Azure AKS Kubernetes - Masterclass | Azure DevOps, Terraform

Kalyan Reddy Daida

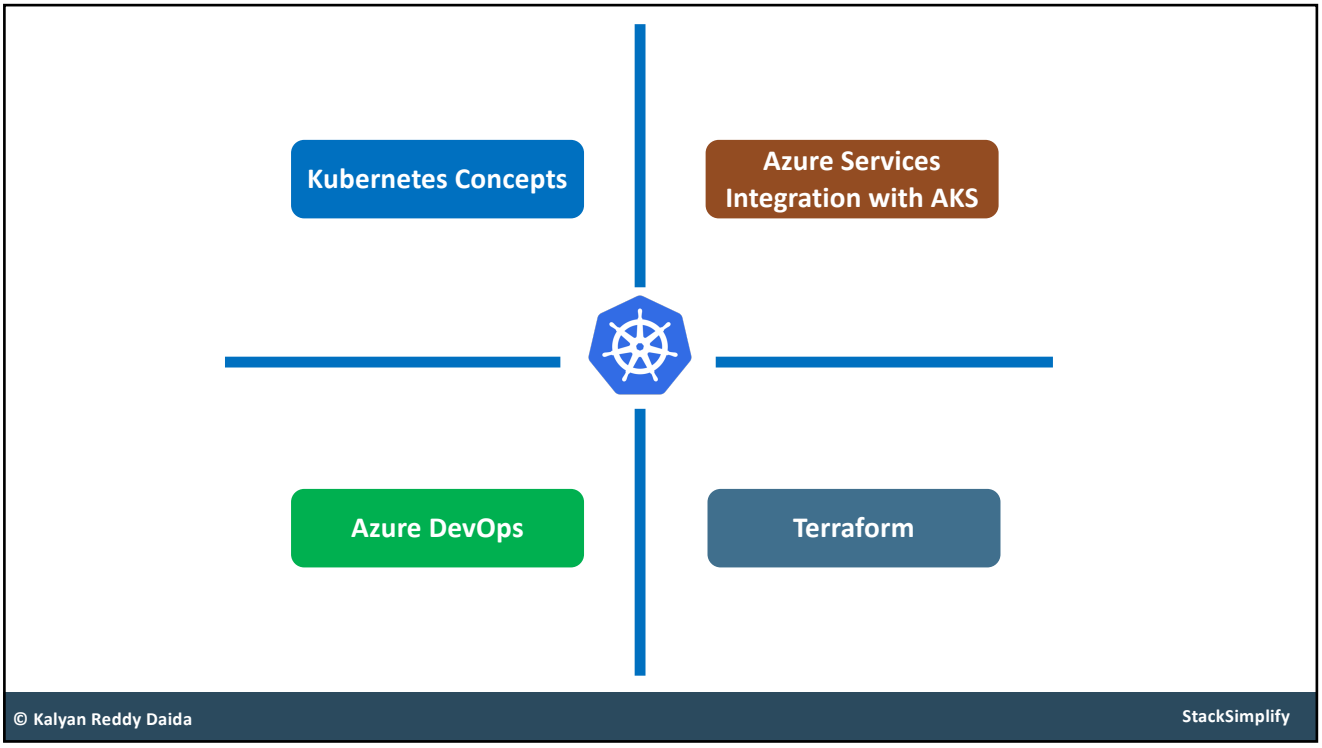
1

STACKSIMPLIFY

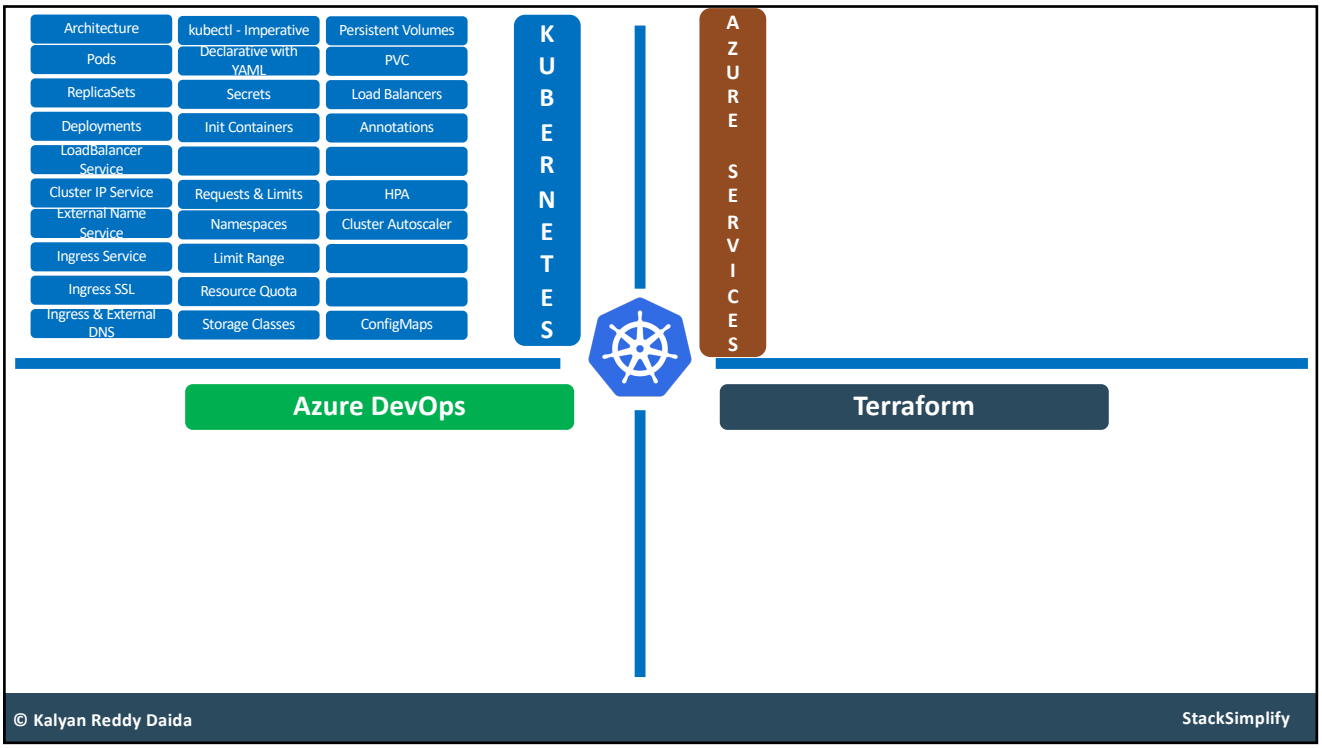
Kubernetes On Cloud Roadmap

- **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**

2

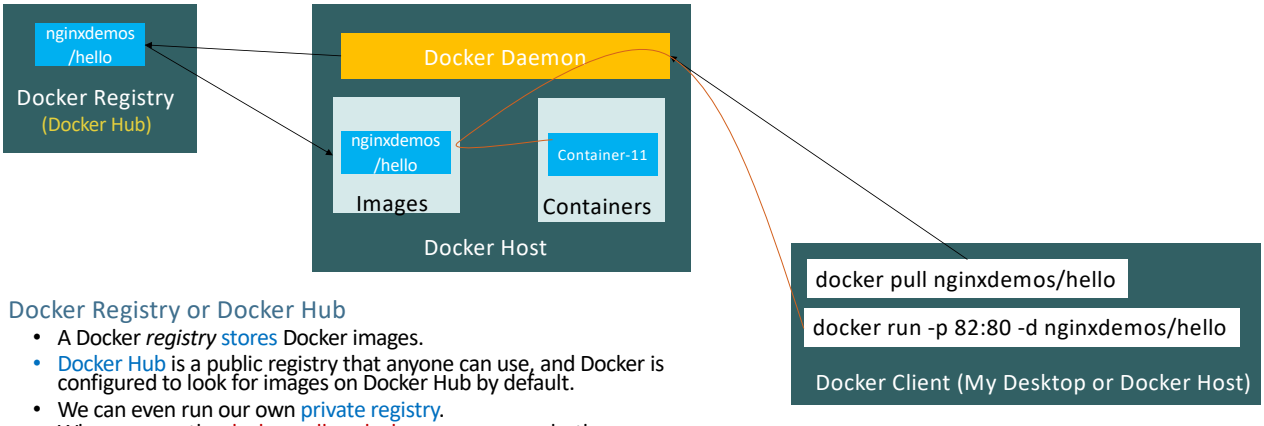

Kubernetes Concepts
Azure Services Integration with AKS
Azure DevOps
Terraform
© Kalyan Reddy Daida
StackSimplify

3


Architecture
Pods
ReplicaSets
Deployments
LoadBalancer Service
Cluster IP Service
External Name Service
Ingress Service
Ingress SSL
Ingress & External DNS
kubectl - Imperative
Declarative with YAML
Secrets
Init Containers
Requests & Limits
Namespaces
Limit Range
Resource Quota
Storage Classes
Persistent Volumes
PVC
Load Balancers
Annotations
HPA
Cluster Autoscaler
ConfigMaps
KUBERNETES
AZURE SERVICES
Azure DevOps
Terraform
© Kalyan Reddy Daida
StackSimplify

4

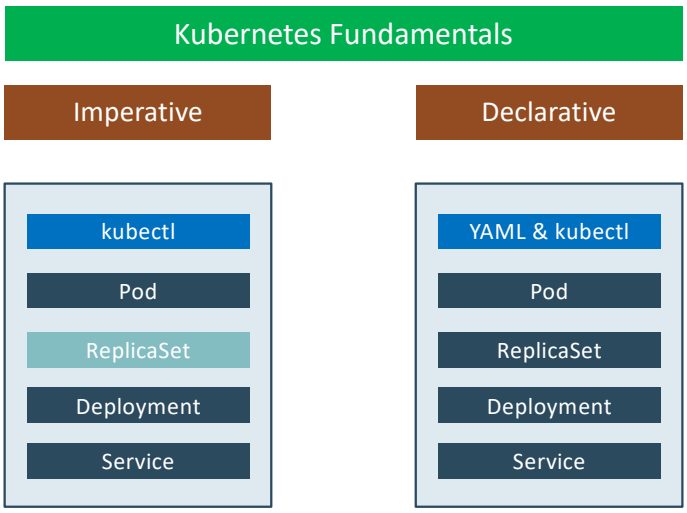
Docker - Fundamentals



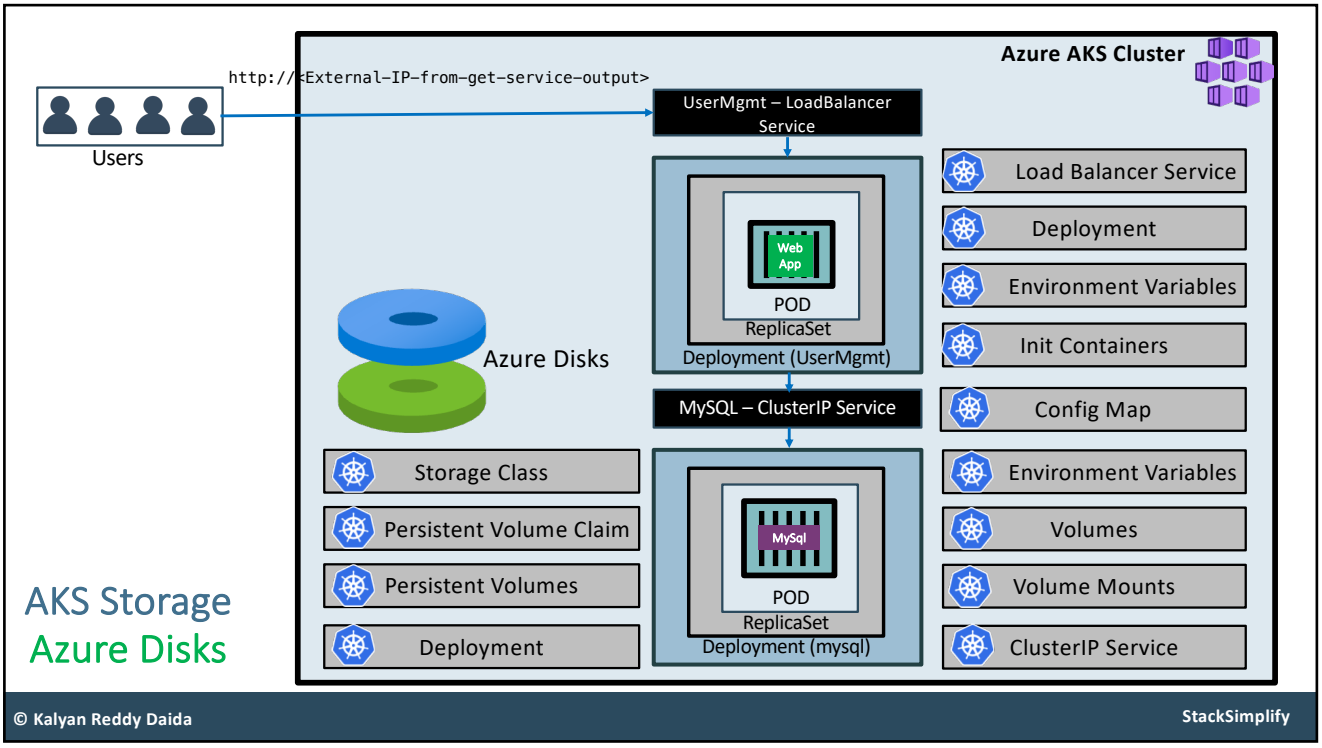
- Docker Registry or Docker Hub
  - A Docker *registry* stores Docker images.
  - Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - We can even run our own private registry.
  - When we use the docker pull or docker run commands,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

5

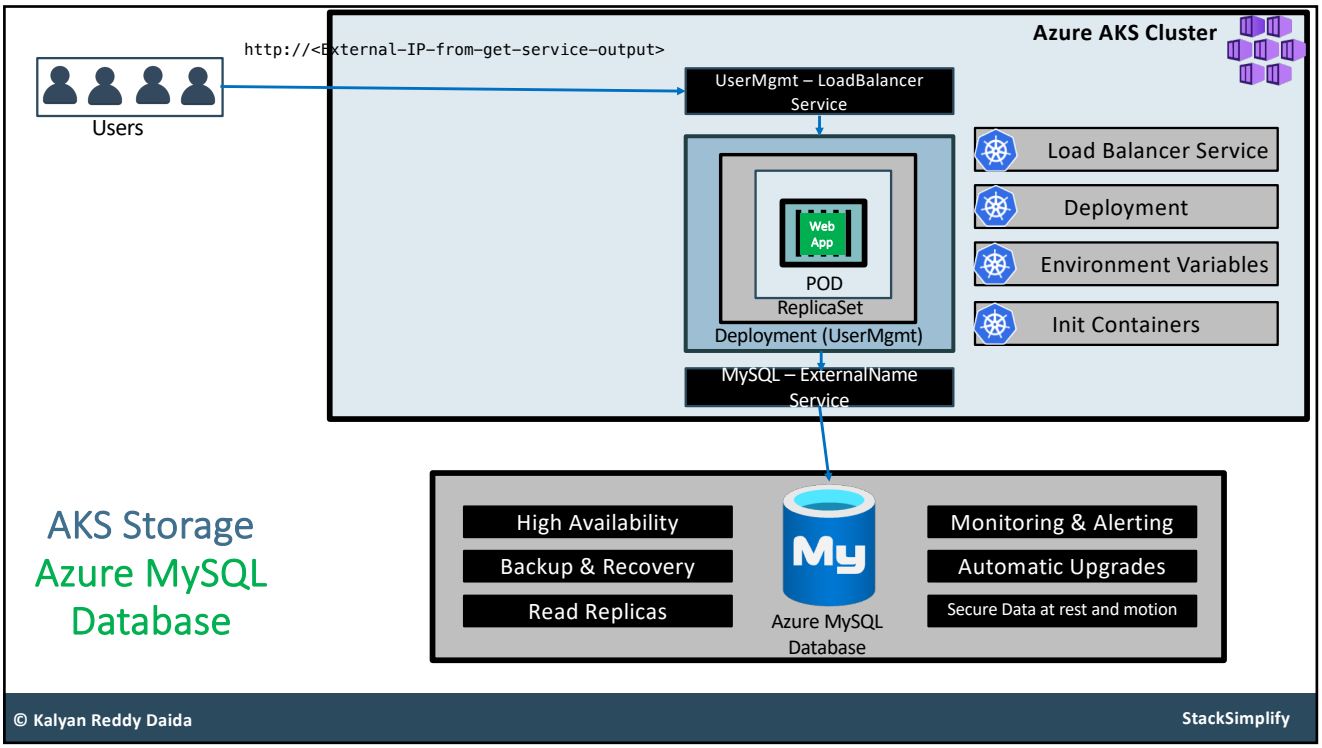
Kubernetes - Imperative & Declarative



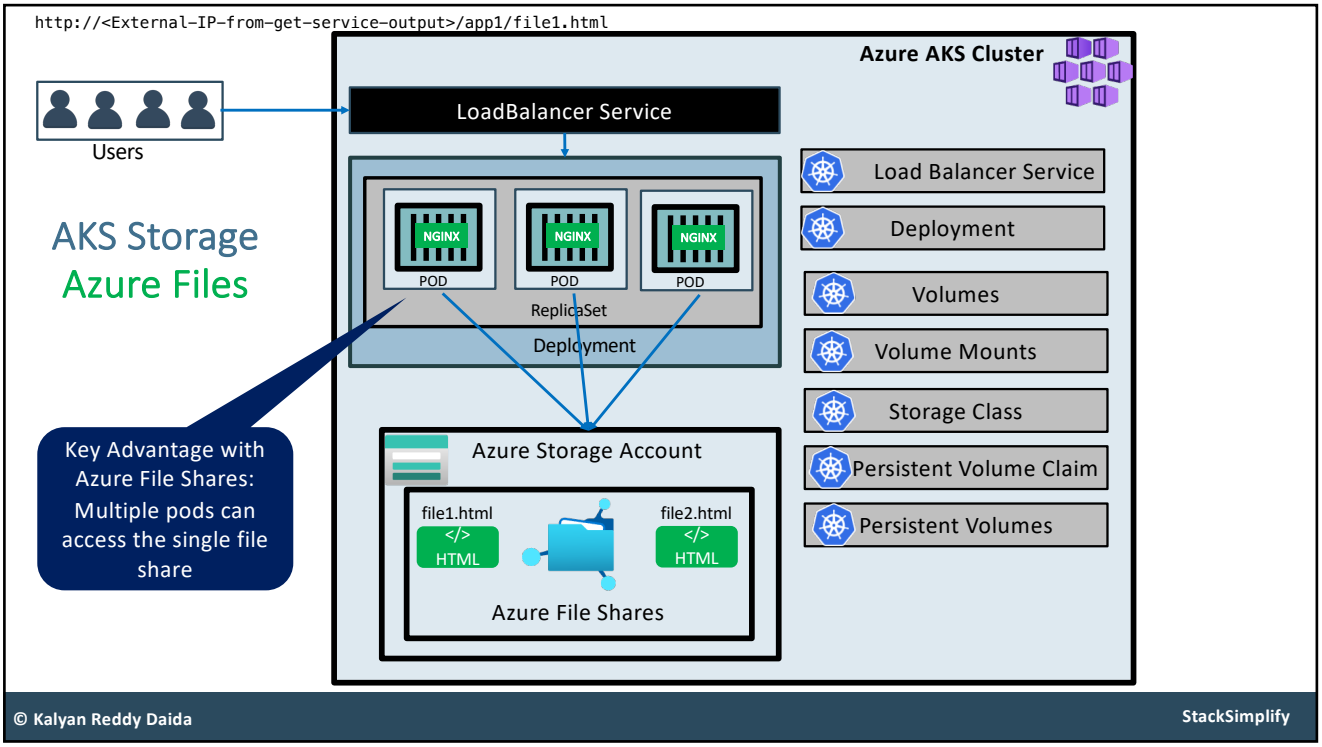
6


Azure AKS Cluster
http://<External-IP-from-get-service-output>
Users
UserMgmt – LoadBalancer Service
Load Balancer Service
Deployment
Environment Variables
Init Containers
Web App
POD
ReplicaSet
Deployment (UserMgmt)
Azure Disks
MySQL – ClusterIP Service
Config Map
Storage Class
Persistent Volume Claim
Persistent Volumes
Deployment
MySql
POD
ReplicaSet
Deployment (mysql)
Environment Variables
Volumes
Volume Mounts
ClusterIP Service
AKS Storage
Azure Disks
© Kalyan Reddy Daida
StackSimplify

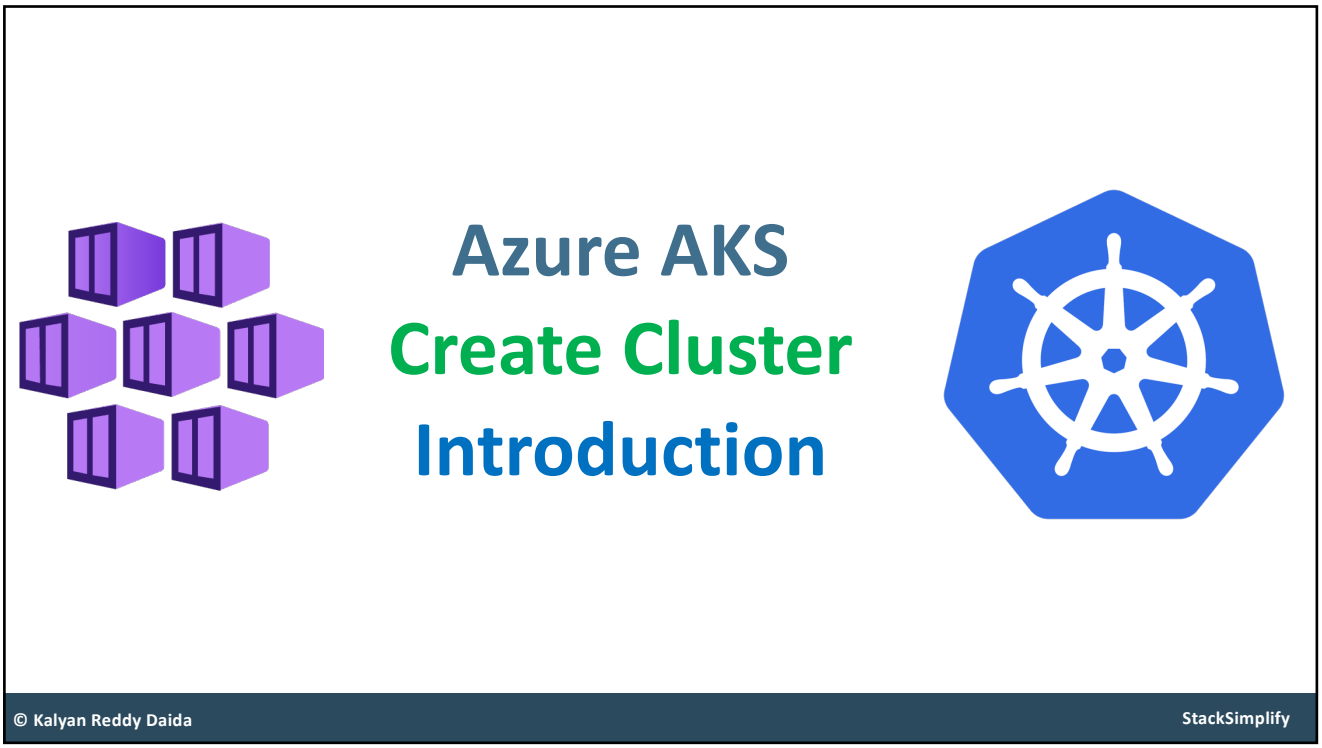
7


Azure AKS Cluster
http://<External-IP-from-get-service-output>
Users
UserMgmt – LoadBalancer Service
Load Balancer Service
Deployment
Environment Variables
Init Containers
Web App
POD
ReplicaSet
Deployment (UserMgmt)
MySQL – ExternalName Service
High Availability
Backup & Recovery
Read Replicas
Azure MySQL Database
Monitoring & Alerting
Automatic Upgrades
Secure Data at rest and motion
AKS Storage
Azure MySQL Database
© Kalyan Reddy Daida
StackSimplify

