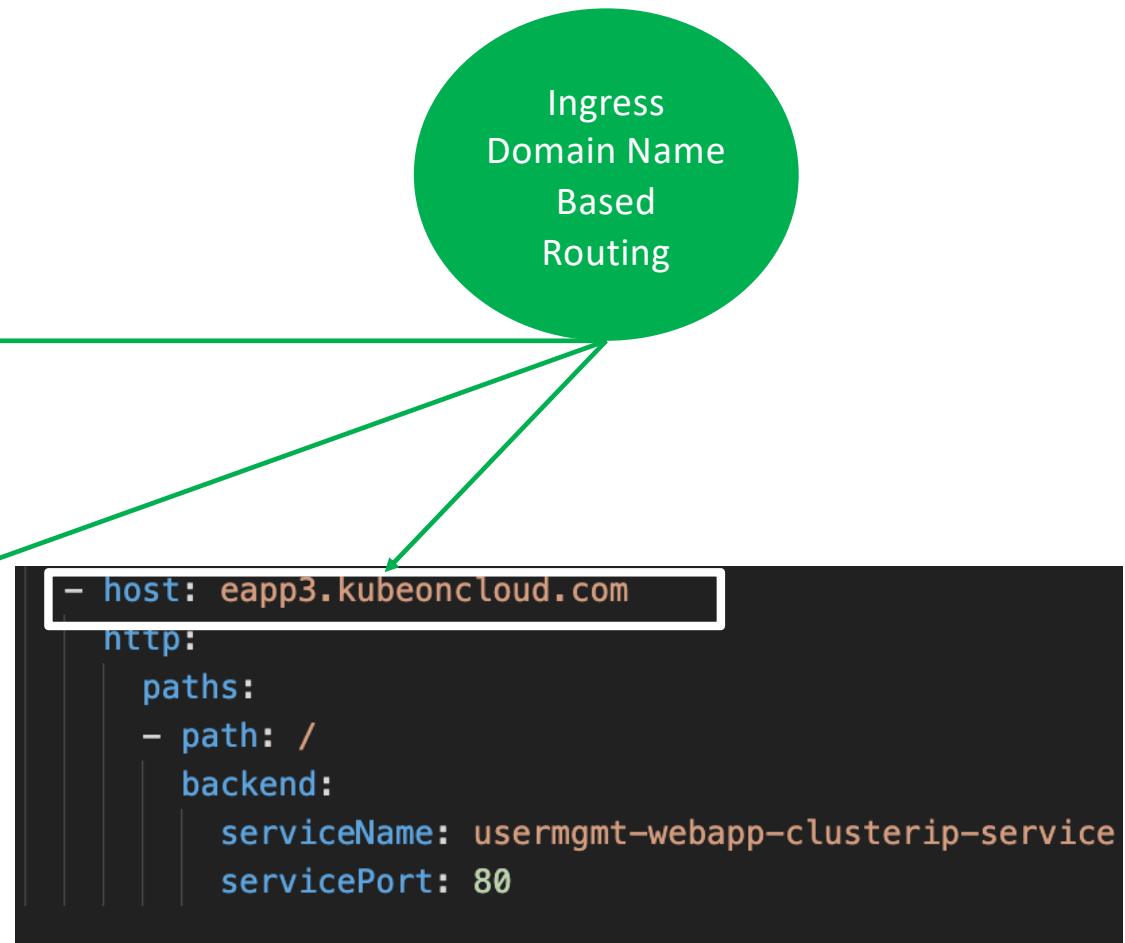


Azure AKS
Nginx Ingress
External DNS
Domain Name based Routing



Ingress with External DNS and Domain Name based Routing

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-app1-app2-app3
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
    - host: eapp1.kubeoncloud.com
      http:
        paths:
          - path: /
            backend:
              serviceName: app1-nginx-clusterip-service
              servicePort: 80
    - host: eapp2.kubeoncloud.com
      http:
        paths:
          - path: /
            backend:
              serviceName: app2-nginx-clusterip-service
              servicePort: 80
```

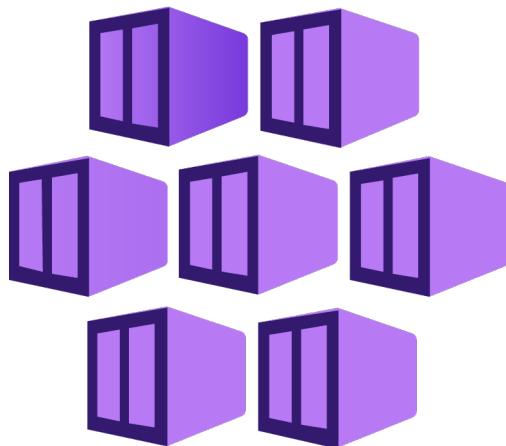




DNS
Zones



Managed
Service
Identity



Azure AKS

Nginx Ingress Service

Ingress SSL

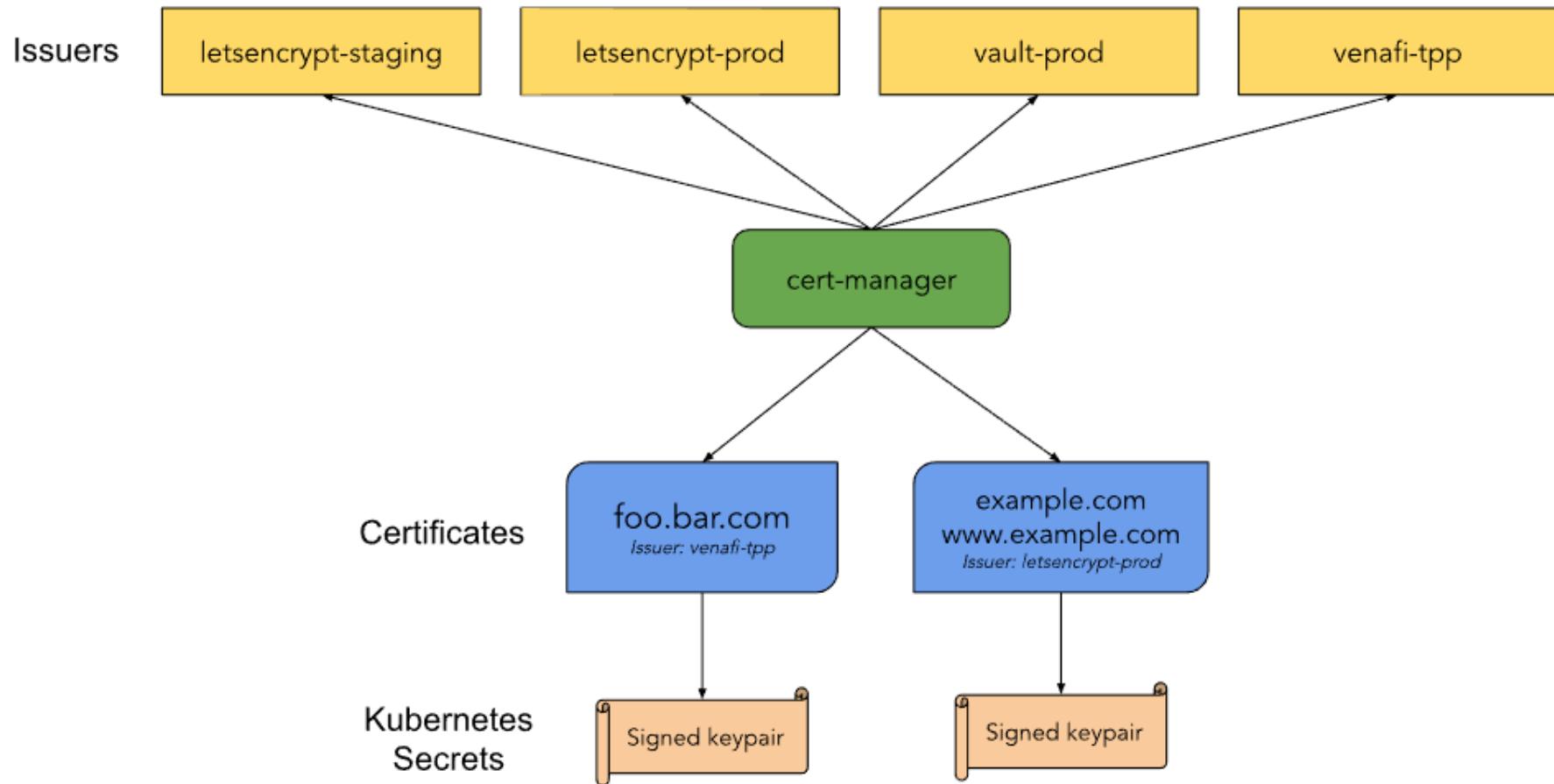
Automatically generate and renew ssl
certificates



Kubernetes Cert Manager

- cert-manager is a **native** Kubernetes certificate management controller.
- It can help with issuing certificates from a variety of sources, such as [Let's Encrypt](#), [HashiCorp Vault](#), [Venafi](#), [a simple signing key pair](#), or [self signed](#).
- It will ensure certificates [are valid and up to date](#), and attempt to [renew certificates](#) at a configured time before expiry.
- Reference: <https://cert-manager.io/docs/>

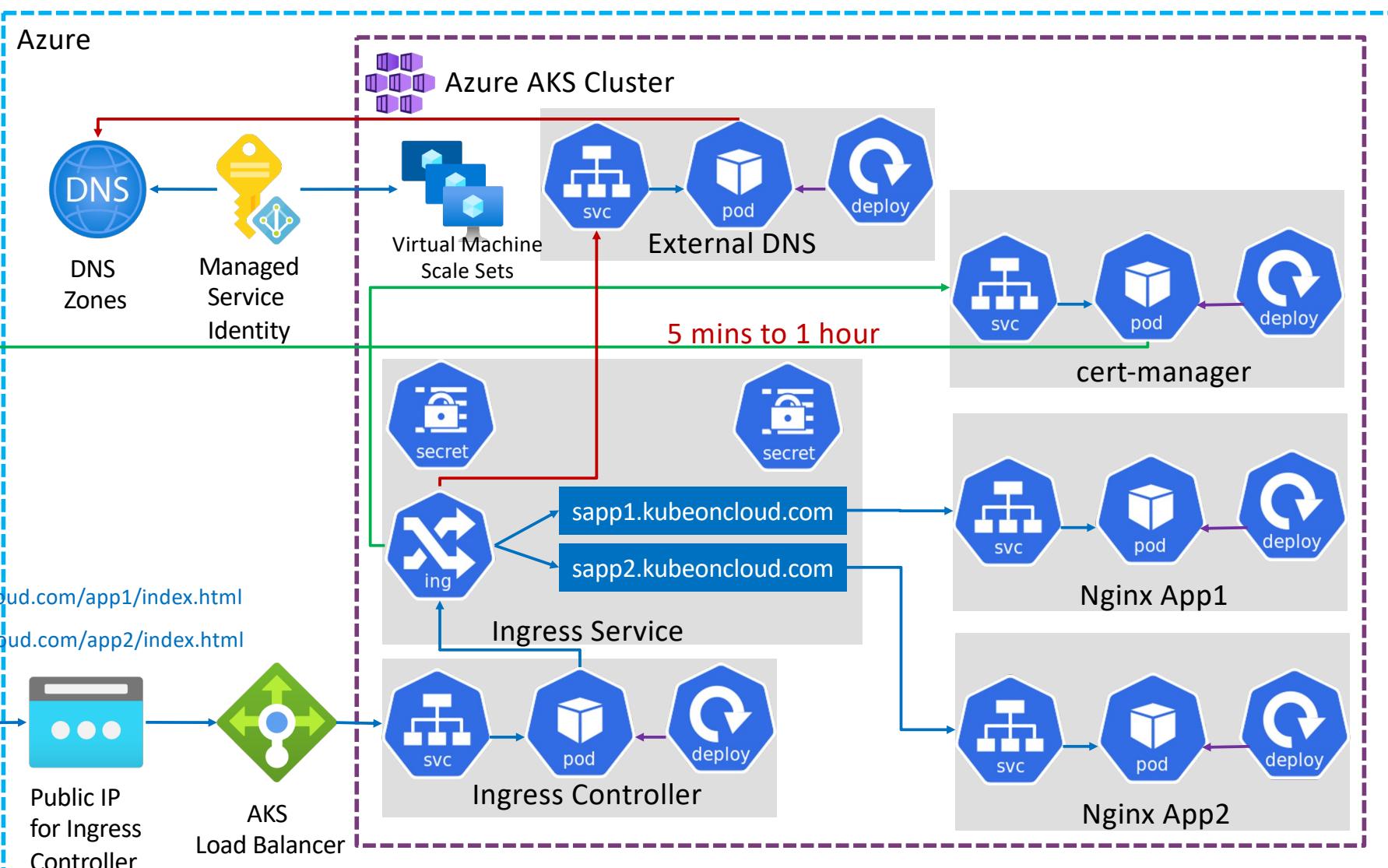
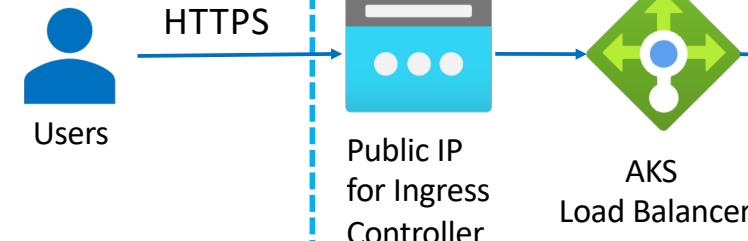
Kubernetes cert-manager



Azure AKS
Nginx Ingress
External DNS
SSL with
Lets Encrypt



<https://sapp1.kubeoncloud.com/app1/index.html>
<https://sapp2.kubeoncloud.com/app2/index.html>



Ingress with External DNS, DNR and SSL

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-ssl
  annotations:
    kubernetes.io/ingress.class: "nginx"
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  rules:
  - host: sapp1.kubeoncloud.com
    http:
      paths:
      - path: /
        backend:
          serviceName: app1-nginx-clusterip-service
          servicePort: 80
```

```
- host: sapp2.kubeoncloud.com
  http:
    paths:
    - path: /
      backend:
        serviceName: app2-nginx-clusterip-service
        servicePort: 80
  tls:
  - hosts:
    - sapp1.kubeoncloud.com
    secretName: sapp1-kubeoncloud-secret
  - hosts:
    - sapp2.kubeoncloud.com
    secretName: sapp2-kubeoncloud-secret
```

Ingress SSL Settings

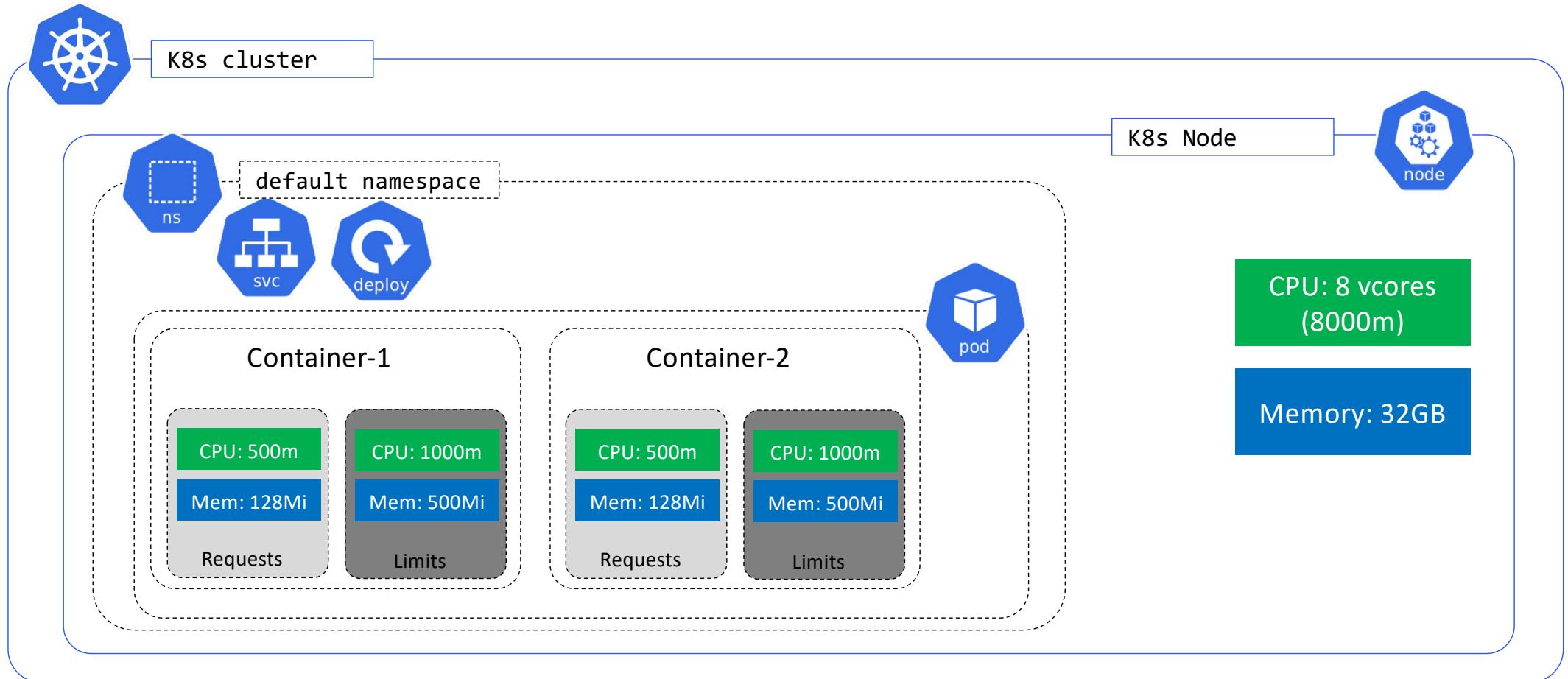
Kubernetes Resources Requests & Limits



Kubernetes - Resources Requests & Limits

- We can specify how much each container in a pod needs the resources like **CPU & Memory**.
- When we provide this information in our **pod definition for a specific container**, the scheduler uses this information to decide **which node to place the Pod on** based on availability of k8s worker Node CPU and Memory Resources.
- When you specify a resource **limit** for a Container, **the kubelet enforces those 'limits'** so that the running container is not allowed to use more of that resource than the **limit** you set.
- The **kubelet also reserves** at least the **request** amount of that system resource specifically for that container to use.

Kubernetes – Requests & Limits



Kubernetes

Requests & Limits

We have defined Requests & Limits for a specific container in a Deployment Manifest.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app1-nginx-deployment
  labels:
    app: app1-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: app1-nginx
  template:
    metadata:
      labels:
        app: app1-nginx
    spec:
      containers:
        - name: app1-nginx
          image: stackimplify/kube-nginxapp1:1.0.0
          imagePullPolicy: Always
          ports:
            - containerPort: 80
# Requests & Limits for usermgmt-webapp Container
resources:
  requests:
    cpu: "100m"
    memory: "128Mi"
  limits:
    cpu: "200m"
    memory: "256Mi"
```

Kubernetes Namespaces



Kubernetes - Namespaces

- Namespaces are also called **Virtual clusters** in our **physical** k8s cluster

- We use this in environments where we have **many users spread** across multiple teams or projects

- Clusters with **tens of users** ideally don't need to use namespaces

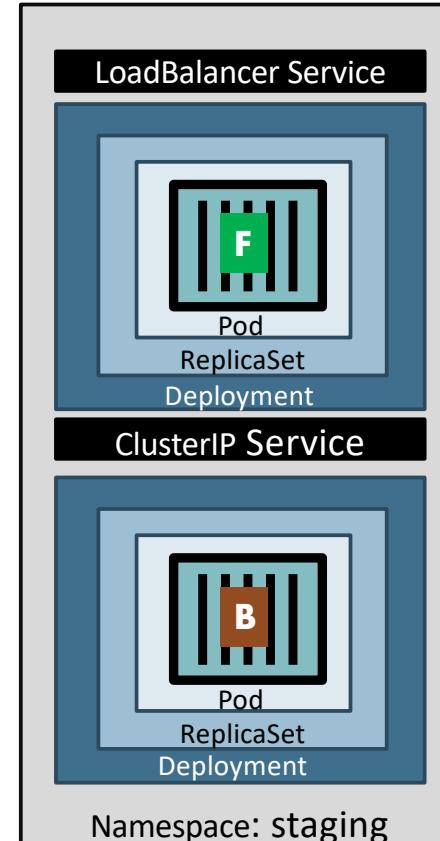
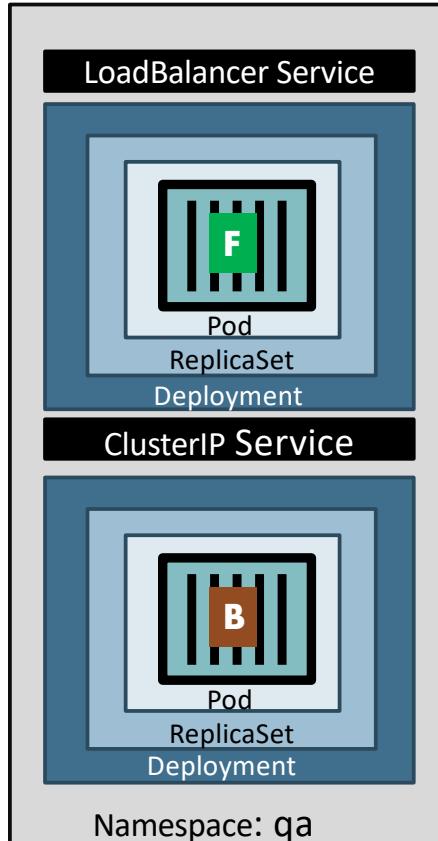
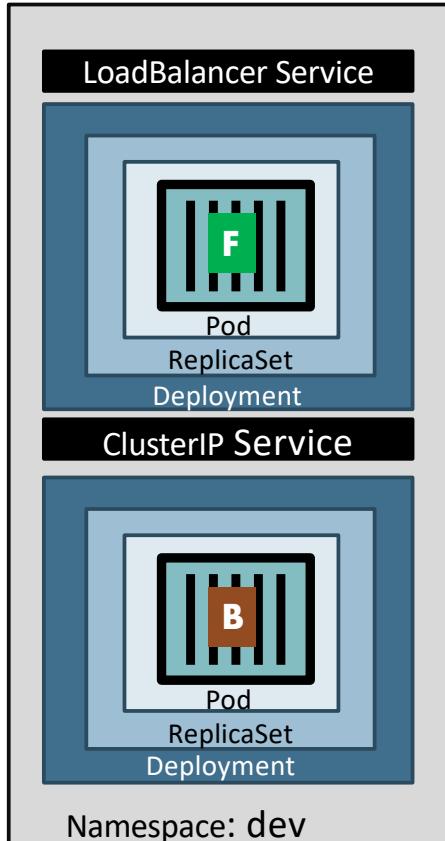
- Benefits

- Creates **isolation boundary** from other k8s objects
- We can limit the resources like **CPU, Memory** on per namespace basis (**Resource Quota**).

```
kubectl get namespace
```

NAME	STATUS	AGE
default	Active	141m
kube-node-lease	Active	141m
kube-public	Active	141m
kube-system	Active	141m

Kubernetes - Namespaces



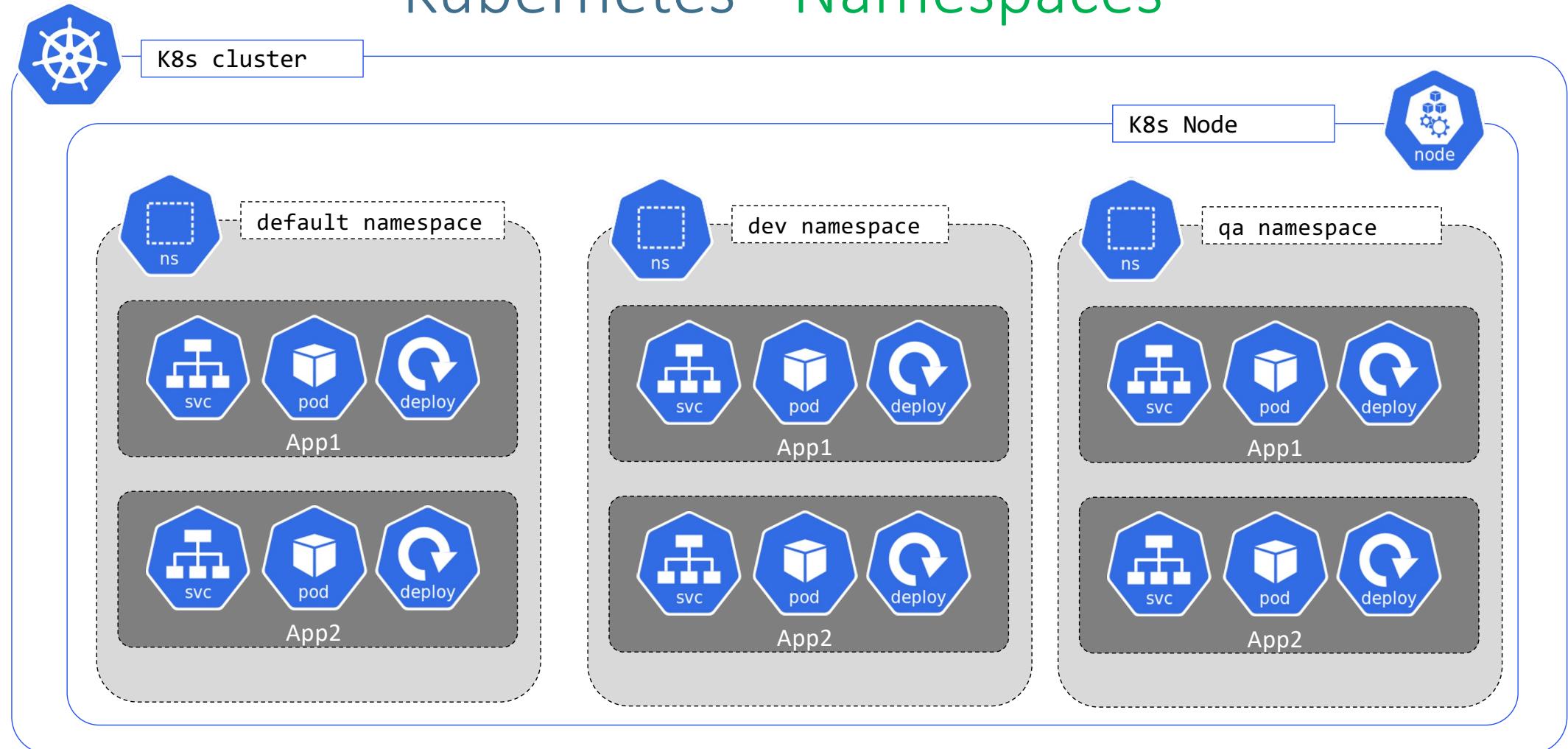
Namespace Manifest - Declarative

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4   name: dev3
```

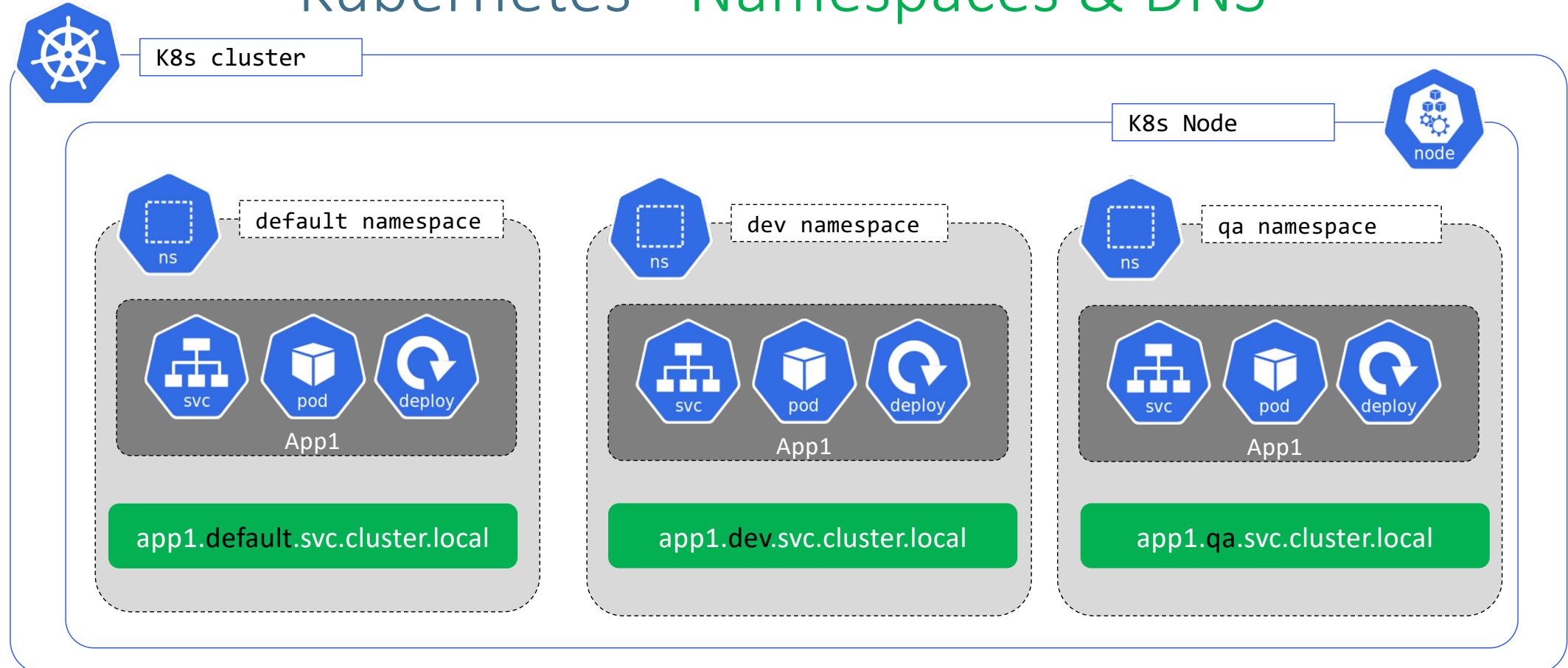
Namespace Manifest - Imperative

```
kubectl create namespace dev1
```

Kubernetes - Namespaces



Kubernetes - Namespaces & DNS



<service-name>.<namespace-name>.svc.cluster.local

Kubernetes Namespaces Limit Range



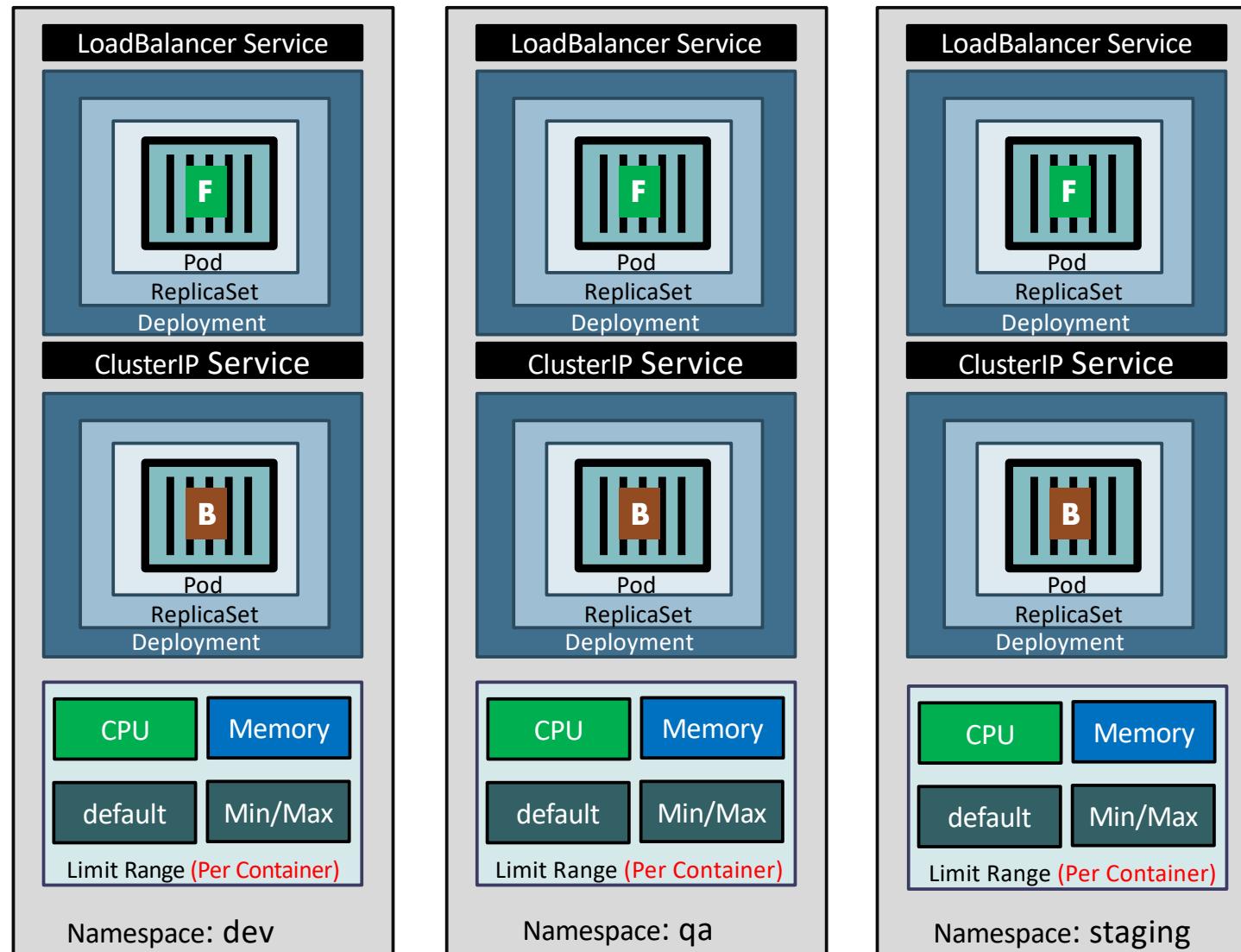
Namespaces – Limit Range

- By default, containers run with **unbounded compute resources** on a Kubernetes cluster.
- With resource quotas, **cluster administrators can restrict** resource consumption and creation on a namespace basis.
- Within a namespace, a Pod or Container can consume as much CPU and memory as defined by the **namespace's resource quota**.
- There is a **concern that** one Pod or Container could monopolize all available resources.
- A **LimitRange is a policy** to constrain resource allocations (to Pods or Containers) in a namespace.

Namespaces – Limit Range

- A *LimitRange* provides constraints that can:
- Enforce **minimum and maximum compute resources** usage per Pod or Container in a namespace.
- Enforce **minimum and maximum storage request** per **PersistentVolumeClaim** in a namespace.
- Enforce a **ratio between request and limit** for a resource in a namespace.
- Set **default request/limit** for compute resources in a namespace and automatically **inject** them to Containers at runtime.

Limit Range



Limit Range Manifest

```
apiVersion: v1
kind: LimitRange
metadata:
  name: default-cpu-mem-limit-range
  namespace: dev3
spec:
  limits:
    - default:
        memory: "512Mi"
        cpu: "500m"
  defaultRequest:
    memory: "256Mi"
    cpu: "300m"
  type: Container
```

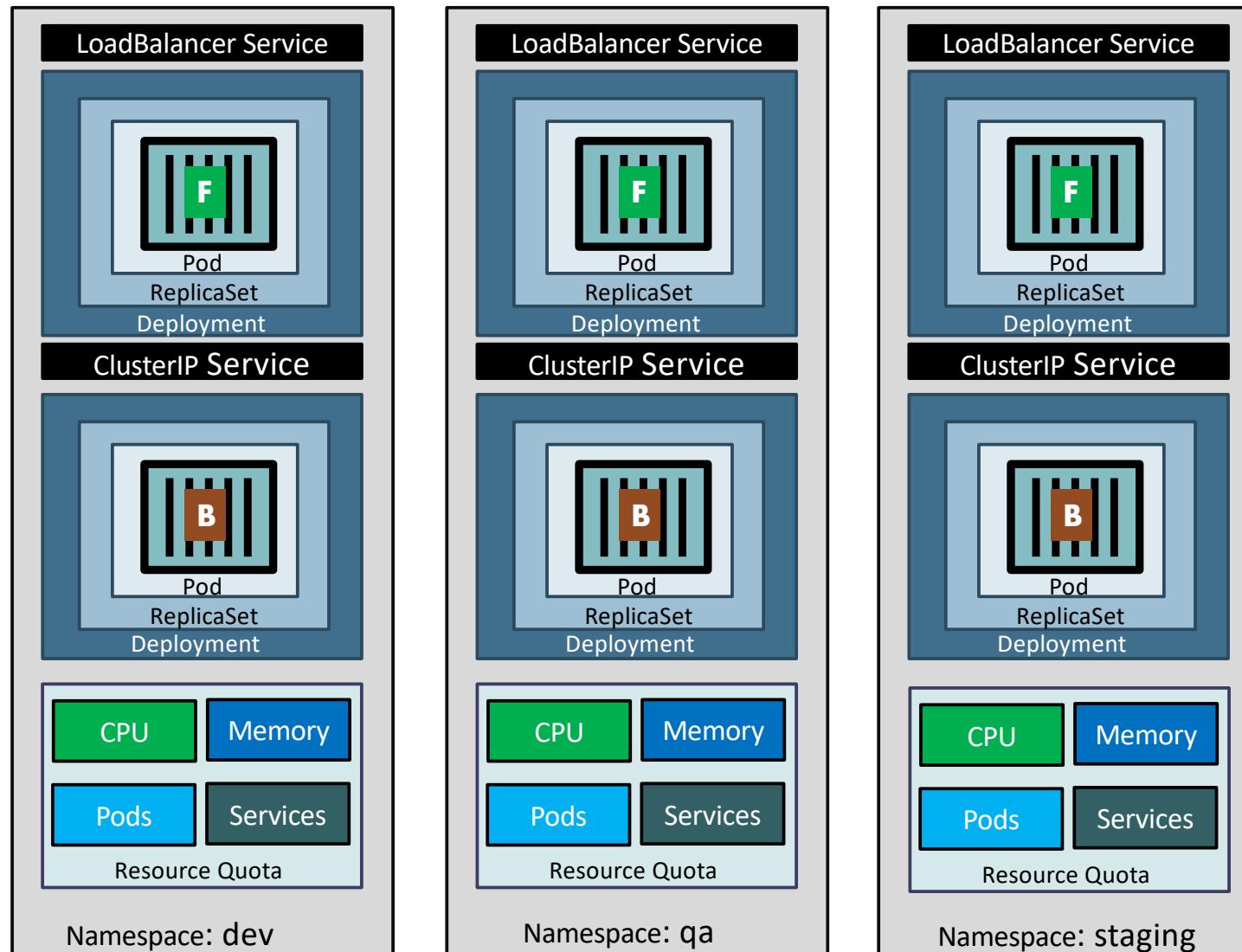
Kubernetes Namespaces Resource Quota



Namespaces - Resource Quota

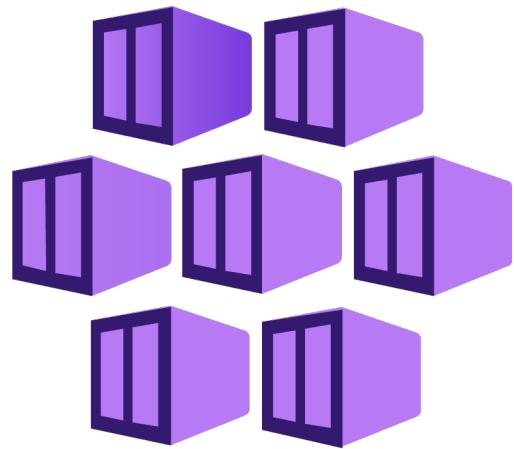
- When several users or teams **share a cluster with a fixed number of nodes**, there is a concern that one team could use more than its fair share of resources.
- Resource quotas are a tool for administrators to **address this concern**.
- A resource quota, defined by a ResourceQuota object, provides constraints that **limit aggregate resource consumption per namespace**.
- It can limit the **quantity of objects** that can be created in a namespace **by type**, as well as the **total amount of compute resources** that may be consumed by resources in that project.

Resource Quota



Resource Quota Manifest

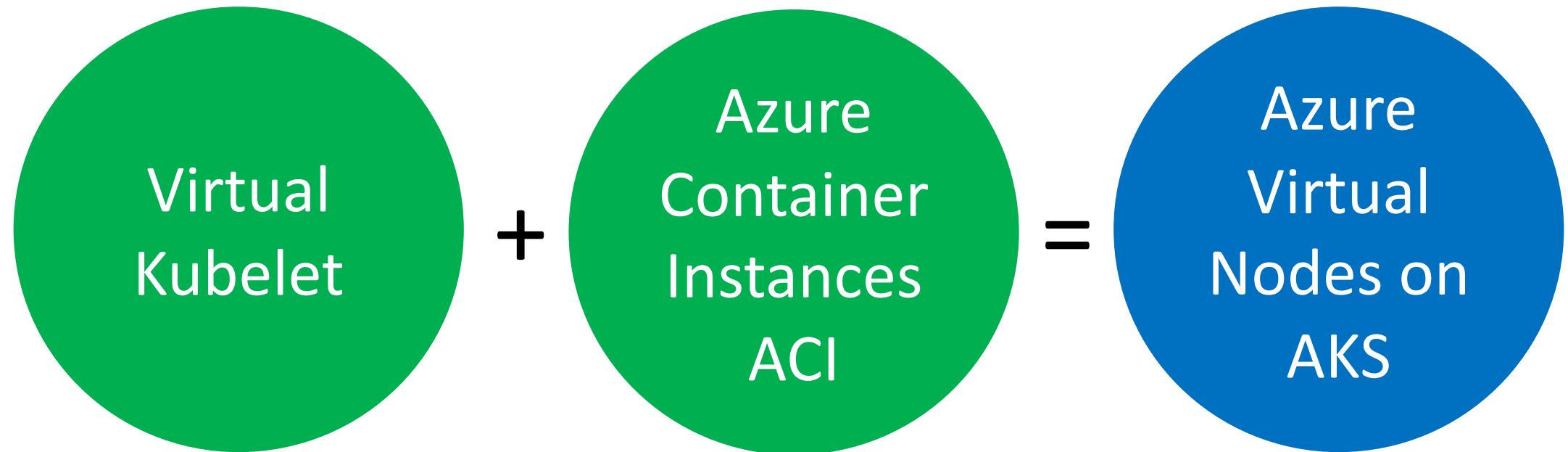
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: ns-resource-quota
  namespace: dev3
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
    pods: "5"
    configmaps: "5"
    persistentvolumeclaims: "5"
    secrets: "5"
    services: "5"
```



Azure AKS Virtual Nodes (Serverless)



Azure AKS Virtual Nodes



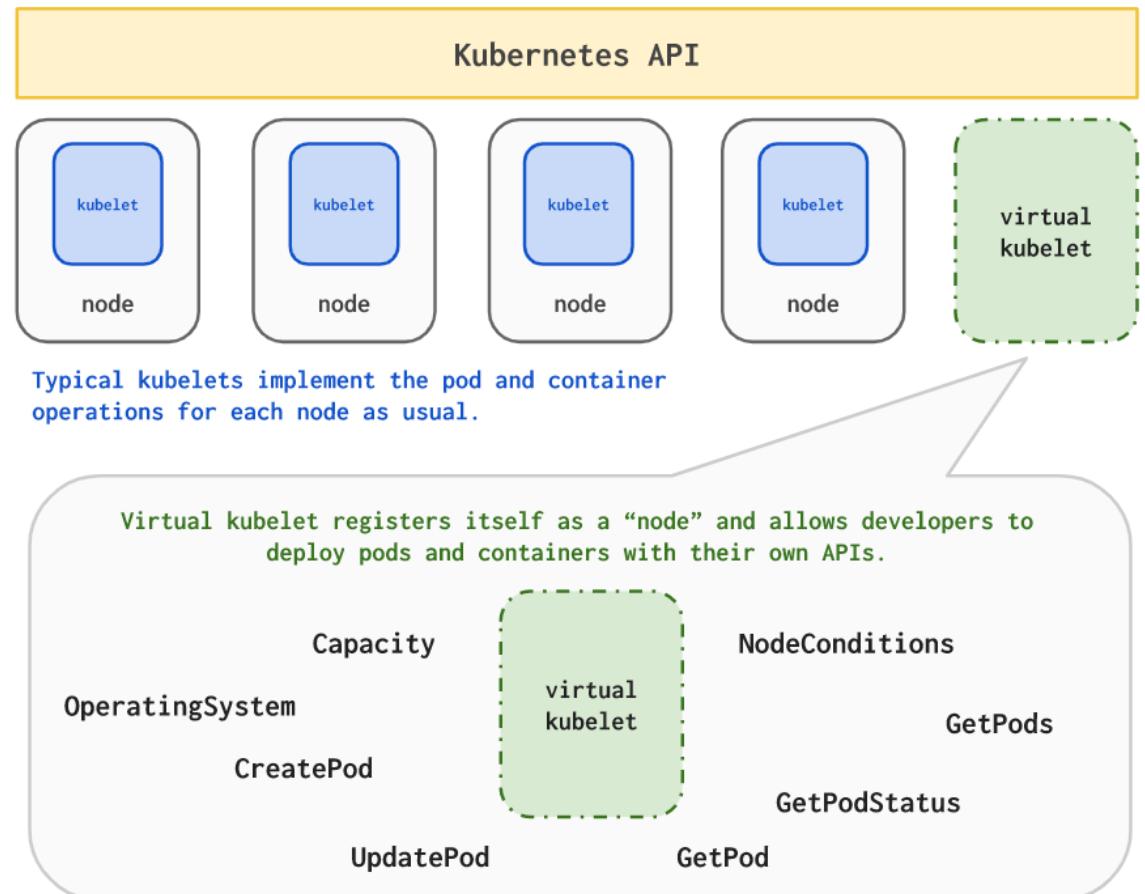
What is Virtual Kubelet?

- Virtual Kubelet is an open source [Kubernetes kubelet implementation](#) that acts as a kubelet for the purposes of connecting Kubernetes to other APIs.
- This allows the [k8s worker nodes](#) to be backed by other services like [Azure ACI](#) and [AWS Fargate](#)
- The primary scenario for VK is enabling the extension of the Kubernetes API into [serverless container platforms](#) like Azure ACI and AWS Fargate

How Virtual-Kubelet works?

Virtual
Kubelet

- Current Features
- create, delete and update pods
- container logs, exec, and metrics
- get pod, pods and pod status
- capacity
- node addresses, node capacity, node daemon endpoints
- operating system
- bring your own virtual network



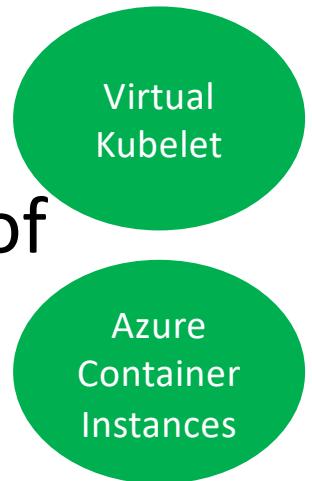
What is ACI (Azure Container Instances)?

- Azure Container Instances (ACI) provide a **hosted environment** for running containers in Azure.
- When using ACI, there is **no need to manage** the **underlying compute infrastructure**, Azure handles this management for you.
- When running containers in ACI, you are charged by **the second** for each running container



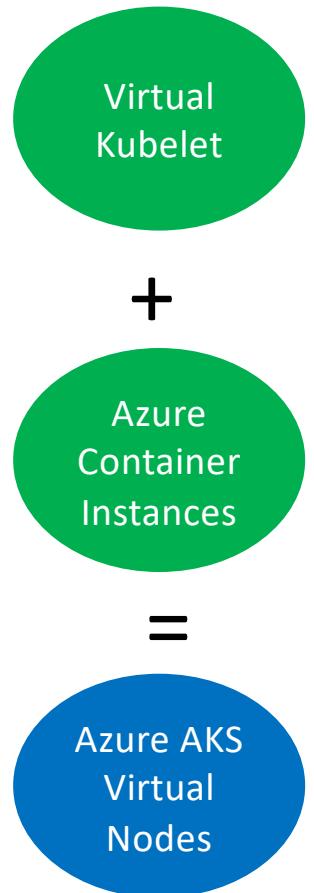
Kubernetes Virtual Kubelet with Azure ACI

- The Azure Container Instances (ACI) provider for the Virtual Kubelet configures an ACI instance as a node in any Kubernetes cluster.
- When using the Virtual Kubelet ACI provider, pods can be scheduled on an ACI instance as if the ACI instance is a standard Kubernetes node.
- This configuration allows you to take advantage of both the capabilities of Kubernetes and the management value and cost benefit of ACI.



Virtual Kubelet's ACI provider - Features

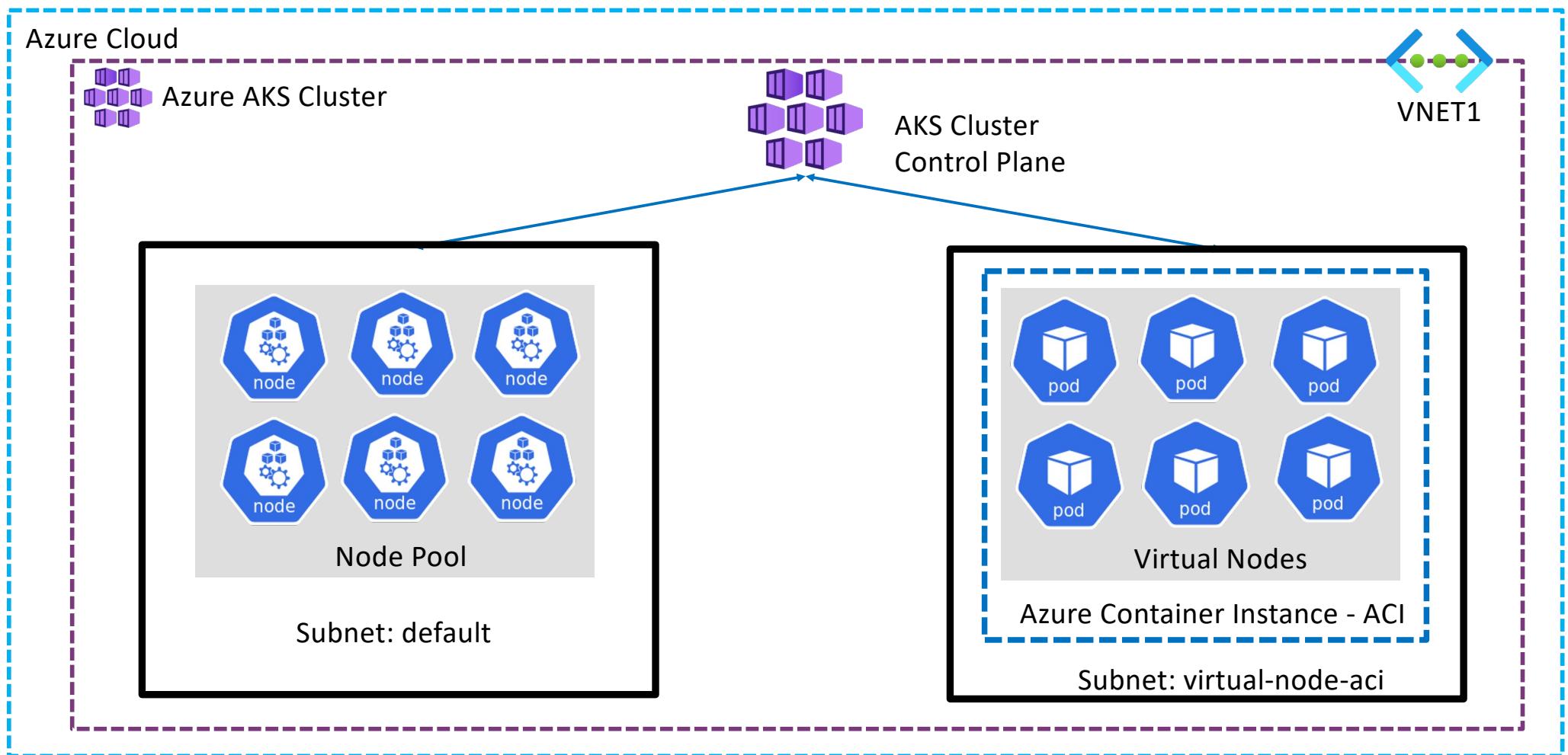
- **Volumes:** empty dir, github repo, Azure Files
- Secure env variables, config maps
- Bring your own **virtual network (VNet)**
- **Network security group support**
- Basic **Azure Networking** support within AKS virtual node
- **Exec support** for container instances
- Azure Monitor integration or formally known as **OMS**



Azure AKS – Virtual Nodes (Serverless)

- We can run our Kubernetes workloads on **Serverless Infrastructure** of Azure which is called **Virtual Nodes**
- **Advantages**
 - Scale our applications rapidly **without any limitations**
 - **Quick Provisioning** of pods using Virtual Nodes when compared to cluster autoscaler.
 - **Cluster Autoscaler** need to provision the Nodes in a managed node pool **first** then only **Kubernetes can schedule pods** on those newly provisioned nodes.
- **Limitations (Huge and Many)**
 - <https://docs.microsoft.com/en-us/azure/aks/virtual-nodes-cli#known-limitations>

Azure AKS - Virtual Nodes

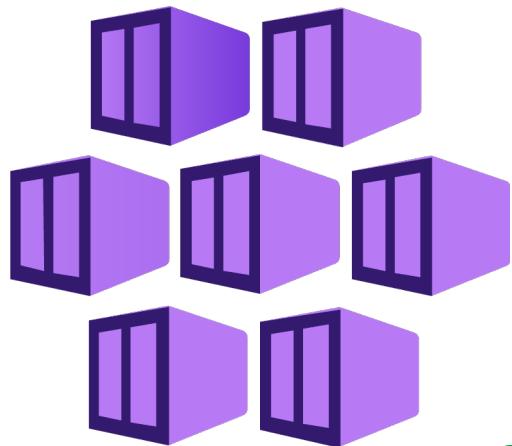


AKS Virtual Nodes

NodeSelector

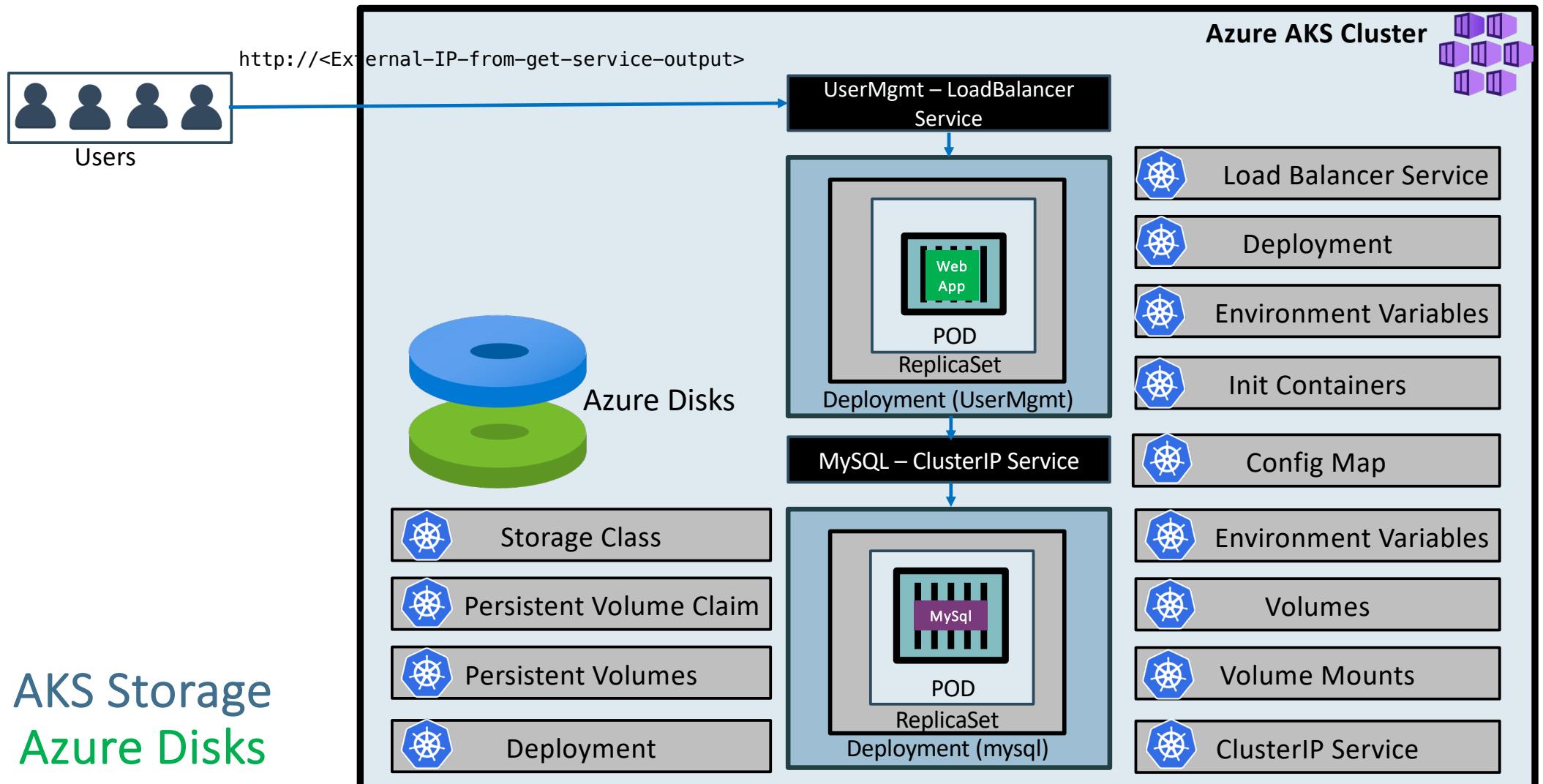
```
spec:  
  containers:  
    - name: app1-nginx  
      image: stacksmplify/kube-nginxapp1:1.0.0  
    ports:  
      - containerPort: 80  
# To schedule pods on Azure Virtual Nodes  
  nodeSelector:  
    kubernetes.io/role: agent  
    beta.kubernetes.io/os: linux  
    type: virtual-kubelet  
  tolerations:  
    - key: virtual-kubelet.io/provider  
      operator: Exists  
    - key: azure.com/aci  
      effect: NoSchedule
```