8


http://<External-IP-from-get-service-output>/app1/file1.html
Azure AKS Cluster
Users
LoadBalancer Service
NGINX
POD
ReplicaSet
Deployment
AKS Storage
Azure Files
Key Advantage with Azure File Shares: Multiple pods can access the single file share
Azure Storage Account
file1.html
HTML
file2.html
HTML
Azure File Shares
Load Balancer Service
Deployment
Volumes
Volume Mounts
Storage Class
Persistent Volume Claim
Persistent Volumes
© Kalyan Reddy Daida
StackSimplify

9


Azure AKS
Create Cluster
Introduction
© Kalyan Reddy Daida
StackSimplify

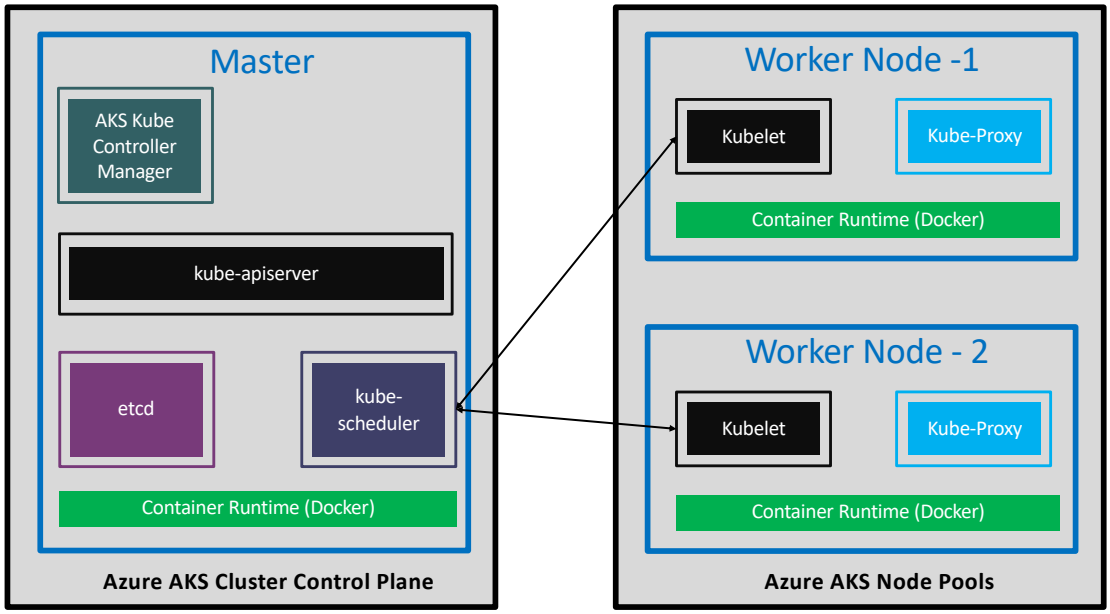
10

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

11

AKS Kubernetes - Architecture

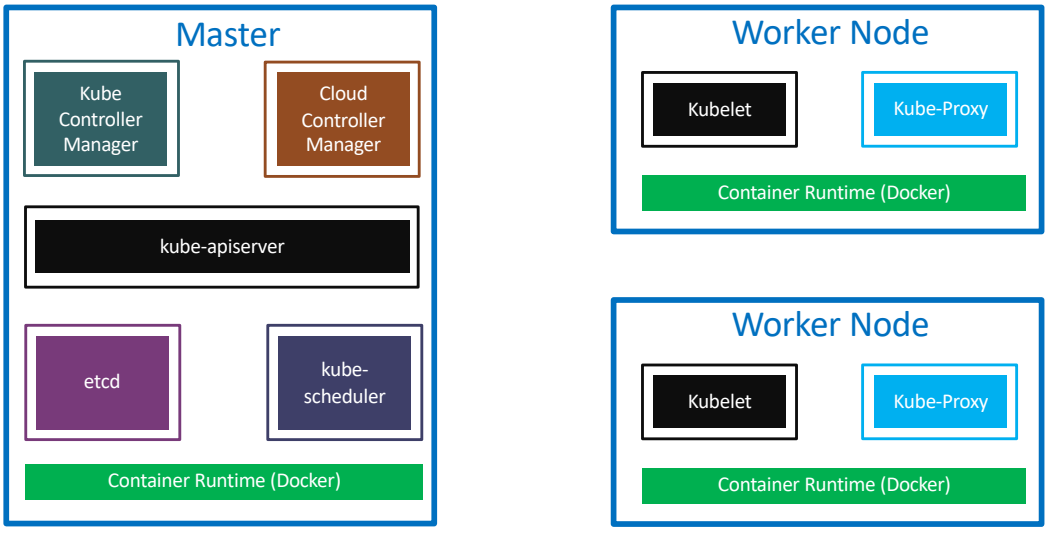


12

Kubernetes Architecture

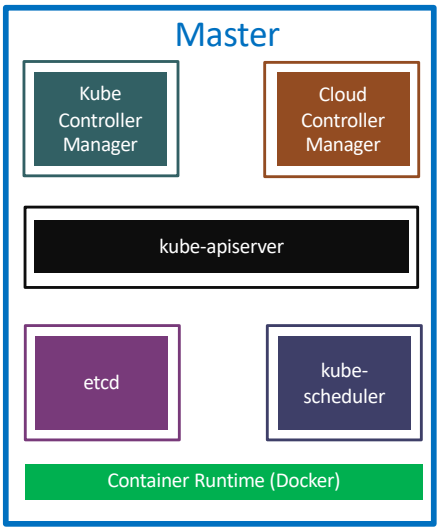
13

Kubernetes - Architecture



14

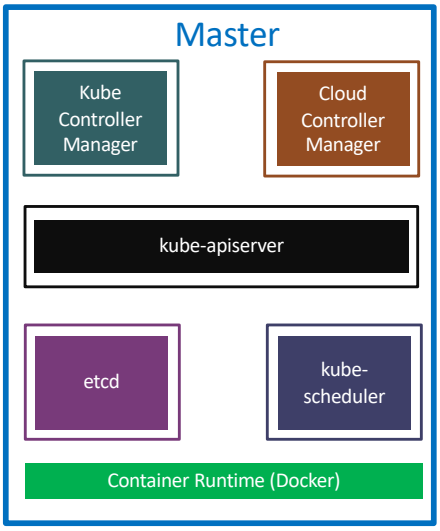
Kubernetes Architecture - Master



- 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.

15

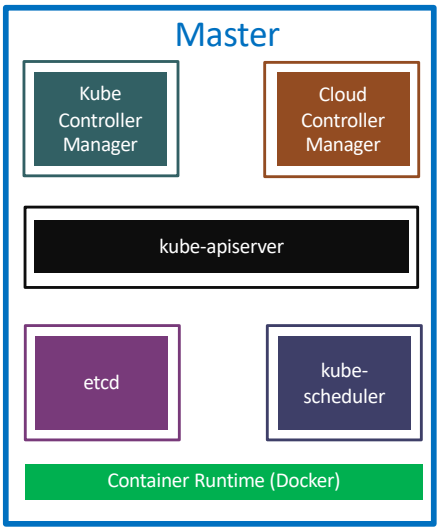
Kubernetes Architecture - Master



- 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.

16

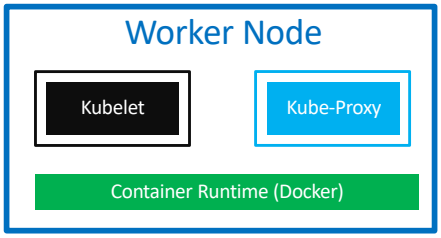
Kubernetes Architecture - Master



- 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

17

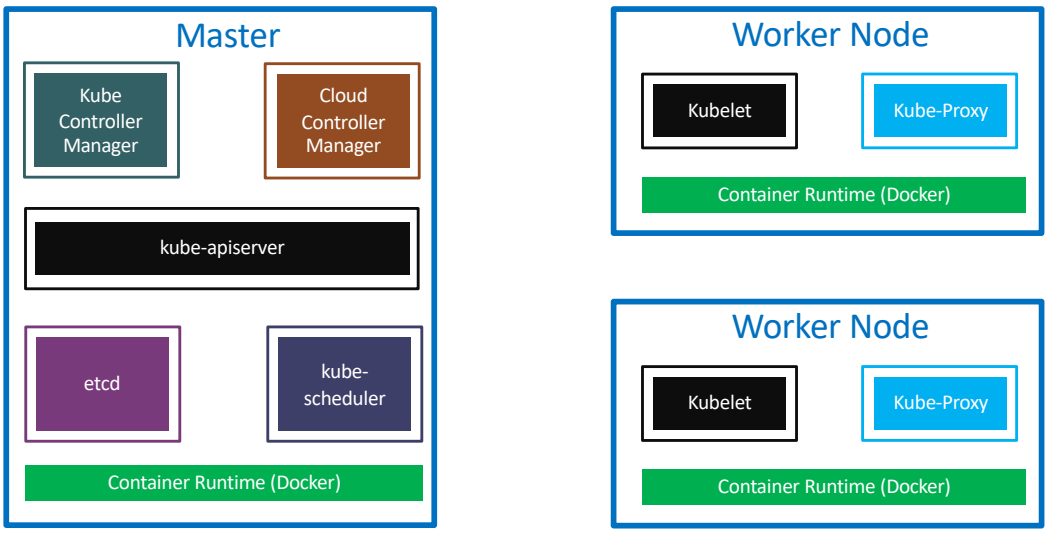
Kubernetes Architecture – Worker Nodes



- 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.

18

Kubernetes - Architecture

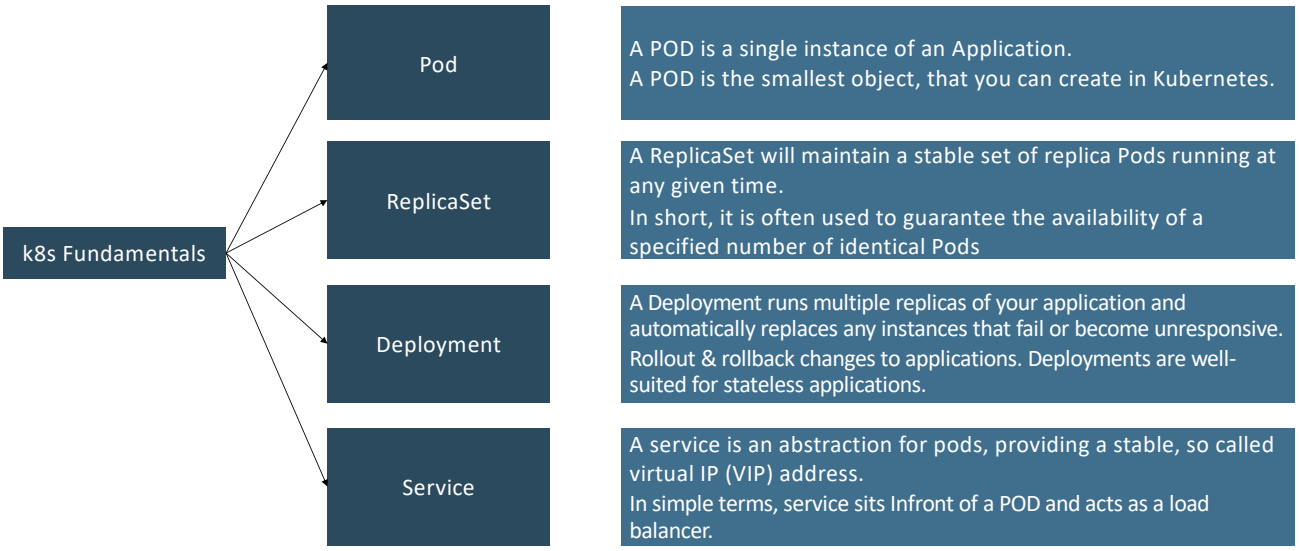


19

Kubernetes
Fundamentals
Pod, ReplicaSet, Deployment & Service

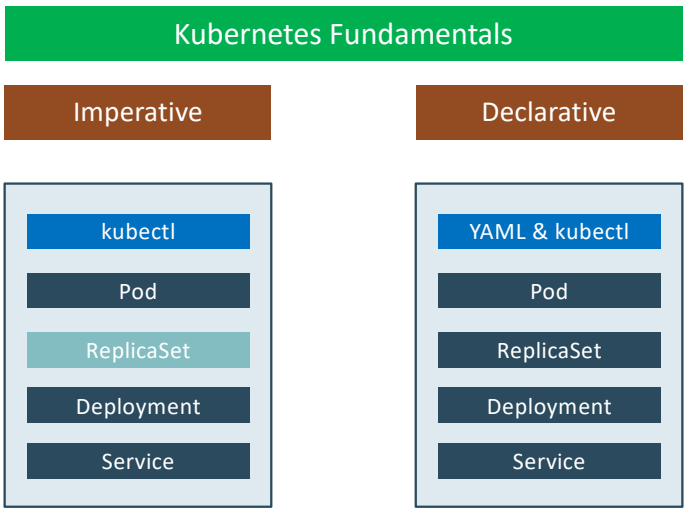
20

Kubernetes - Fundamentals



21

Kubernetes - Imperative & Declarative



22

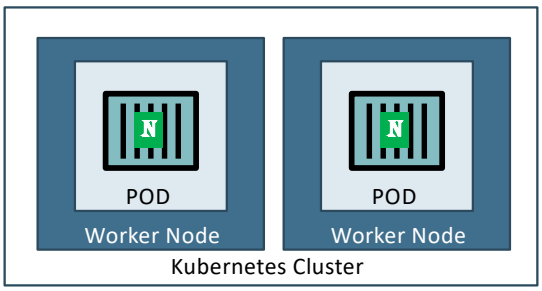
Kubernetes
POD

23

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.

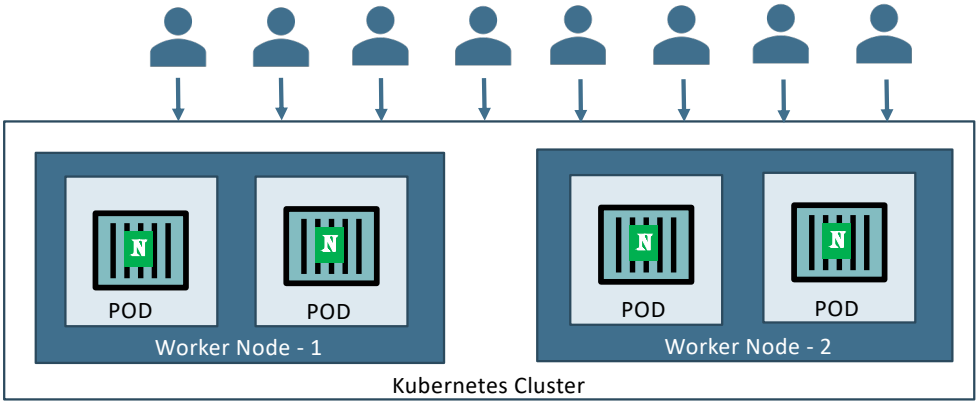
Nginx Container Image



24

Kubernetes - POD

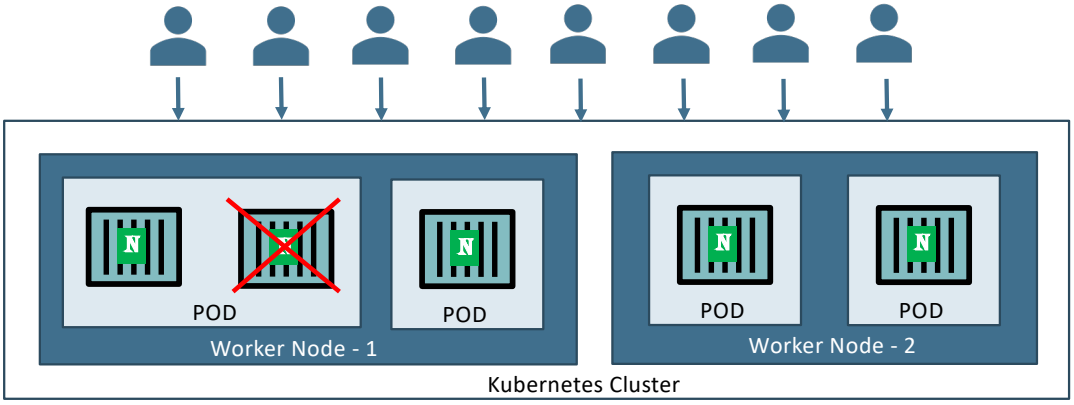
- PODs generally have one to one relationship with containers.
- To scale up we create new POD and to scale down we delete the POD.



25

Kubernetes – PODs

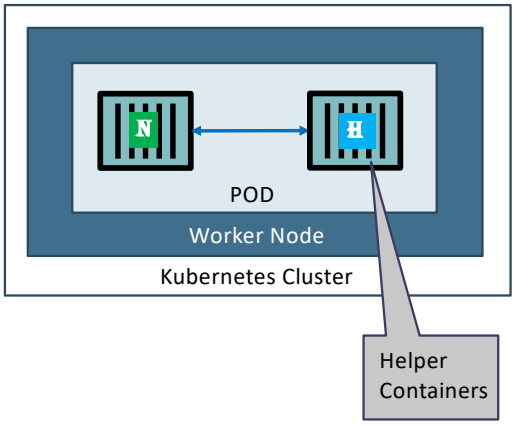
- We cannot have multiple containers of same kind in a single POD.
- Example: Two NGINX containers in single POD serving same purpose is not recommended.



26

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.



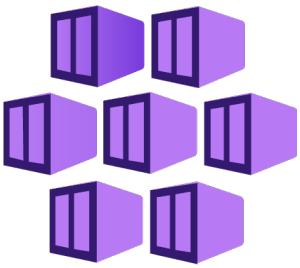
27

Kubernetes PODs Demo

28

Azure Public
IP Address

Azure Standard
Load Balancer

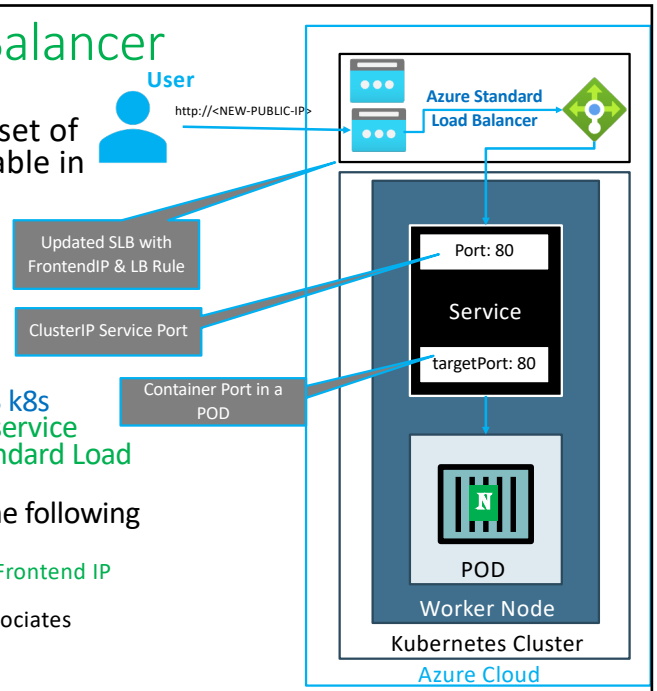


Kubernetes
Services
Load Balancer

29

Kubernetes – Service - LoadBalancer

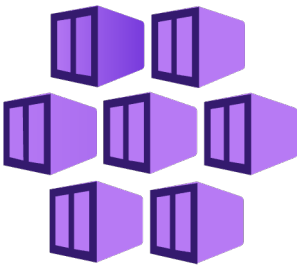
- We can expose an application running on a set of PODs using different types of Services available in k8s.
  - ClusterIP Service(Internal to k8s cluster)
  - NodePort Service (To internet)
  - LoadBalancer Service (To internet)
  - Ingress Service (To internet)
- LoadBalancer Service
  - To access our 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



30

Azure Public
IP Address

Azure Standard
Load Balancer



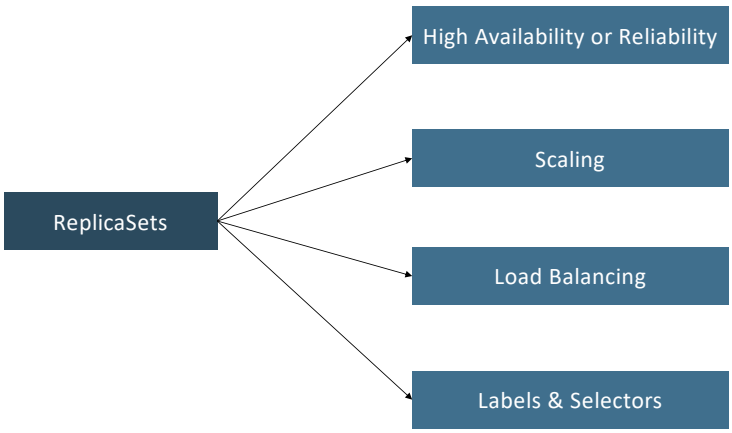
Kubernetes
POD & Load Balancer Service Demo

31

Kubernetes
ReplicaSets

32

Kubernetes - ReplicaSets

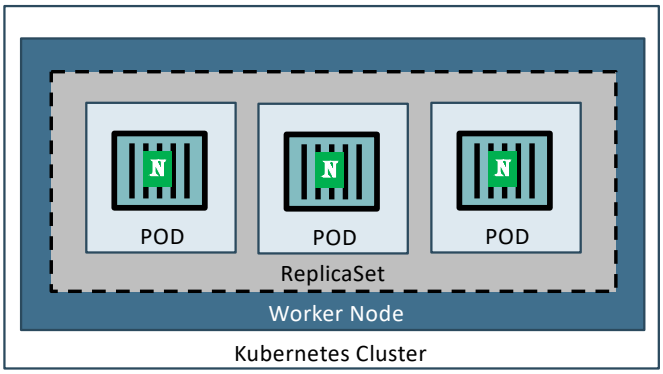


33

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.