```
01-NginxApp1-Deployment.yml ✘  
githubcontent > azure-aks-kubernetes-masterclass > 17-Azure-Virtual-Nodes-for-AKS > kube-manifests > 01-NginxAp...  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: app1-nginx-deployment  
5    labels:  
6      app: app1-nginx  
7  spec:  
8    replicas: 2  
9    selector:  
10      matchLabels:  
11        app: app1-nginx  
12    template:  
13      metadata:  
14        labels:  
15          app: app1-nginx  
16      spec:  
17        containers:  
18          - name: app1-nginx  
19            image: stacksmplify/kube-nginxapp1:1.0.0  
20          ports:  
21            - containerPort: 80  
# To schedule pods on Azure Virtual Nodes  
22  nodeSelector:  
23    kubernetes.io/role: agent  
24    beta.kubernetes.io/os: linux  
25    type: virtual-kubelet  
26  tolerations:  
27    - key: virtual-kubelet.io/provider  
28      operator: Exists  
29    - key: azure.com/aci  
30      effect: NoSchedule  
31  
32
```

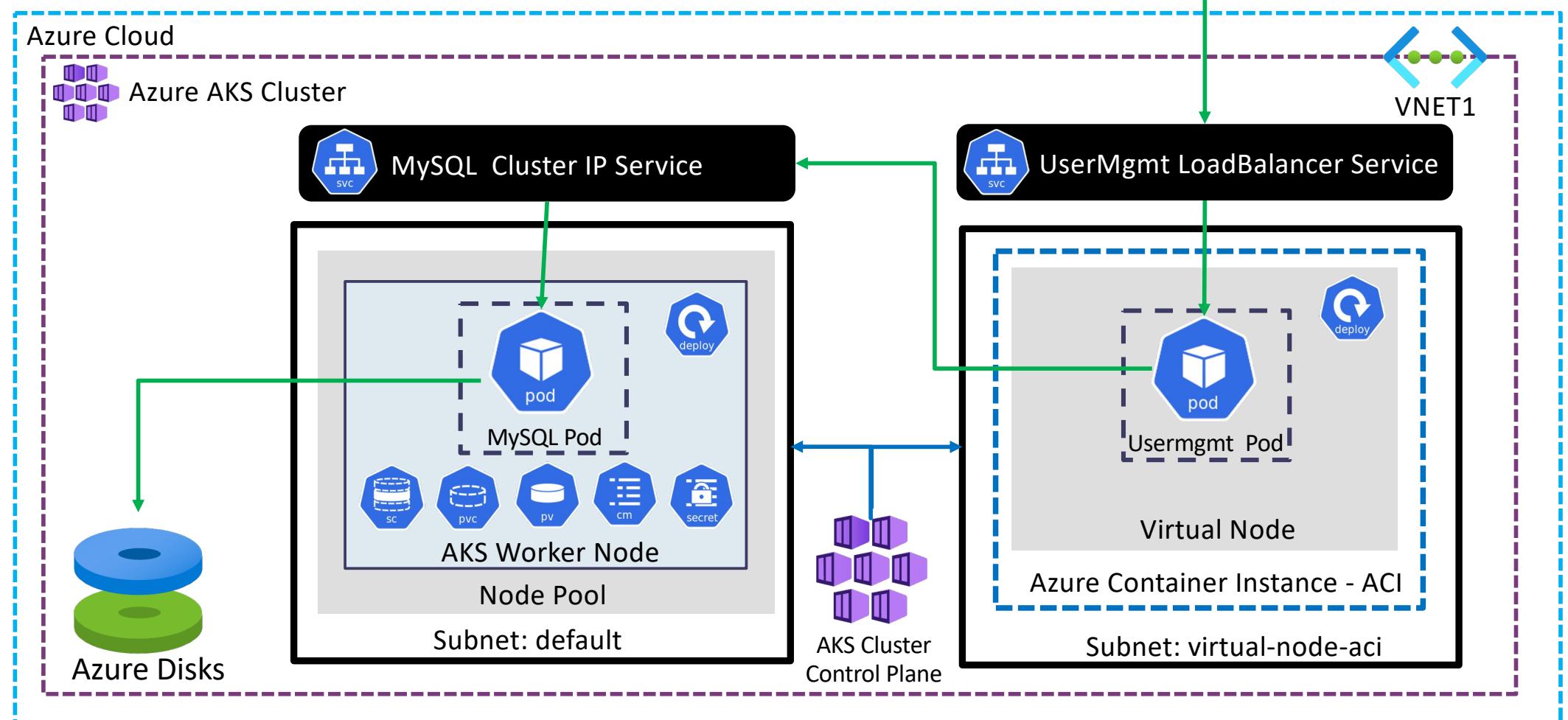


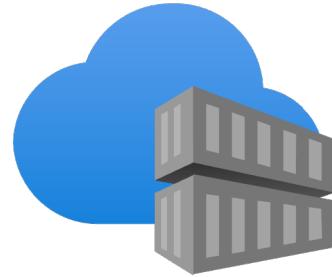
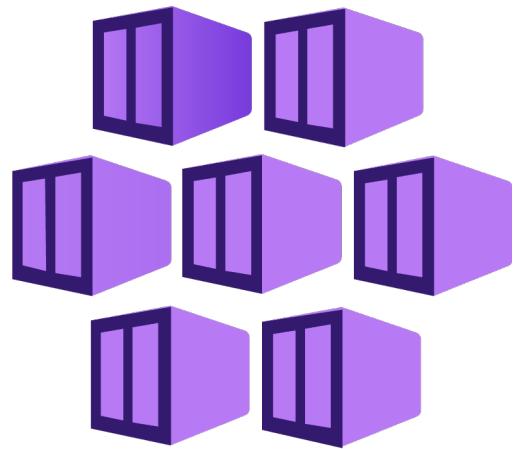
Azure AKS Virtual Nodes (Serverless) **Mixed Mode Deployments**





Azure AKS - Virtual Nodes Mixed Mode Deployments





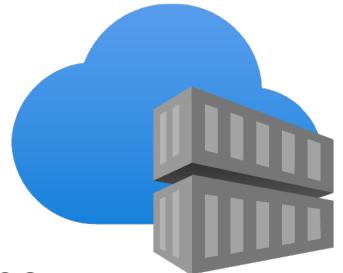
Azure AKS

Azure Container

Registry (ACR)



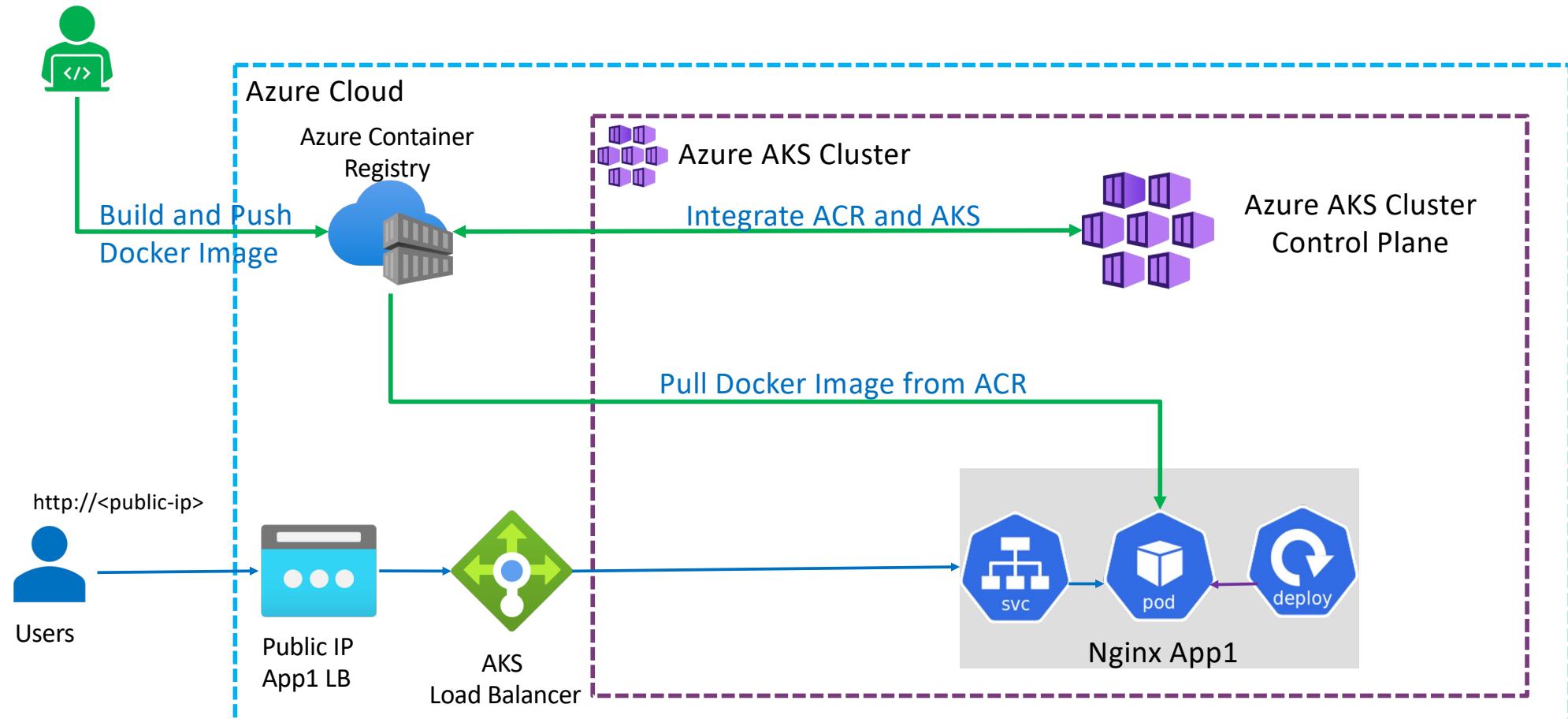
Azure Container Registry - Introduction



- Azure Container Registry is a [managed, private Docker registry](#) service
- We can [create and maintain](#) Azure container registries to store and manage our private Docker container images.
- With ACR, we can simplify our [container lifecycle management](#).
- [Geo-replication](#) to efficiently manage a single registry across multiple regions
- [Automated container building](#) and patching including base image updates and task scheduling
- [Integrated security](#) with Azure Active Directory (Azure AD) authentication, role-based access control, [Docker Content Trust](#) and virtual network integration

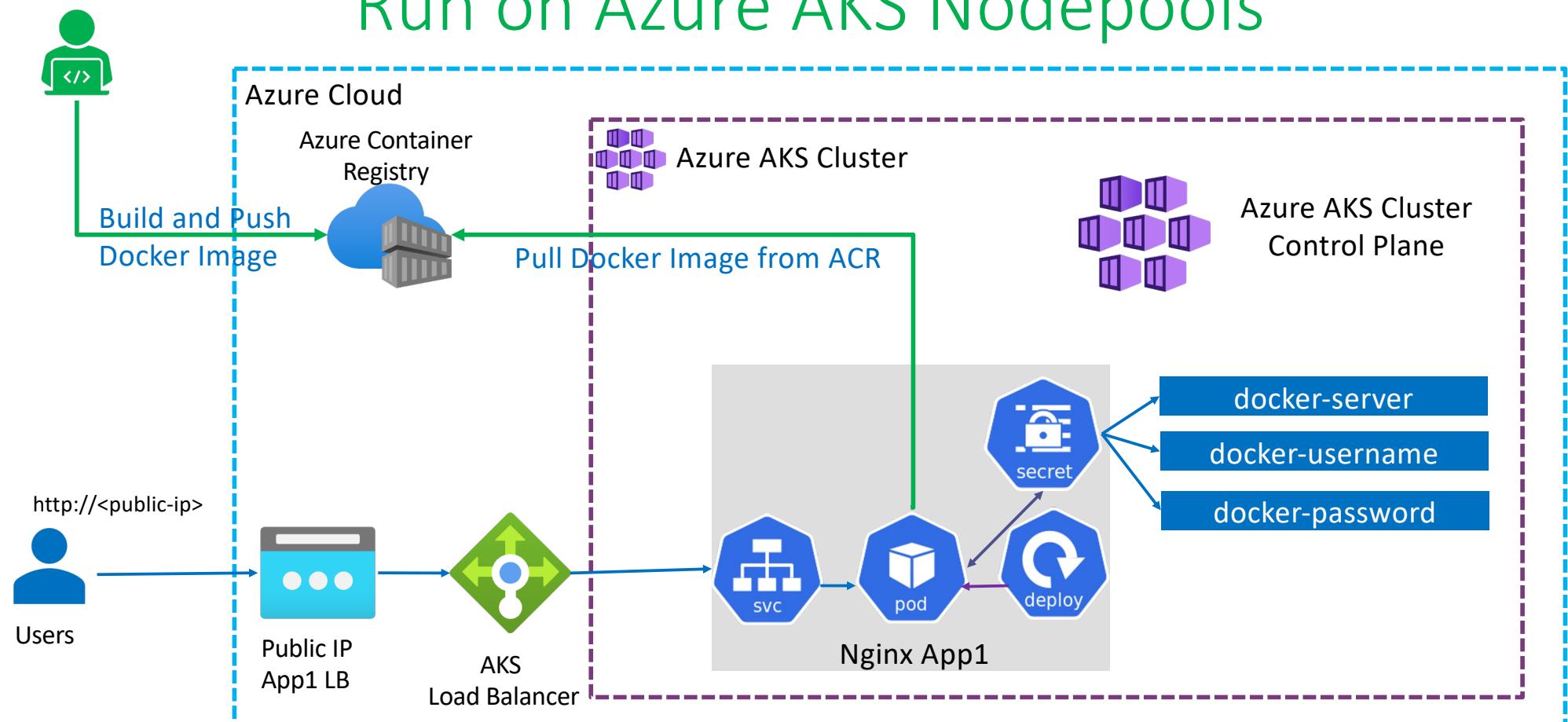
Azure ACR and AKS - Integration

Docker Developer



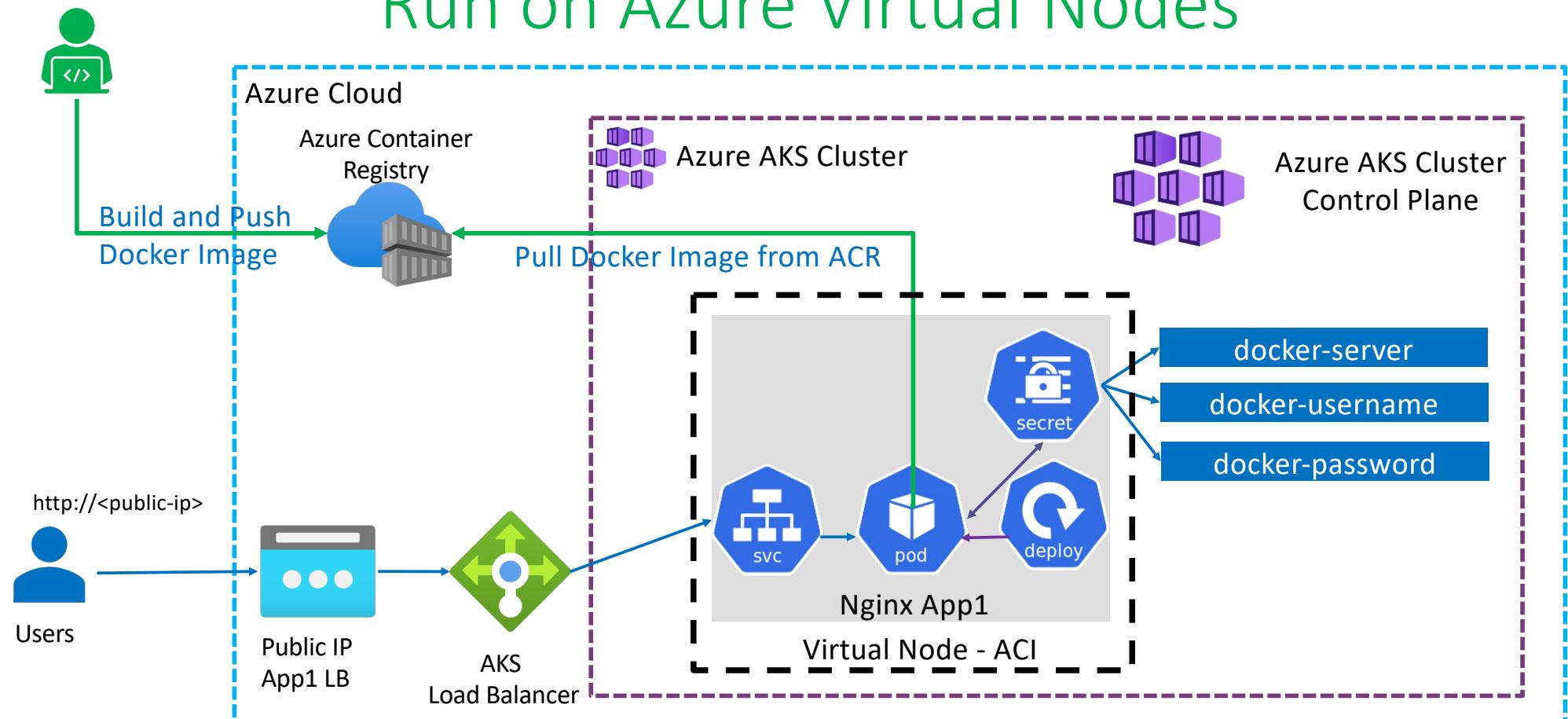
Pull Docker Images from ACR using Service Principal Run on Azure AKS Nodepools

Docker Developer



Pull Docker Images from ACR using Service Principal Run on Azure Virtual Nodes

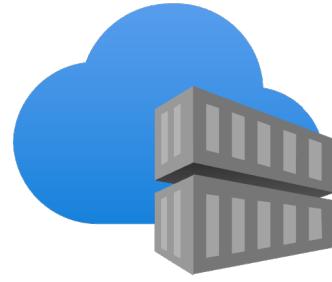
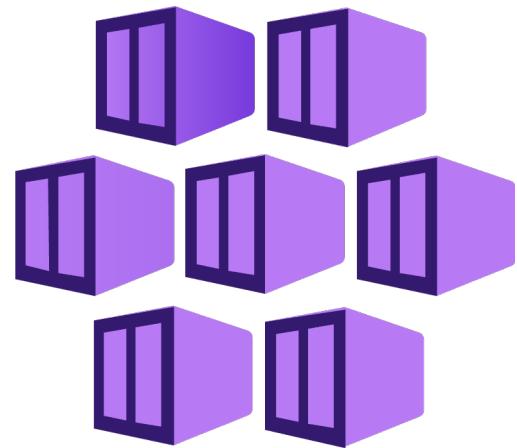
Docker Developer



Azure Container Registry – Pricing Tiers

Basic	Standard	Premium
Enabled as Public Endpoint (Public Docker Registry)	Enabled as Public Endpoint (Public Docker Registry)	Enabled as Public & Private Endpoint (Private Docker Registry)
No Customer Managed Key for Encryption	No Customer Managed Key for Encryption	Can use Customer Managed Key for Encryption
Storage: 10GB (Max Up to 20TB)	Storage: 100GB (Max Up to 20TB)	Storage: 500GB (Max Up to 20TB)
No Geo Replication & Content Trust	No Geo Replication & Content Trust	Supports Geo Replication & Content Trust
Download Speed: 30MBps Upload Speed: 10MBps	Download Speed: 60MBps Upload Speed: 20MBps	Download Speed: 100MBps Upload Speed: 50MBps
\$0.167 per day + Other Charges	\$0.667 per day + Other Charges	\$1.667 per day + Other Charges

Reference: <https://docs.microsoft.com/en-us/azure/container-registry/container-registry-skus#service-tier-features-and-limits>



Azure AKS

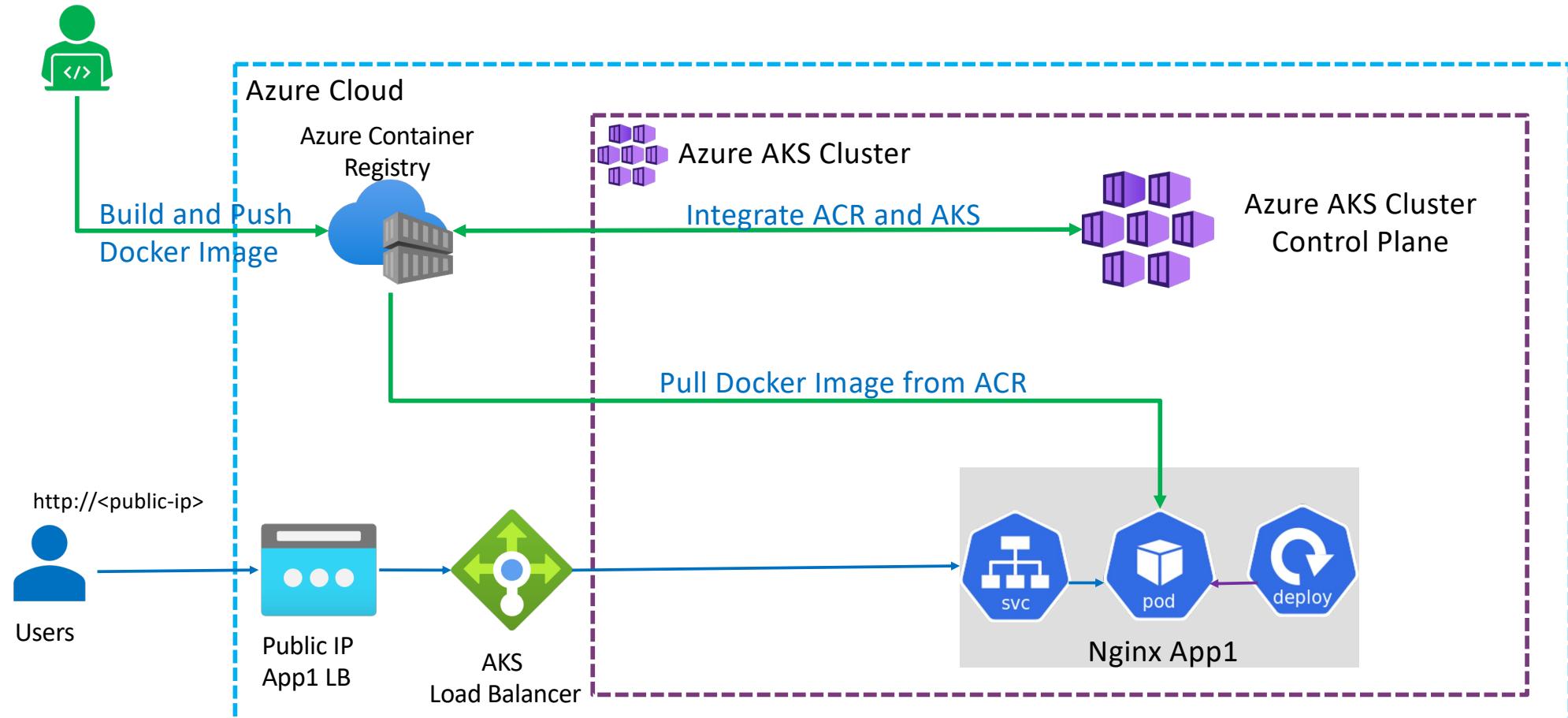
Azure Container Registry (ACR)

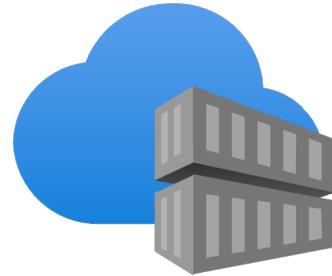
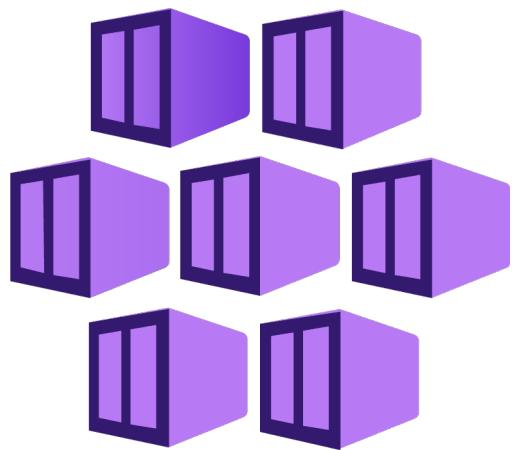
Integrate ACR & AKS



Azure ACR and AKS - Integration

Docker Developer





Run
AKS Node
Pools



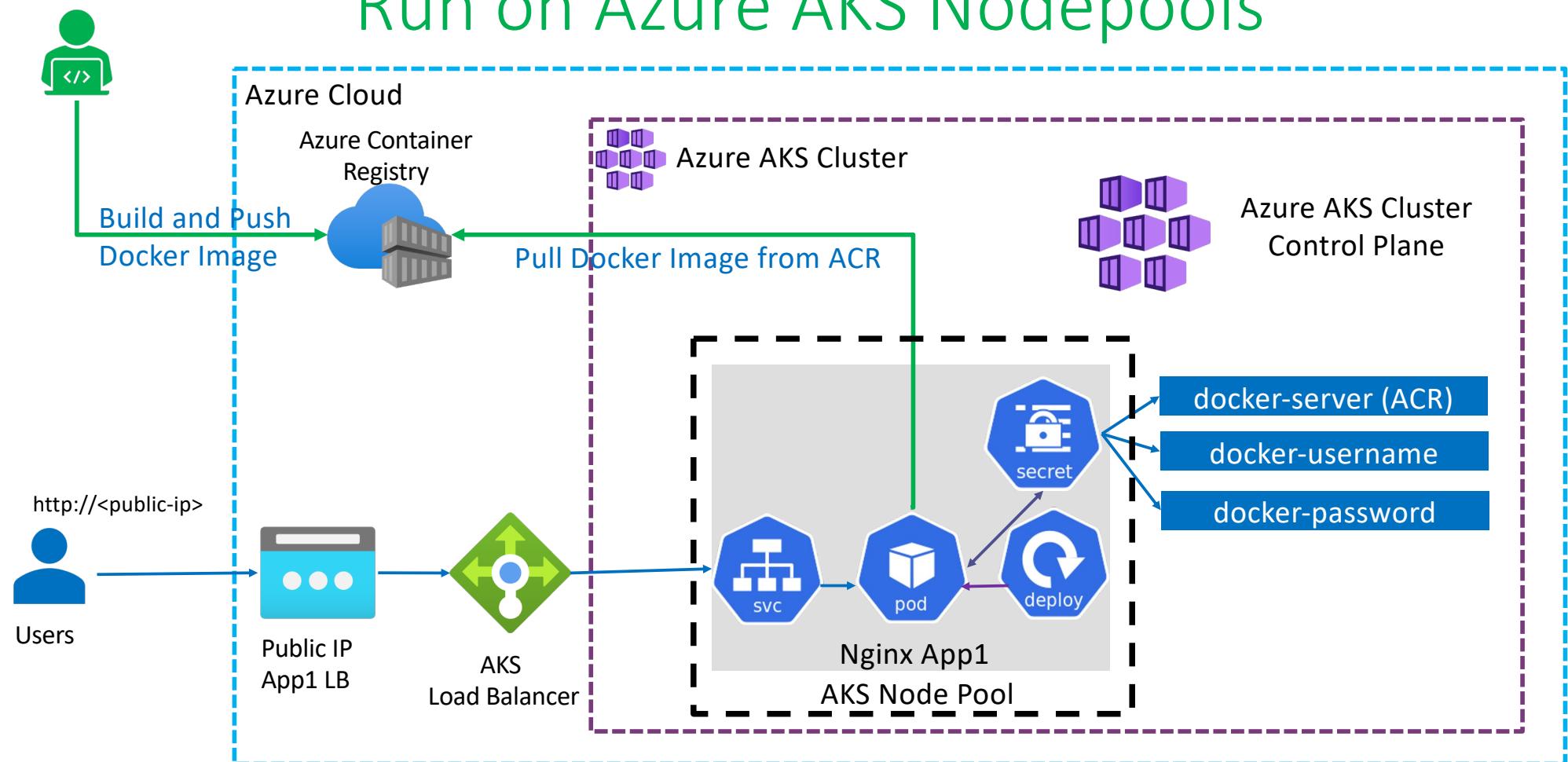
Azure AKS

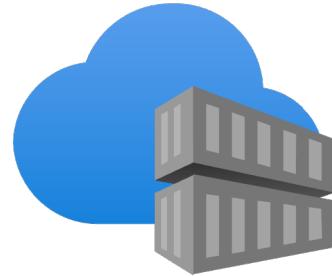
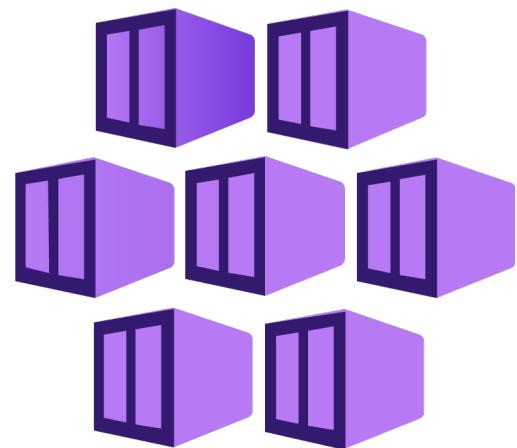
Azure Container Registry (ACR)

Pull Images from ACR using Service Principal

Pull Docker Images from ACR using Service Principal Run on Azure AKS Nodepools

Docker Developer





Schedule
on
AKS Virtual
Nodes

Azure AKS

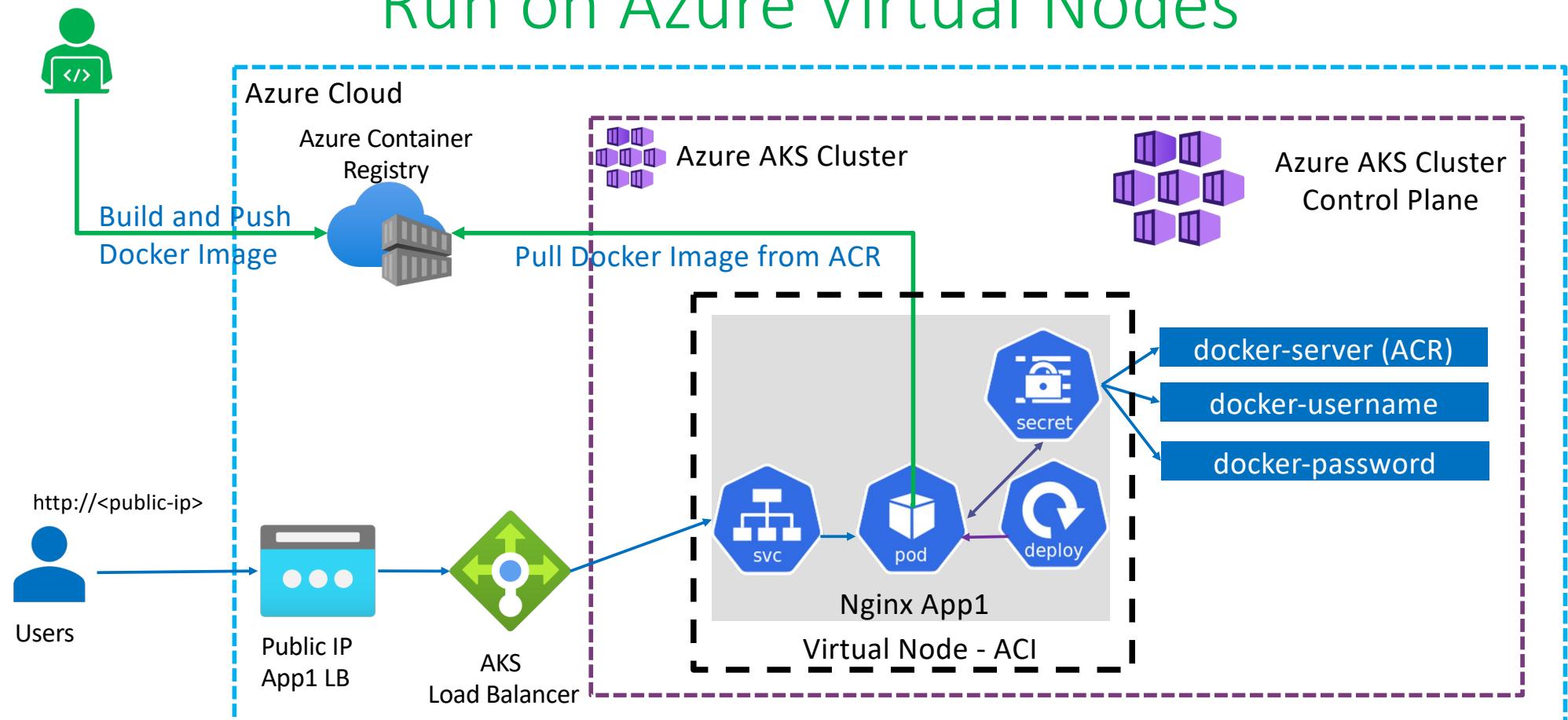
Azure Container Registry (ACR)

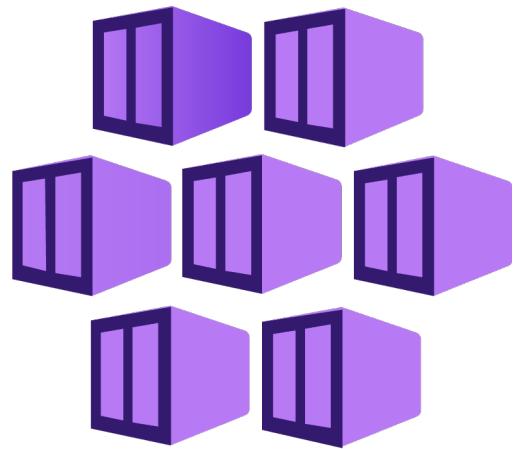
Pull Images from ACR using Service Principal



Pull Docker Images from ACR using Service Principal Run on Azure Virtual Nodes

Docker Developer





Azure AKS

Azure DevOps

Introduction





Github



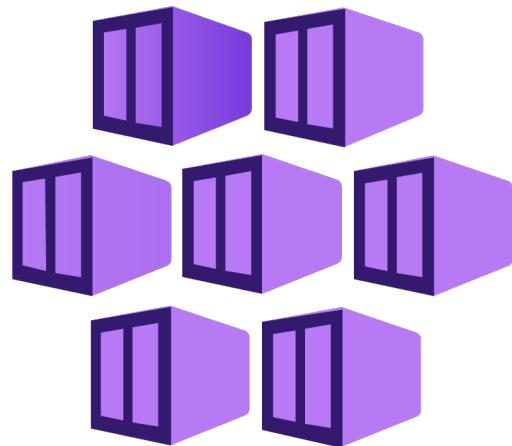
Azure
DevOps



Azure
Pipelines



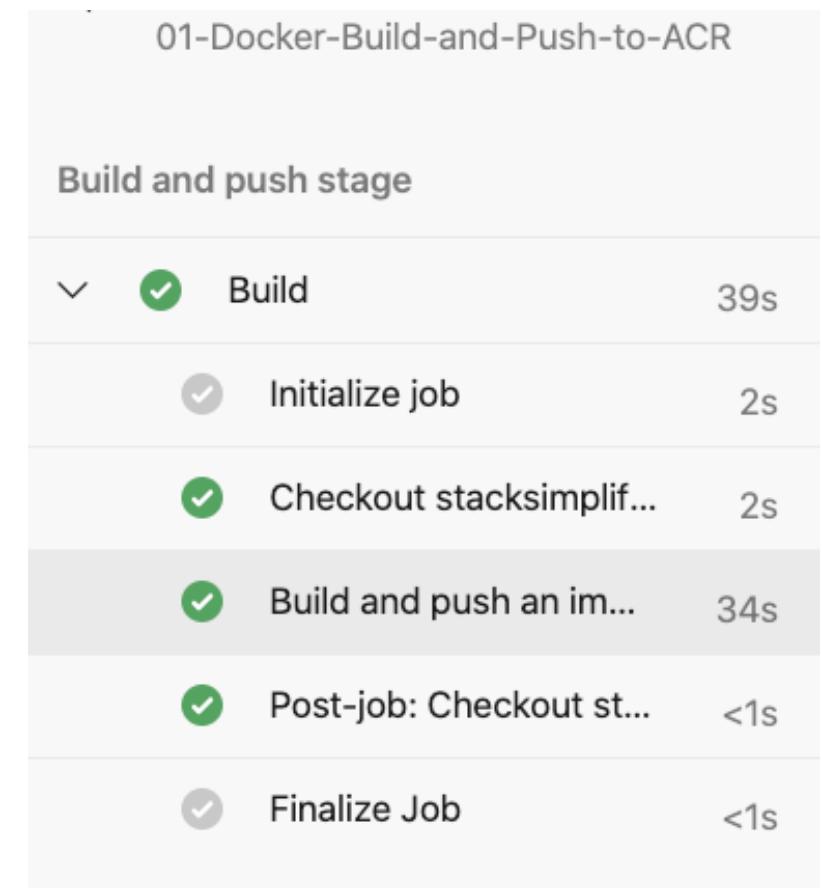
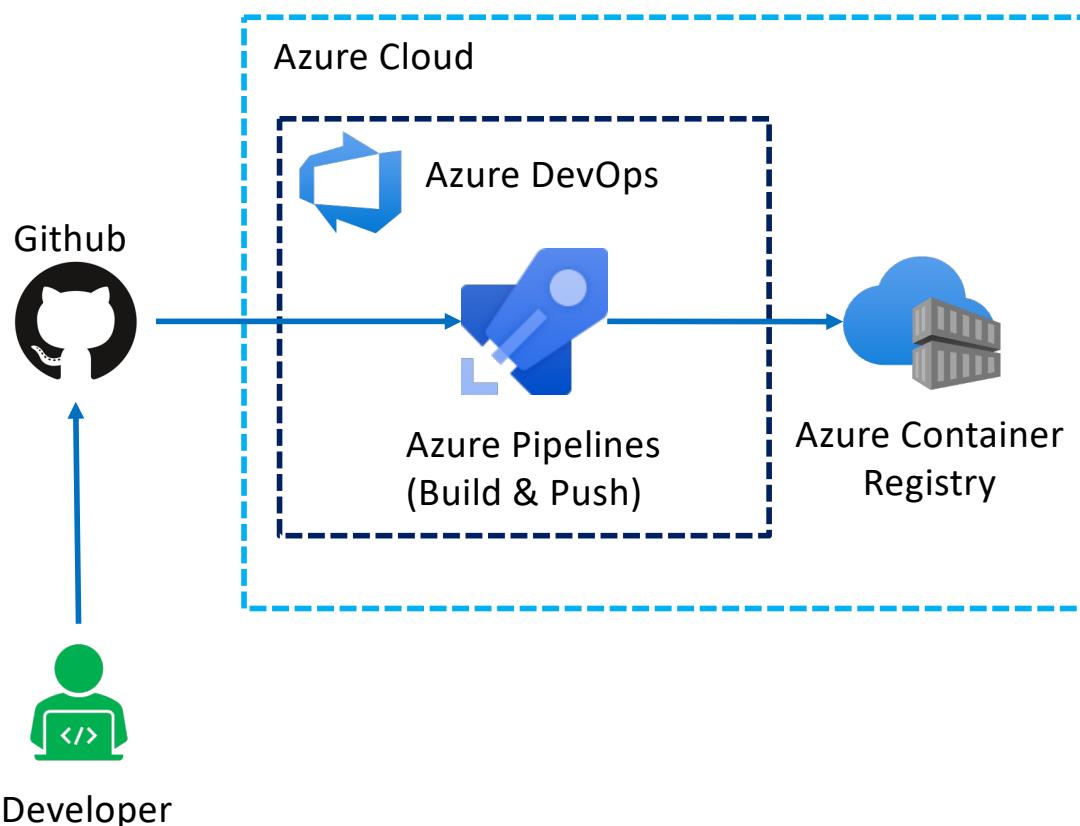
Azure
Container Registry



Azure AKS Azure DevOps Build & Push Docker Image to ACR



Azure DevOps Pipelines – Build & Push Docker Image to ACR



Azure DevOps Pipelines – Build & Push Docker Image to ACR

The screenshot shows the Azure DevOps interface for a pipeline named "01-Docker-Build-and-Push-to-ACR". The pipeline has run successfully on 2020-10-21 at 18:18:08 UTC. The "Build and push stage" contains a single "Build" job, which includes steps for initializing the job, checking out code from GitHub, building the Docker image, pushing it to the container registry, and finalizing the job. The logs for the "Build and push an image to container registry" task show the Docker build command being executed.

Step	Description	Time
1	Starting: Build and push an image to container registry	
2	====	
3	Task : Docker	
4	Description : Build or push Docker images, login or logout, start or stop containers	
5	Version : 2.176.0	
6	Author : Microsoft Corporation	
7	Help : https://aka.ms/azpipes-docker-tsg	
8	====	
9	/usr/bin/docker build -f /home/vsts/work/1/s/Dockerfile --label com.azure.dev.	
10	Sending build context to Docker daemon 98.82kB	
11		
12	Step 1/11 : FROM nginx	
13	latest: Pulling from library/nginx	
14	bb79b6b2107f: Pulling fs layer	



Github



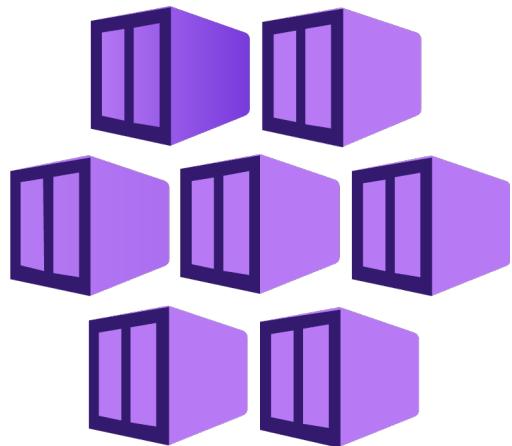
Azure
DevOps



Azure
Pipelines



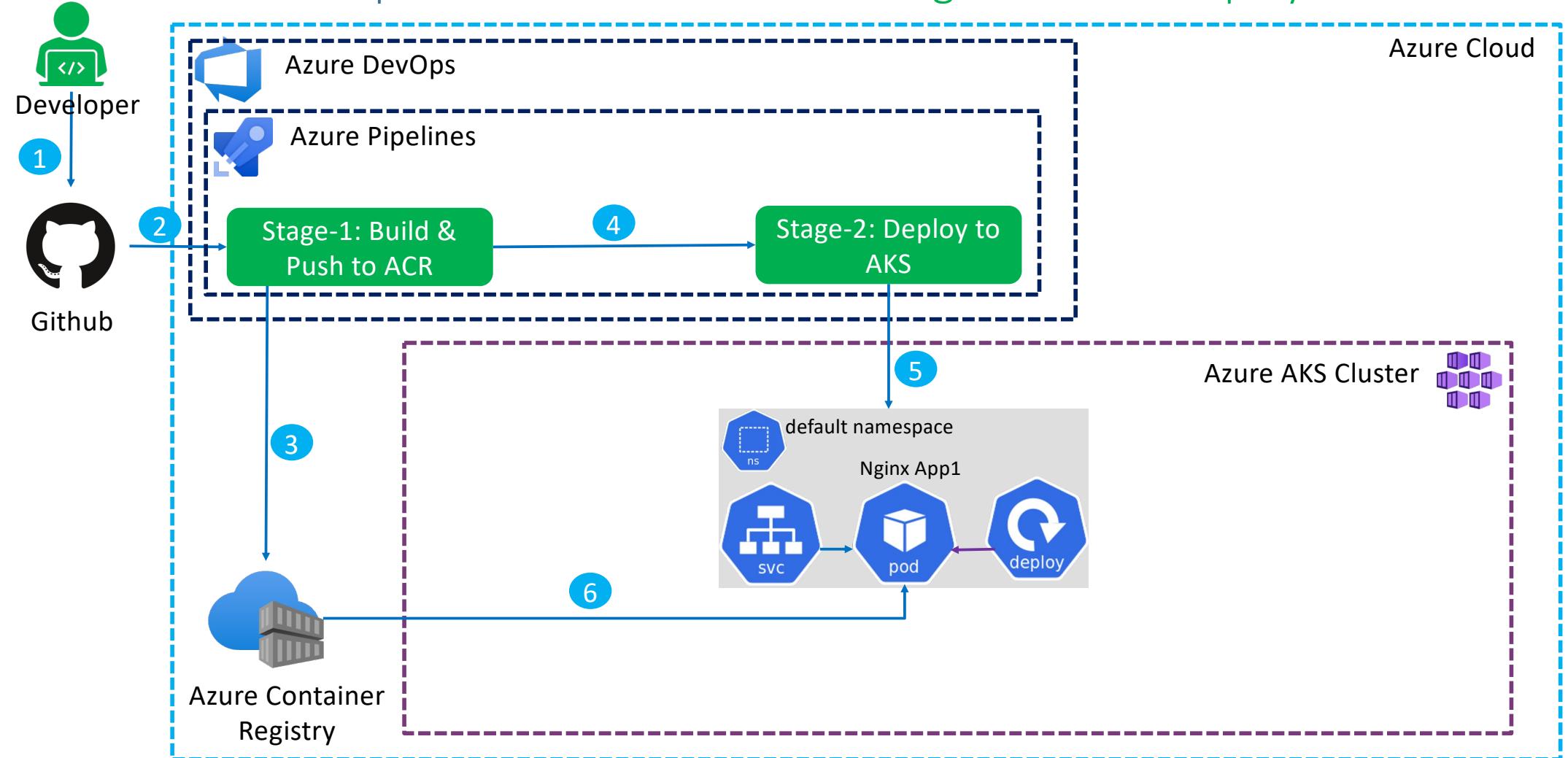
Azure
Container Registry



Azure AKS
Azure DevOps
Build & Push
Docker Image to ACR
Deploy to AKS



Azure DevOps – Build & Push Docker Image to ACR & Deploy to AKS





Github



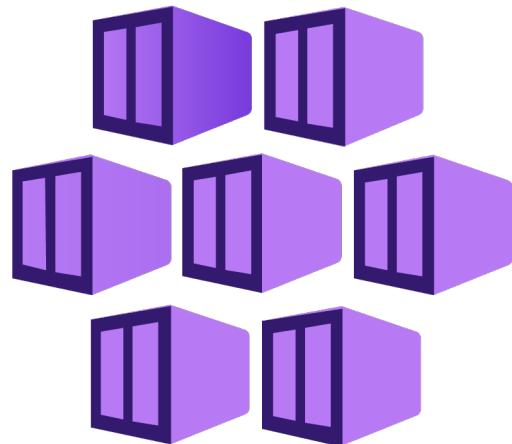
Azure
DevOps



Azure
Pipelines



Azure
Container Registry



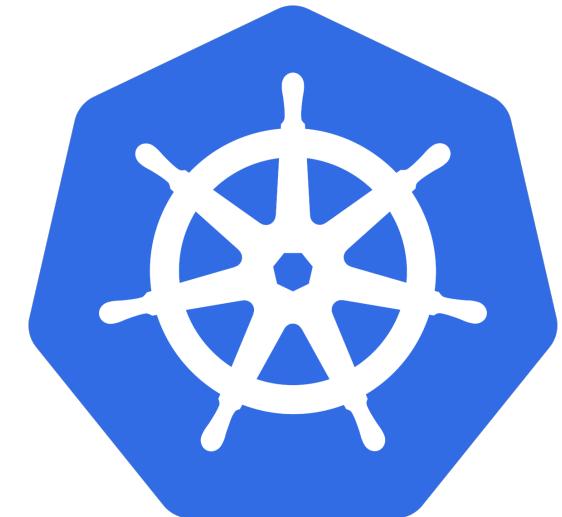
Azure AKS

Azure DevOps

Build & Push

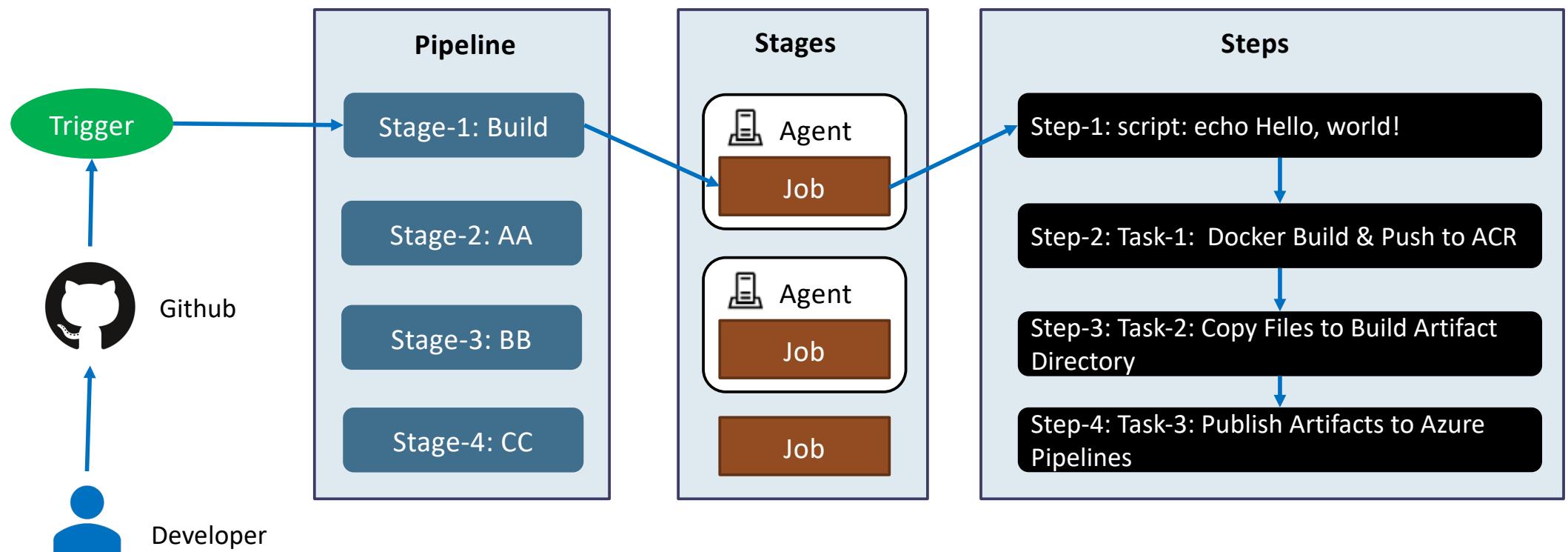
Docker Image to ACR, Publish Artifacts

STARTER PIPELINE

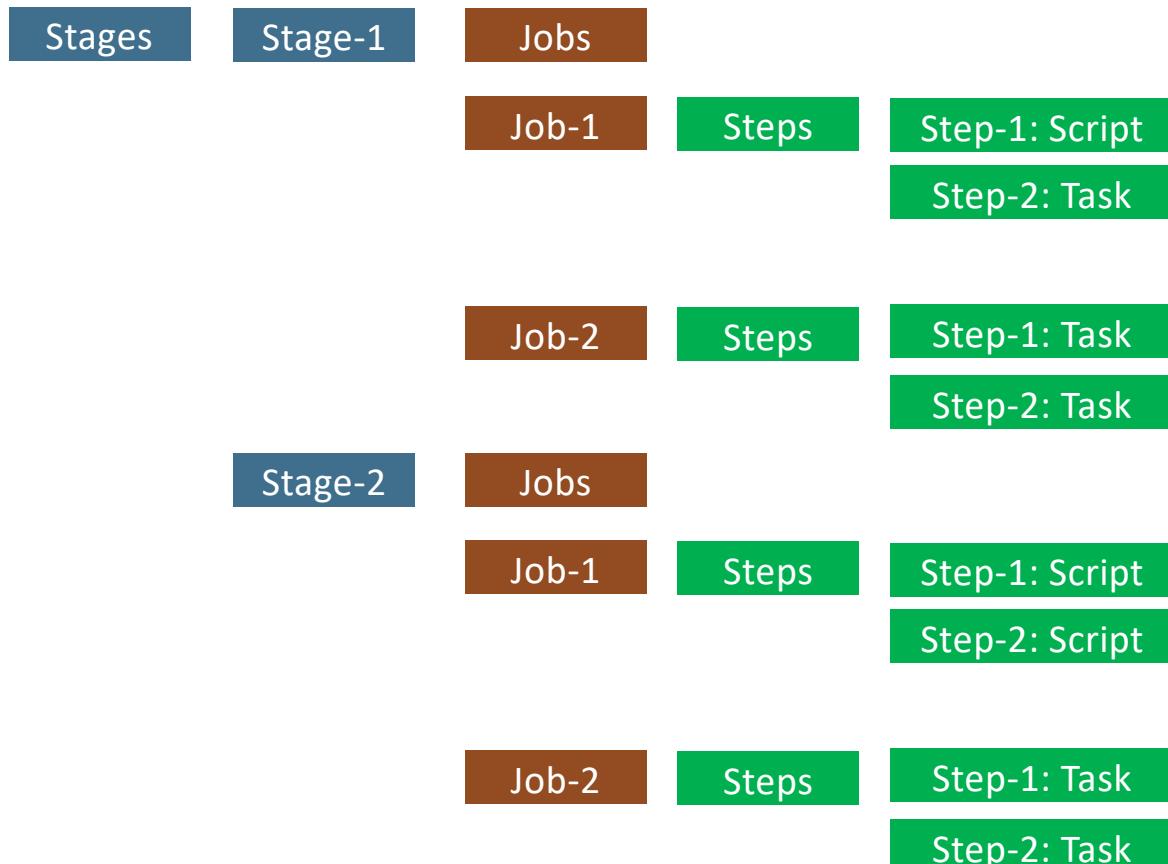




Azure Pipelines – Key Concepts



Azure Pipelines – Key Concepts



```
stages:  
- stage: Stage-1  
  jobs:  
    - job: Job-1  
      steps:  
        - script: echo Step-1  
        - script: echo Step-2  
    - job: Job-2  
      steps:  
        - task: some task step-1  
        - task: some task step-2  
- stage: Stage-2  
  jobs:  
    - job: Job-1  
      steps:  
        - task: some task step-1  
        - task: some task step-2  
    - job: Job-2  
      steps:  
        - script: echo Step-1  
        - script: echo Step-2
```



Azure Pipelines – Starter Pipeline

Goal

Create a Pipeline that will build docker images, push them to Azure Container Registry and Publish Kubernetes Manifests to Azure Pipelines

Part-1

Semi Customized

Task-1

Use pre-defined **Docker Build & Push Pipeline**

Task-2

Customize Pipeline to Use **Copy Files Task**

Task-3

Customize Pipeline to Use **Publish Build Artifacts Task**

Part-2

Fully Customized using Starter Pipeline

Task-1

Start using **Starter pipeline** and use **Docker Build or Push Docker Images Task**

Task-2

Customize Pipeline to Use **Copy Files Task**

Task-3

Customize Pipeline to Use **Publish Build Artifacts Task**

Azure DevOps – Build Pipeline

Task-1

Docker Image Build and Push to Azure Container Registry

Task-2

Copy files (kube-manifests folder) from System default working Directory to Build Artifact Directory

Task-3

Publish Build Artifacts to Azure Pipelines, so that we can use them in Release Pipelines



Github



Azure
DevOps



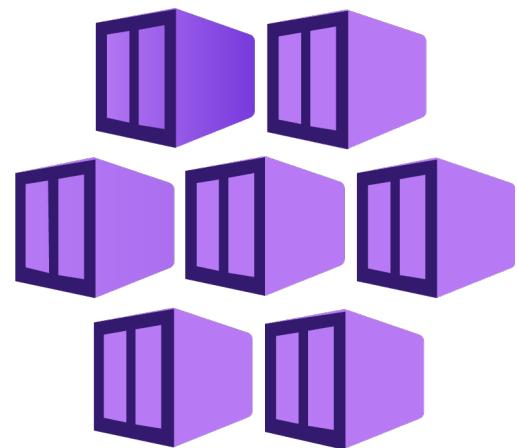
Azure
Pipelines



Azure
Container Registry



Azure
Release Pipelines



Azure AKS

Azure DevOps

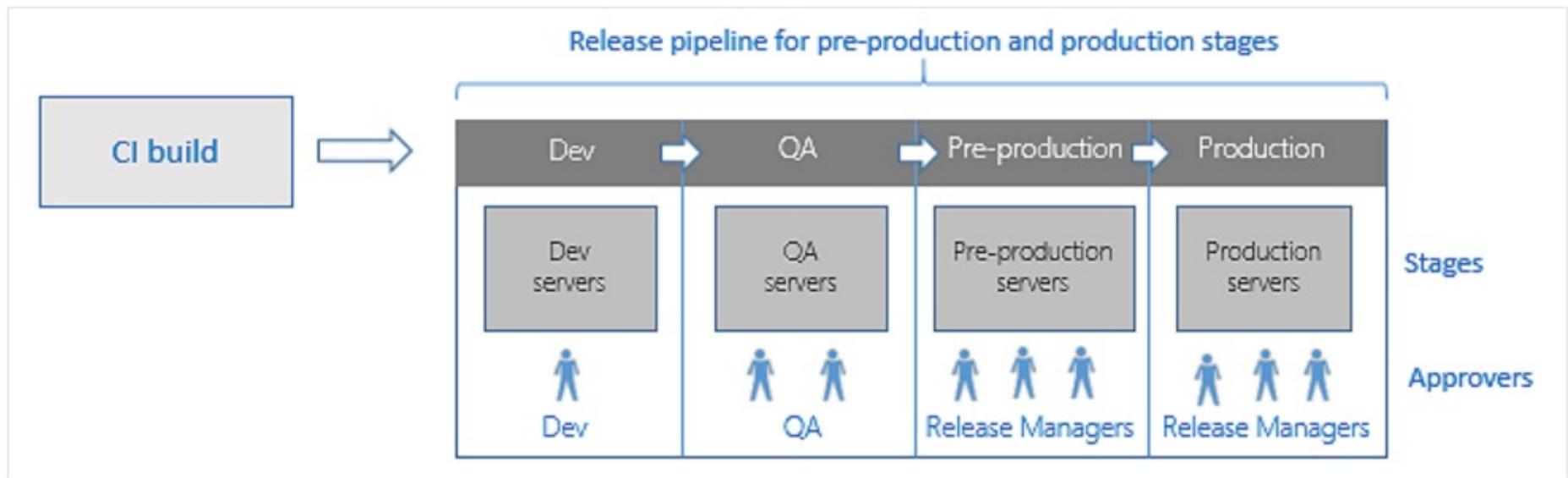
Release Pipelines

Continuous Delivery

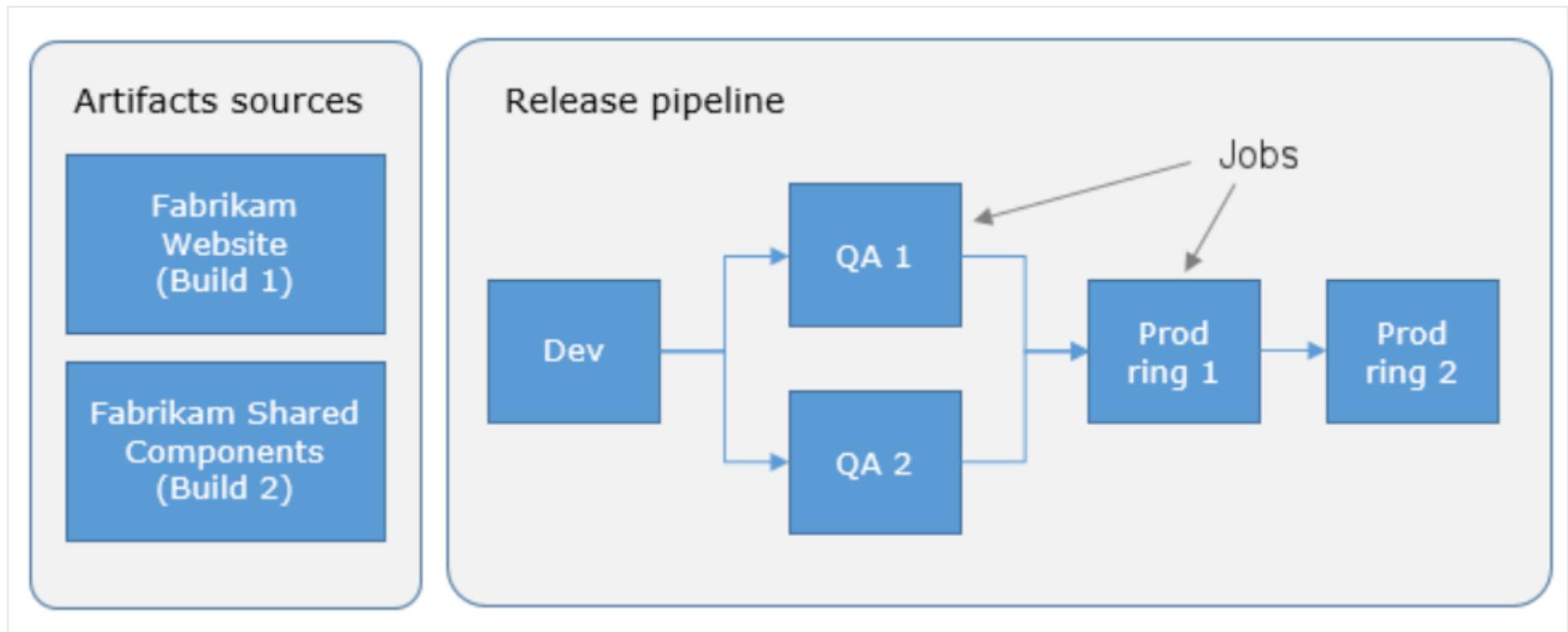


Azure DevOps – Release Pipelines

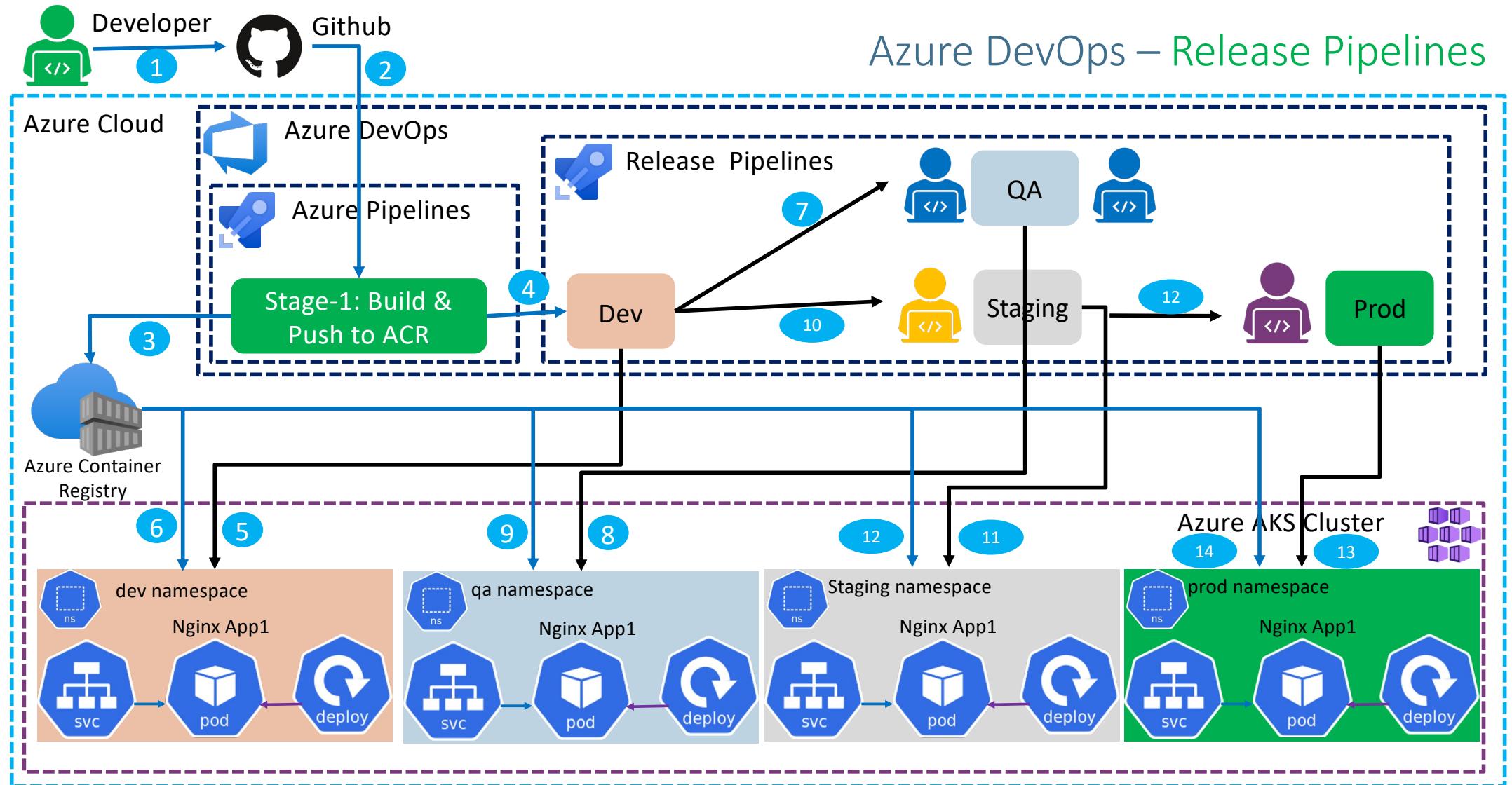
To achieve **Continuous Delivery** we use Release Pipelines



Azure DevOps – Release Pipelines



Azure DevOps – Release Pipelines

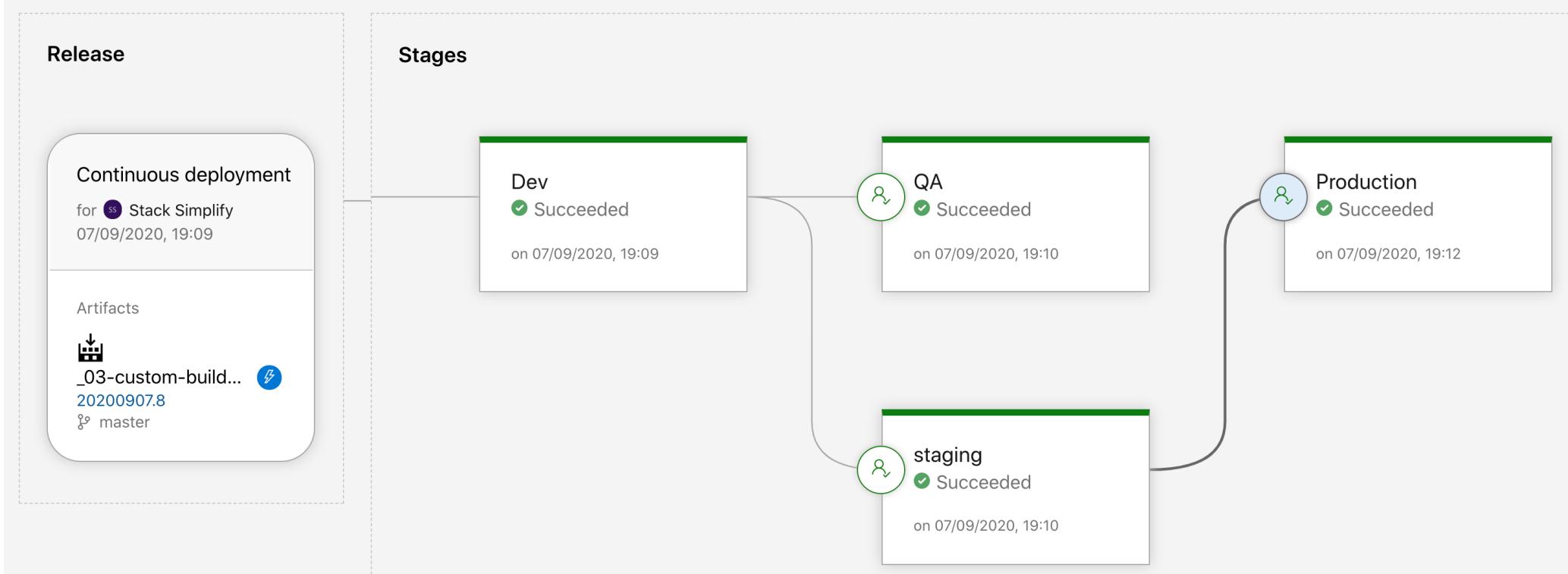


Azure Release Pipelines

↑ 01-Release-Pipeline > Release-4 ▾

Pipeline Variables History

+ Deploy ▾ ⚙ Cancel ⏪ Refresh ⌛ Edit ▾ ...



Azure Release Pipelines - Releases

The screenshot shows the Azure DevOps interface for managing releases. On the left, there's a sidebar with various icons and a search bar for pipelines. The main area displays the '01-Release-Pipeline' under the 'Releases' tab. It lists four releases: 'Release-4', 'Release-3', 'Release-2', and 'Release-1'. Each release is associated with a creation date and a set of stages: Dev, QA, staging, and Production. The 'Production' stage is marked with a checkmark in all cases.

Release	Created	Stages
Release-4 2020090... master	07/09/2020, 19:09:02	Dev, QA, staging, Production
Release-3 2020090... master	07/09/2020, 19:03:32	Dev, QA, staging, Production
Release-2 2020090... master	07/09/2020, 18:43:39	Dev
Release-1 2020090... master	07/09/2020, 18:38:48	Dev



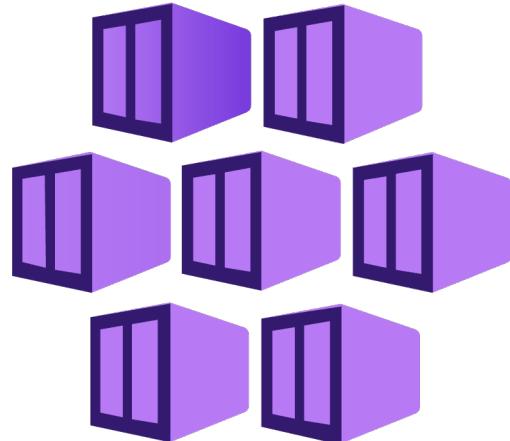
DNS
Zones



Managed
Service
Identity



AKS
Addon

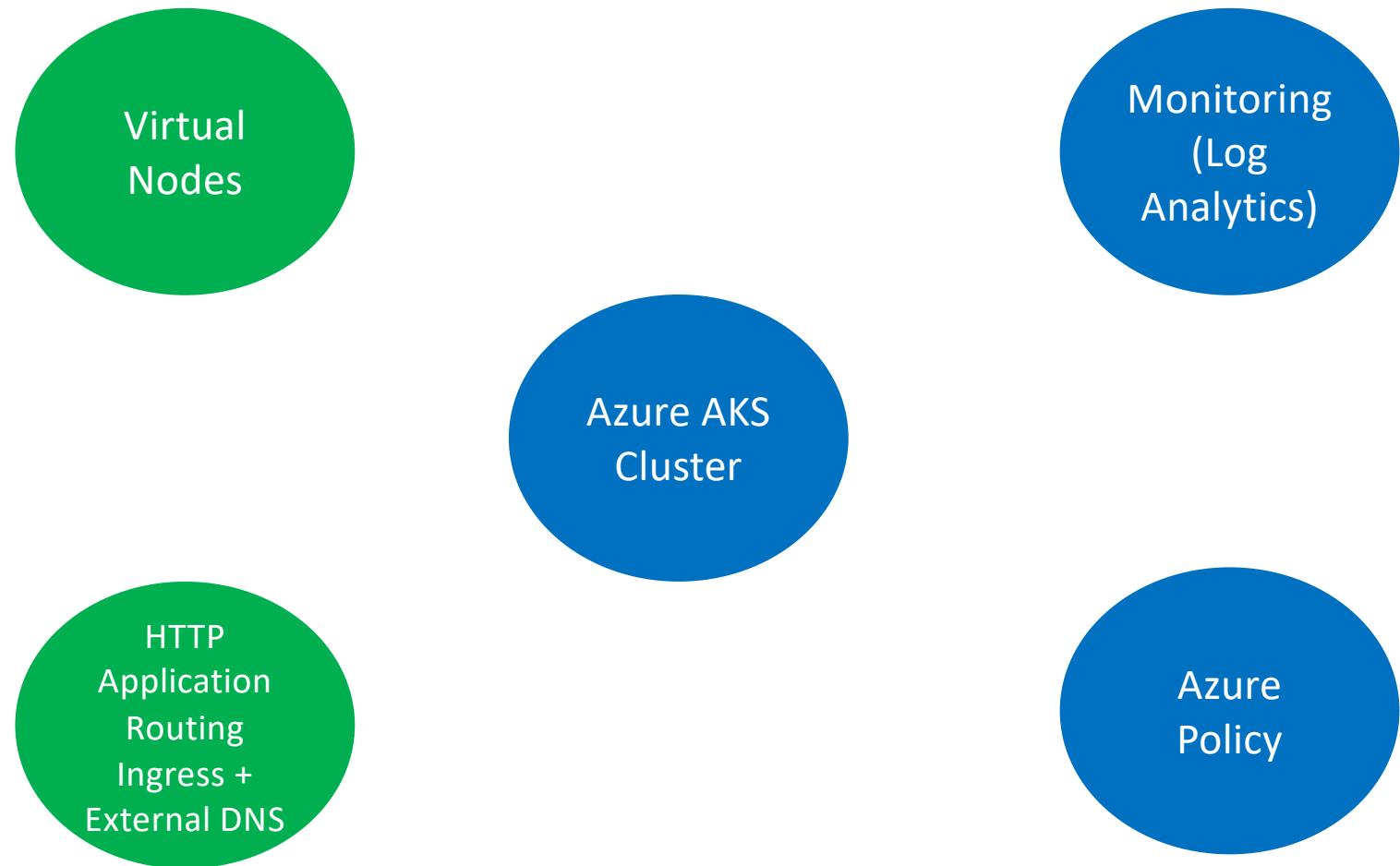


Azure AKS HTTP Application Routing Add On

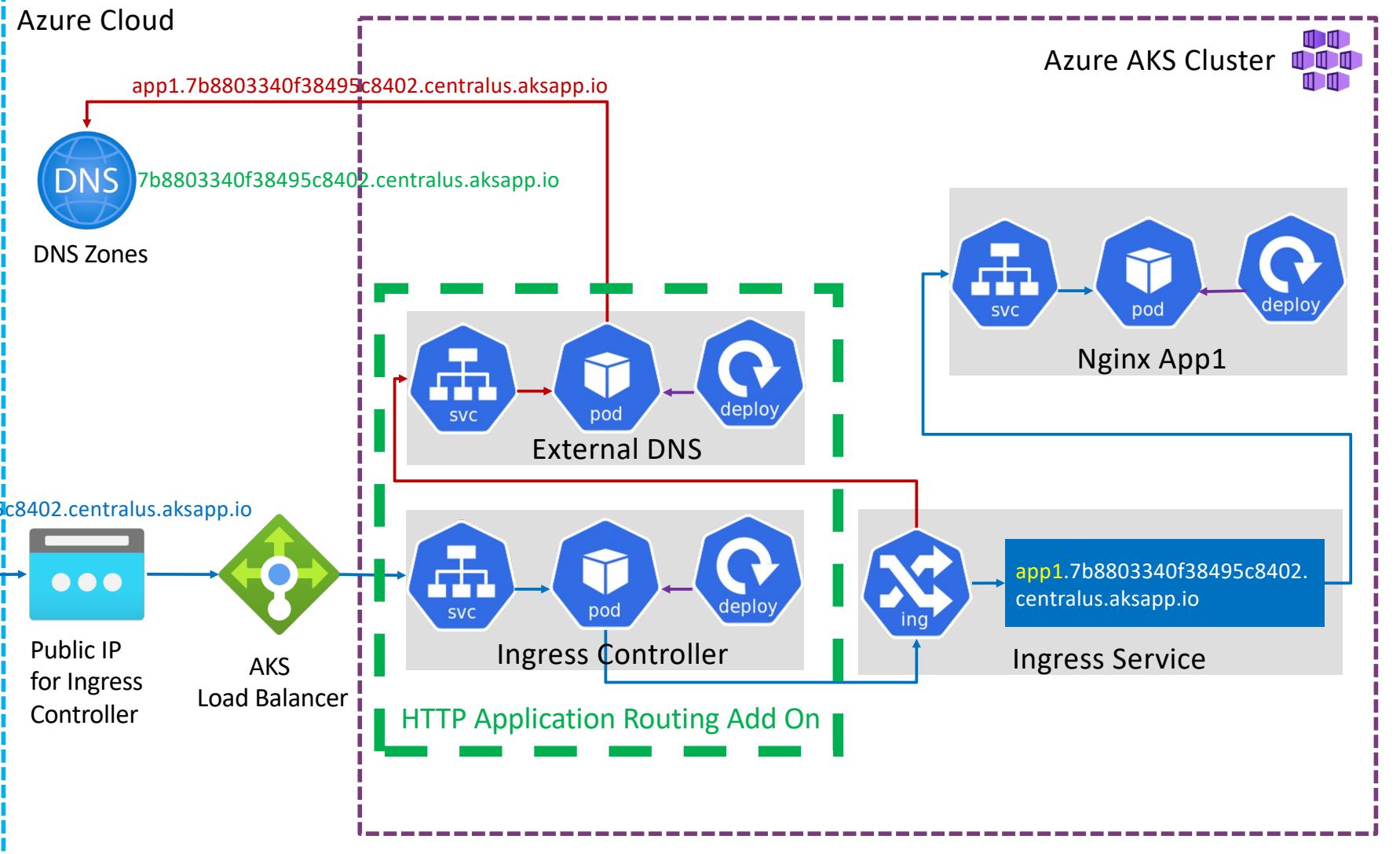
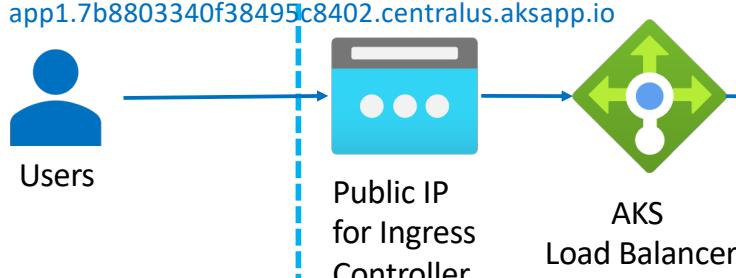


Ingress Controller + External DNS

Azure AKS - Addons



Azure AKS
HTTP
Application
Routing
Add On
Ingress +
External DNS
(Automatic Install)

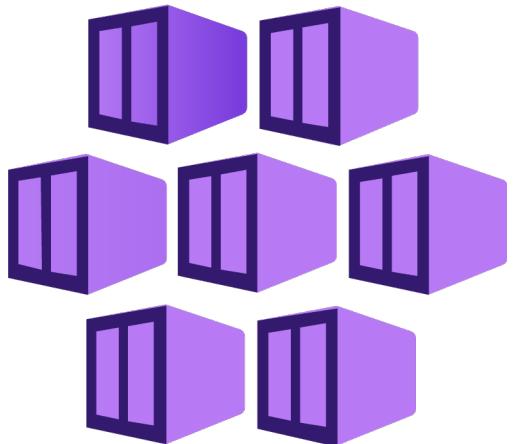


Ingress with External DNS

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-demo
  annotations:
    kubernetes.io/ingress.class: addon-http-application-routing
spec:
  rules:
  - host: app1.7b8803340f38495c8402.centralus.aksapp.io
    http:
      paths:
      - path: /
        backend:
          serviceName: app1-nginx-clusterip-service
          servicePort: 80
```