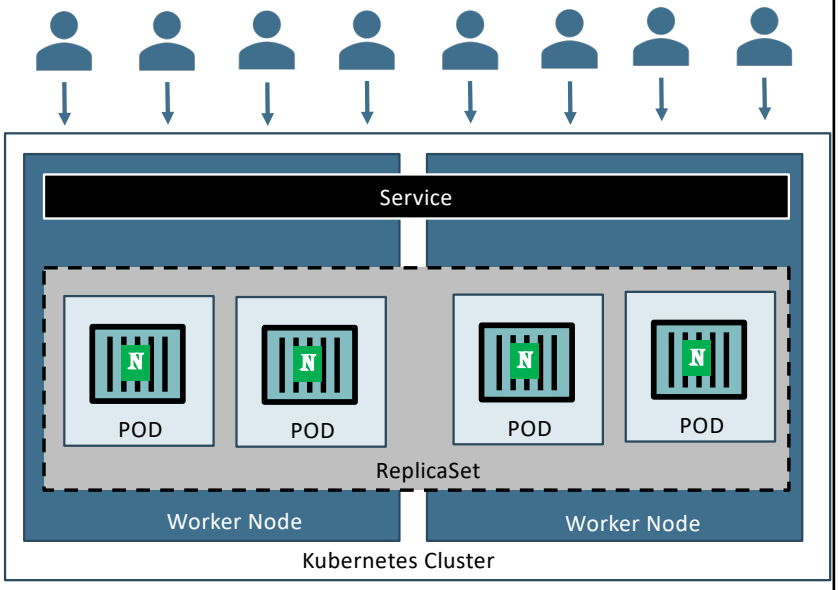
Reliability
Or
High Availability



34

Kubernetes – ReplicaSet

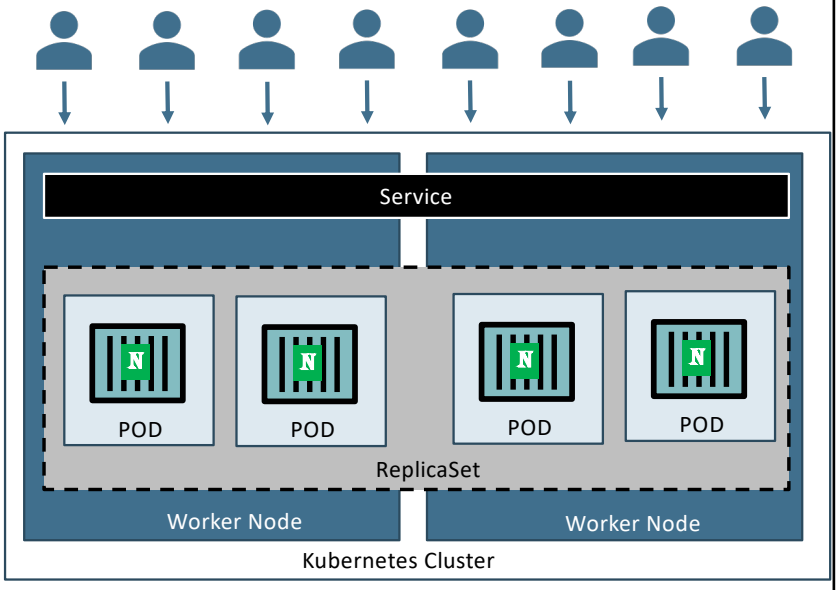
- Load Balancing
- To avoid overloading of traffic to single pod we can use load balancing.
- Kubernetes provides pod load balancing out of the box using Services for the pods which are part of a ReplicaSet
- Labels & Selectors are the key items which ties all 3 together (Pod, ReplicaSet & Service),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



35

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.



36

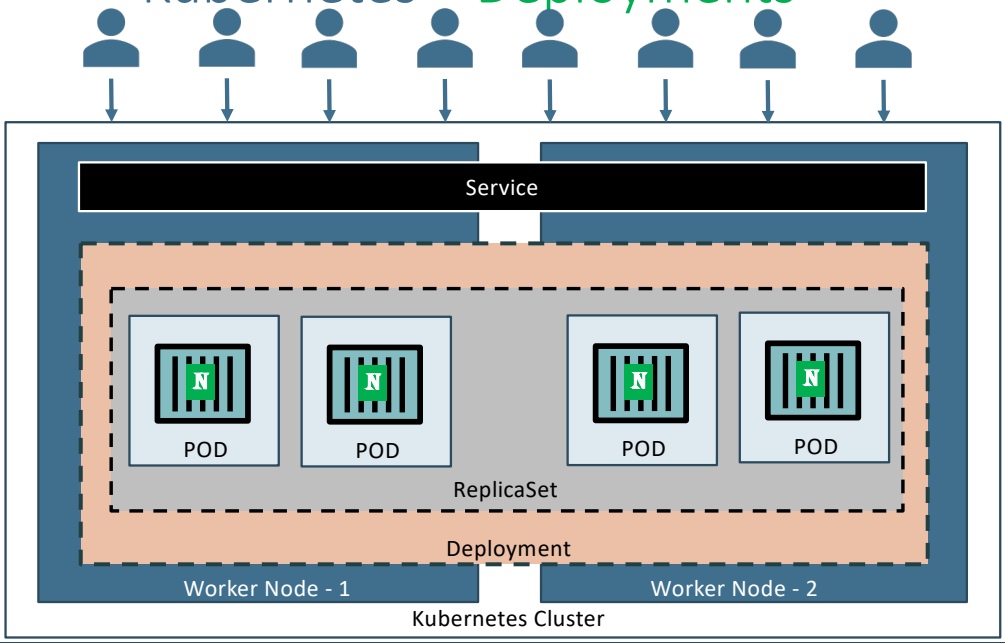
Kubernetes ReplicaSets Demo

37

Kubernetes Deployments

38

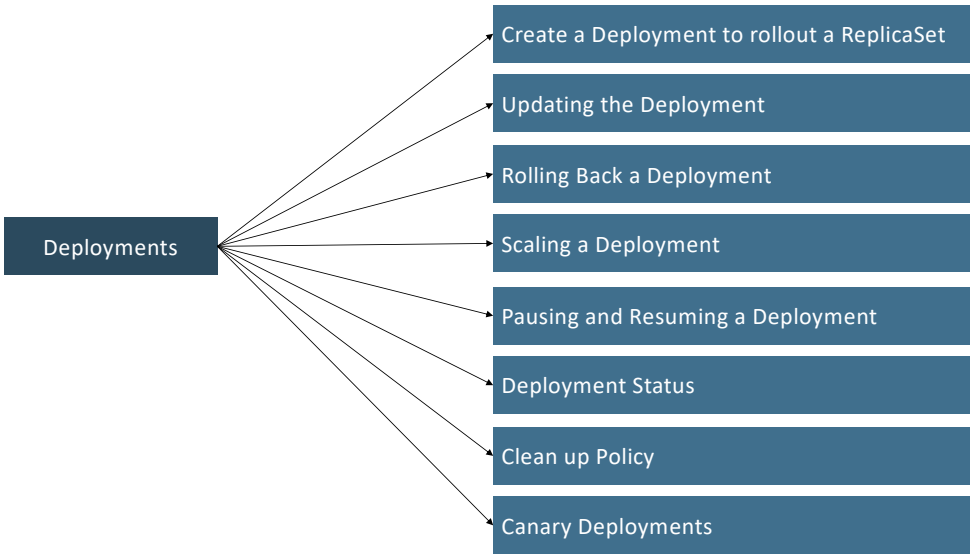
Kubernetes – Deployments



 StackSimplify

39

Kubernetes - Deployment



 StackSimplify

40

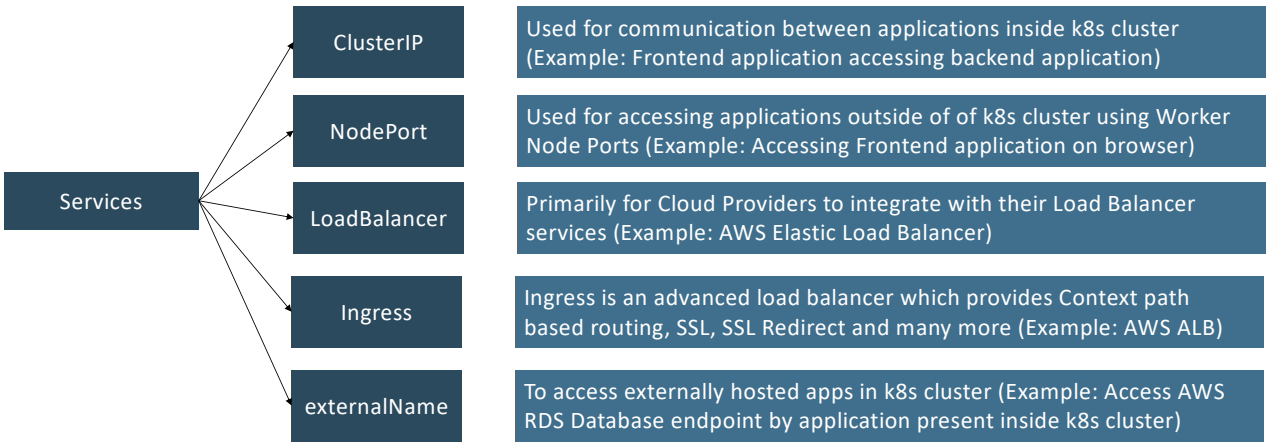
Kubernetes Deployments Demo

41

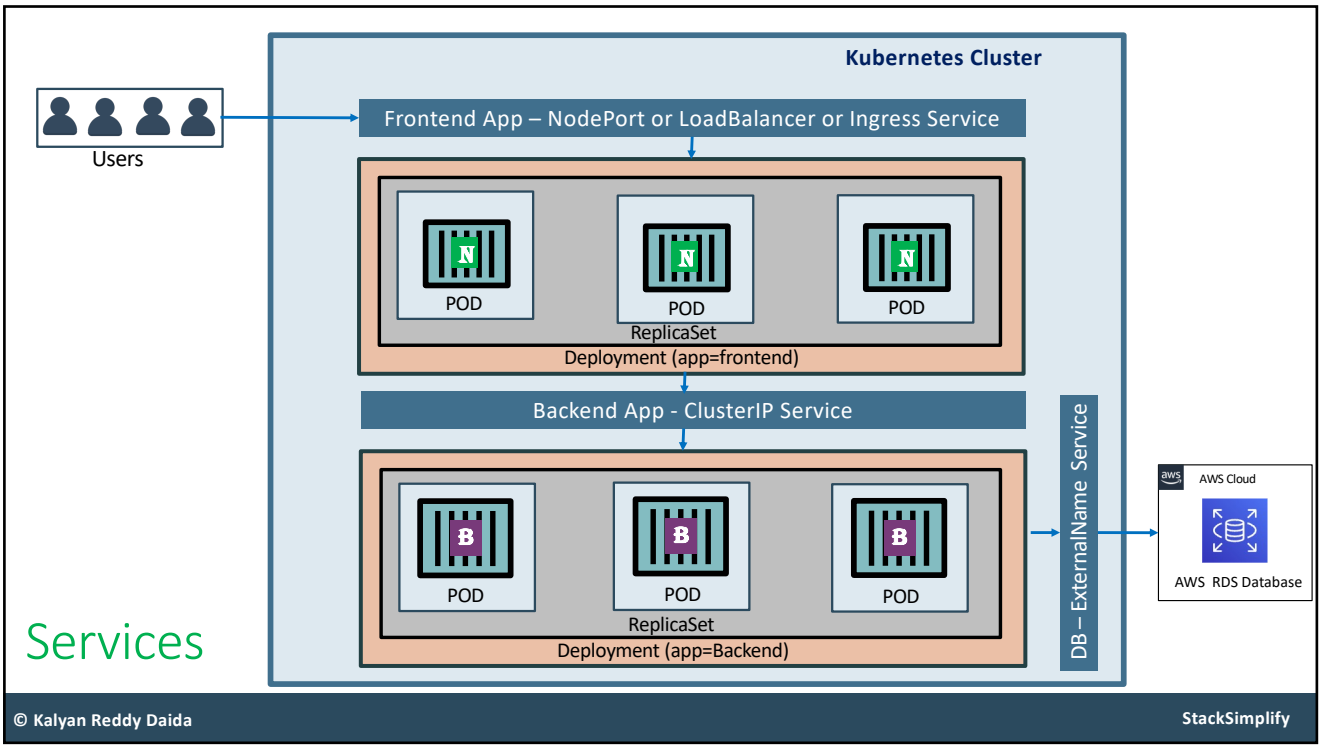
Kubernetes Services

42

Kubernetes - Services



43

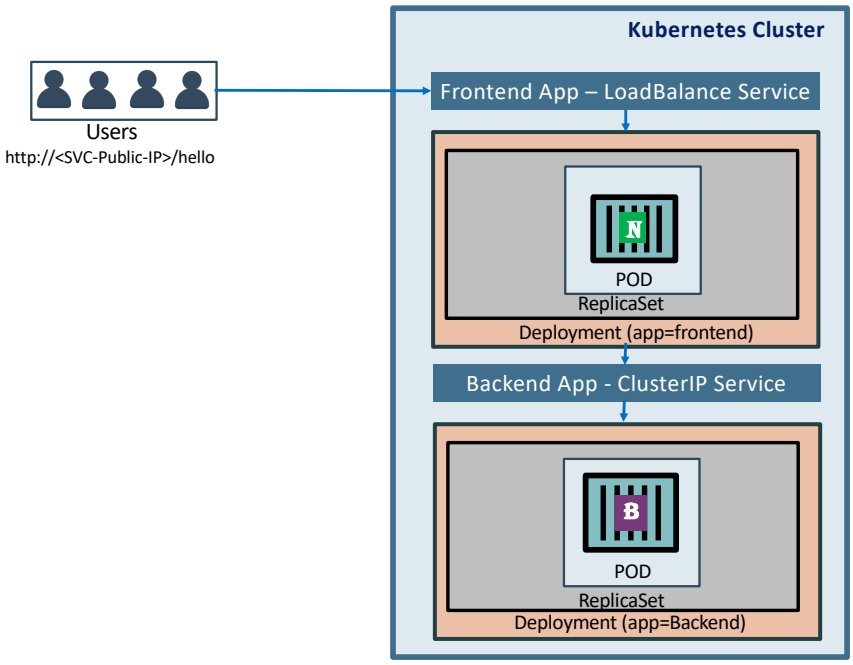


44

Kubernetes Services Demo

45

Services Demo



46

Kubernetes
YAML Basics

47

YAML Basics

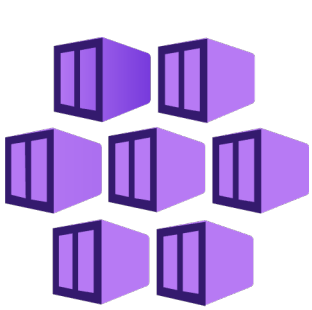
- YAML is not a Markup Language
- YAML is used to store information about different things
- We can use YAML to define key, Value pairs like variables, lists and objects
- YAML is very similar to JSON (Javascript Object Notation)
- YAML primarily focuses on readability and user friendliness
- YAML is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

48

YAML Basics

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

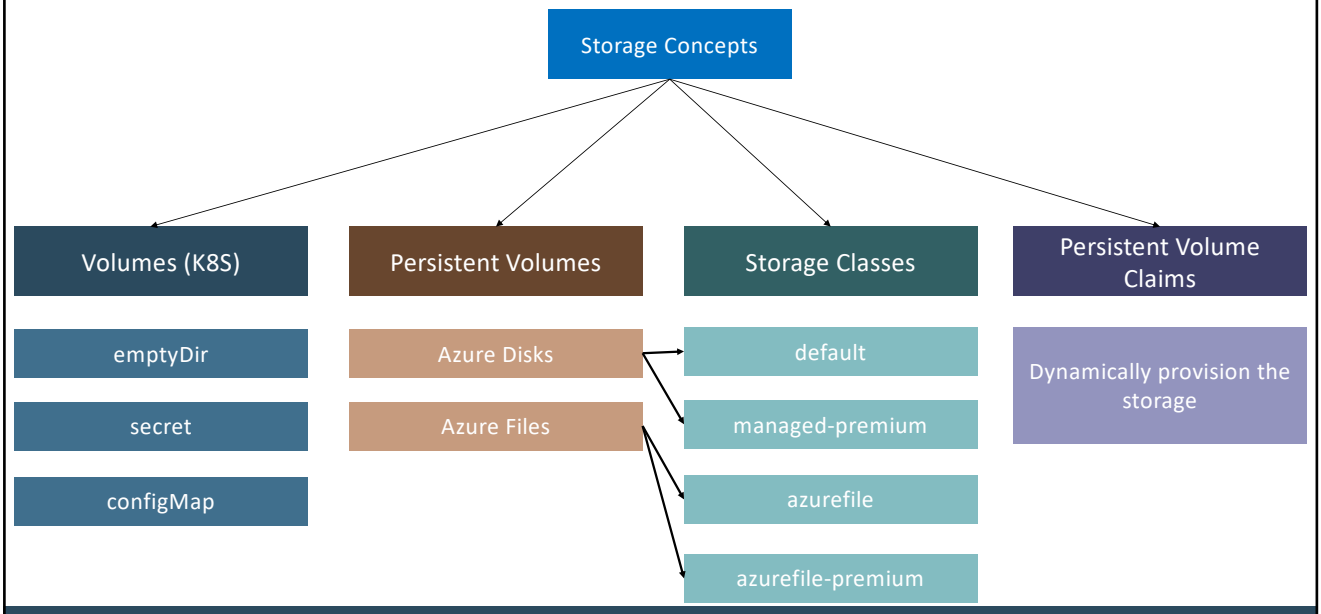
49



Azure AKS Storage with Azure Disks & Files

50

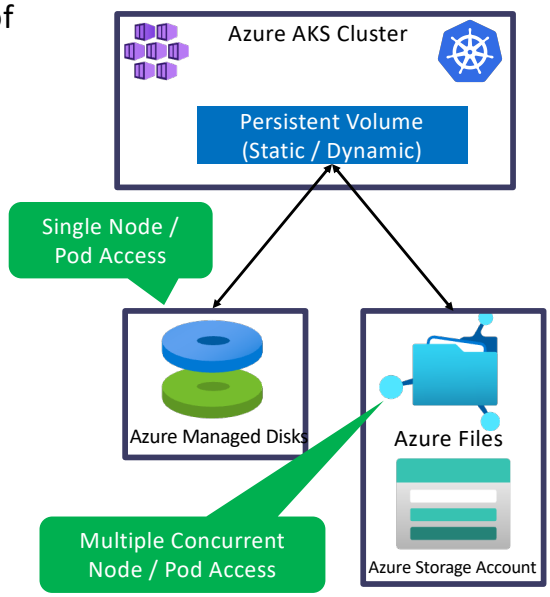
AKS Storage Concepts



51

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.



52

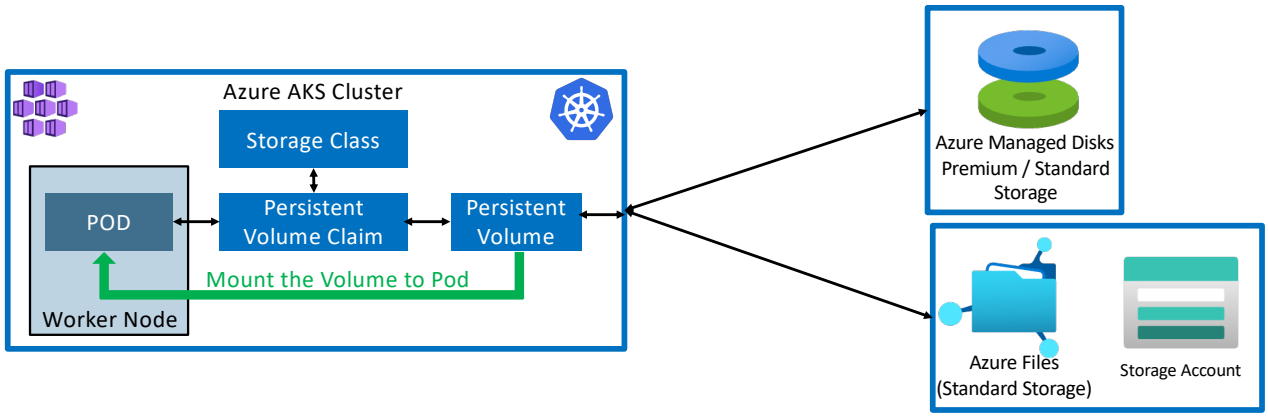
Persistent Volume Claims – How it works?

create a pod

PVC will request storage based on specifics from Storage class (Disk or File)

Kube API Server will dynamically provision the Persistent Volume PV requested

Once the PV is available it will get mounted to Pod for usage.