This annotation will ensure the ingress service we are creating will be part of the Ingress Controller created using HTTP Application Routing Add On

This will help us to add DNS Record Set in Azure DNS Zones using k8s External DNS



Azure AKS Configure Access To **Multiple AKS Clusters**



Azure AKS Cluster Access

```
# Configure AKSDEM03 & 4 Cluster Access for kubectl  
az aks get-credentials --resource-group aks-rg3 --name aksdemo3  
az aks get-credentials --resource-group aks-rg4 --name aksdemo4
```

```
# View kubeconfig  
kubectl config view
```



AKS Admin

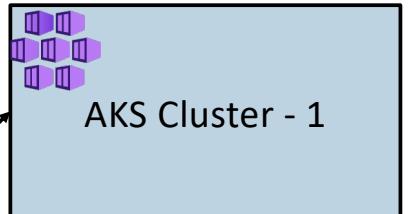
```
# View the current context for kubectl  
kubectl config current-context
```

```
# Switch Context  
kubectl config use-context aksdemo3
```

Current Context

AKS Cluster-1

AKS Cluster-2





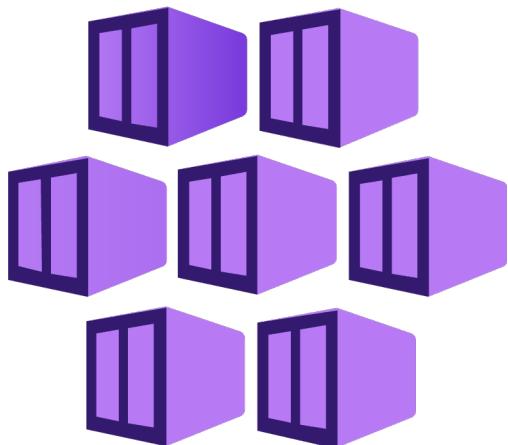
Azure Active Directory



Azure AD Groups



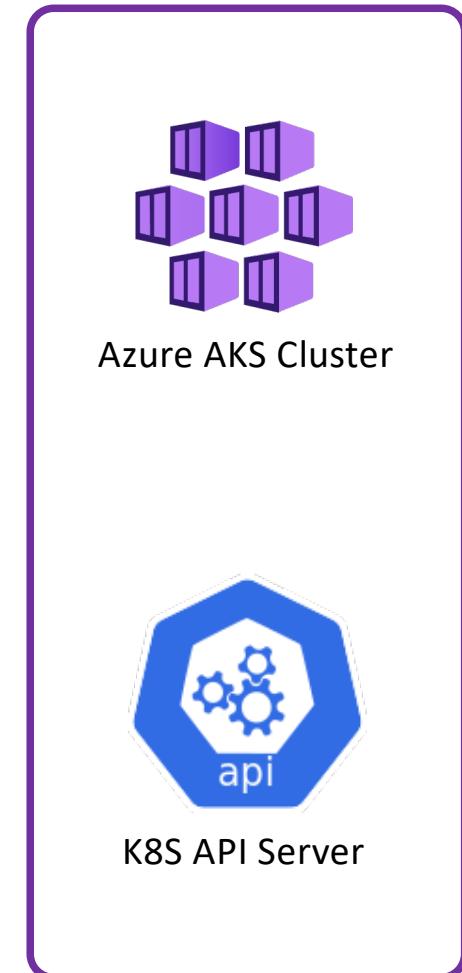
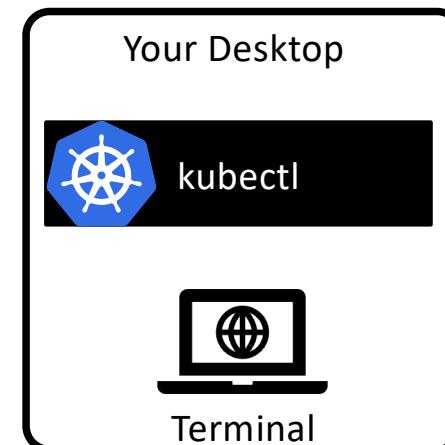
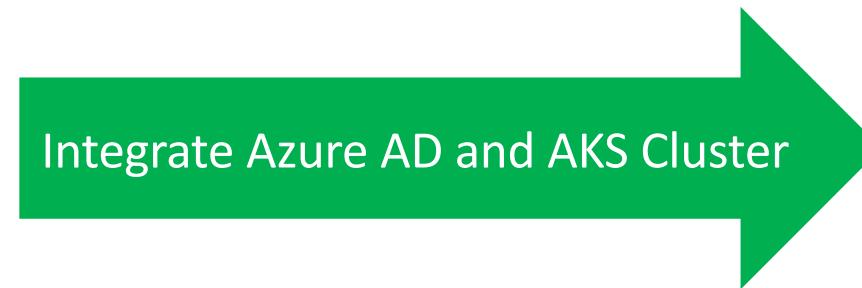
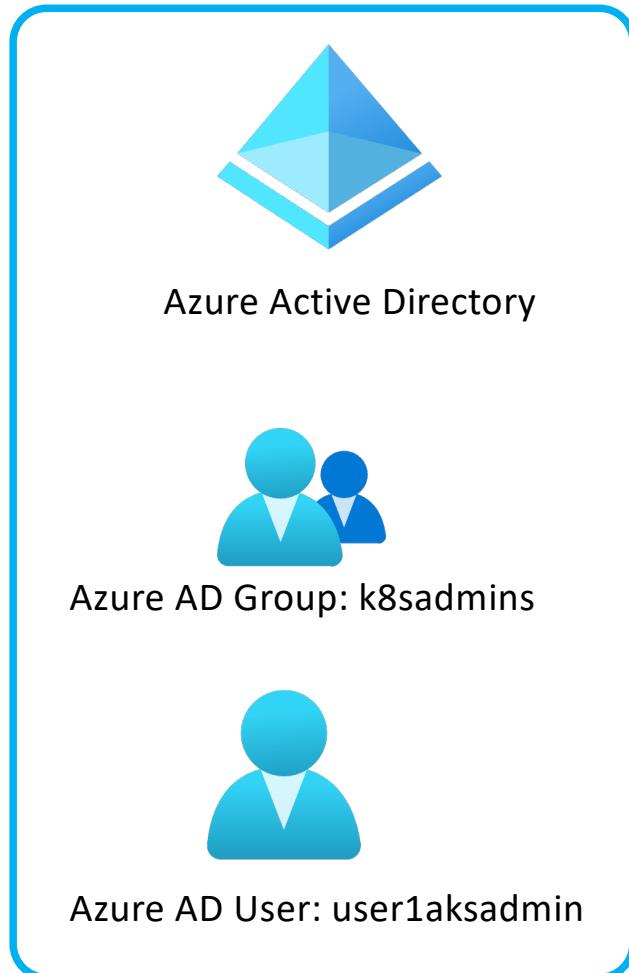
Azure AD Users



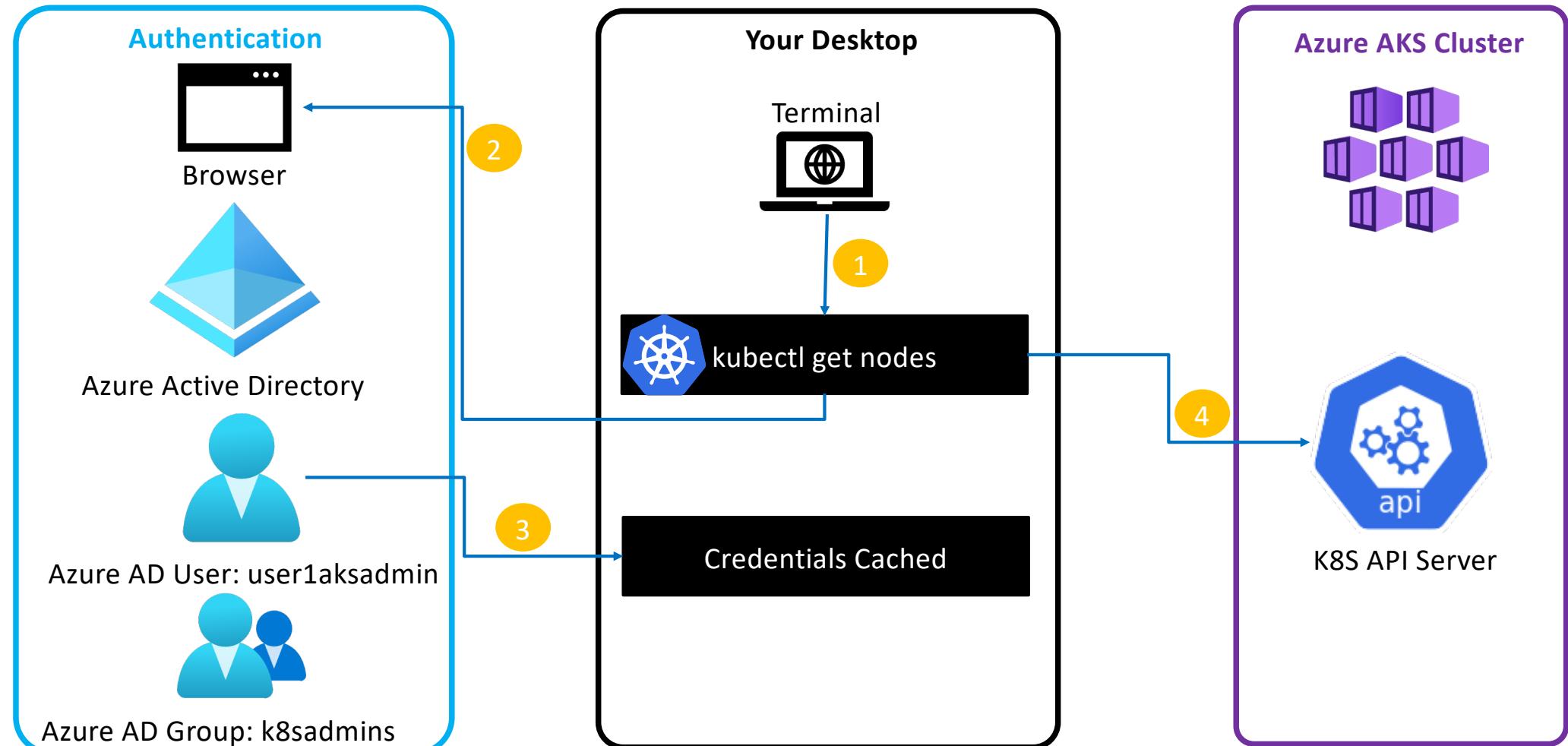
Azure AKS Azure AD Authentication for AKS Admins



Azure Active Directory Authentication for AKS Admins



Azure Active Directory Authentication for AKS Admins





Role



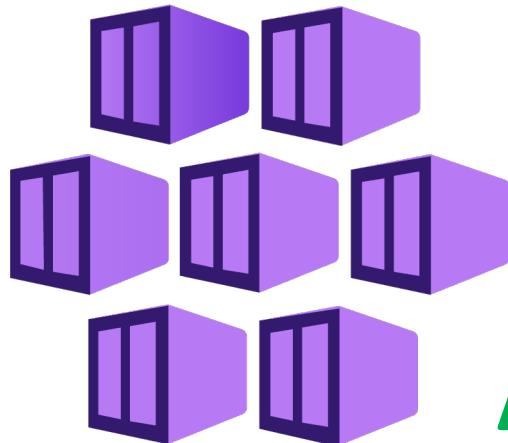
Role Binding



Cluster Role



Cluster Role Binding



Azure AKS Kubernetes RBAC Azure Active Directory



Active Directory



AD Groups



AD Users

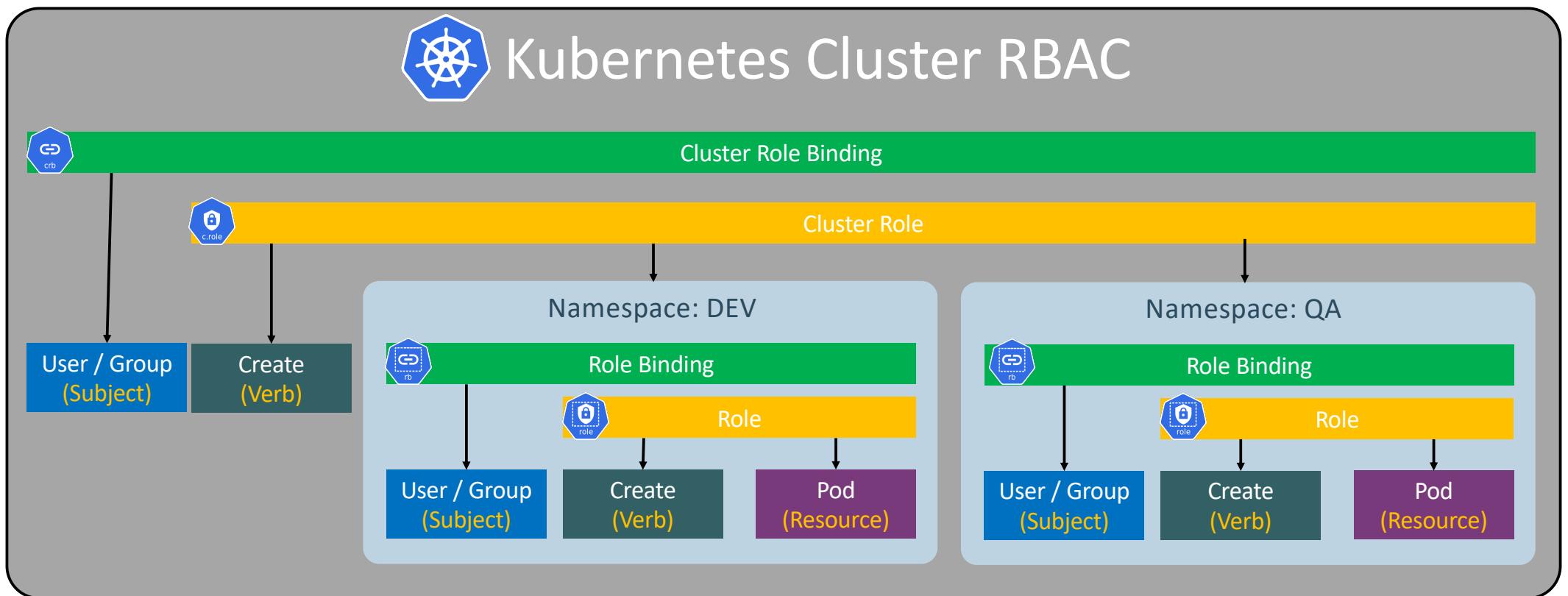
Kubernetes RBAC

Kubernetes RBAC - Fundamentals

Subjects	API Groups	Resources	Verbs
Users or processes that need access to the Kubernetes API	The k8s API objects that we grant access to		List of actions that can be taken on a resource
Kind: Group, User	core	Pods	Create Patch
Kind: Service Account	extensions	Deployments	List get
	apps	Services	Watch Replace
	batch	StatefulSets	Delete Read

Reference: <https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19>

Kubernetes RBAC

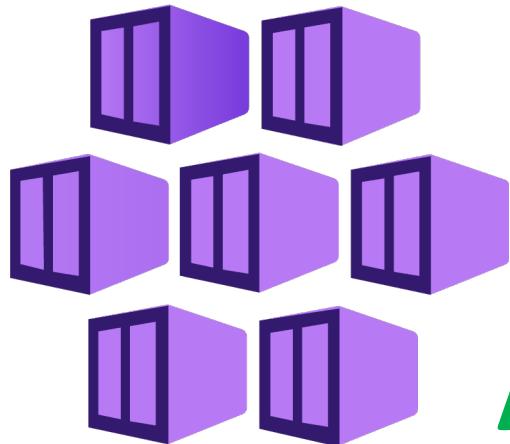




Role



Role Binding



Active Directory

Azure AKS

Kubernetes RBAC

Azure Active Directory



AD Groups



AD Users

Kubernetes RBAC Role



A **Role** can only be used to grant access to resources within a single namespace.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dev-user-full-access-role
  namespace: dev
rules:
- apiGroups: [ "", "extensions", "apps"]
  resources: ["*"]
  verbs: ["*"]
- apiGroups: ["batch"]
  resources:
  - jobs
  - cronjobs
  verbs: [ "*" ]
```

Namespace: Creating this role in dev namespace

API Groups: core, extensions, apps

Resources: pods, deployments, services

Verbs: Create, List, Delete, Patch

Kubernetes RBAC Role Binding

A **Role Binding** is used to tie the **Role** and **Subject** together



```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dev-user-access-rolebinding
  namespace: dev
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: dev-user-full-access-role
subjects:
- kind: Group
  namespace: dev
  name: "e6dcdae4-e9ff-4261-81e6-0d08537c4cf8"
```

Role

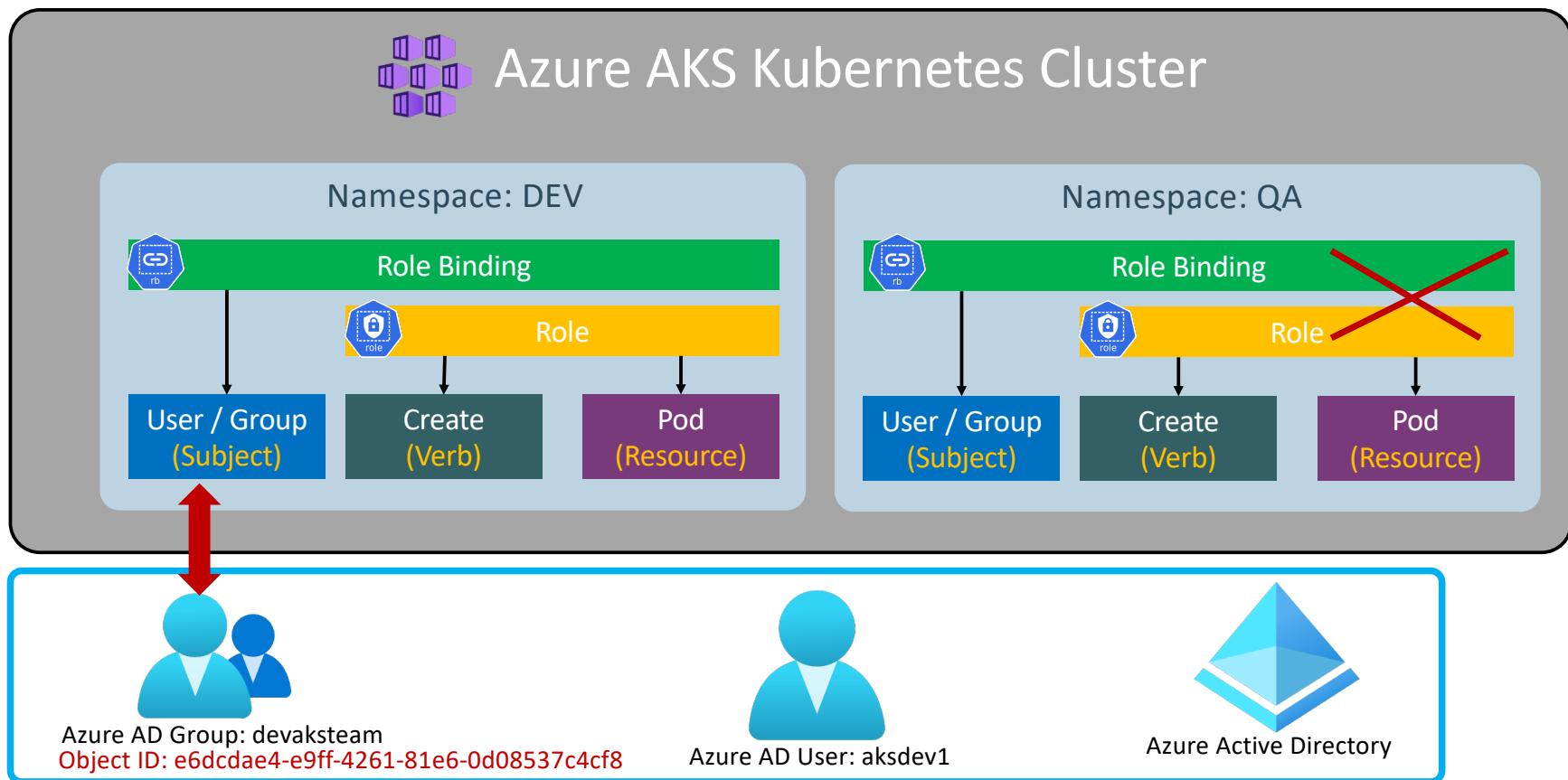
Subjects

Namespace: Creating this role binding in dev namespace

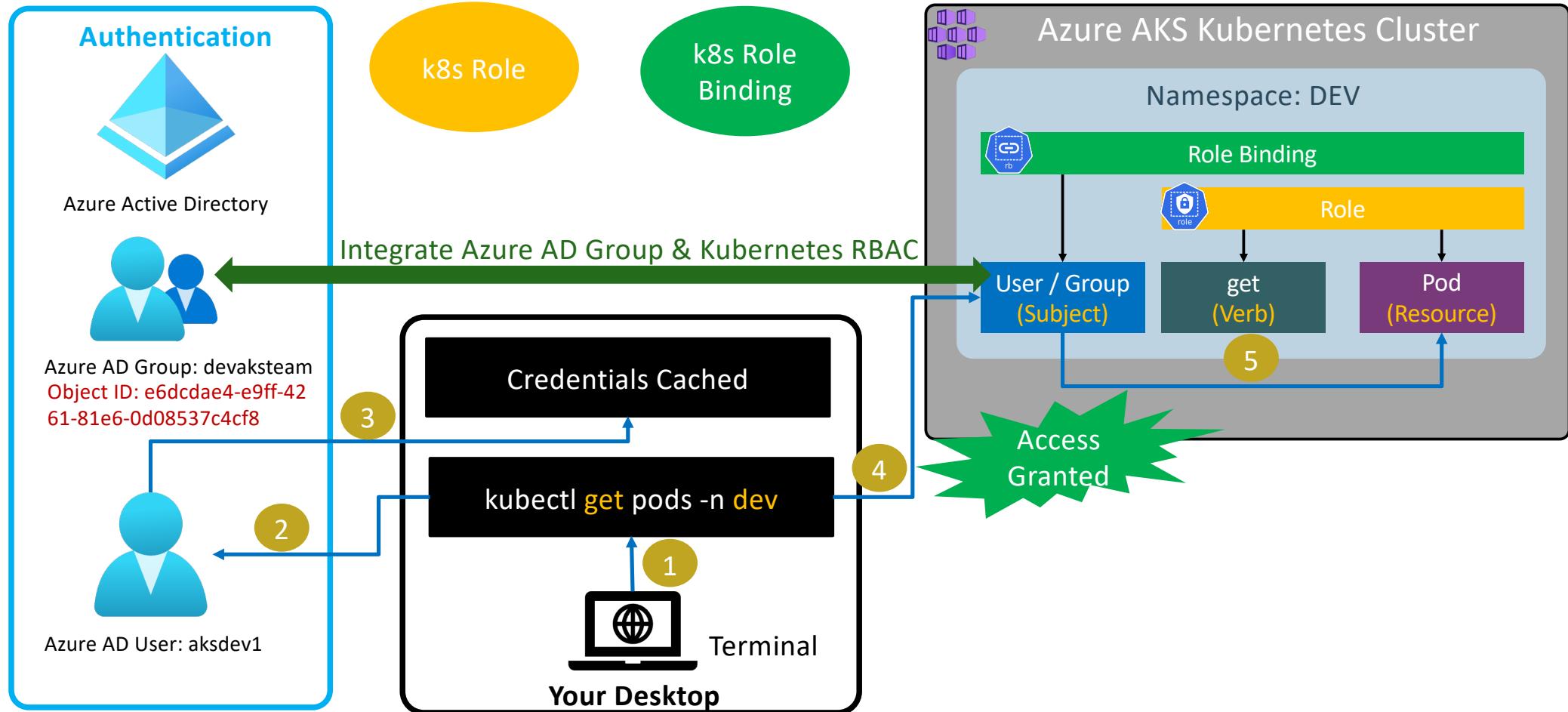
Role: Mapped role here

Subjects: Group: Azure AD Group with Object ID xxx has full access to namespace dev in AKS Cluster

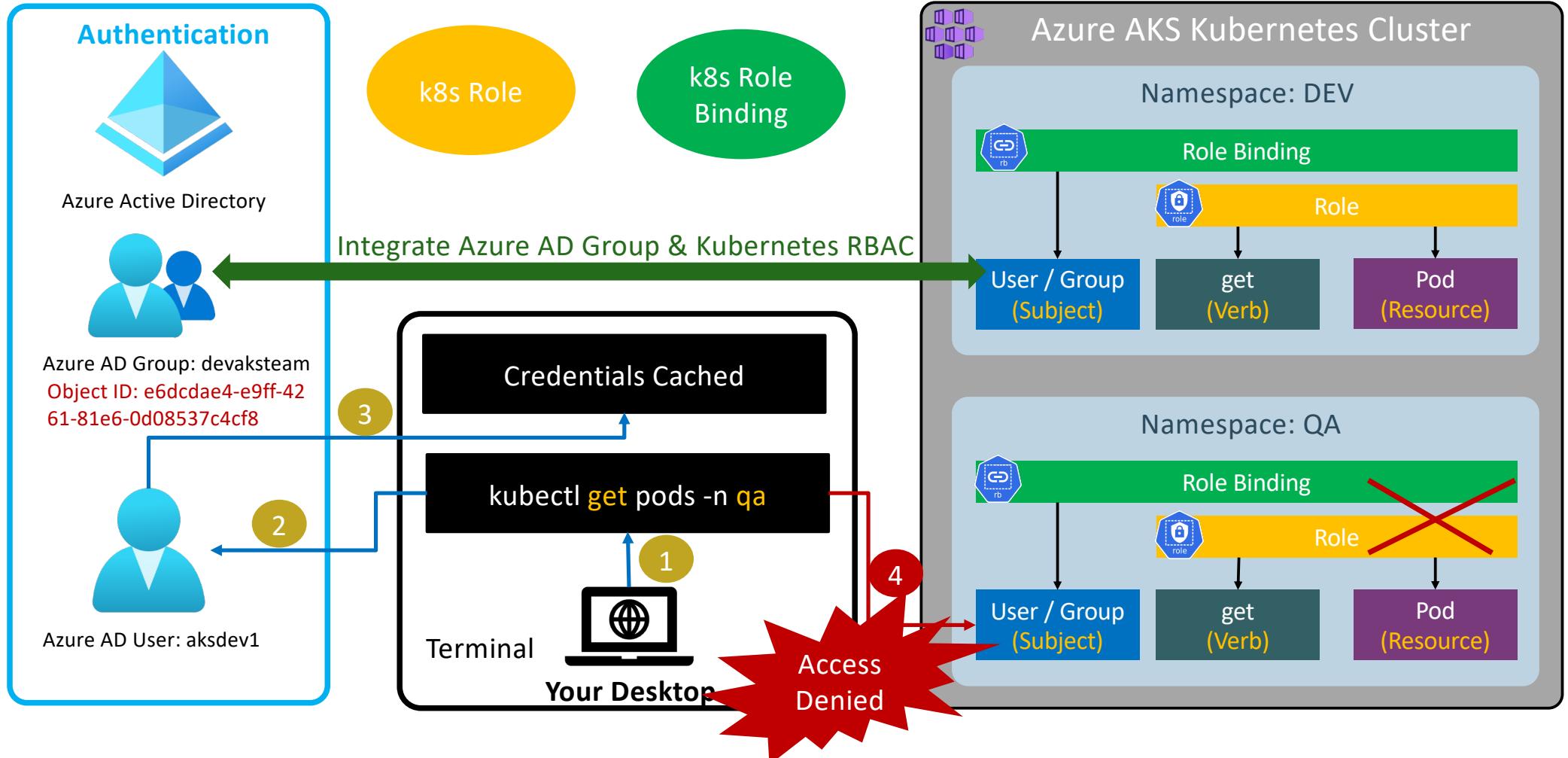
Azure Active Directory & Kubernetes RBAC



Azure Active Directory & Kubernetes RBAC



Azure Active Directory & Kubernetes RBAC

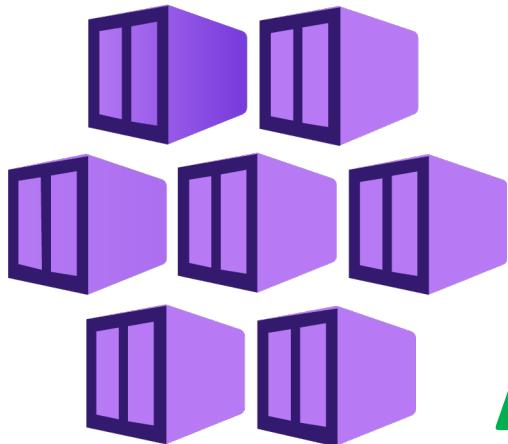




Cluster Role



Cluster Role Binding



Active Directory



Azure AKS Kubernetes RBAC Azure Active Directory



AD Groups



AD Users

Kubernetes RBAC Cluster Role

Cluster Role



A **ClusterRole** can be used to grant the same permissions as a Role, but because they are cluster-scoped, access will be granted across cluster

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: aks-cluster-readonly-role
rules:
- apiGroups: [ "", "extensions", "apps"]
  resources: [ "*" ]
  verbs: [ "get", "list", "watch" ]
- apiGroups: [ "batch" ]
  resources:
    - jobs
    - cronjobs
  verbs: [ "get", "list", "watch" ]
```

Read Only access to AKS Cluster

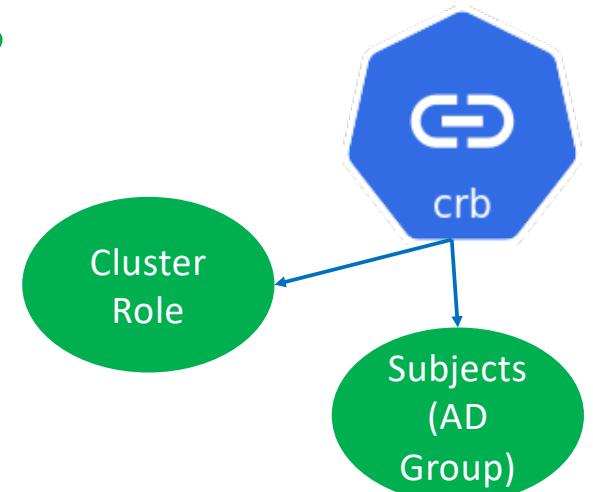
Read Only access to AKS Cluster

Kubernetes RBAC Cluster Role Binding

A Cluster Role Binding is used to tie the Cluster Role and Subject together

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: aks-cluster-readonly-rolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: aks-cluster-readonly-role
subjects:
- kind: Group
  name: "e808215d-d159-49ba-8bb6-9661ba478842"
```

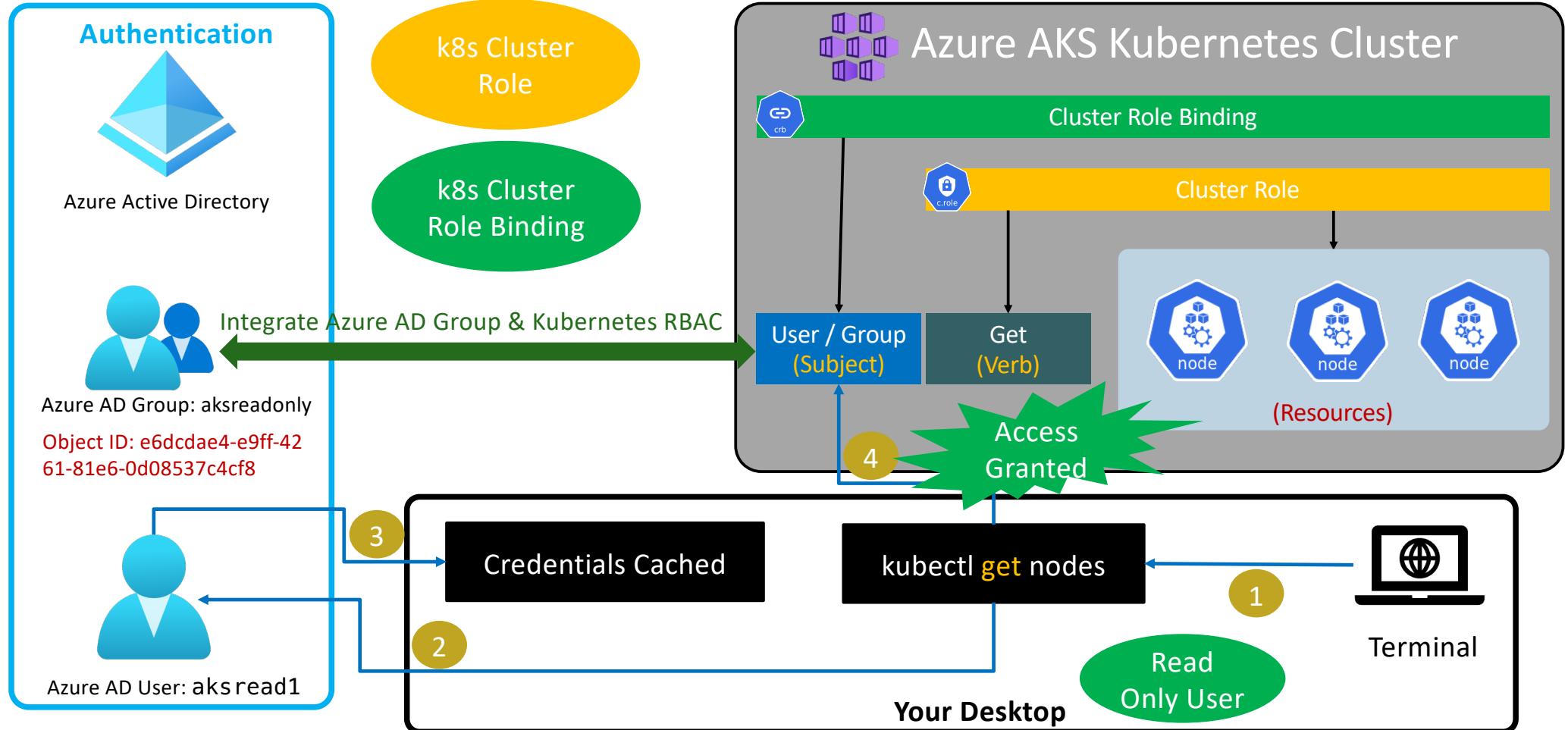
Cluster Role Binding



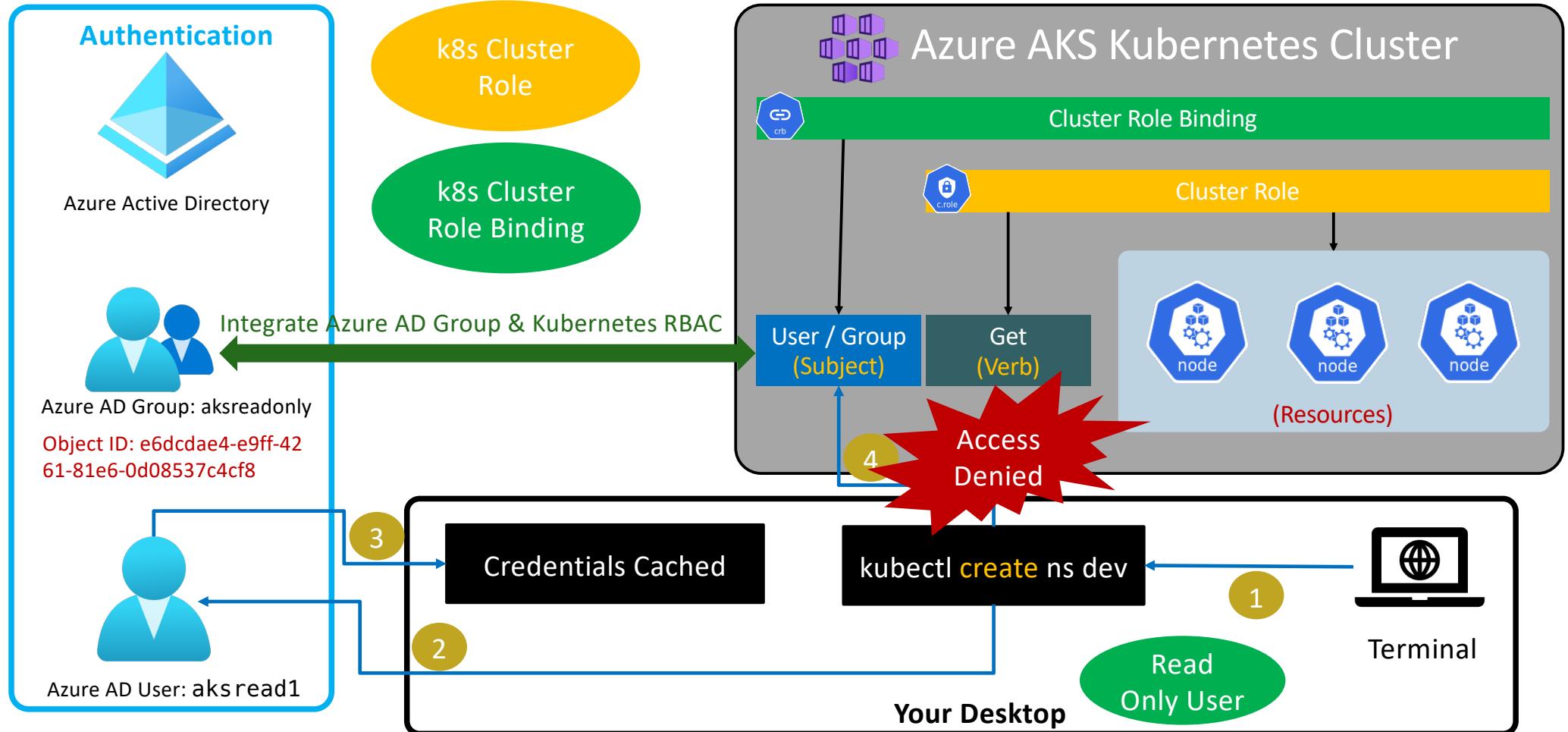
Role: Mapped role here

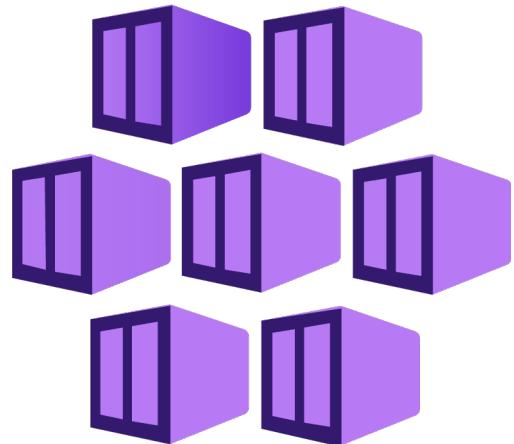
Subjects: Group: Azure AD Group
with Object ID xxx has read only
access to AKS Cluster

Azure Active Directory & Kubernetes RBAC



Azure Active Directory & Kubernetes RBAC

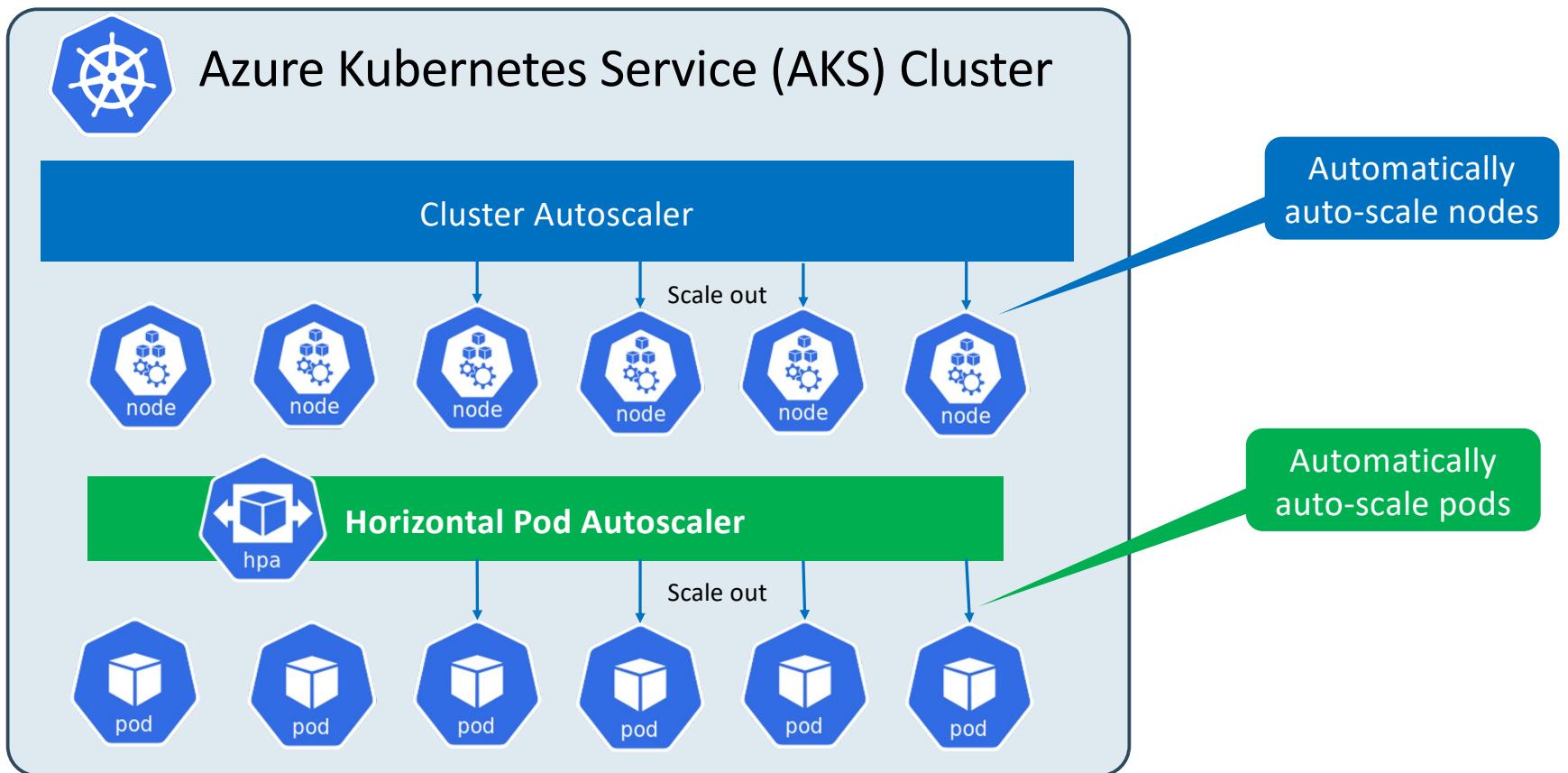


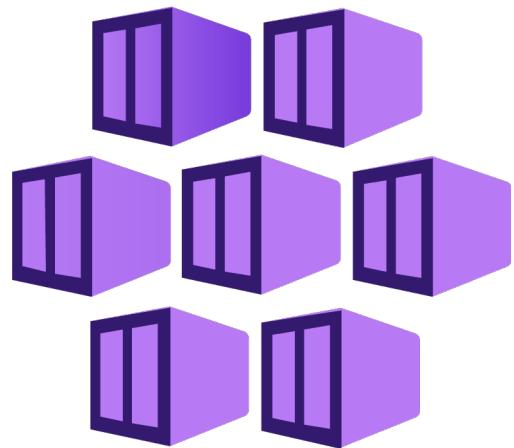


Azure AKS Autoscaling Nodes & Pods



Azure AKS – Autoscaling Nodes & Pods



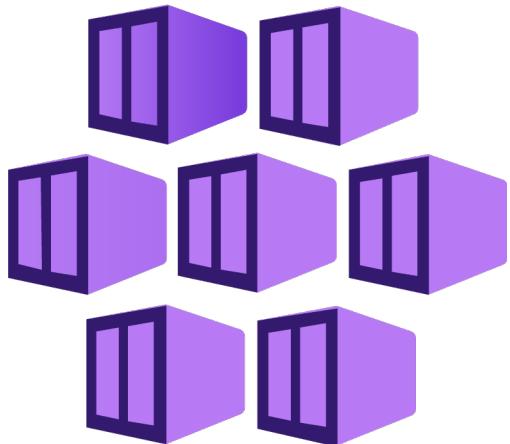


Azure AKS Autoscaling Cluster Autoscaler

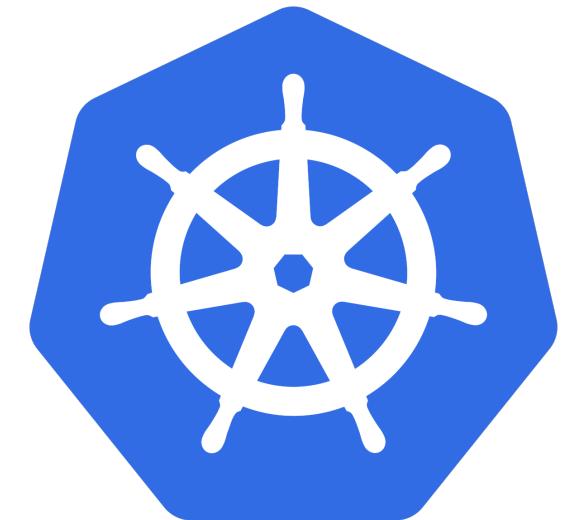


Cluster Autoscaler - Introduction

- Cluster Autoscaler is a tool that automatically adjusts the size of a Kubernetes cluster when one of the following conditions is true:
- There are pods that failed to run in the cluster due to insufficient resources.
- There are nodes in the cluster that have been underutilized for an extended period of time and their pods can be placed on other existing nodes.
- The Cluster Autoscaler modifies our nodepools so that they scale out when we need more resources and scale in when we have underutilized resources.



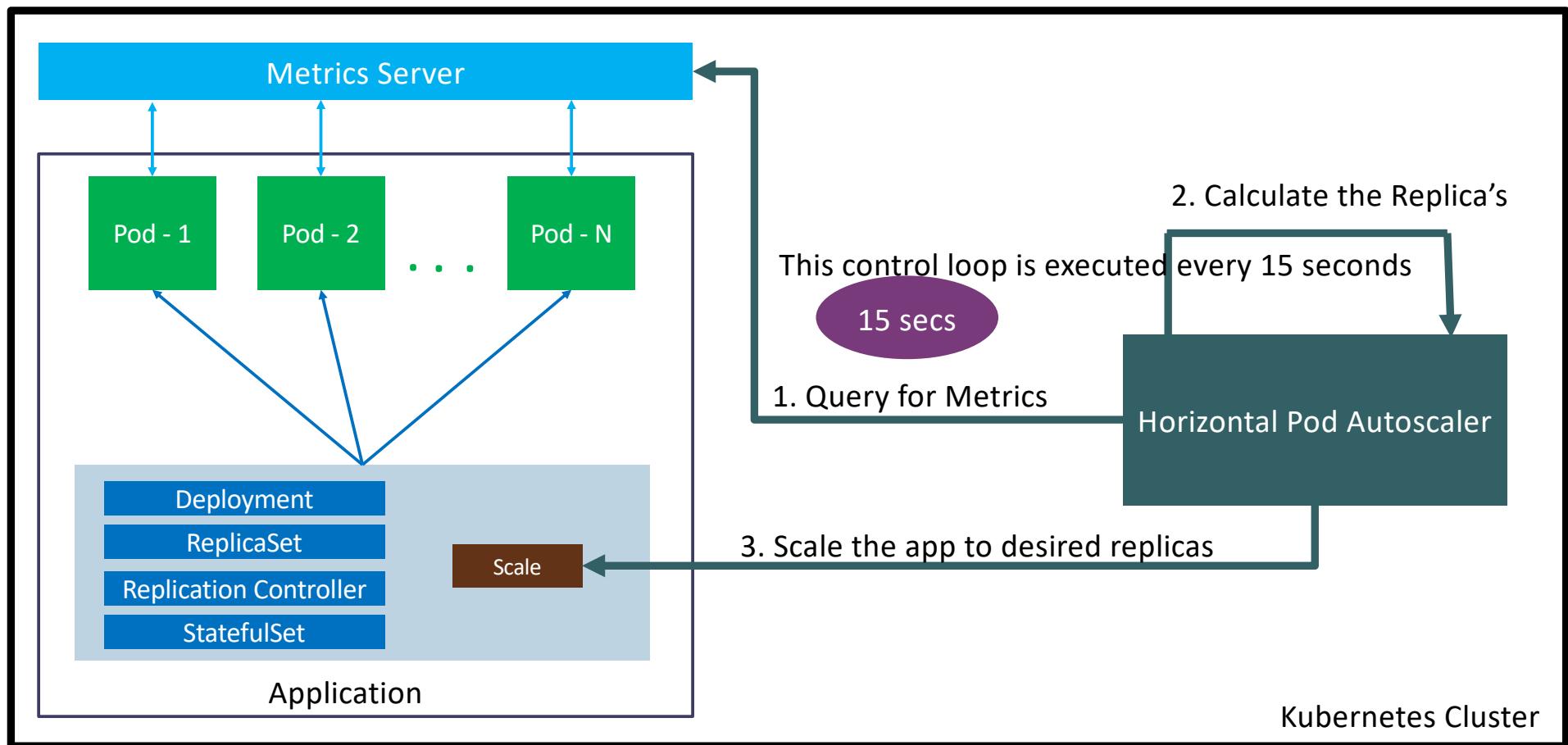
Azure AKS Autoscaling Horizontal Pod Autoscaling



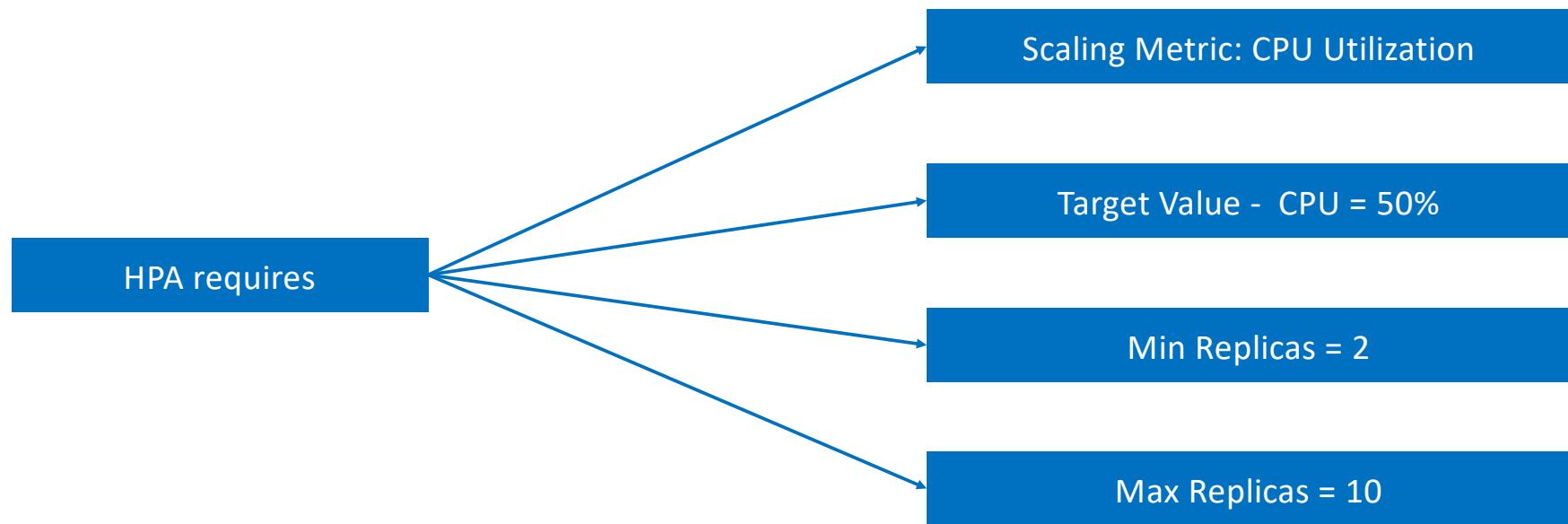
Horizontal Pod Autoscaler – HPA - Introduction

- In a very simple note Horizontal Scaling means **increasing and decreasing** the number of **Replicas (Pods)**
- HPA **automatically scales** the number of pods in a deployment, replication controller, or replica set, stateful set based on that resource's **CPU utilization**.
- This can help our applications **scale out to meet increased demand** or **scale in when resources are not needed**, thus freeing up your worker nodes for other applications.
- When we set a **target CPU utilization percentage**, the HPA scales our application in or out to try to meet that **target**.
- HPA needs **Kubernetes metrics server** to verify CPU metrics of a pod.
- We **do not need** to **deploy or install the HPA** on our cluster to begin scaling our applications, its out of the box available as a **default Kubernetes API** resource.

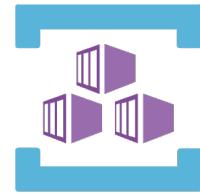
How HPA works?



How is HPA configured?