53

Azure AKS Storage with Azure Disks

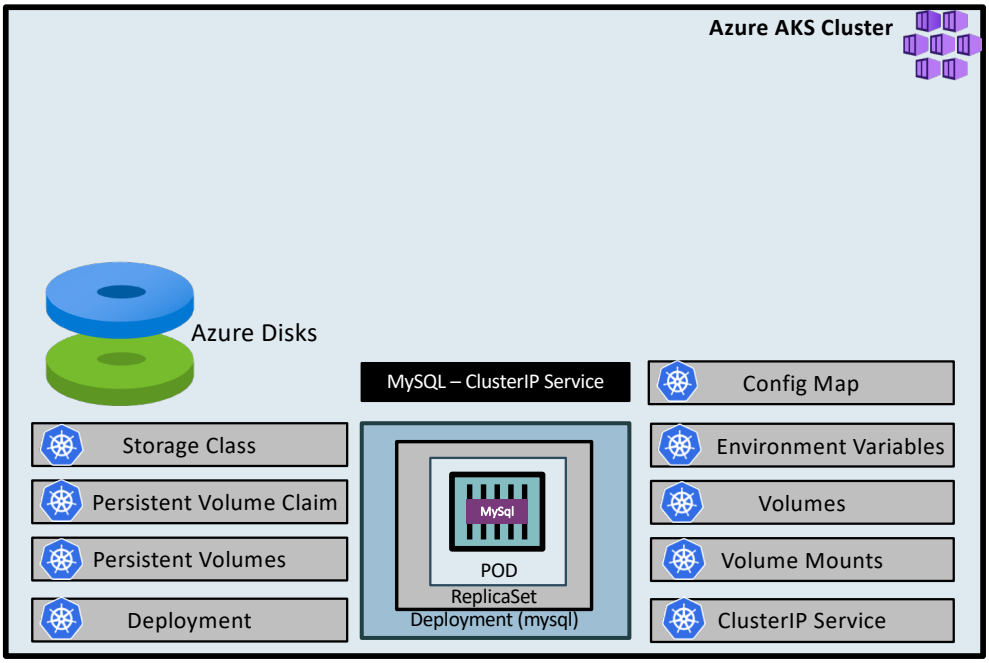
54

Azure Disks - Introduction

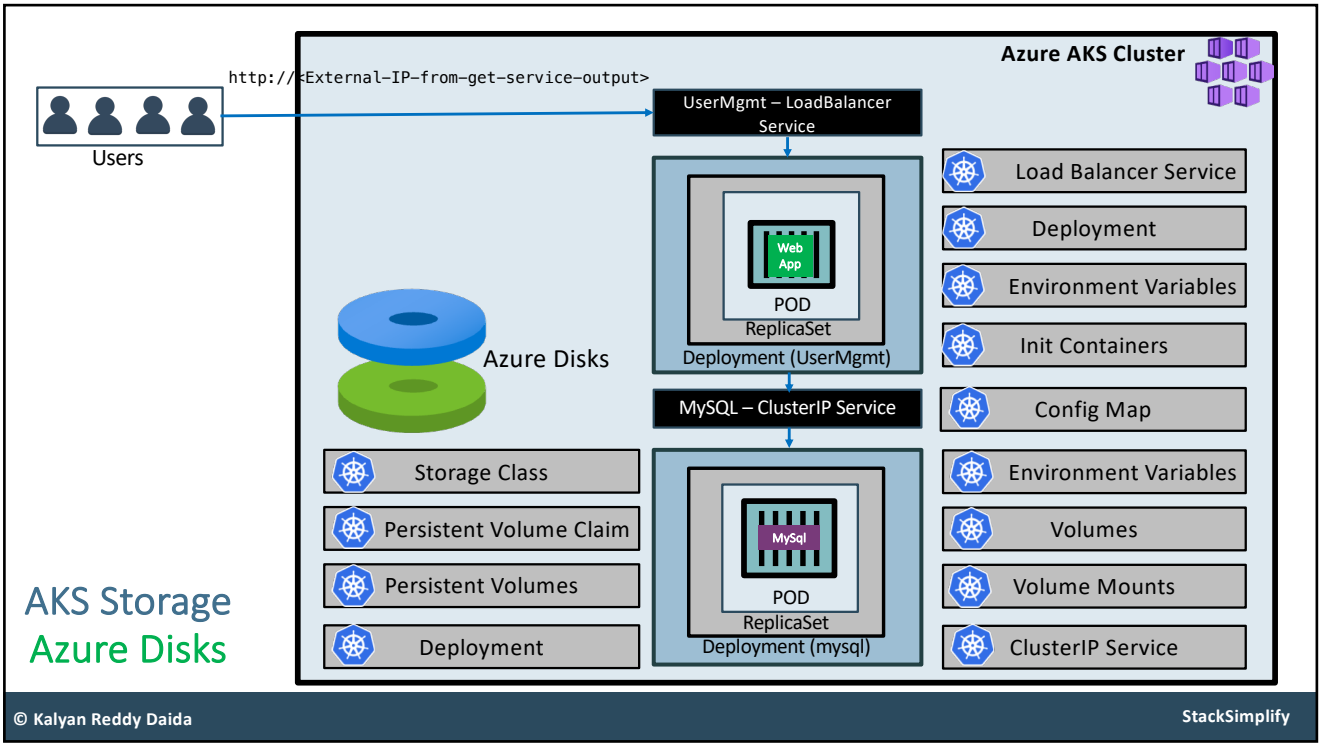
- Azure Disk Storage offers high-performance, highly durable block storage for our mission- and business-critical workloads
- We can mount these volumes as devices on our Virtual Machines & Container instances.
- **Cost-effective storage**
  - Built-in bursting capabilities to handle unexpected traffic 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

55

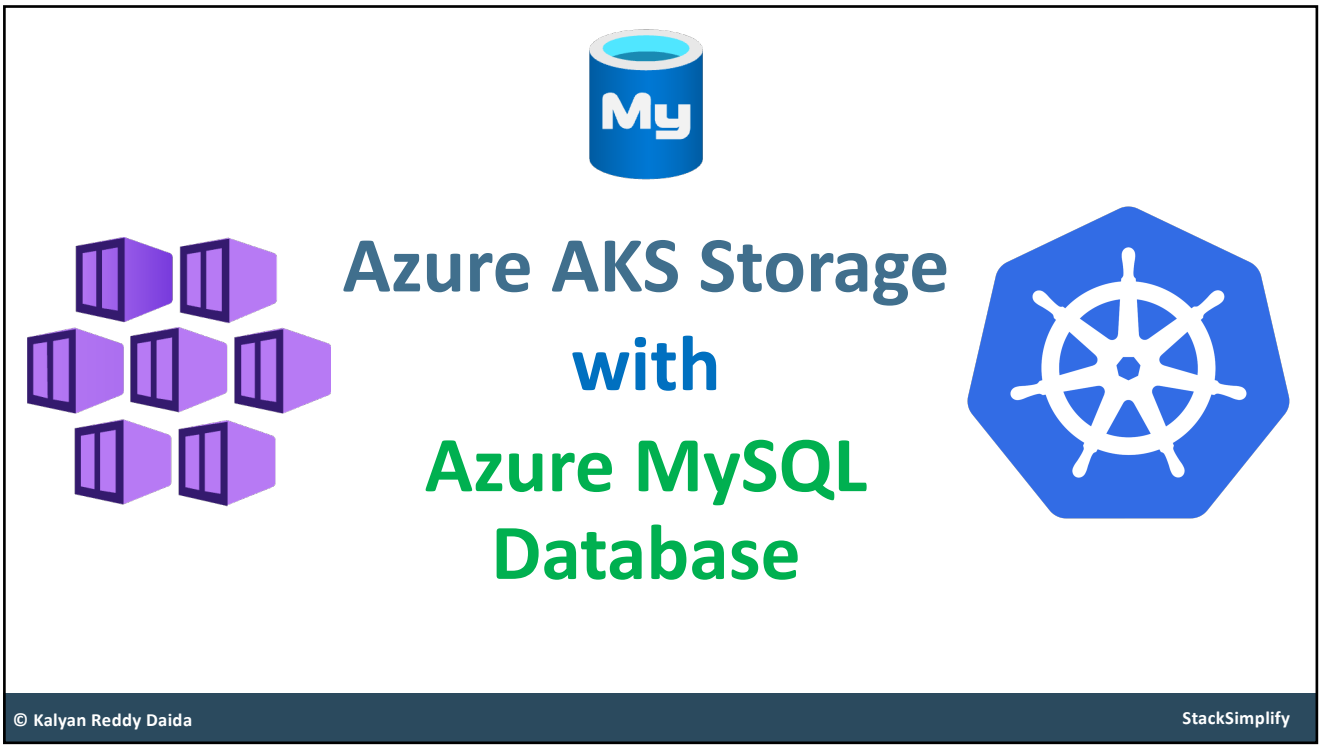
AKS Storage Azure Disks



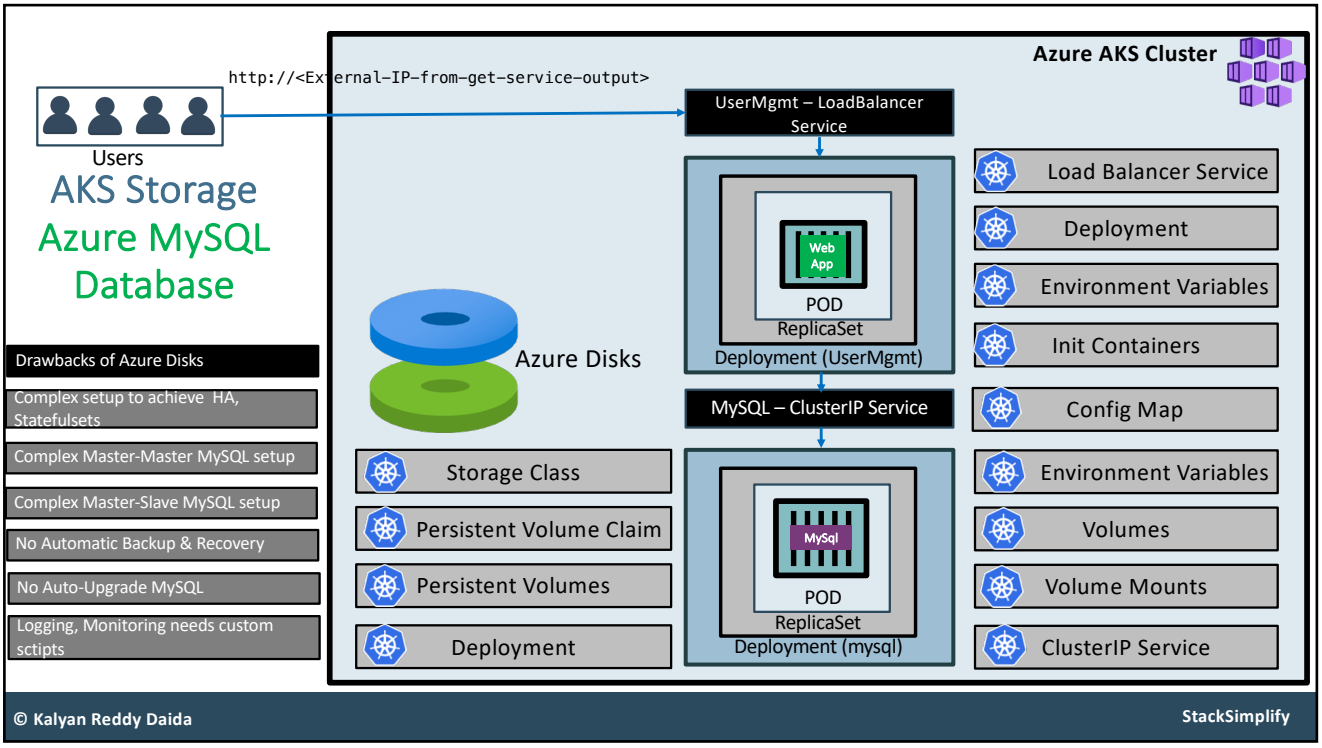
56


Azure AKS Cluster
http://<External-IP-from-get-service-output>
Users
UserMgmt – LoadBalancer Service
Load Balancer Service
Deployment
Environment Variables
Init Containers
Web App
POD
ReplicaSet
Deployment (UserMgmt)
Azure Disks
MySQL – ClusterIP Service
Config Map
Storage Class
Persistent Volume Claim
Persistent Volumes
Deployment
MySql
POD
ReplicaSet
Deployment (mysql)
Environment Variables
Volumes
Volume Mounts
ClusterIP Service
AKS Storage
Azure Disks
© Kalyan Reddy Daida
StackSimplify

57


Azure AKS Storage
with
Azure MySQL
Database
© Kalyan Reddy Daida
StackSimplify

58

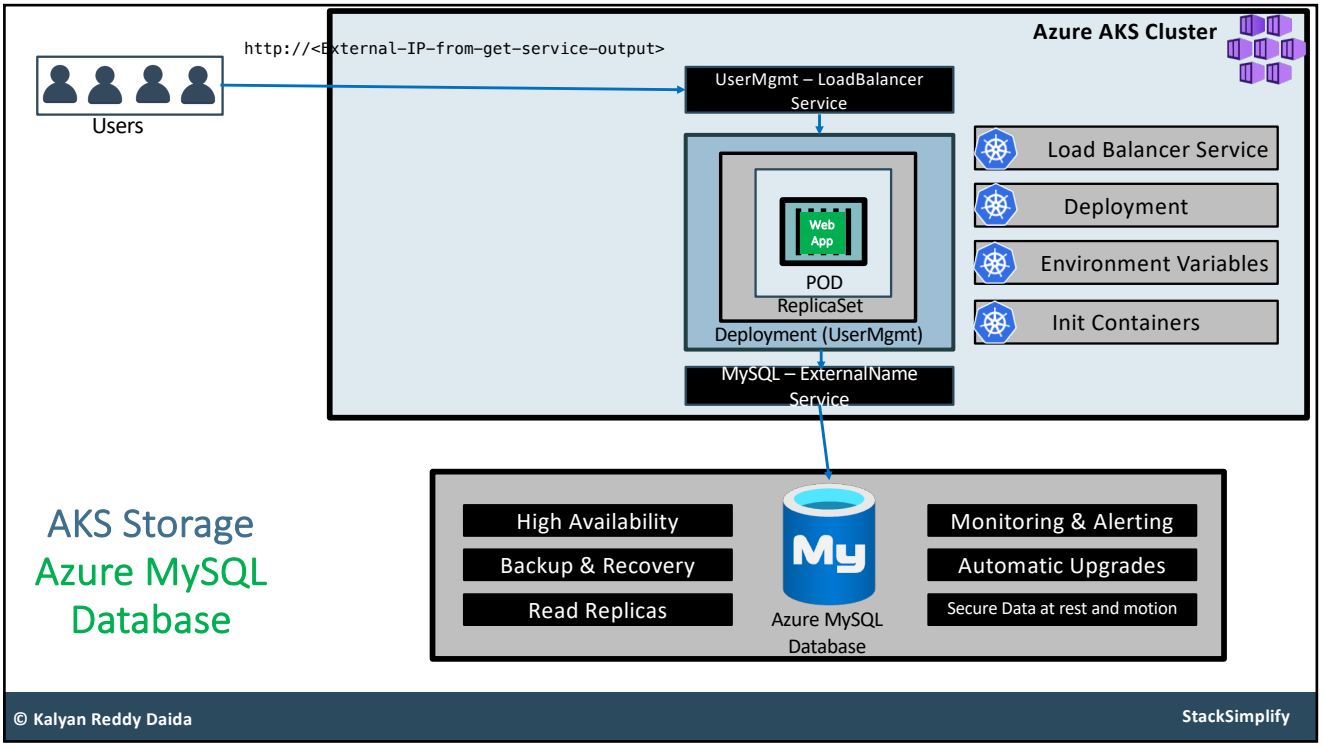


59

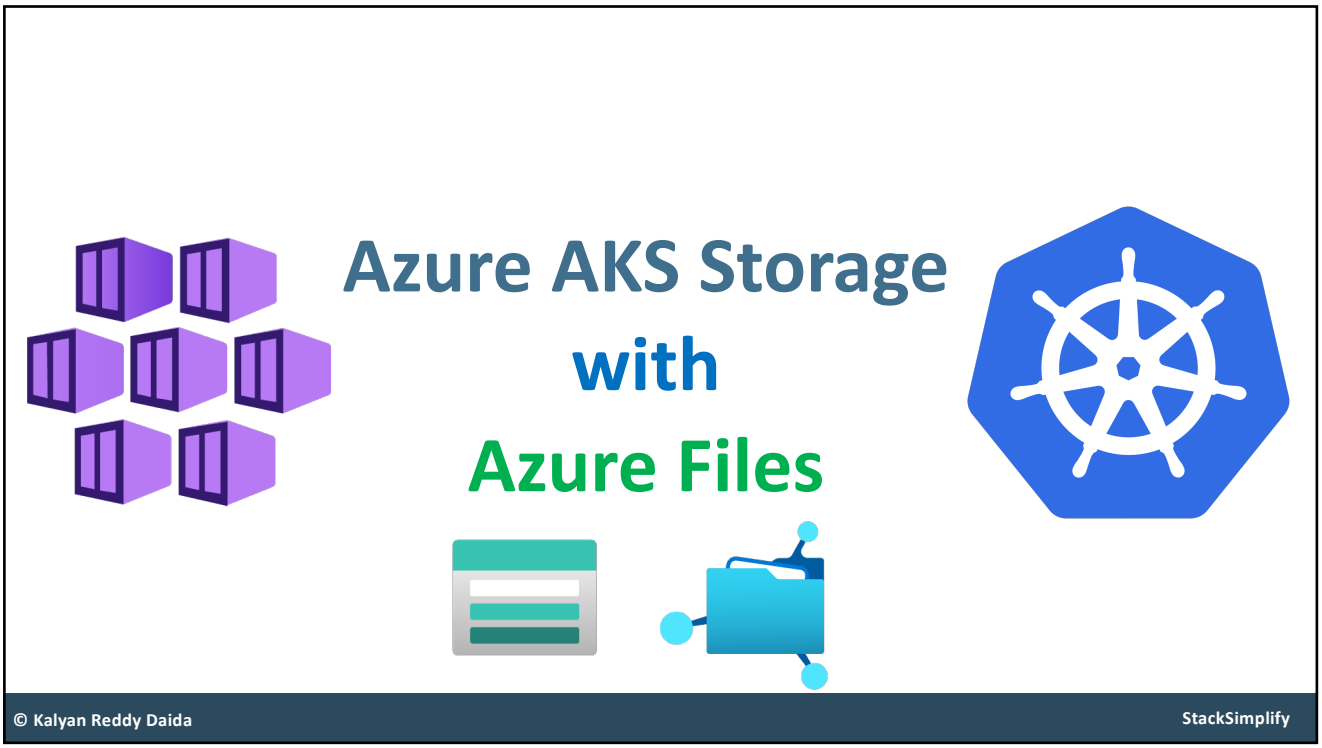
Azure MySQL Database

- **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.

60

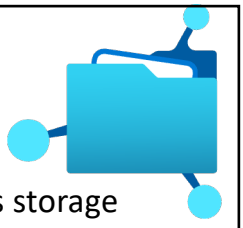

Azure AKS Cluster
http://<External-IP-from-get-service-output>
Users
UserMgmt – LoadBalancer Service
Web App
POD
ReplicaSet
Deployment (UserMgmt)
MySQL – ExternalName Service
Load Balancer Service
Deployment
Environment Variables
Init Containers
AKS Storage
Azure MySQL Database
High Availability
Backup & Recovery
Read Replicas
Azure MySQL Database
Monitoring & Alerting
Automatic Upgrades
Secure Data at rest and motion
© Kalyan Reddy Daida
StackSimplify

61


Azure AKS Storage
with
Azure Files
© Kalyan Reddy Daida
StackSimplify

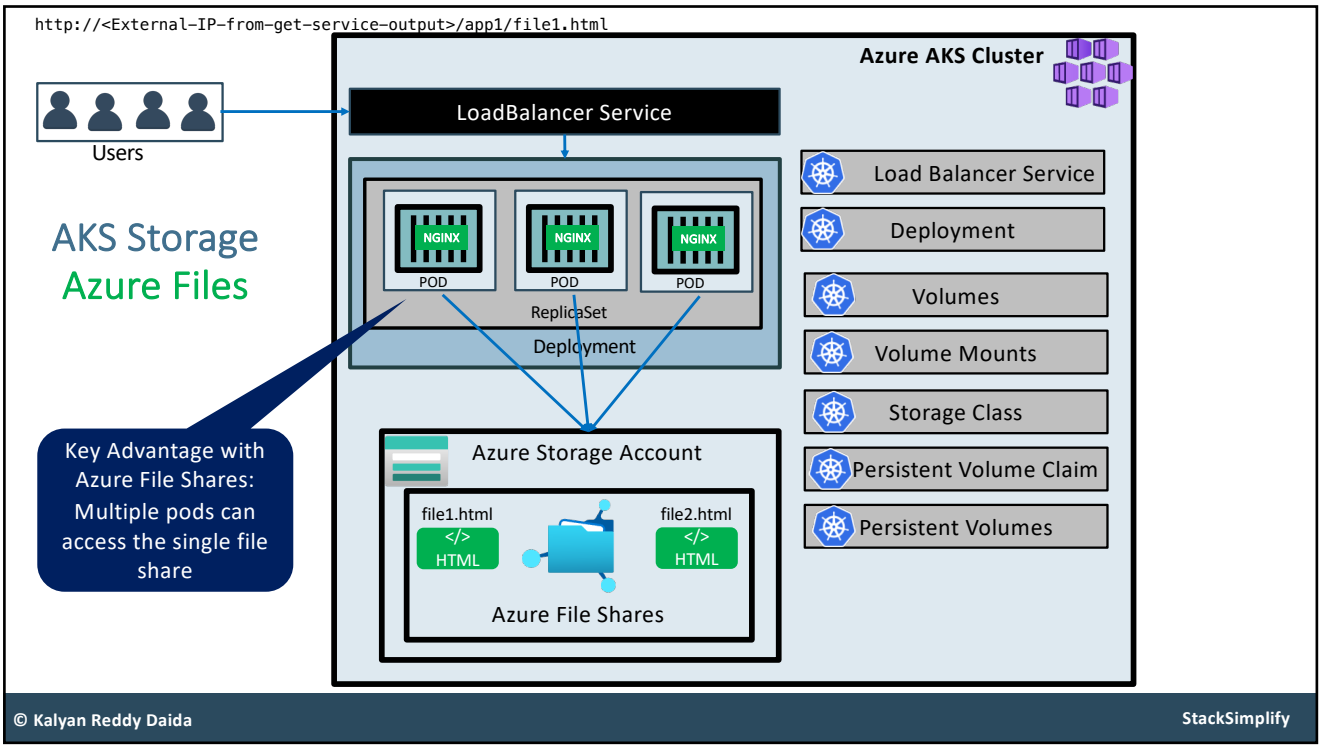
62

Azure Files

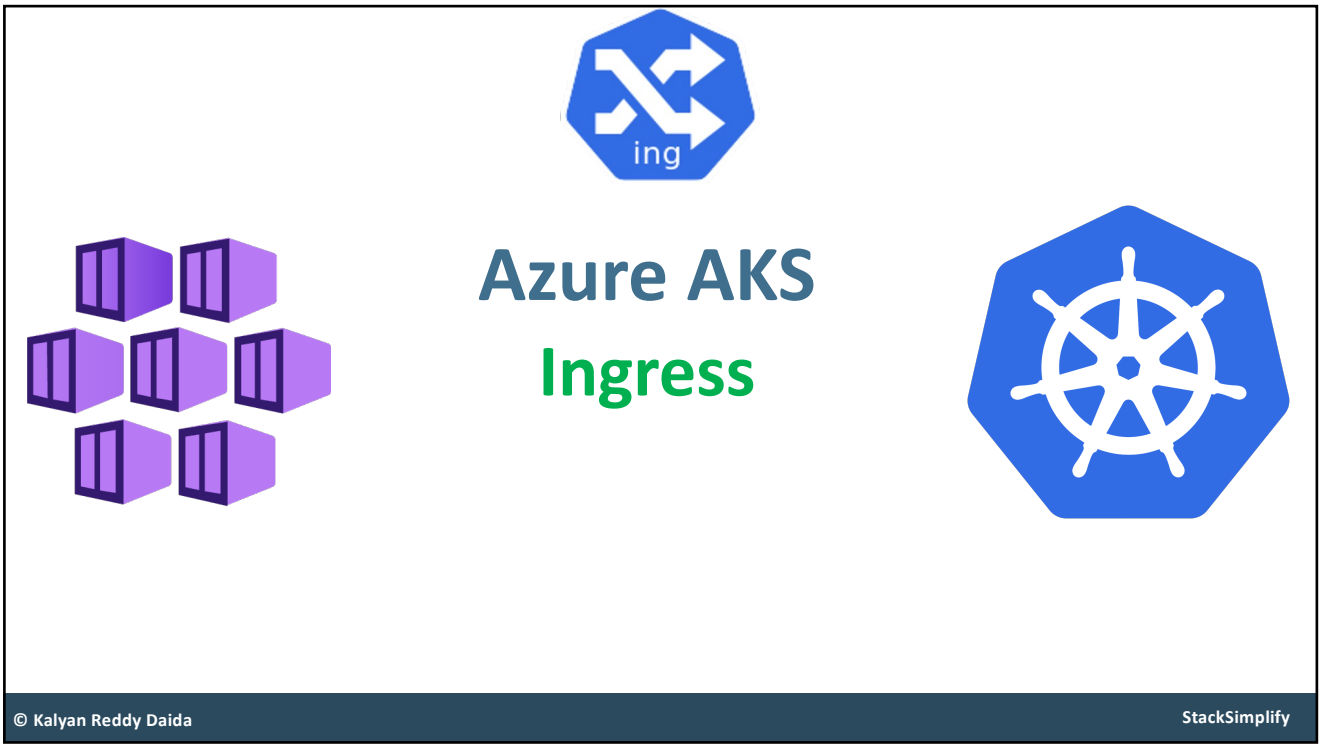


- These are simple, secure and fully managed cloud file shares
- We can secure data at rest and in-transit using SMB 3.0 and HTTPS
- We can create high-performance file shares using the Premium Files storage tier
- We can replace or supplement on-premises file servers
- Scripting and tooling: PowerShell 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.

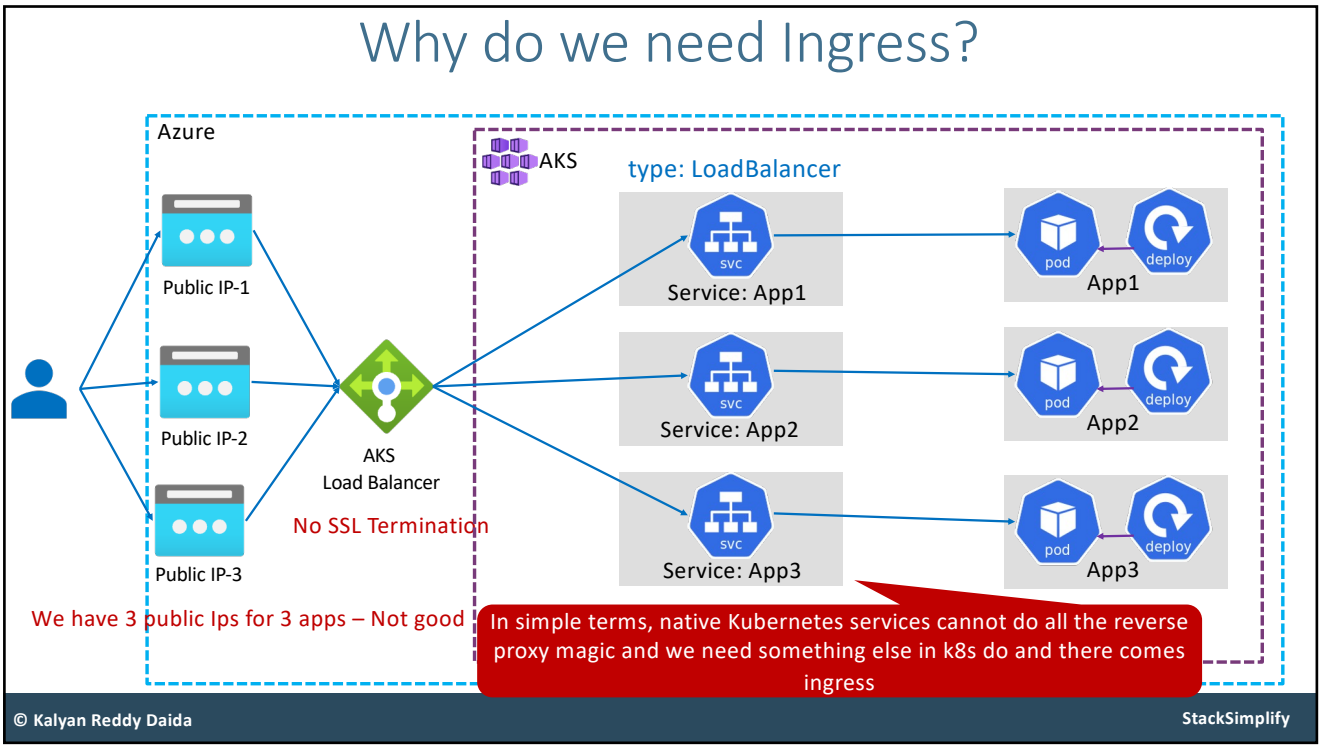
63



64


ing
Azure AKS
Ingress
© Kalyan Reddy Daida
StackSimplify

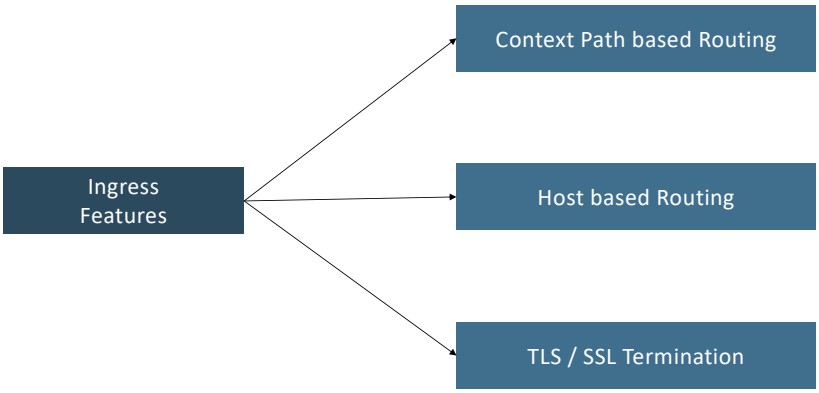
65


Why do we need Ingress?
Azure
AKS
type: LoadBalancer
Public IP-1
Public IP-2
Public IP-3
AKS
Load Balancer
No SSL Termination
svc
Service: App1
Service: App2
Service: App3
pod
deploy
App1
App2
App3
We have 3 public Ips for 3 apps – Not good
In simple terms, native Kubernetes services cannot do all the reverse proxy magic and we need something else in k8s do and there comes ingress
© Kalyan Reddy Daida
StackSimplify

66

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.



67

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.

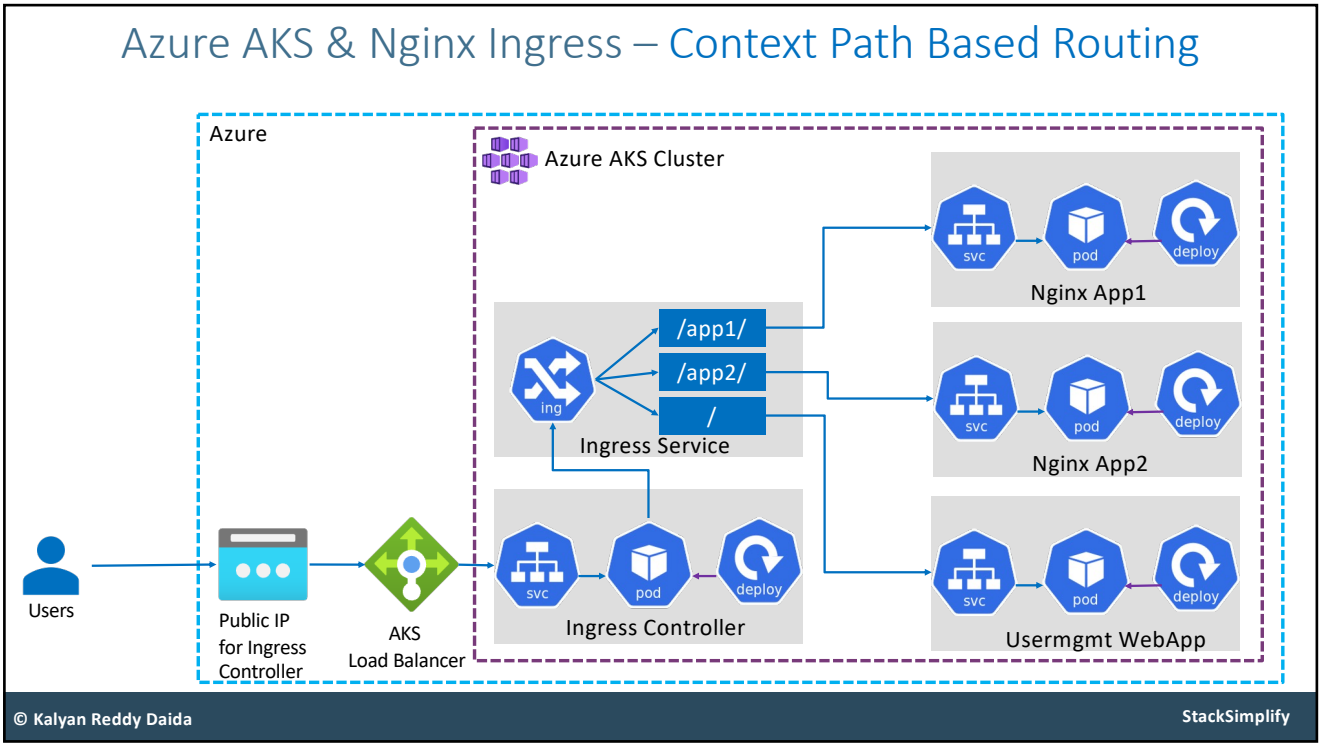
Ingress Resource or Service

Equivalent to Kubernetes Services but with lot of additional features

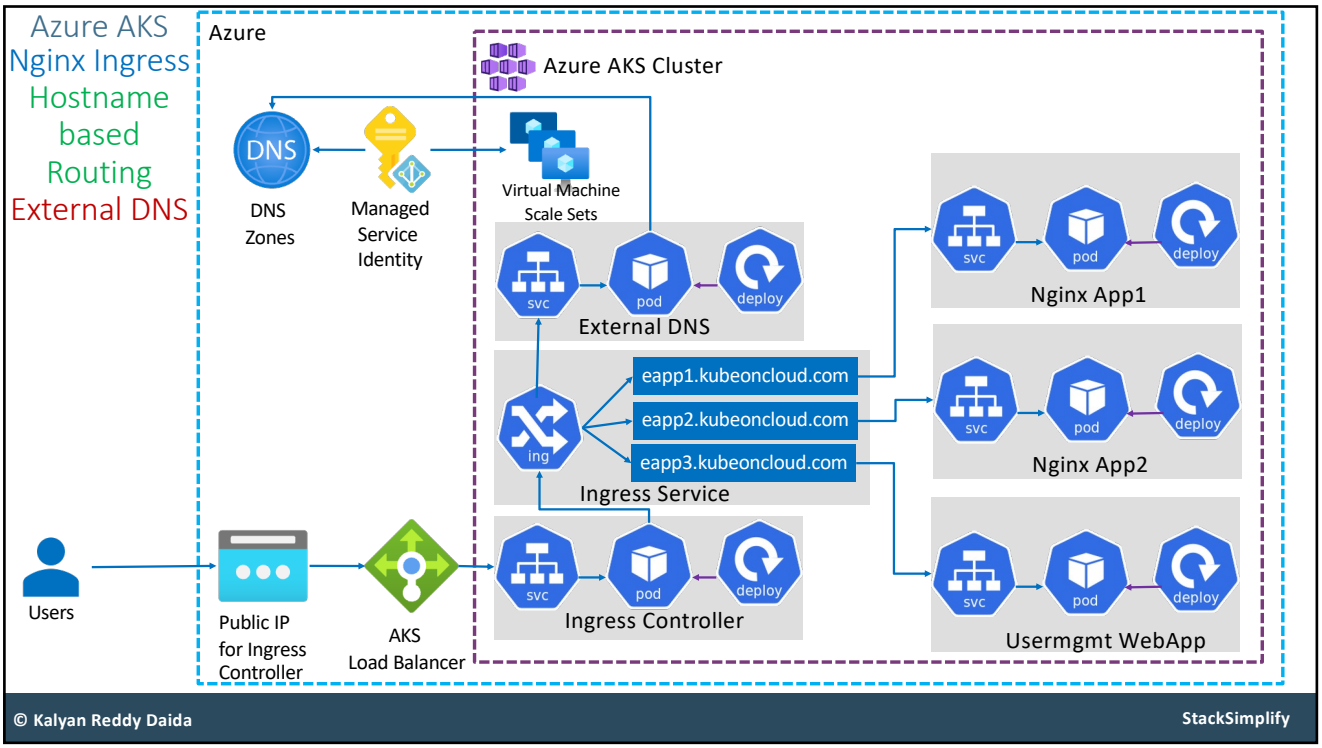
We can define routing rules based on URI or Hostname in Ingress Resource Manifest

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

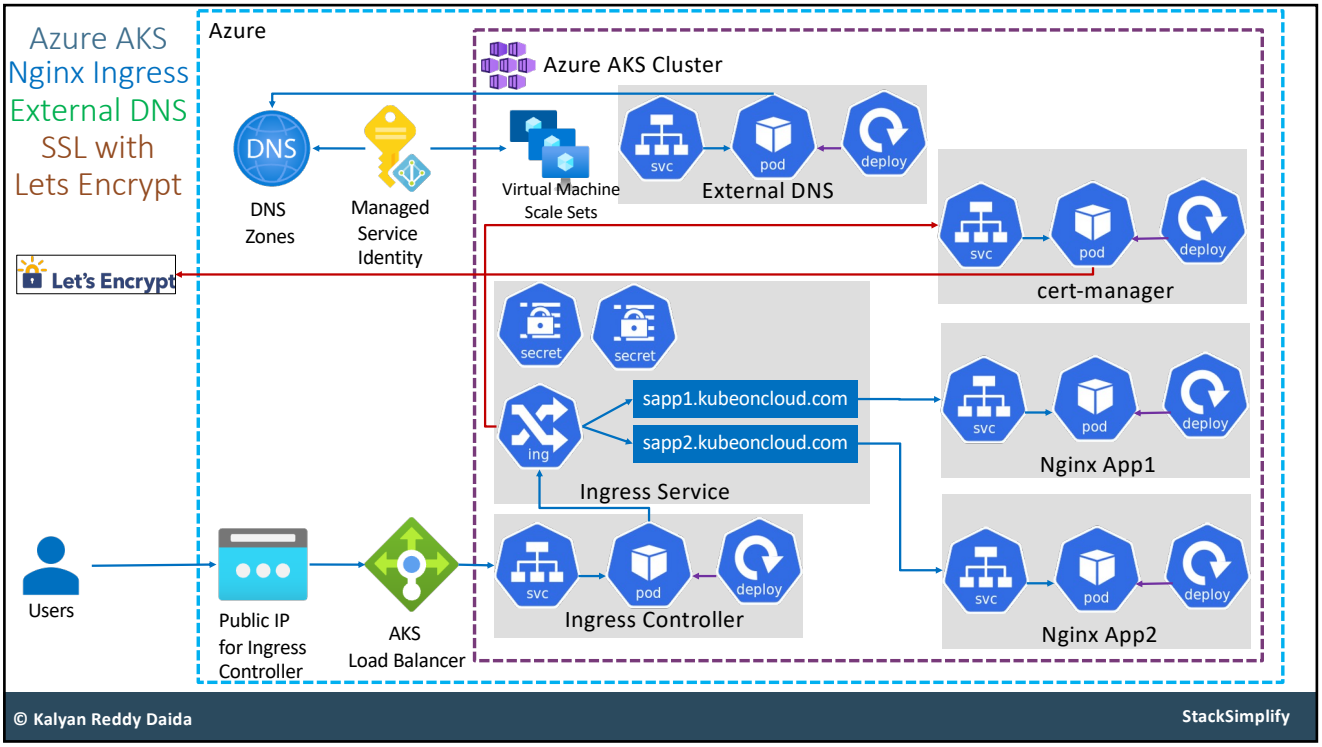
68


Azure AKS & Nginx Ingress – Context Path Based Routing
Azure
Azure AKS Cluster
Nginx App1
Nginx App2
Usermgmt WebApp
/app1/
/app2/
/
Ingress Service
Ingress Controller
Users
Public IP for Ingress Controller
AKS Load Balancer
svc
pod
deploy
ing
© Kalyan Reddy Daida
StackSimplify

69


Azure AKS Nginx Ingress Hostname based Routing External DNS
Azure
Azure AKS Cluster
DNS
DNS Zones
Managed Service Identity
Virtual Machine Scale Sets
External DNS
Nginx App1
Nginx App2
Usermgmt WebApp
eapp1.kubeoncloud.com
eapp2.kubeoncloud.com
eapp3.kubeoncloud.com
Ingress Service
Ingress Controller
Users
Public IP for Ingress Controller
AKS Load Balancer
svc
pod
deploy
ing
© Kalyan Reddy Daida
StackSimplify

70



71

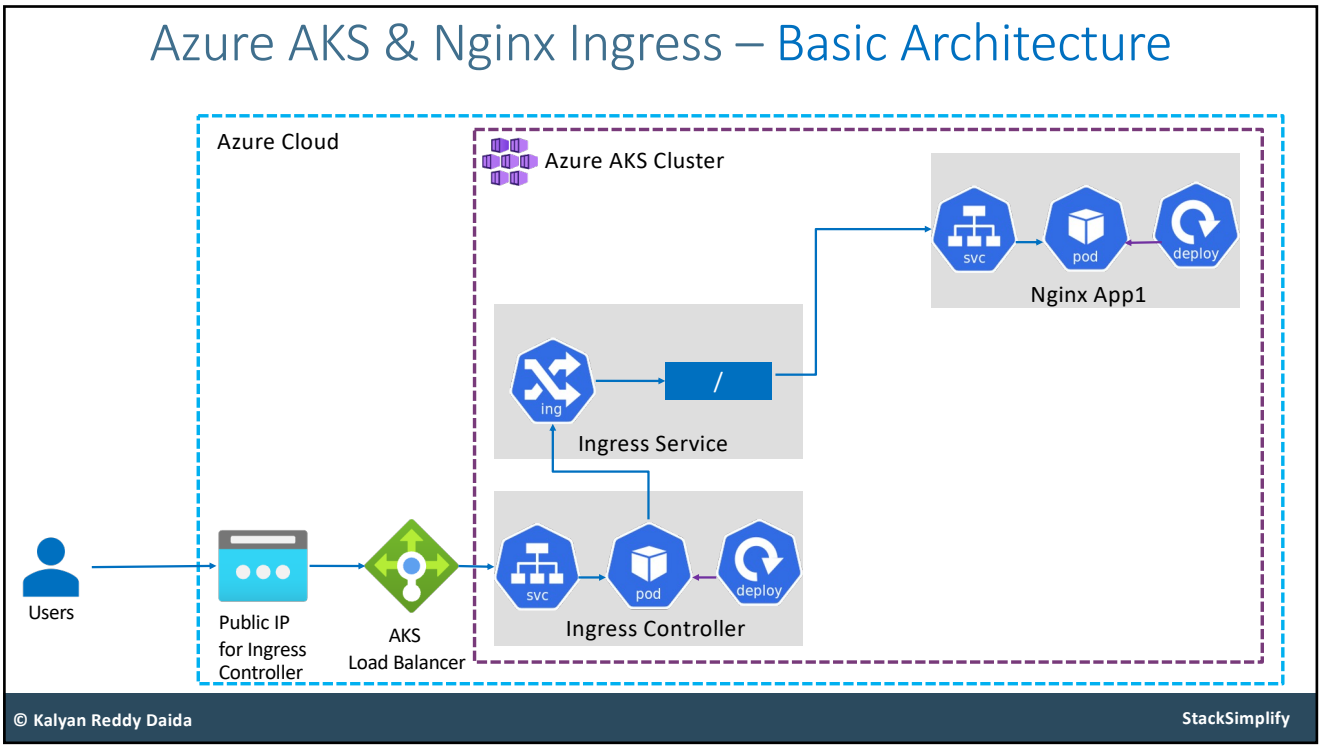
Nginx Ingress Controller - References

- 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/

72


ing
Azure AKS
Nginx Ingress Service
Install & Implement
© Kalyan Reddy Daida
StackSimplify

73


Azure AKS & Nginx Ingress – Basic Architecture
Azure Cloud
Azure AKS Cluster
svc
pod
deploy
Nginx App1
ing
/
Ingress Service
Users
Public IP
for Ingress
Controller
AKS
Load Balancer
svc
pod
deploy
Ingress Controller
© Kalyan Reddy Daida
StackSimplify

74

Basic Ingress Manifest

Deprecated

03-Ingress-Basic.yml

azure-aks-kubernetes-masterclass > 09-Ingress-Basic > kube-manifests > 03-Ingress-Basic.yml > apiVersion