```
kubectl autoscale deployment demo-deployment --cpu-percent=50 --min=1 --max=10
```



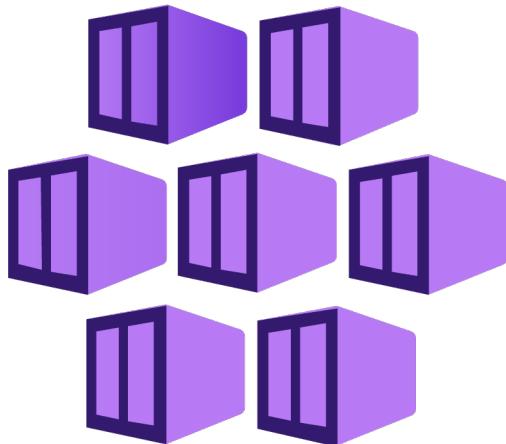
AKS Linux NodePools



AKS Windows NodePools

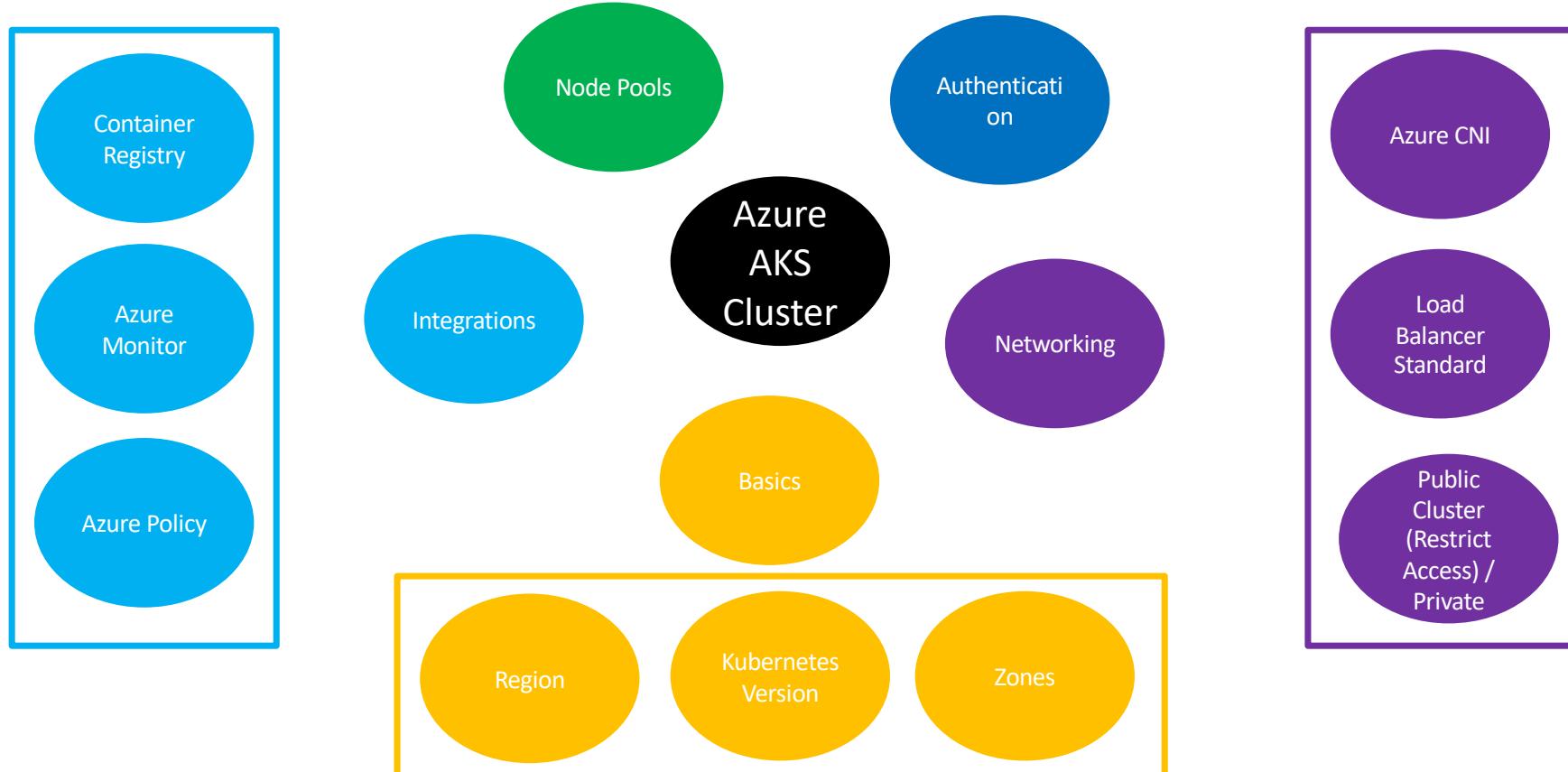
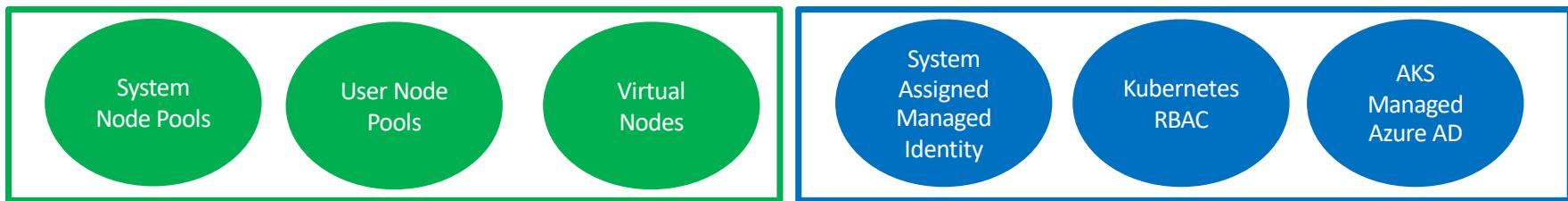


AKS Virtual Nodes

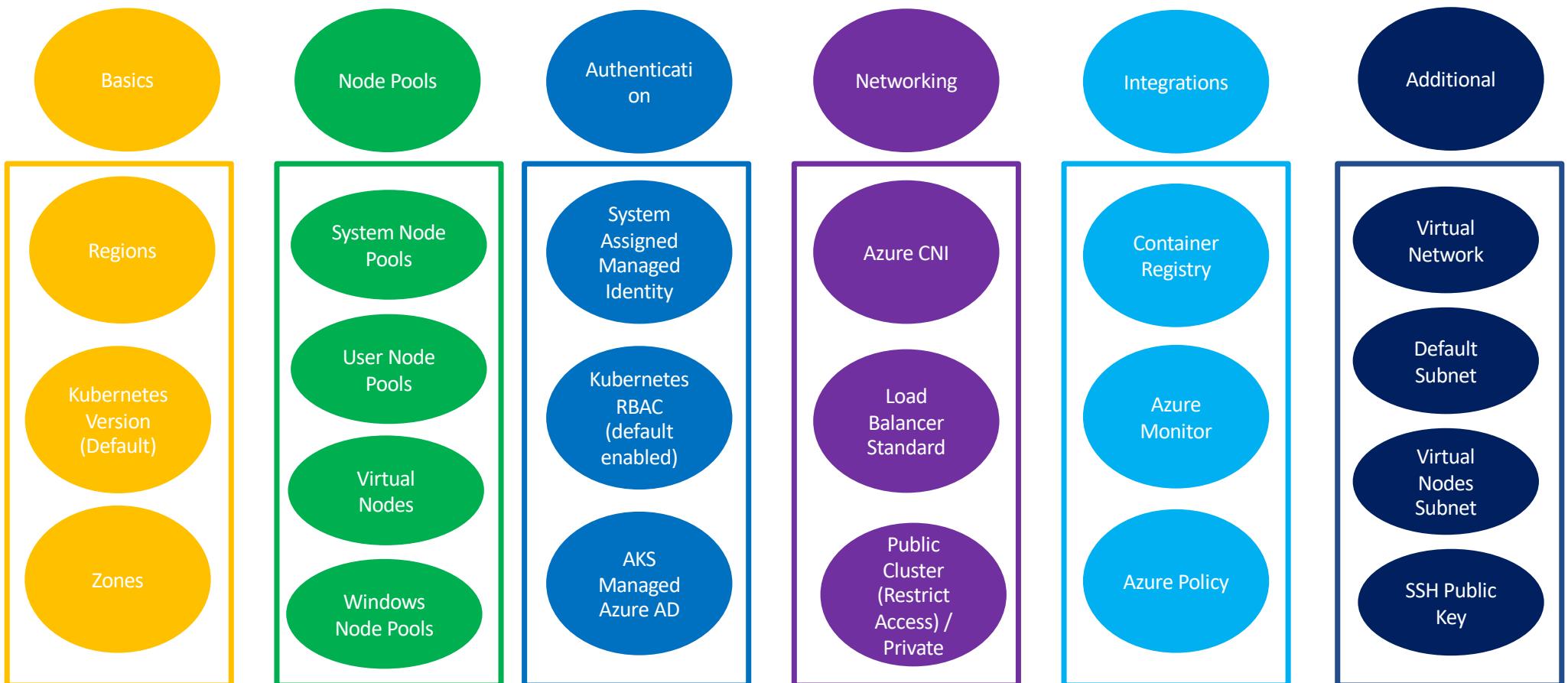


Azure AKS Design Production Grade AKS Cluster **az aks cli**





Azure AKS Cluster



Section-1

Create AKS Cluster

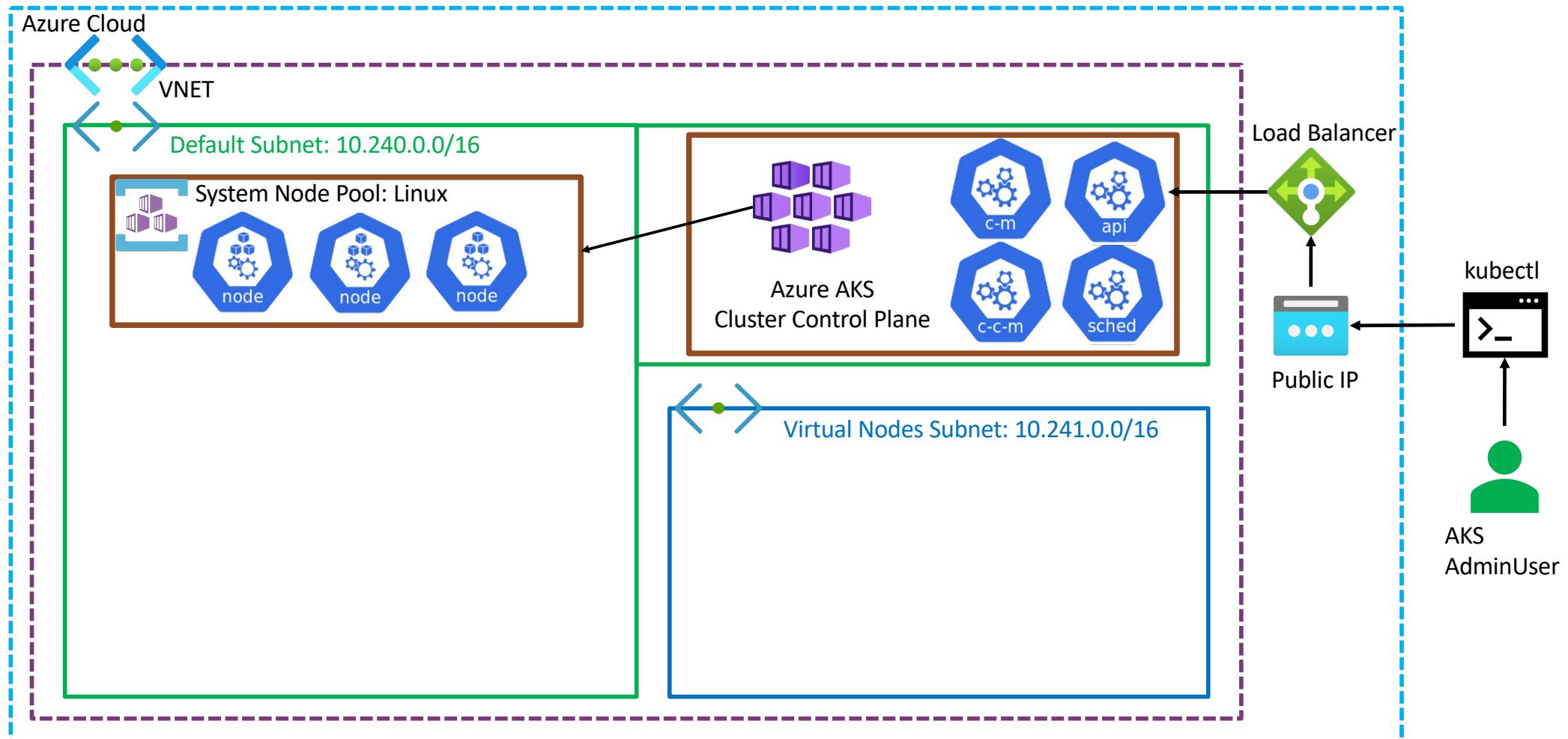
Section-2

Create
Windows, Linux
User Node
Pools and
Virtual Nodes

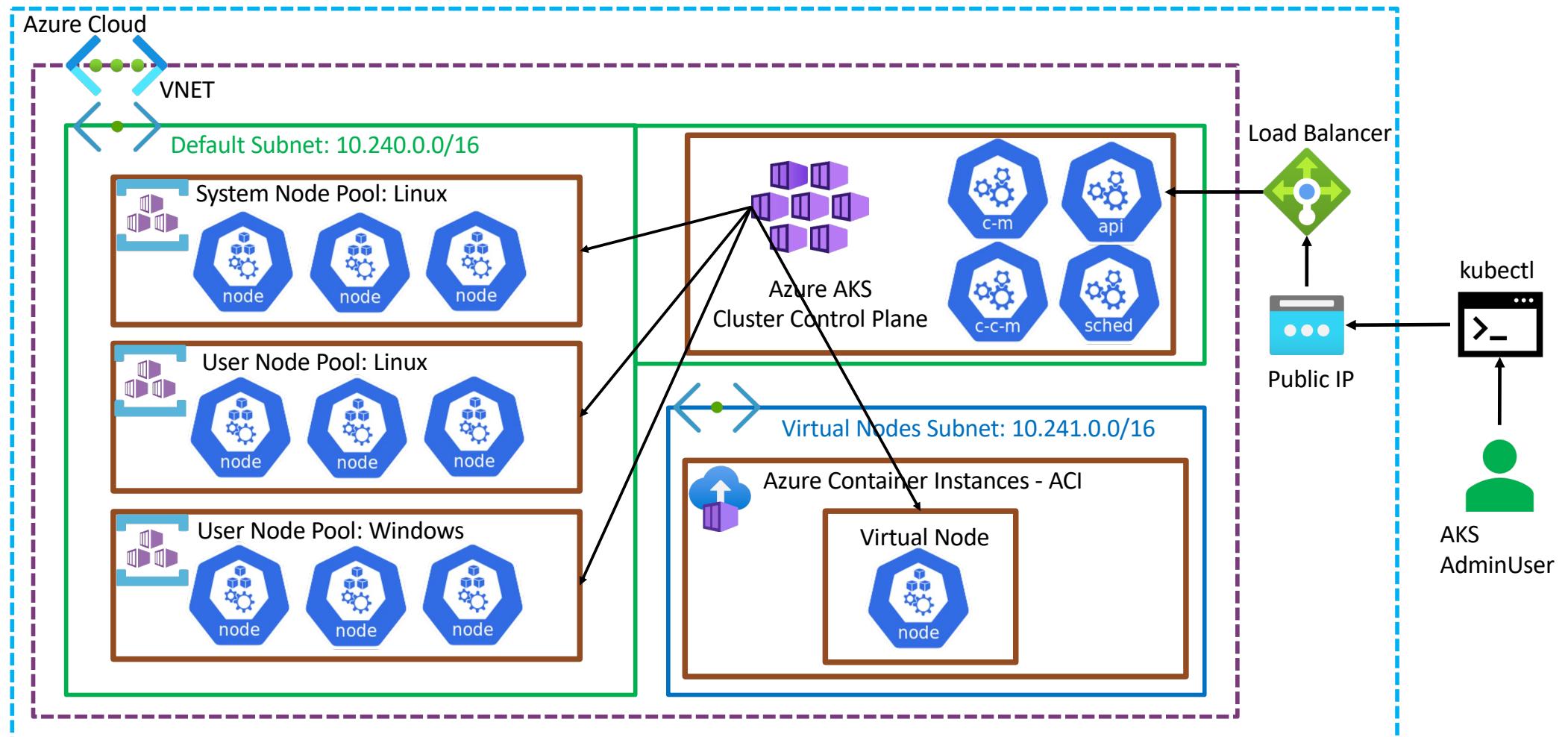
Section-3

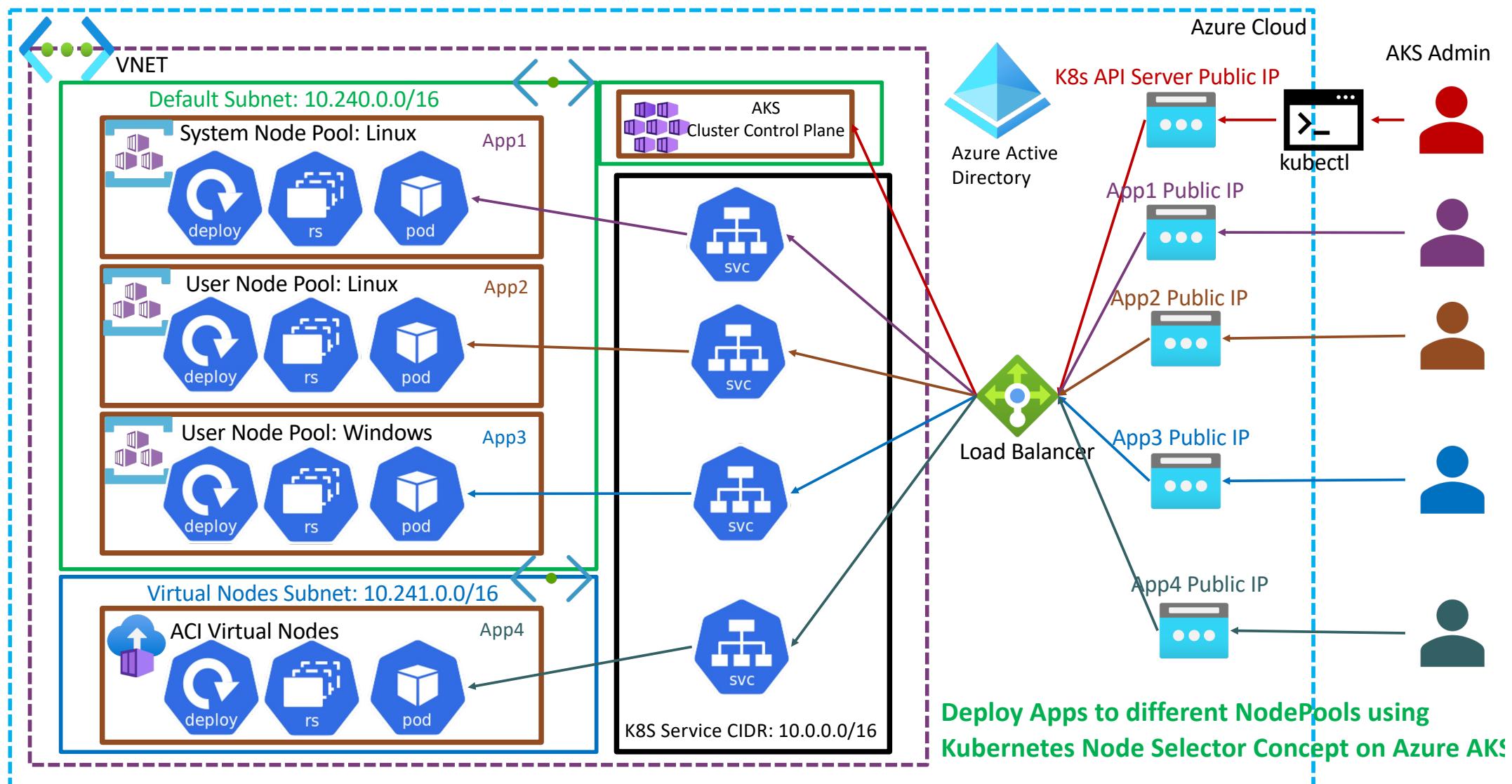
Deploy Apps to
all 4 node pools
using
Kubernetes
Node Selector
concept

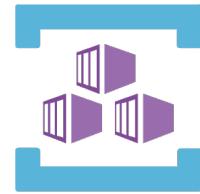
Azure AKS Cluster – Create AKS Cluster using CLI



Azure AKS Cluster – Node Pools (Linux, Windows, Virtual)







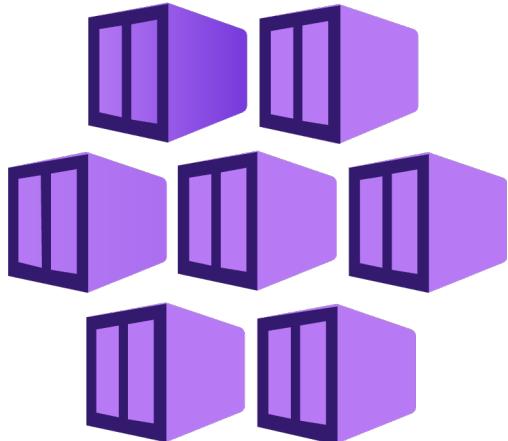
AKS Linux NodePools



AKS Windows NodePools



AKS Virtual Nodes



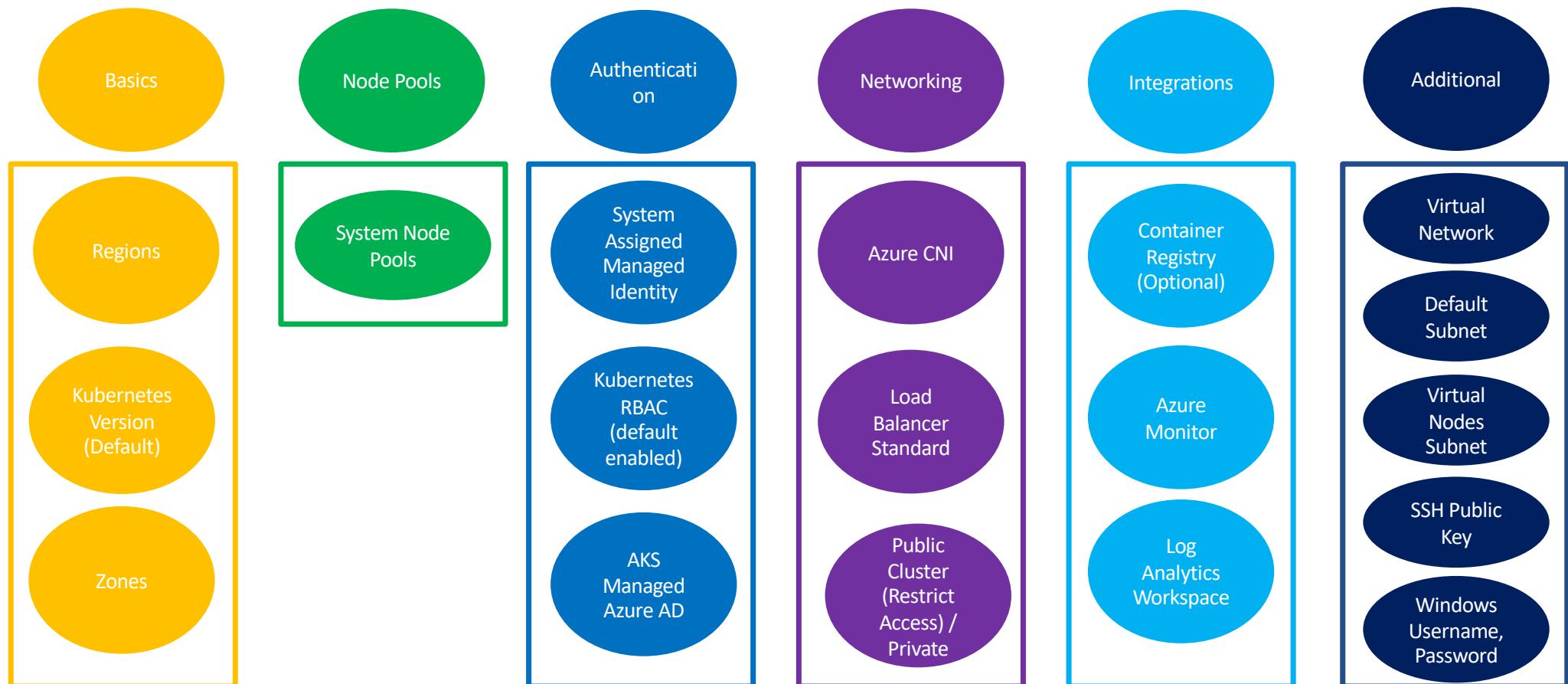
Azure AKS

Create AKS Cluster using **az aks cli**

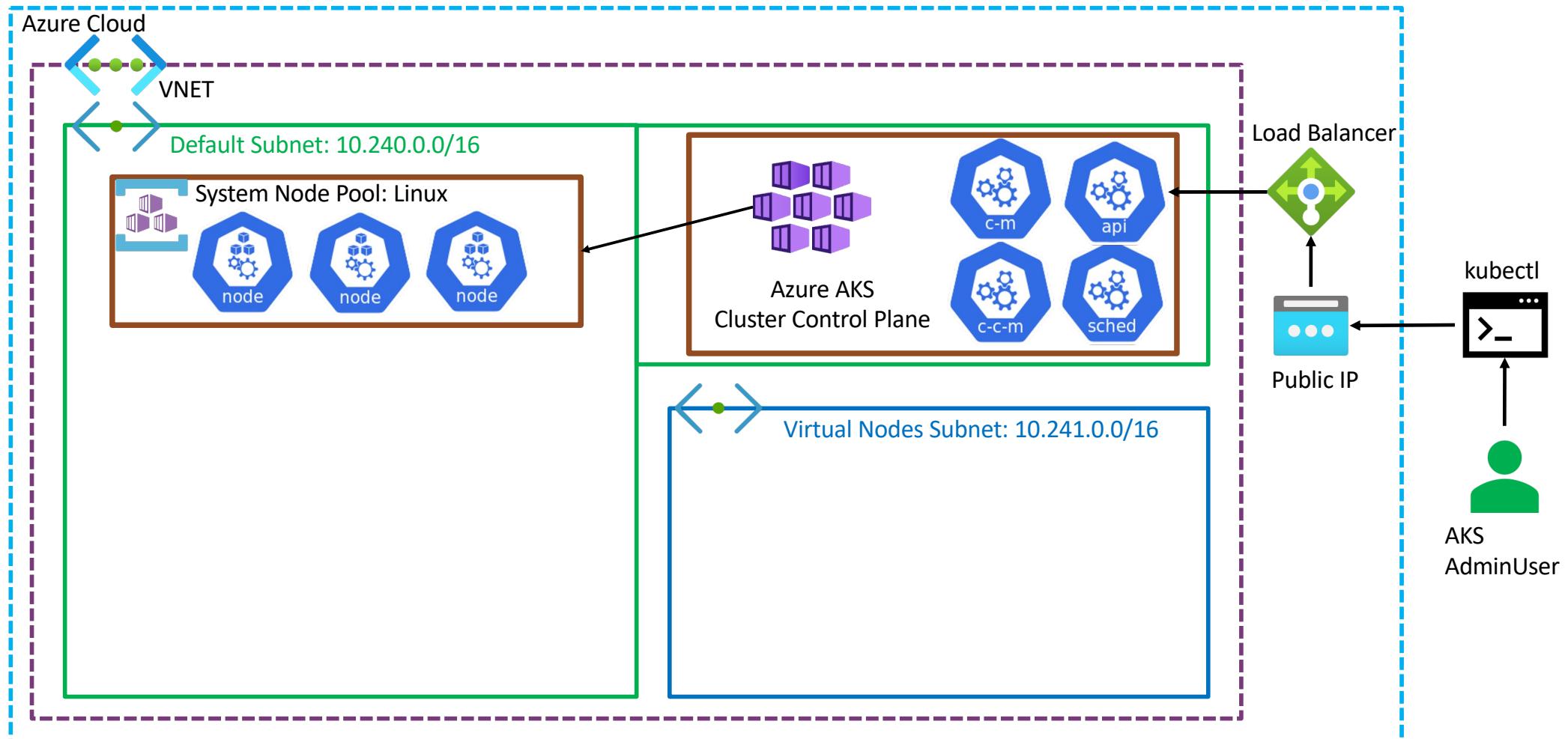


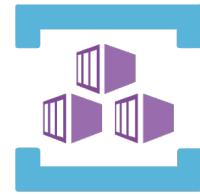
Module-1: Create AKS Cluster

Azure AKS Cluster



Azure AKS Cluster – Create AKS Cluster using CLI





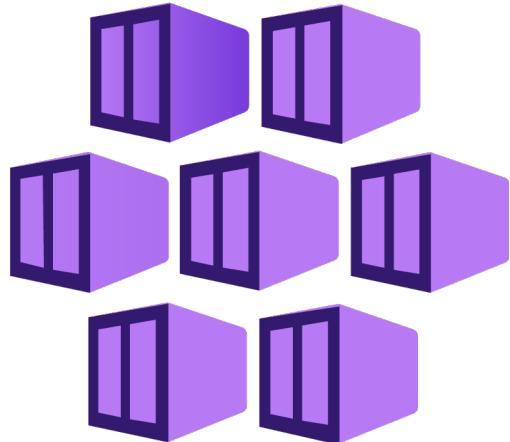
AKS Linux NodePools



AKS Windows NodePools



AKS Virtual Nodes



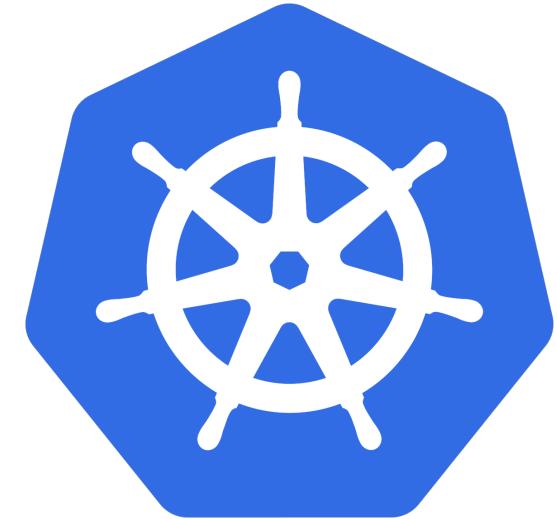
Azure AKS

Create Node Pools

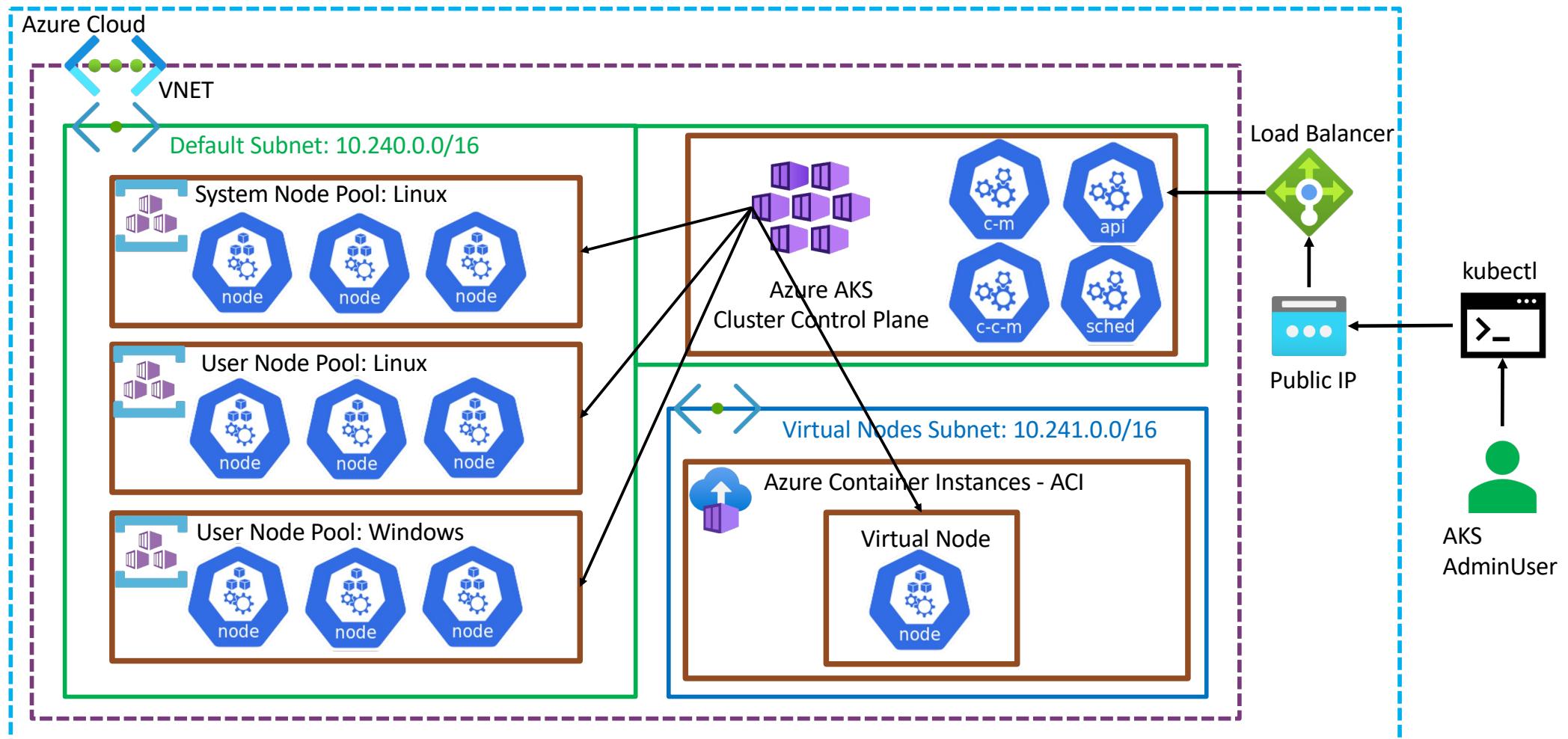
Windows, Linux and

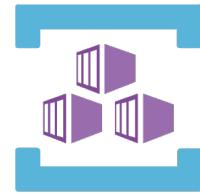
Virtual Nodes using

az aks cli



Azure AKS Cluster – Node Pools (Linux, Windows, Virtual)





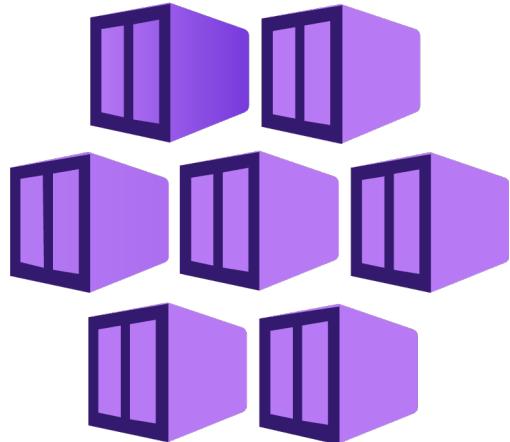
AKS Linux NodePools



AKS Windows NodePools



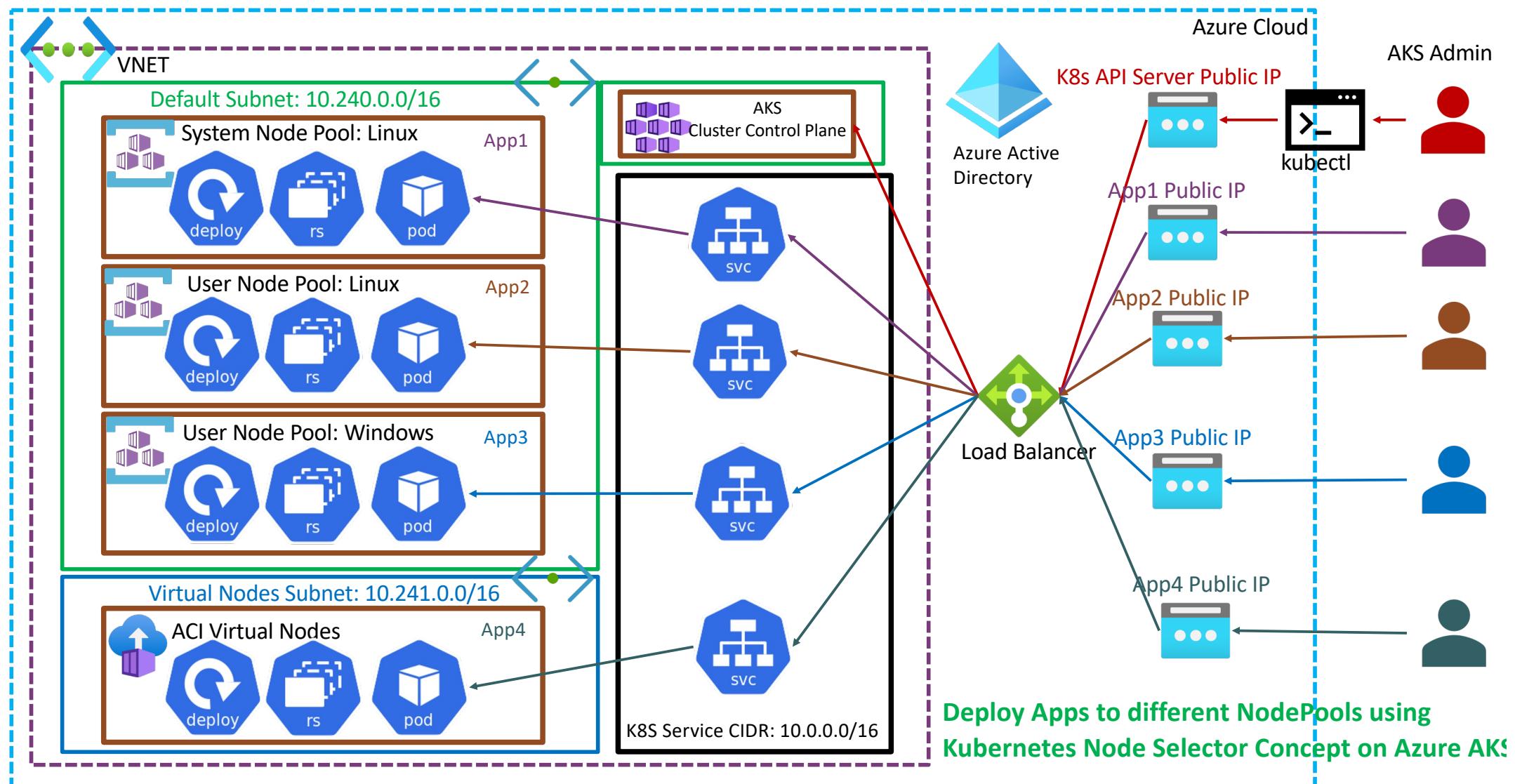
AKS Virtual Nodes

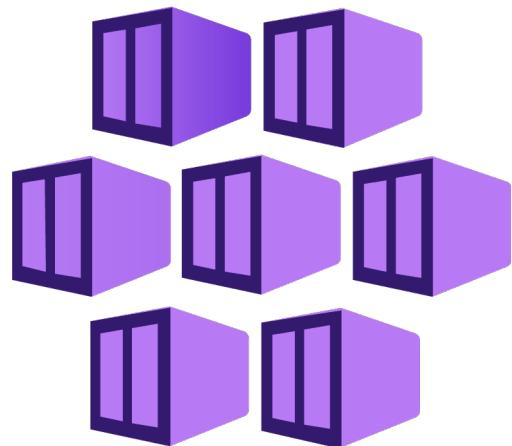


Azure AKS
Deploy Apps to
different NodePools
using Node Selectors

az aks cli

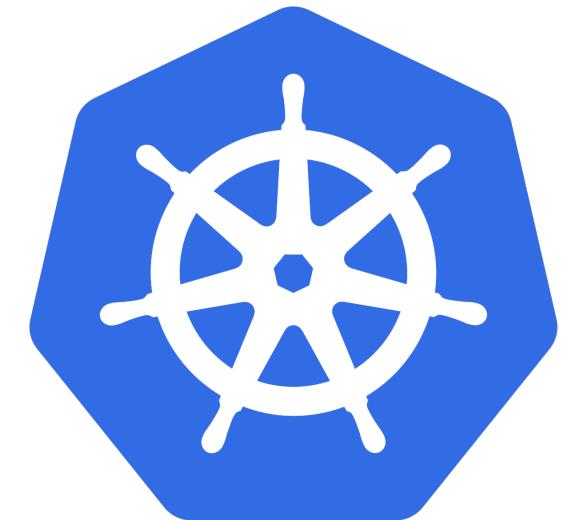


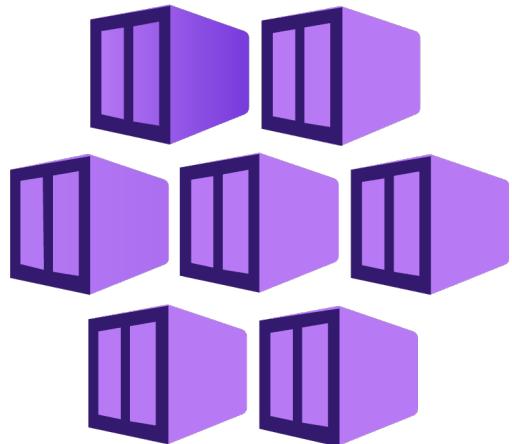




Azure AKS

Terraform Basics

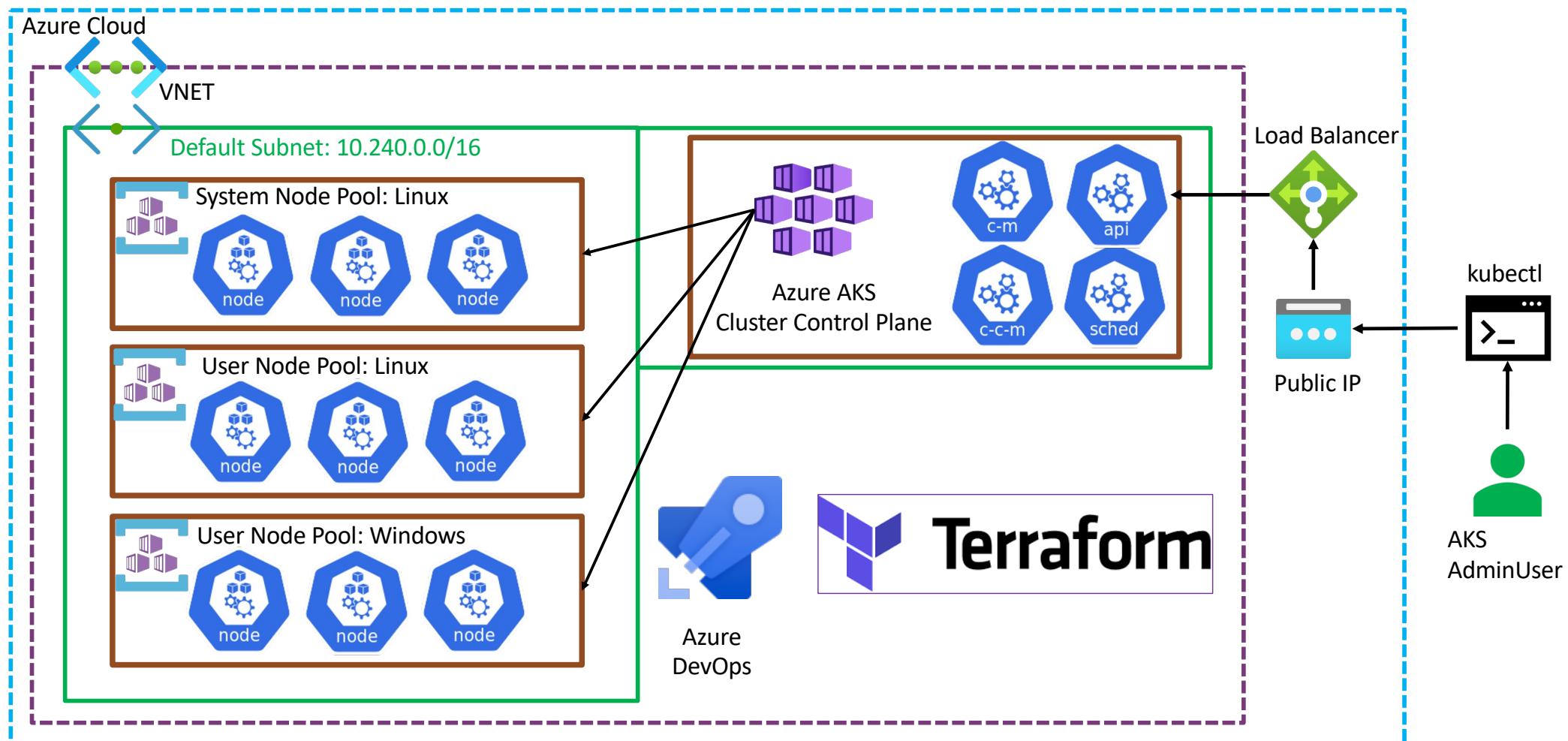




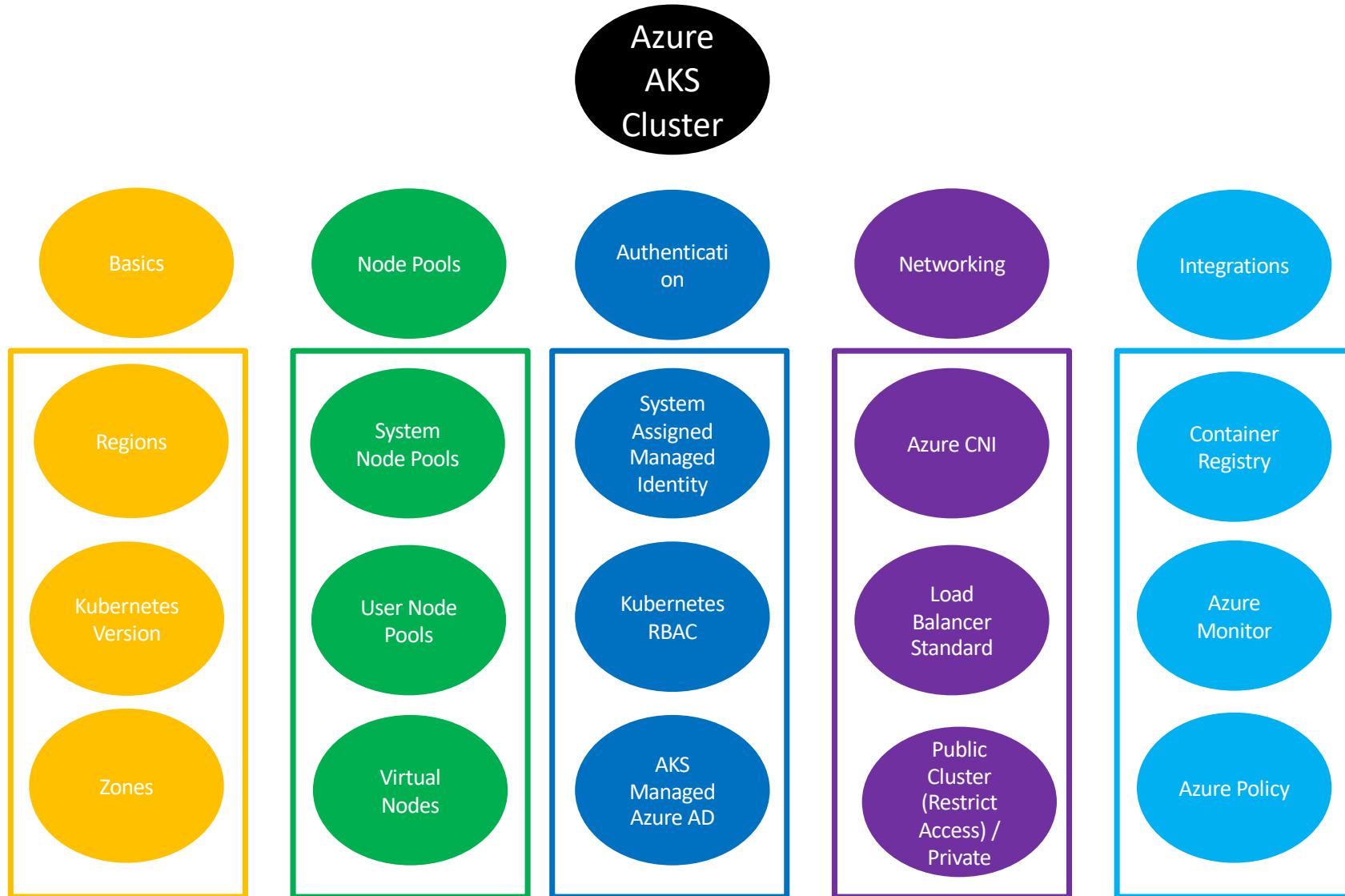
Azure AKS
Azure DevOps
Terraform



Azure AKS Cluster – Node Pools (Linux, Windows)



BEFORE THIS ARE LIVE SLIDES