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

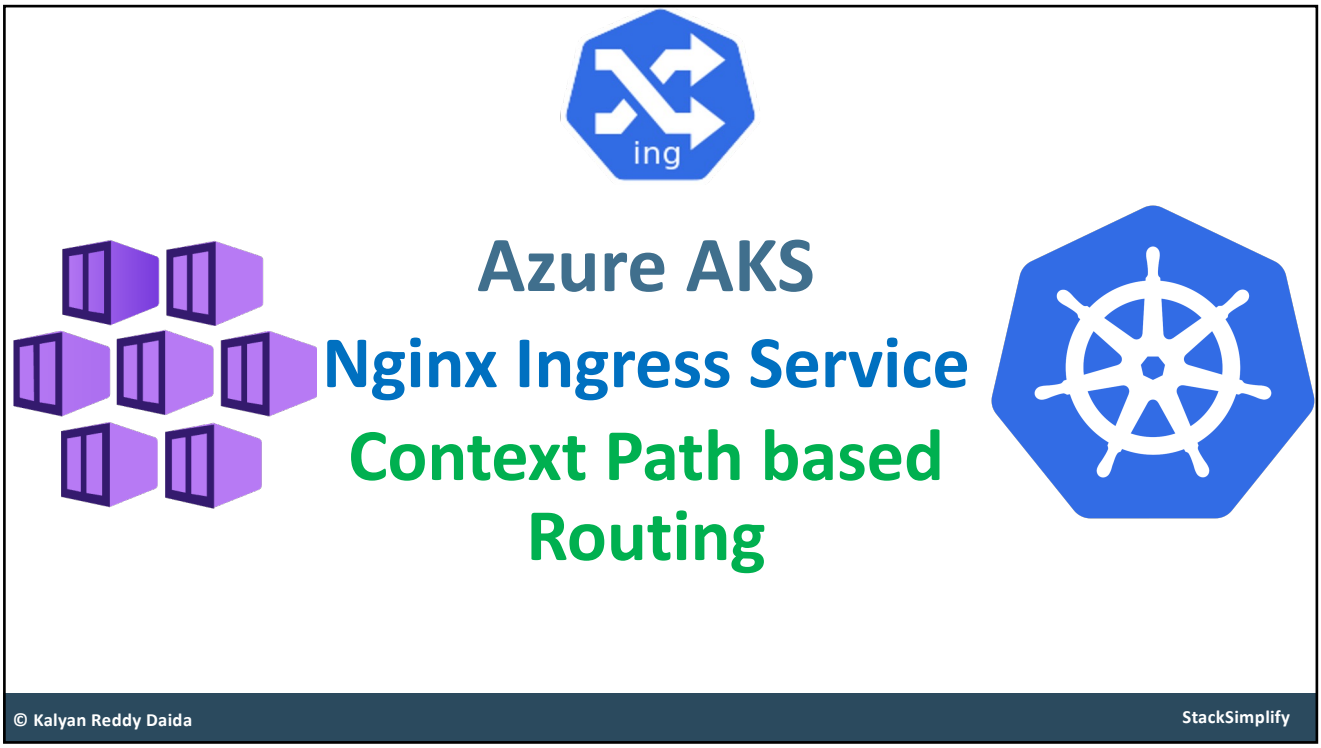

75

Basic Ingress Manifest

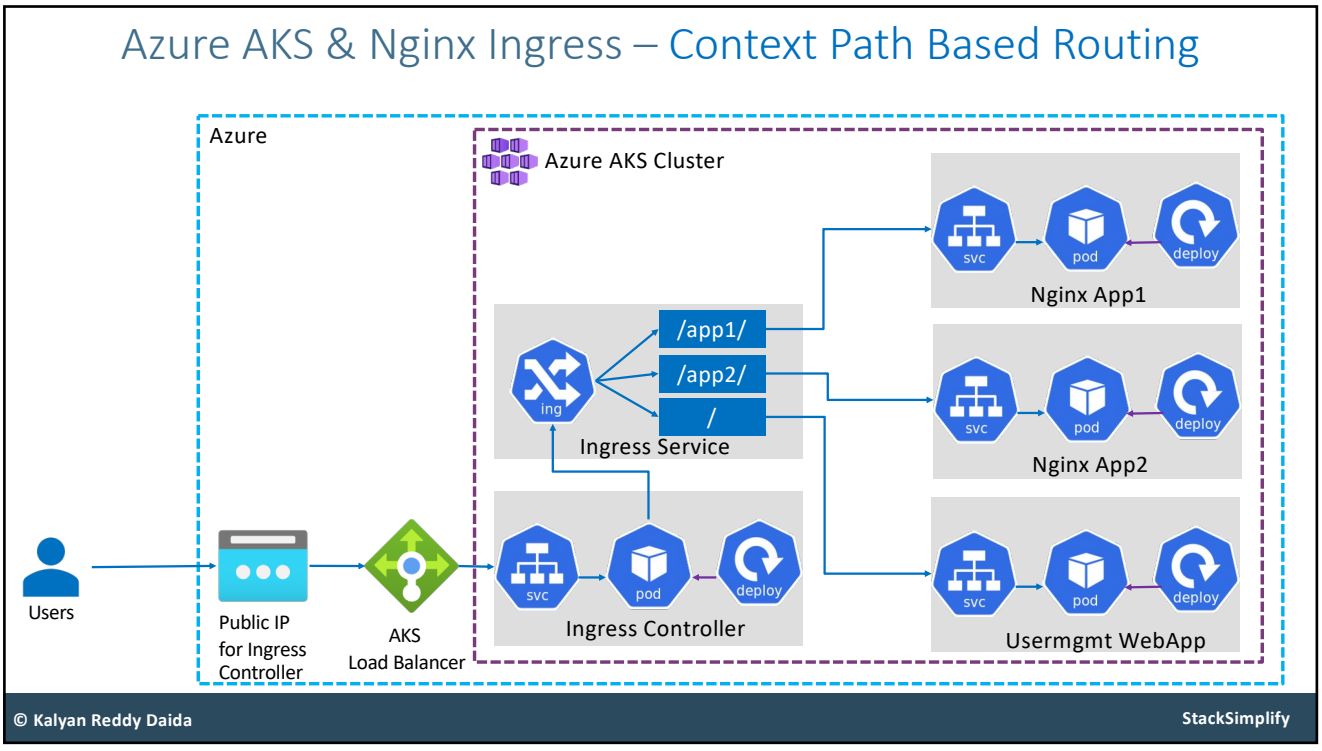
NEW

```yaml
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
```


76


ing
Azure AKS
Nginx Ingress Service
Context Path based
Routing
© Kalyan Reddy Daida
StackSimplify

77


Azure AKS & Nginx Ingress – Context Path Based Routing
Azure
Azure AKS Cluster
svc
pod
deploy
Nginx App1
/app1/
/app2/
/
ing
Ingress Service
Nginx App2
Users
Public IP
for Ingress
Controller
AKS
Load Balancer
Ingress Controller
Usermgmt WebApp
© Kalyan Reddy Daida
StackSimplify

78

Ingress CPR Manifest

```yaml
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
```


79

80

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.

81

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.stacksimplify.com (for a mail server)
  - www.stacksimplify.com (for a website).
  - courses.stacksimplify.com (Sub domains to host other applications)

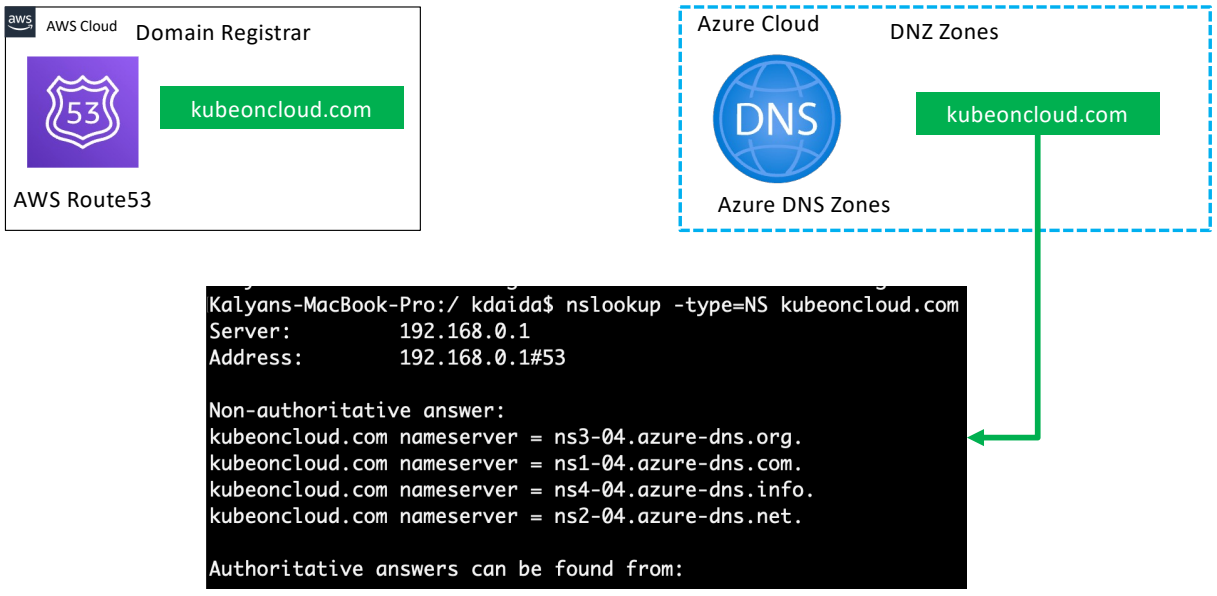
82

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

83

Delegate Domain to Azure DNS



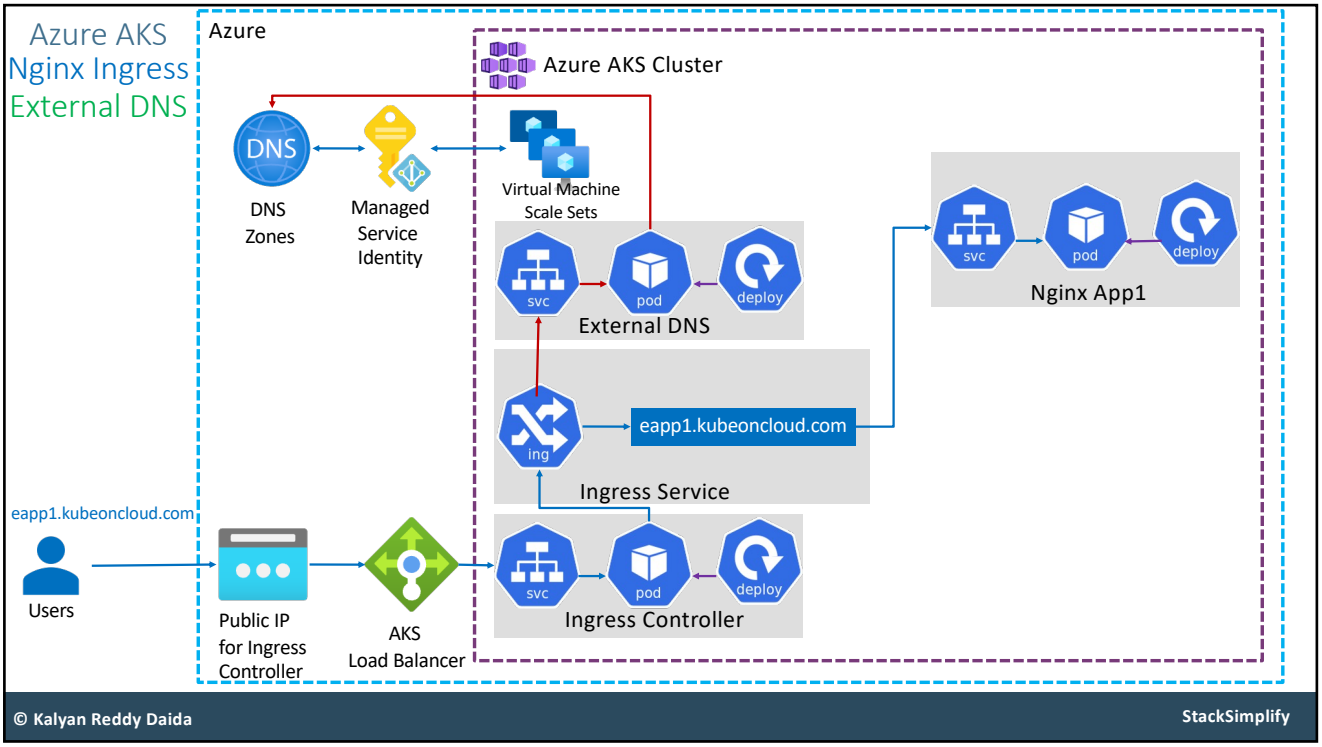
84

85

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

86



87

Ingress with External DNS

```yaml
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
```

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

88

External DNS Updates

66 lines - 4 Removals Copy all

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: external-dns
rules:
- apiGroups: [""]
  resources: ["services","endpoints","pods"]
  verbs: ["get","watch","list"]
- apiGroups: ["extensions","networking.k8s.io"]
  resources: ["ingresses"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
- kind: ServiceAccount
  name: external-dns
  namespace: default
---
```

66 lines + 4 Additions Copy all

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: external-dns
rules:
- apiGroups: [""]
  resources: ["services","endpoints","pods", "nodes"]
  verbs: ["get","watch","list"]
- apiGroups: ["extensions","networking.k8s.io"]
  resources: ["ingresses"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
- kind: ServiceAccount
  name: external-dns
  namespace: default
---
```


89

External DNS Updates

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: external-dns
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: external-dns
  template:
    metadata:
      labels:
        app: external-dns
    spec:
      serviceAccountName: external-dns
      containers:
      - name: external-dns
        image: registry.opensource.zalan.do/teapot/external-dns:latest
        args:
        - --source=service
        - --source=ingress
        #- --domain-filter=example.com # (optional) limit to only example.com domains; change to match the zone created above.
        - --provider=azure
        #- --azure-resource-group=externaldns # (optional) use the DNS zones from the specific resource group
        volumeMounts:
        - name: azure-config-file
          mountPath: /etc/kubernetes
          readOnly: true
      volumes:
      - name: azure-config-file
        secret:
          secretName: azure-config-file
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: external-dns
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: external-dns
  template:
    metadata:
      labels:
        app: external-dns
    spec:
      serviceAccountName: external-dns
      containers:
      - name: external-dns
        image: k8s.gcr.io/external-dns/external-dns:v0.11.0
        args:
        - --source=service
        - --source=ingress
        #- --domain-filter=example.com # (optional) limit to only example.com domains; change to match the zone created above.
        - --provider=azure
        #- --azure-resource-group=externaldns # (optional) use the DNS zones from the specific resource group
        volumeMounts:
        - name: azure-config-file
          mountPath: /etc/kubernetes
          readOnly: true
      volumes:
      - name: azure-config-file
        secret:
          secretName: azure-config-file
```

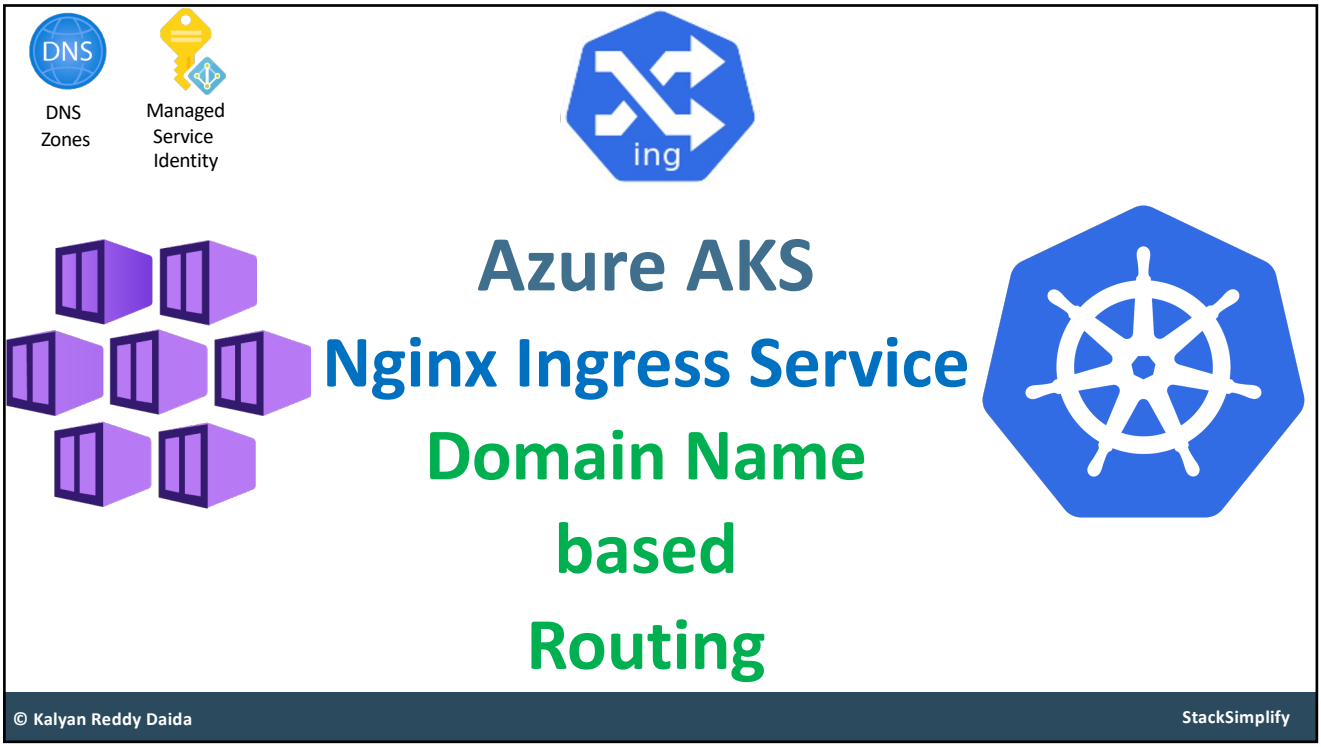

90

Ingress Service Updates

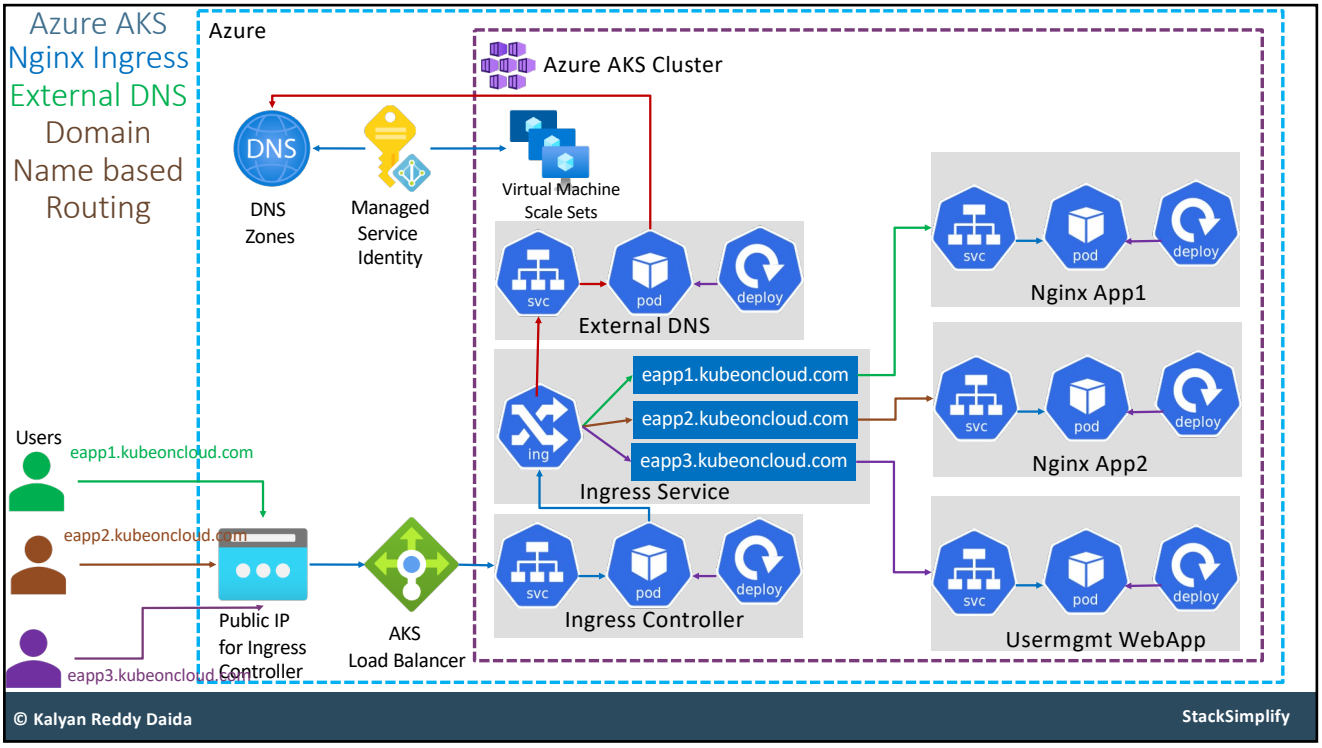
azure-aks-kubernetes-masterclass > 12-ExternalDNS-for-AzureDNS-on-AKS > kube-manifests > 02-NginxApp1 > ! 03-Ingress-with-Exte

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginxapp1-ingress-service
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
    - host: eapp1.kubeoncloud.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: app1-nginx-clusterip-service
                port:
                  number: 80
```


91



92



93

Ingress with External DNS and Domain Name based Routing

```yaml
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
```

Ingress Domain Name Based Routing

```yaml
  - host: eapp3.kubeoncloud.com
    http:
      paths:
      - path: /
        backend:
          serviceName: usermgmt-webapp-clusterip-service
          servicePort: 80
```

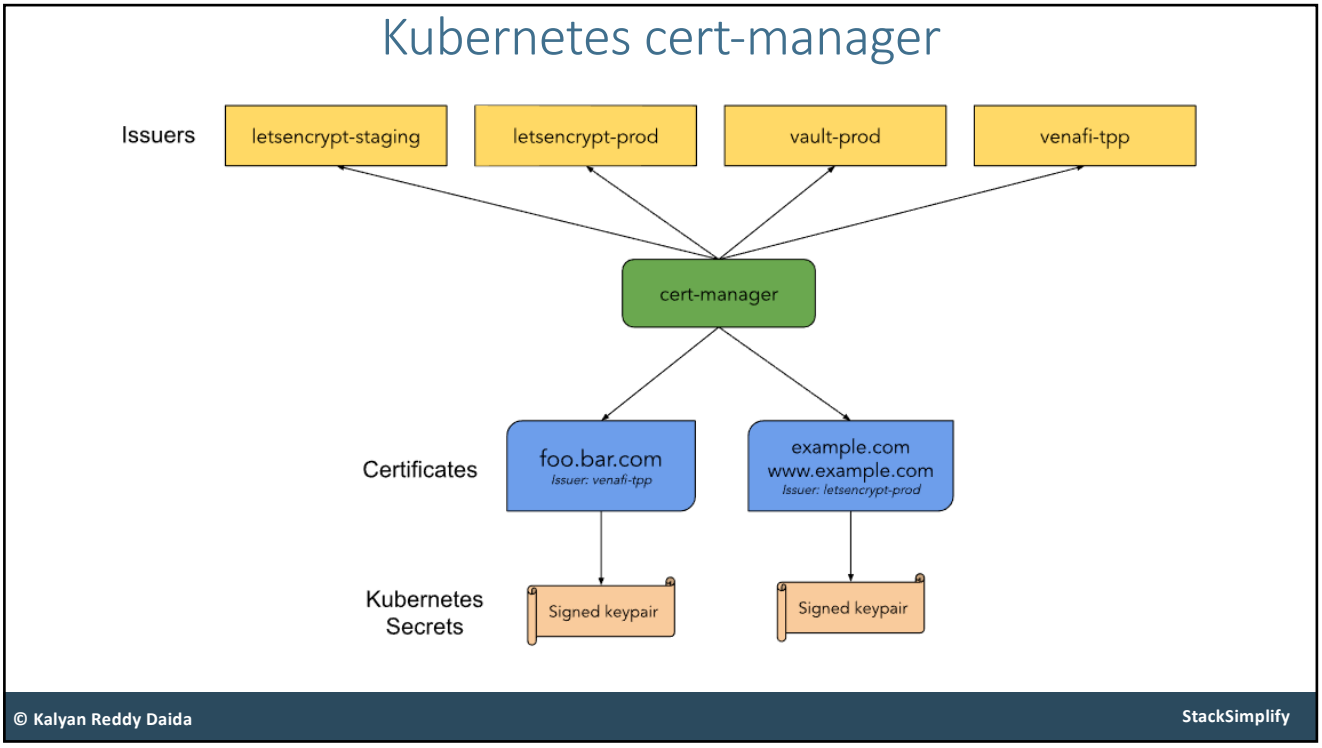

94

95

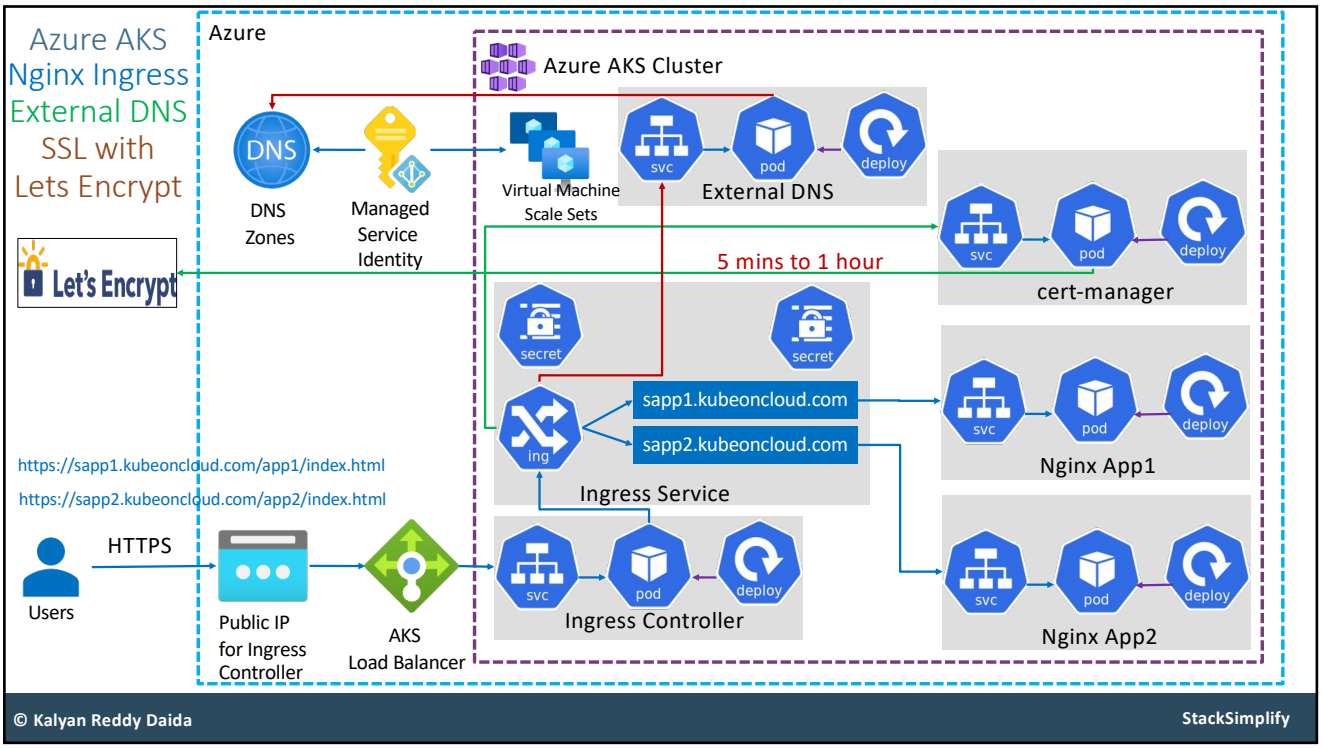
Kubernetes Cert Manager

- 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/

96


Kubernetes cert-manager
Issuers
letsencrypt-staging
letsencrypt-prod
vault-prod
venafi-tpp
cert-manager
Certificates
foo.bar.com
Issuer: venafi-tpp
example.com
www.example.com
Issuer: letsencrypt-prod
Kubernetes Secrets
Signed keypair
Signed keypair
© Kalyan Reddy Daida
StackSimplify

97


Azure AKS Nginx Ingress External DNS SSL with Lets Encrypt
Azure
Azure AKS Cluster
DNS
DNS Zones
Managed Service Identity
Virtual Machine Scale Sets
svc
pod
deploy
External DNS
Let's Encrypt
5 mins to 1 hour
cert-manager
secret
secret
ing
sapp1.kubeoncloud.com
sapp2.kubeoncloud.com
Ingress Service
Nginx App1
https://sapp1.kubeoncloud.com/app1/index.html
https://sapp2.kubeoncloud.com/app2/index.html
HTTPS
Users
Public IP for Ingress Controller
AKS Load Balancer
Ingress Controller
Nginx App2
© Kalyan Reddy Daida
StackSimplify

98

Ingress with External DNS, DNR and SSL

```yaml
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

99

Kubernetes
Resources
Requests & Limits

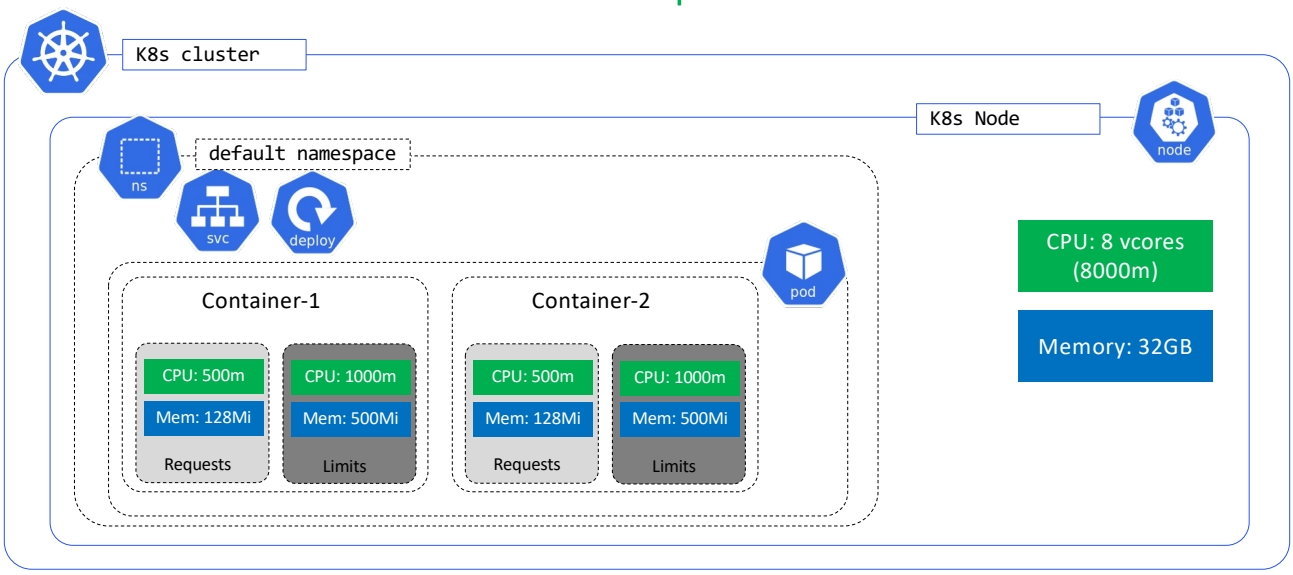
100

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.

101

Kubernetes – Requests & Limits



102

Kubernetes
Requests & Limits

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

```yaml
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: stacksimplify/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"
```


103

Kubernetes
Namespaces

104

Kubernetes - Namespaces

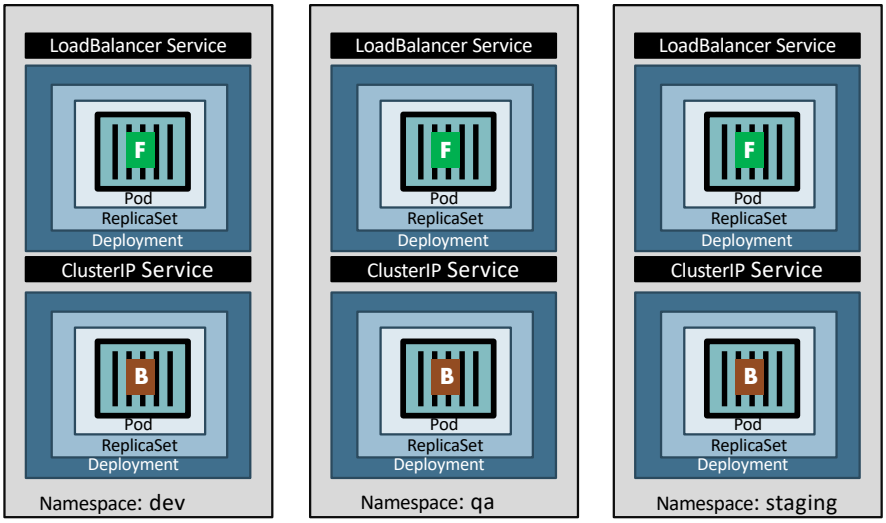
- Namespaces are also called Virtual clusters in our physical k8s cluster
- We use this in environments where we have many users spread across multiple teams or projects
- Clusters with tens of users ideally don't need to use namespaces
- Benefits
  - Creates isolation boundary from other k8s objects
  - 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
```


105

Kubernetes - Namespaces



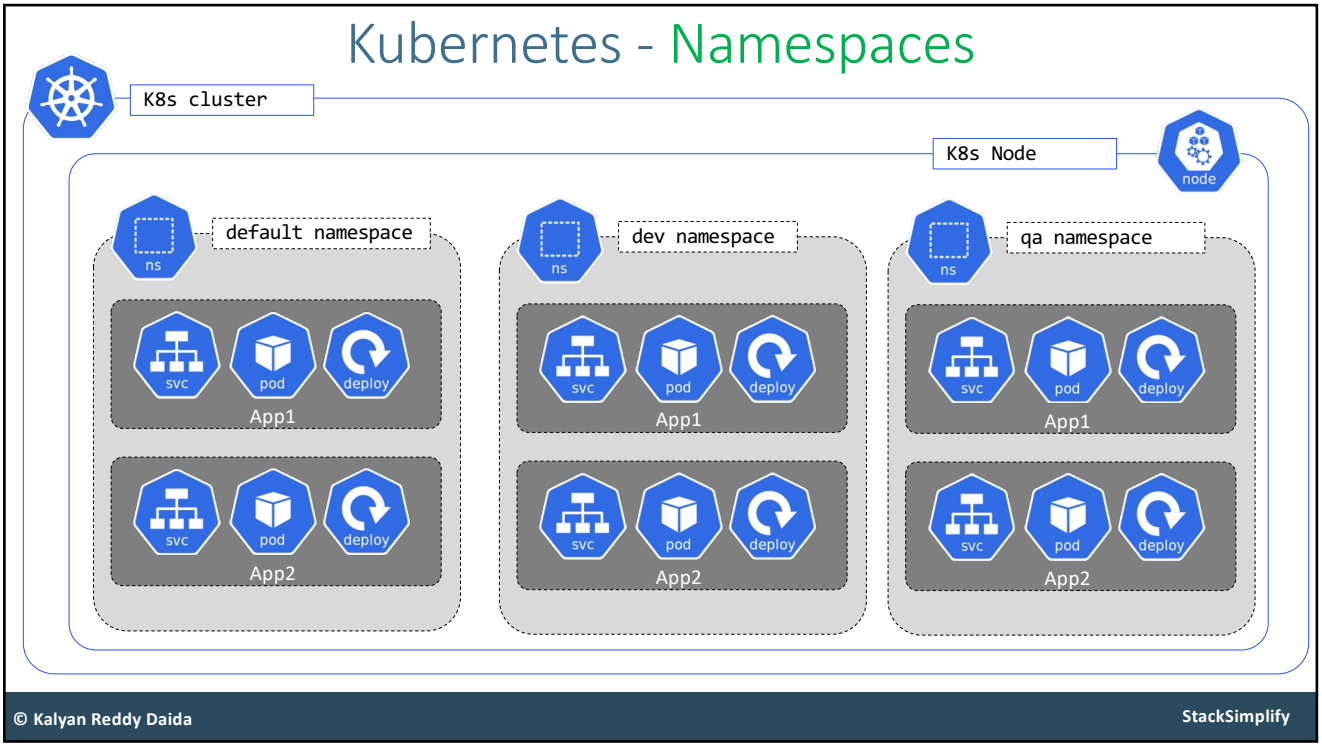
Namespace Manifest - Declarative

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dev3
```

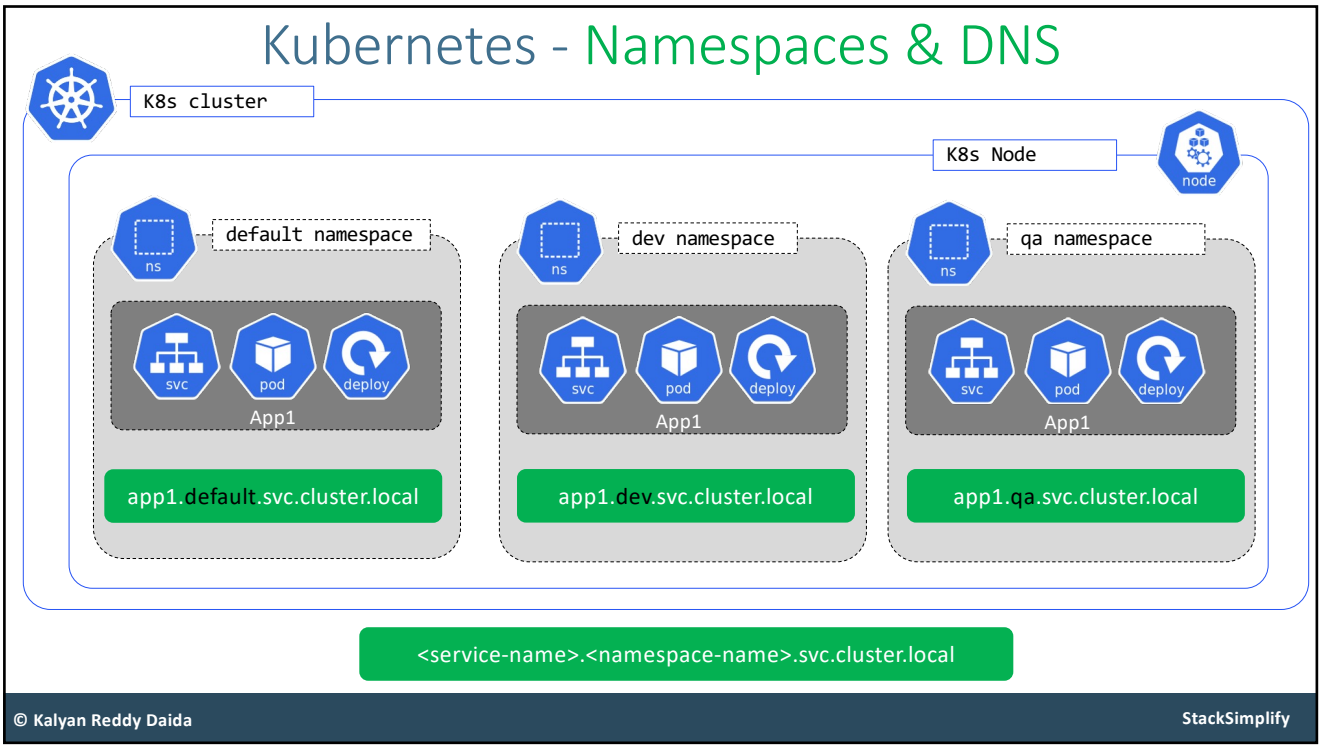
Namespace Manifest - Imperative

```
kubectl create namespace dev1
```


106


Kubernetes - Namespaces
K8s cluster
K8s Node
node
default namespace
dev namespace
qa namespace
ns
svc
pod
deploy
App1
App2
© Kalyan Reddy Daida
StackSimplify

107


Kubernetes - Namespaces & DNS
K8s cluster
K8s Node
node
default namespace
dev namespace
qa namespace
ns
svc
pod
deploy
App1
app1.default.svc.cluster.local
app1.dev.svc.cluster.local
app1.qa.svc.cluster.local
<service-name>.<namespace-name>.svc.cluster.local
© Kalyan Reddy Daida
StackSimplify

108

Kubernetes Namespaces Limit Range

109

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.

110

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.

111

Limit Range

Limit Range Manifest