AKS Cluster - Features

Node Pools

auto_scaler_profile

default_node_pool

identity

network_profile

load_balancer_profile

role_based_access_control



AKS Cluster

role_based_access_control

azure_active_directory

windows_profile

Add Ons

aci_connector_linux

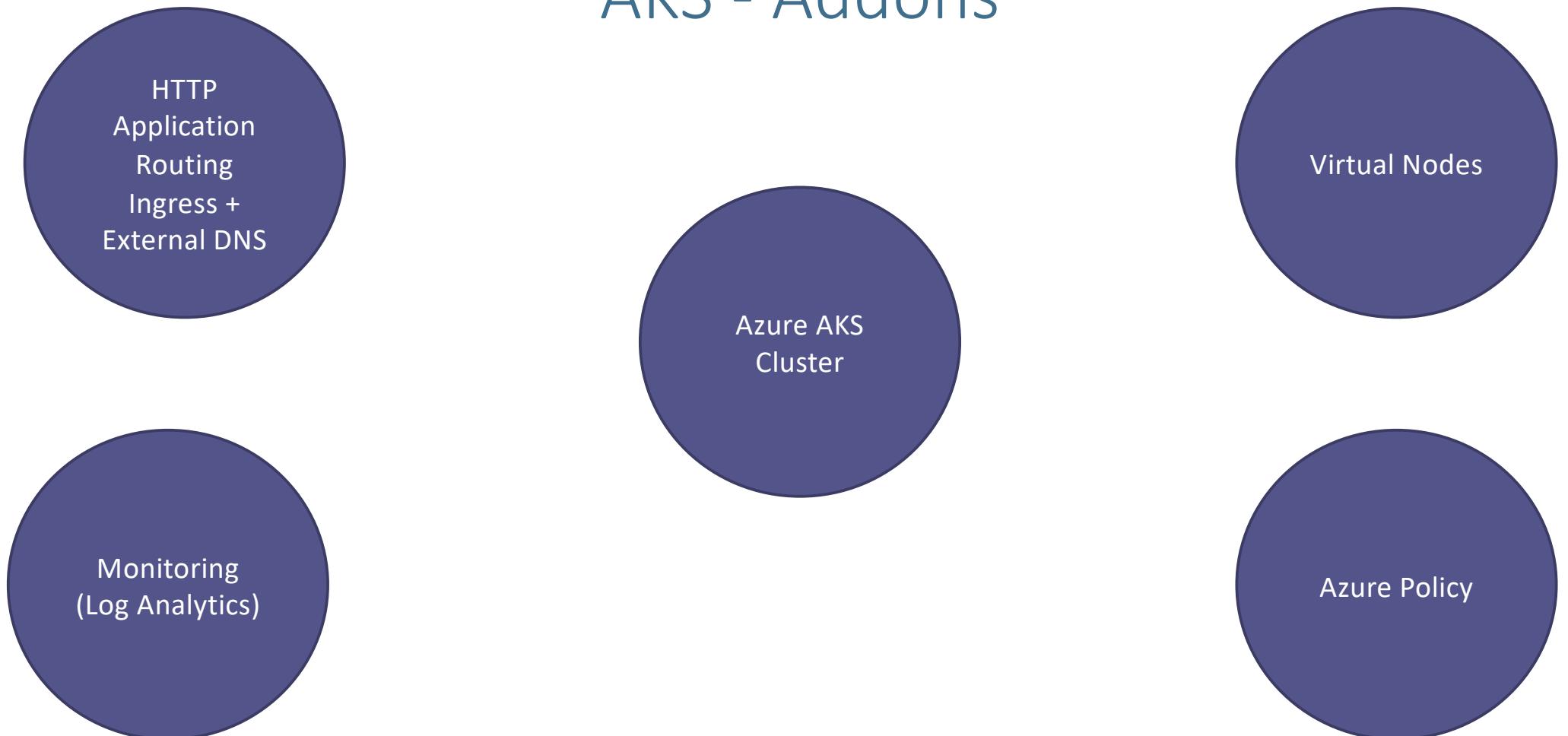
azure_policy

http_application_routing

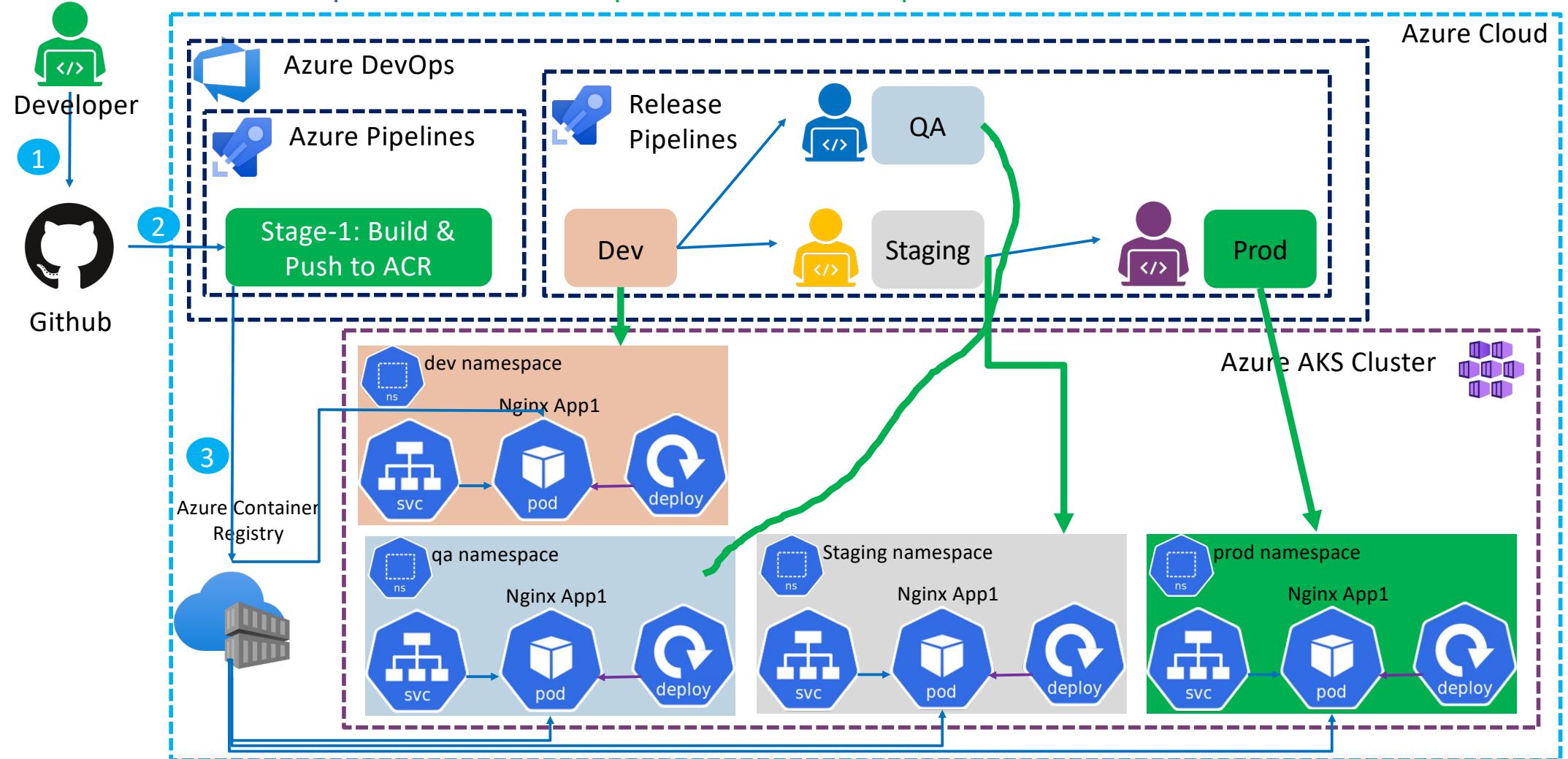
kube_dashboard

oms_agent

AKS - Addons



Azure DevOps – Release Pipelines - Backup



Which Ingress Controller to Use?

Nginx Ingress Controller (Open Source)

Free of cost, only uses AKS memory and cpu resources

Stable and very very good open source community support.

Good for smaller workloads and smaller deployment foot print

Azure Application Gateway Ingress Controller (AGIC)

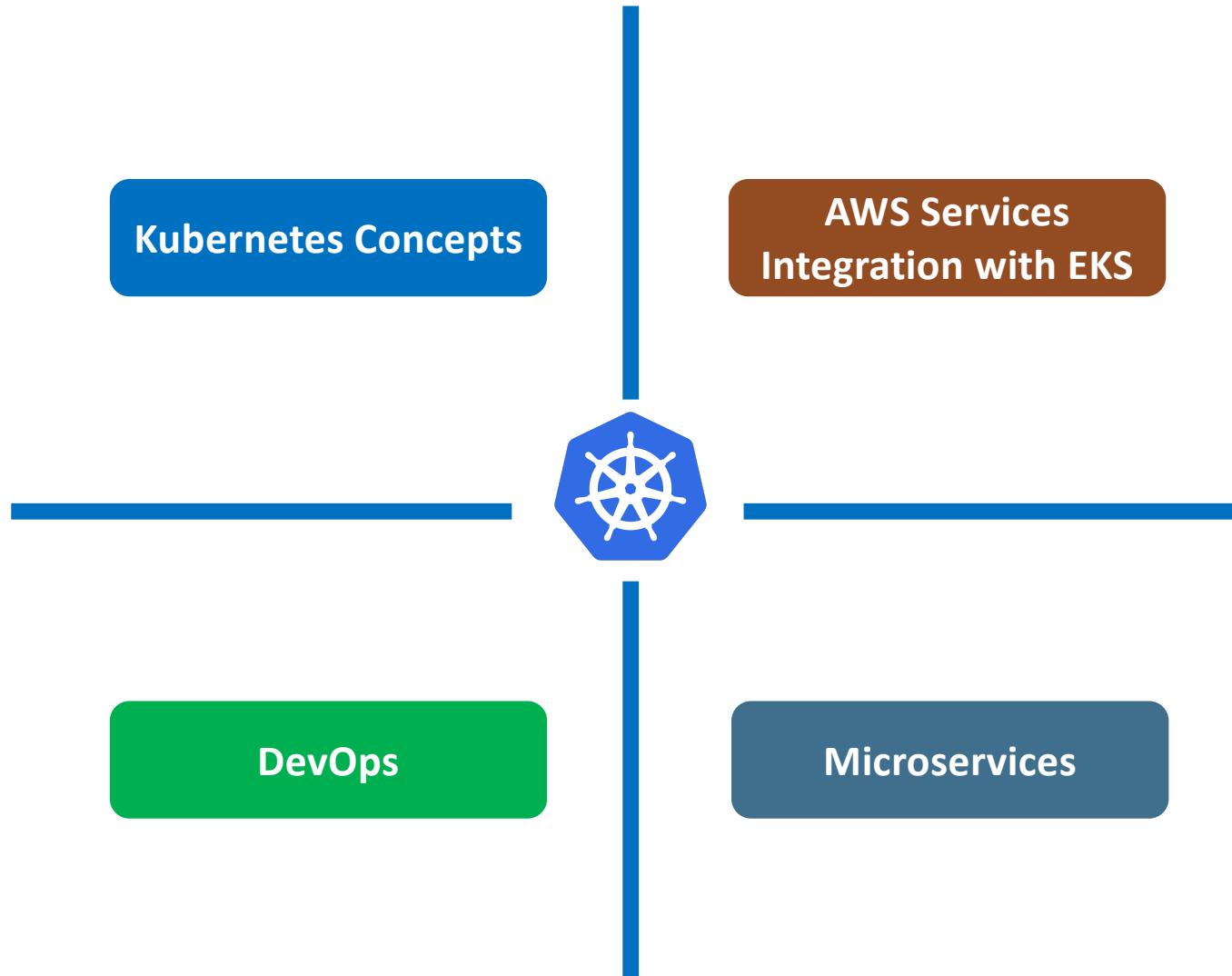
Very costly and still uses AKS memory and cpu resources

Recently launched and most of the features are in preview mode.

Good for large scale workloads, at the same time who can afford the price

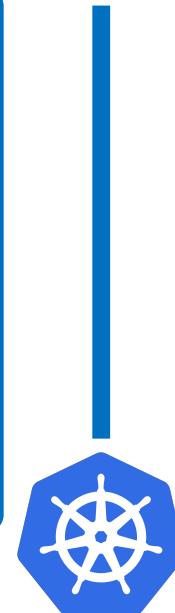


Backup Slides



Architecture	kubectl - Imperative	Persistent Volumes
Pods	Declarative with YAML	PVC
ReplicaSets	Secrets	Load Balancers
Deployments	Init Containers	Annotations
Node Port Service	Probes	Canary Deployments
Cluster IP Service	Requests & Limits	HPA
External Name Service	Namespaces	VPA
Ingress Service	Limit Range	DaemonSets
Ingress SSL	Resource Quota	Fluentd for logs
Ingress & External DNS	Storage Classes	ConfigMaps

K
U
B
E
R
N
E
T
S



A
W
S

S
E
R
V
I
C
E



DevOps

AWS Developer Tools



AWS
CodeCommit



AWS CodeBuild



AWS CodePipeline

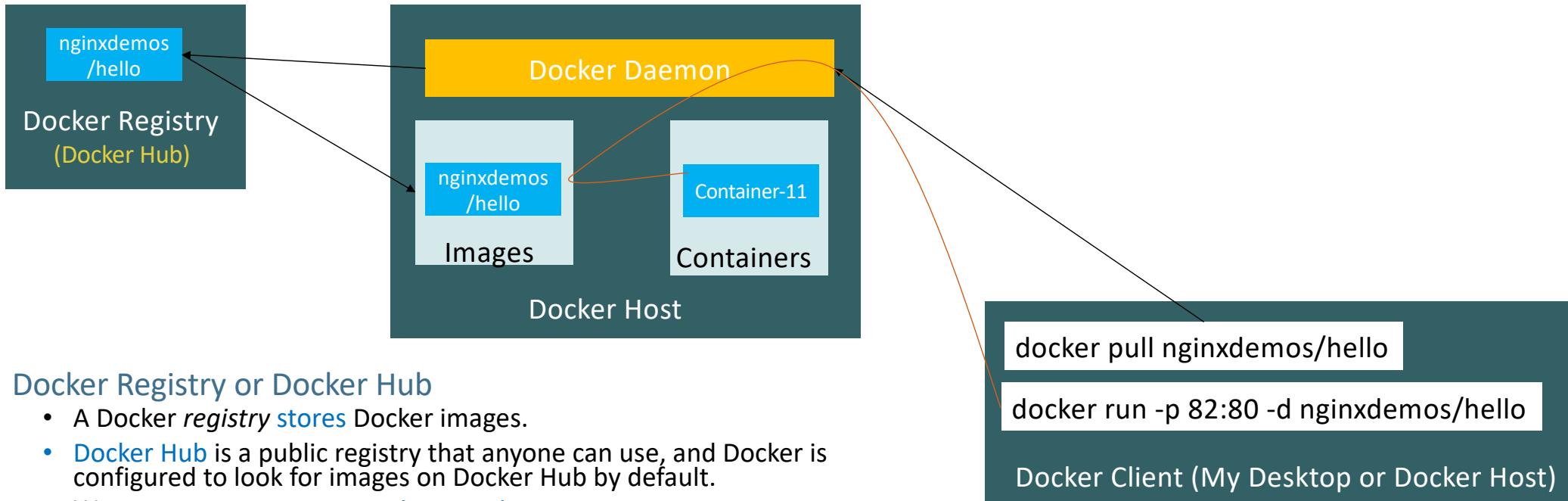
Microservices

Service Discovery

Distributed Tracing

Canary Deployments

Docker - Fundamentals



- **Docker Registry or Docker Hub**

- A Docker *registry* stores Docker images.
- **Docker Hub** is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
- We can even run our own **private registry**.
- When we use the **docker pull** or **docker run** commands, the required images are pulled from our configured registry.
- When we use the **docker push** command, our image is pushed to our configured registry.

Kubernetes - Imperative & Declarative