```yaml
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
```


112

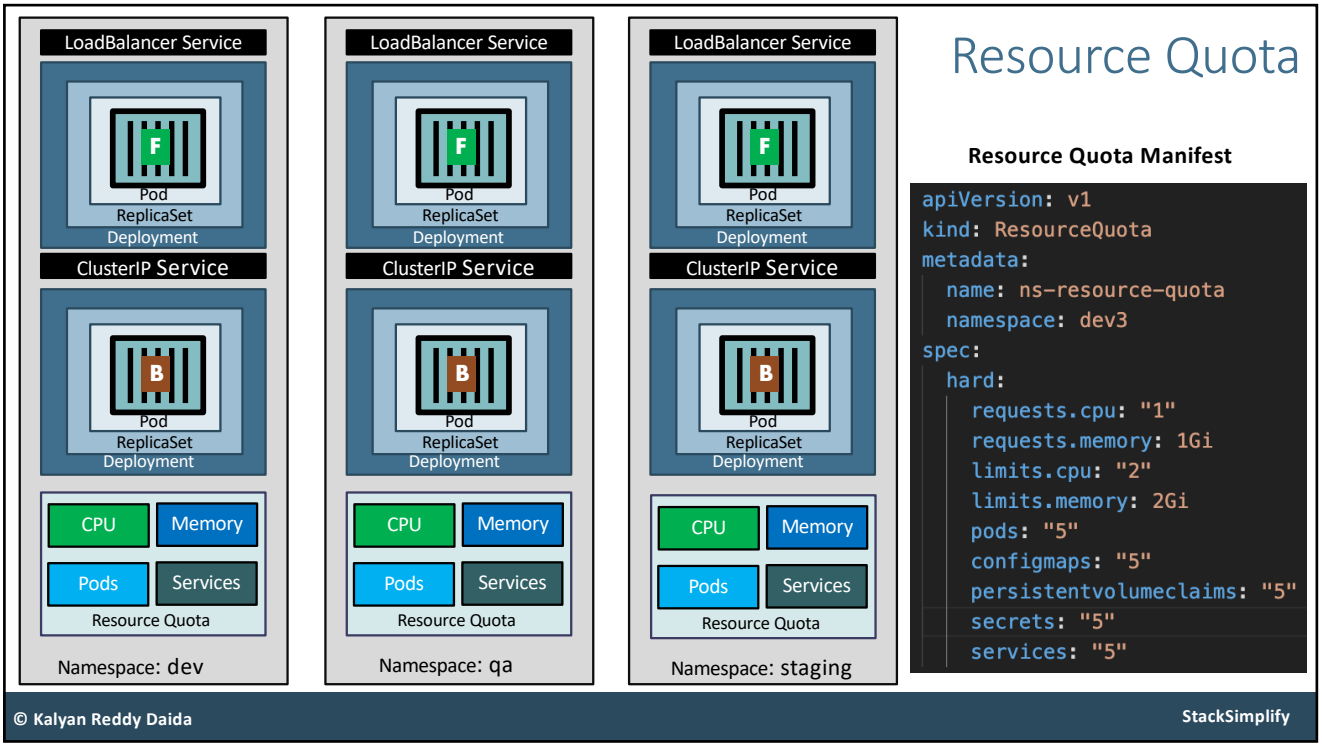
Kubernetes
Namespaces
Resource Quota

113

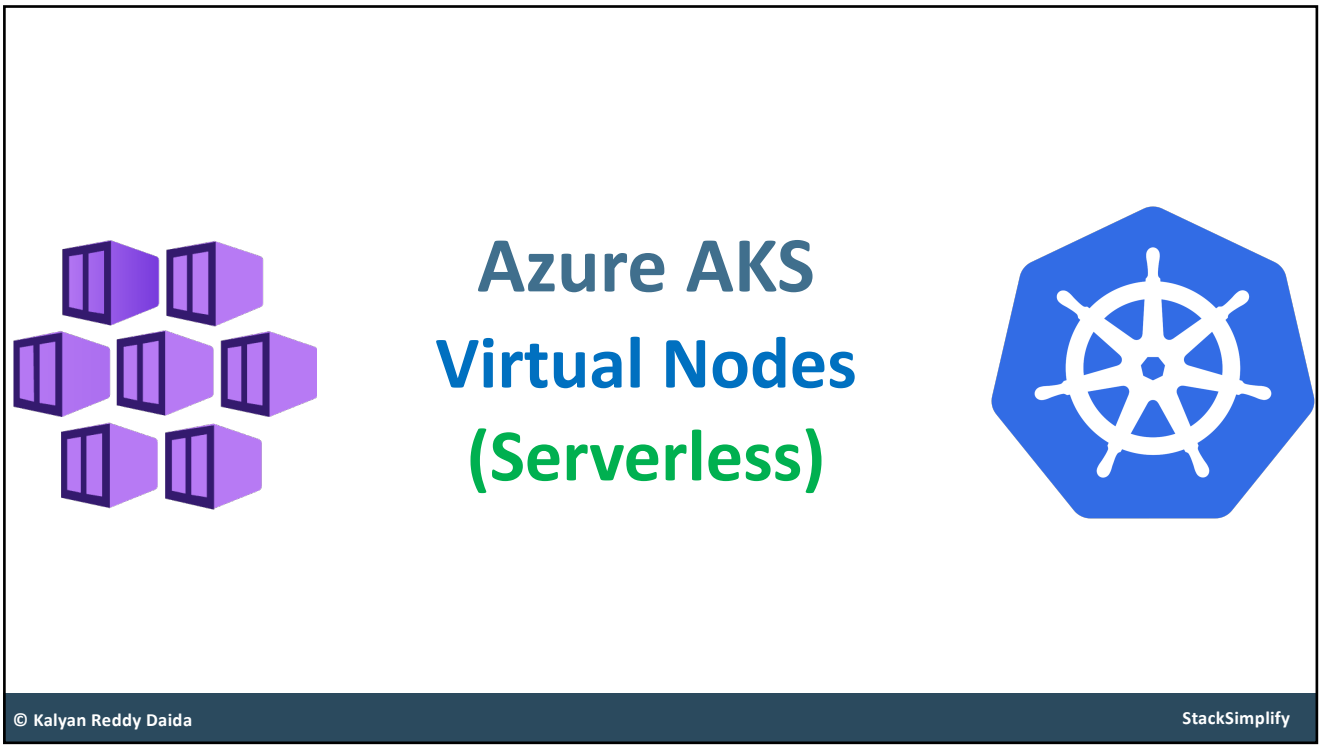
Namespaces - Resource Quota

- 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.

114

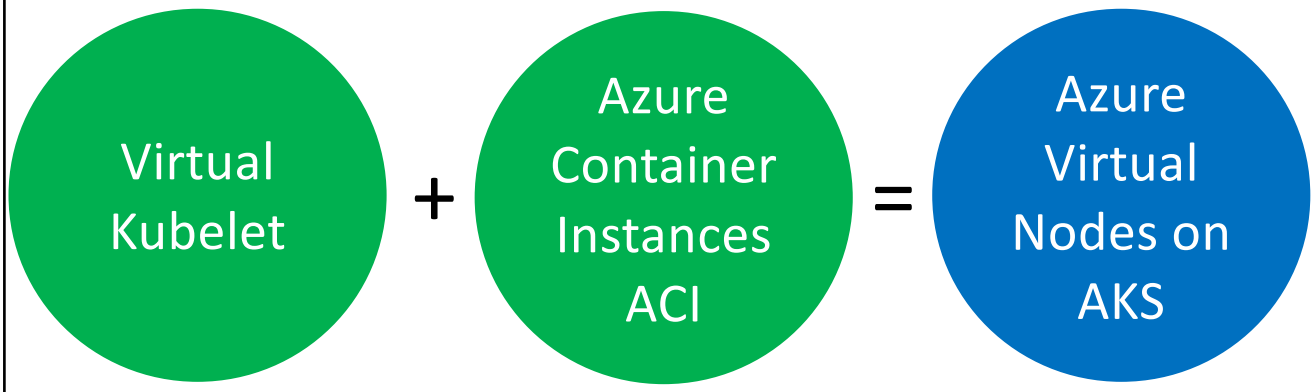

LoadBalancer Service
F
Pod
ReplicaSet
Deployment
ClusterIP Service
B
Pod
ReplicaSet
Deployment
CPU
Memory
Pods
Services
Resource Quota
Namespace: dev
Namespace: qa
Namespace: staging
Resource Quota
Resource Quota Manifest
apiVersion: v1
kind: ResourceQuota
metadata:
name: ns-resource-quota
namespace: 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"
© Kalyan Reddy Daida
StackSimplify

115


Azure AKS
Virtual Nodes
(Serverless)
© Kalyan Reddy Daida
StackSimplify

116

Azure AKS Virtual Nodes



117

What is Virtual Kubelet?

Virtual Kubelet

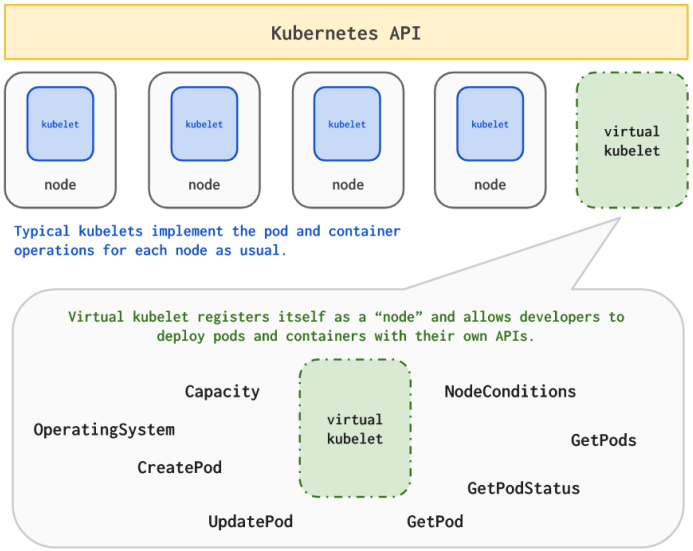
- Virtual Kubelet is an open source Kubernetes kubelet implementation that acts as a kubelet for the purposes of connecting Kubernetes to other APIs.
- This allows the k8s worker nodes to be backed by other services like Azure ACI and AWS Fargate
- The primary scenario for VK is enabling the extension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

118

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



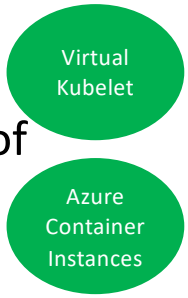
What is ACI (Azure Container Instances)?

- 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

Azure Container Instances

Kubernetes Virtual Kubelet with Azure ACI

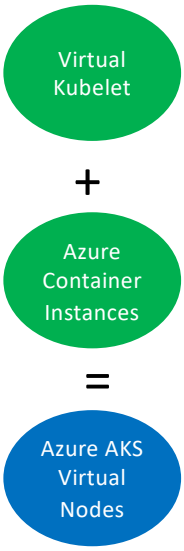
- The Azure Container Instances (ACI) provider for the Virtual Kubelet configures an ACI instance as a node in any Kubernetes cluster.
- When using the Virtual Kubelet ACI provider, pods can be scheduled on an ACI instance as if the ACI instance is a standard Kubernetes node.
- This configuration allows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



121

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



122

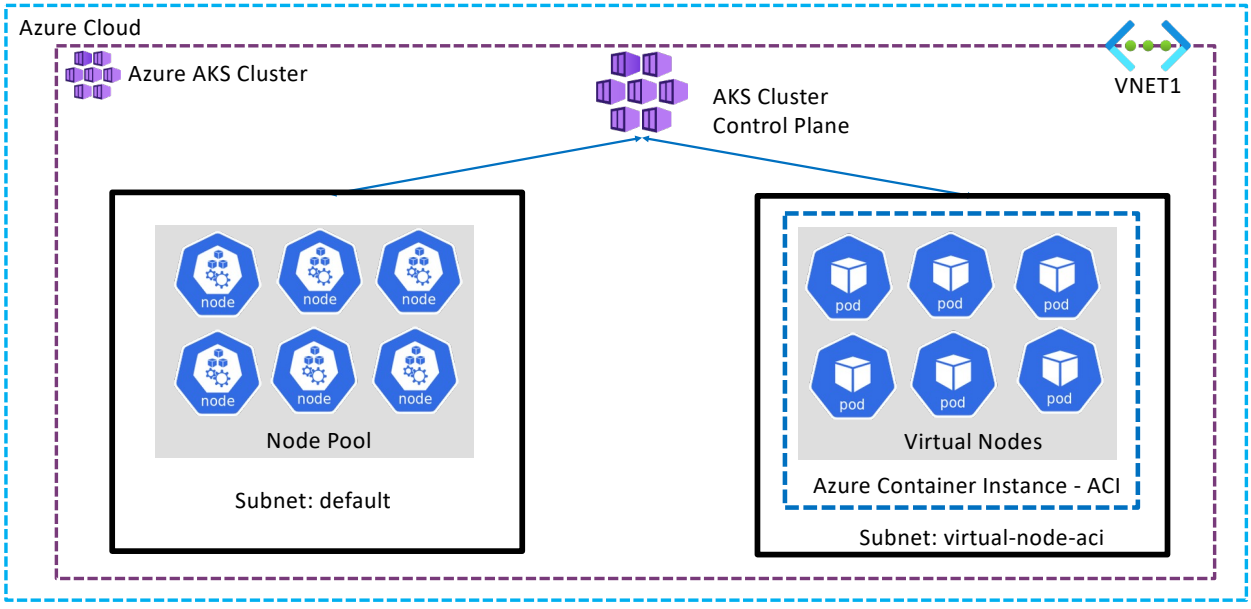
Azure AKS – Virtual Nodes (Serverless)

Azure Virtual Nodes on AKS

- 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

123

Azure AKS - Virtual Nodes



124

AKS Virtual Nodes
NodeSelector

```yaml
    spec:
      containers:
        - name: app1-nginx
          image: stacksimplify/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
```

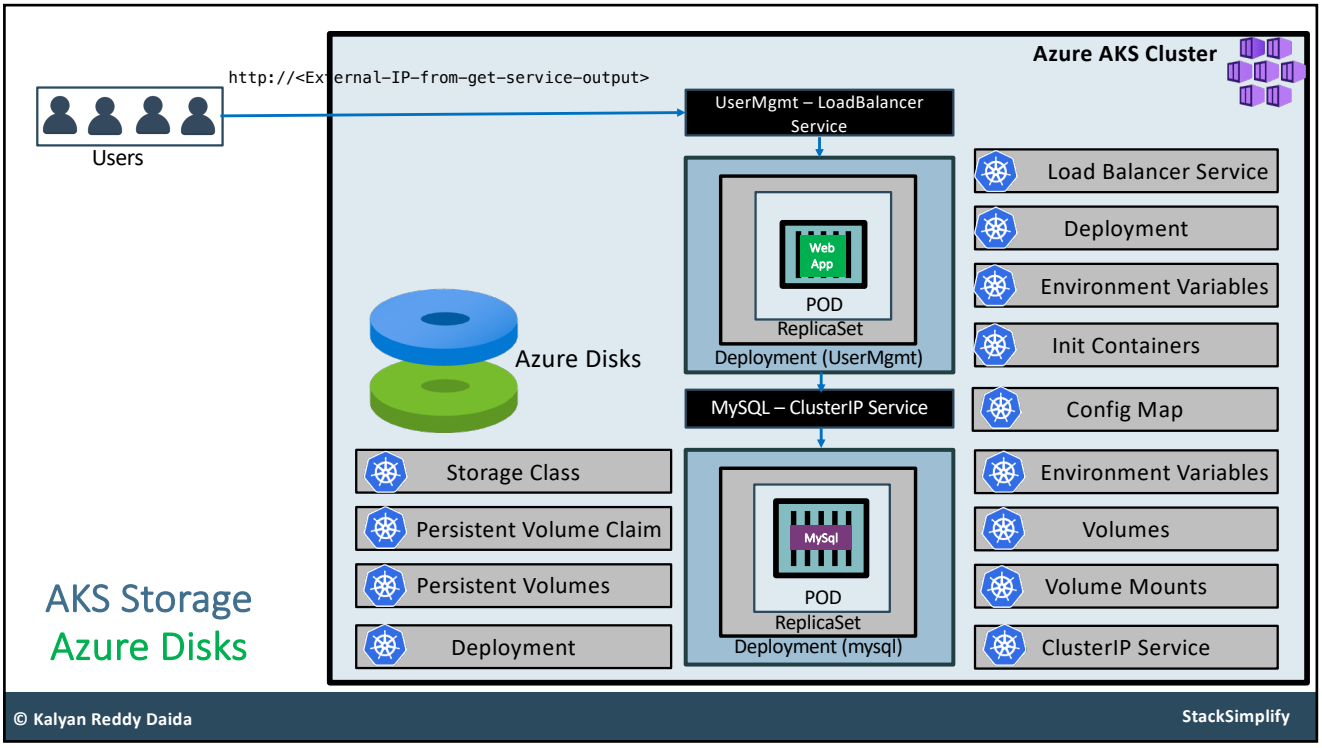
01-NginxApp1-Deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app1-nginx-deployment
  labels:
    app: app1-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: app1-nginx
  template:
    metadata:
      labels:
        app: app1-nginx
    spec:
      containers:
        - name: app1-nginx
          image: stacksimplify/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
```

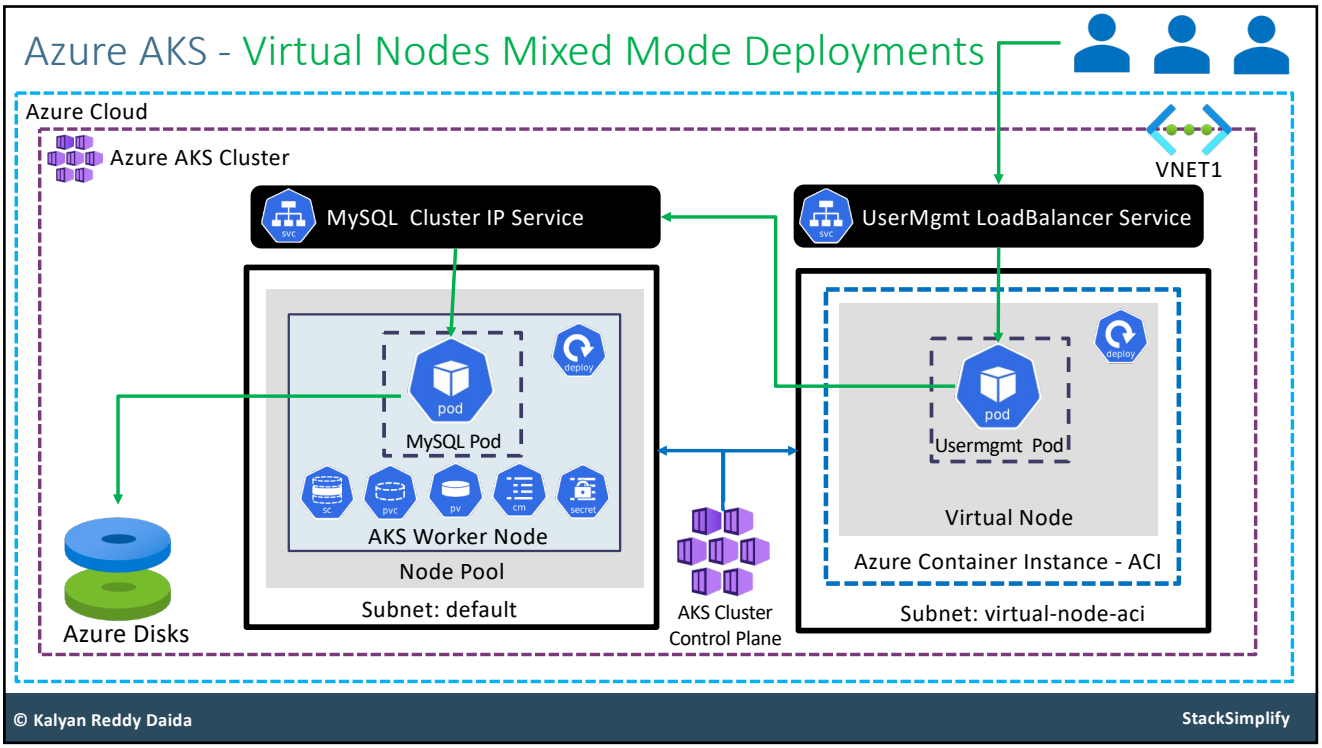

125

Azure AKS
Virtual Nodes
(Serverless)
Mixed Mode Deployments

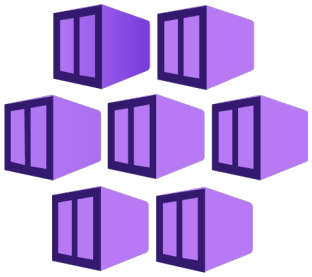
126


Azure AKS Cluster
http://<External-IP-from-get-service-output>
Users
UserMgmt – LoadBalancer Service
Load Balancer Service
Deployment
Environment Variables
Init Containers
Web App
POD
ReplicaSet
Deployment (UserMgmt)
Azure Disks
MySQL – ClusterIP Service
Config Map
Storage Class
Environment Variables
Persistent Volume Claim
Volumes
Persistent Volumes
Volume Mounts
MySql
POD
ReplicaSet
Deployment (mysql)
Deployment
ClusterIP Service
AKS Storage
Azure Disks
© Kalyan Reddy Daida
StackSimplify

127


Azure AKS - Virtual Nodes Mixed Mode Deployments
Azure Cloud
Azure AKS Cluster
VNET1
MySQL Cluster IP Service
UserMgmt LoadBalancer Service
pod
MySQL Pod
deploy
sc
pvc
pv
cm
secret
AKS Worker Node
Node Pool
Subnet: default
Azure Disks
AKS Cluster
Control Plane
pod
Usermgmt Pod
deploy
Virtual Node
Azure Container Instance - ACI
Subnet: virtual-node-aci
© Kalyan Reddy Daida
StackSimplify

128



Azure AKS

Azure Container Registry (ACR)

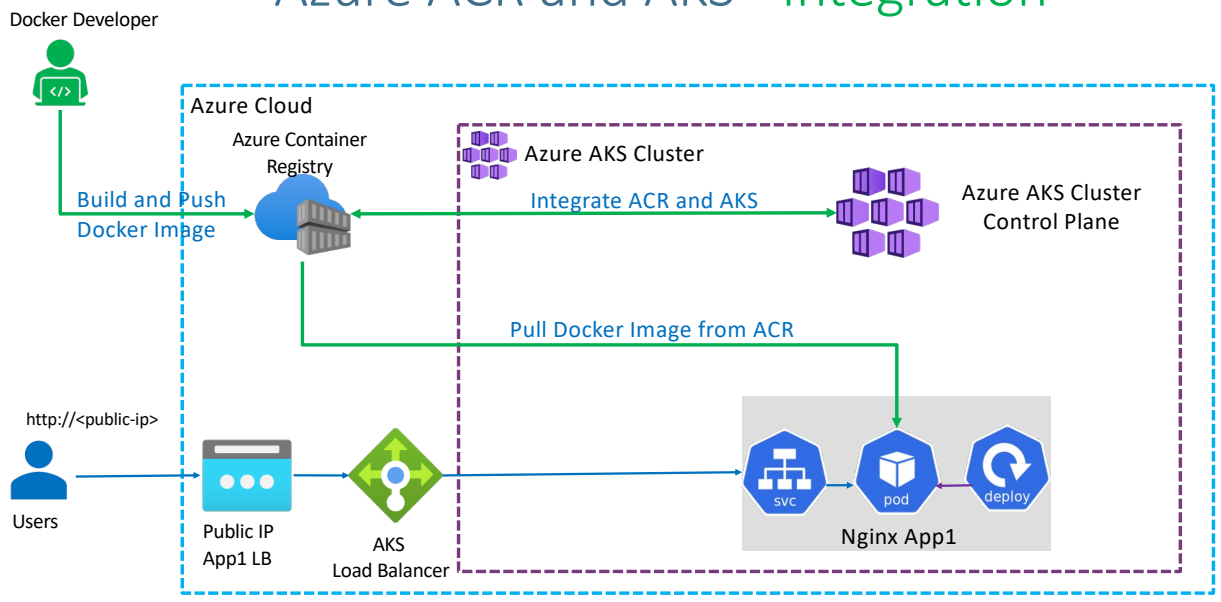
129

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

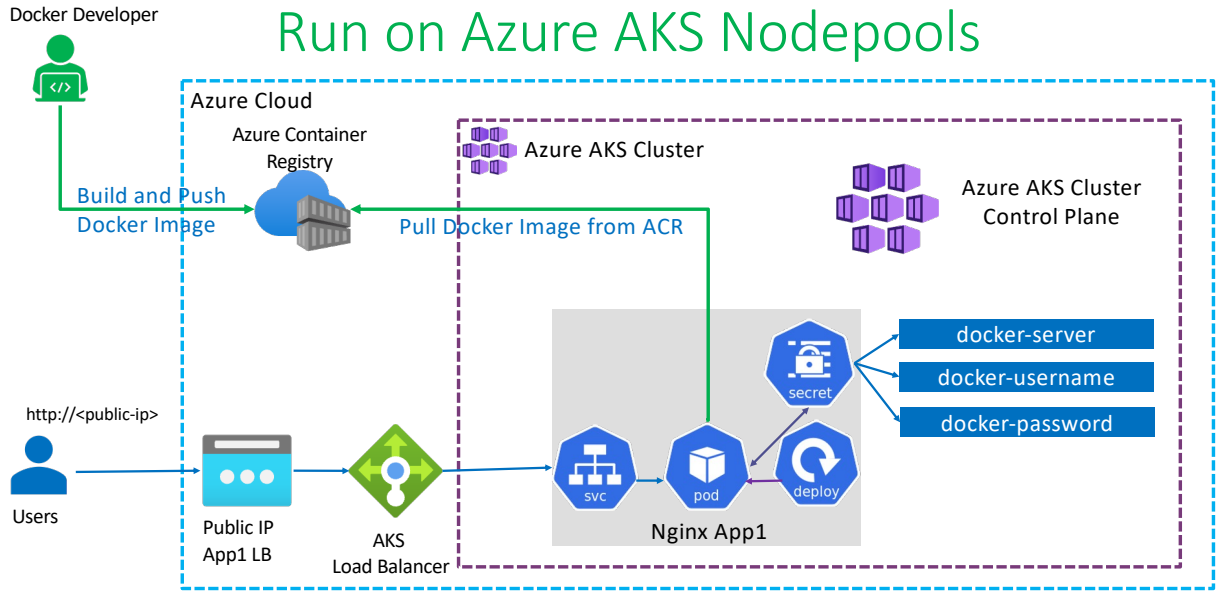
130

Azure ACR and AKS - Integration



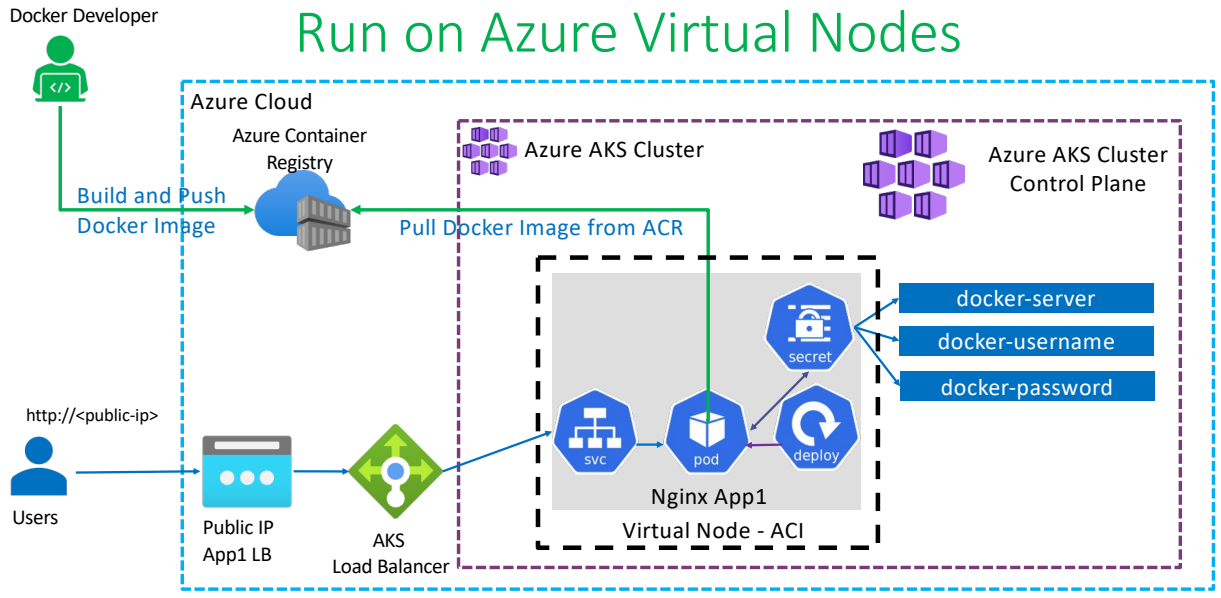
131

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



132

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



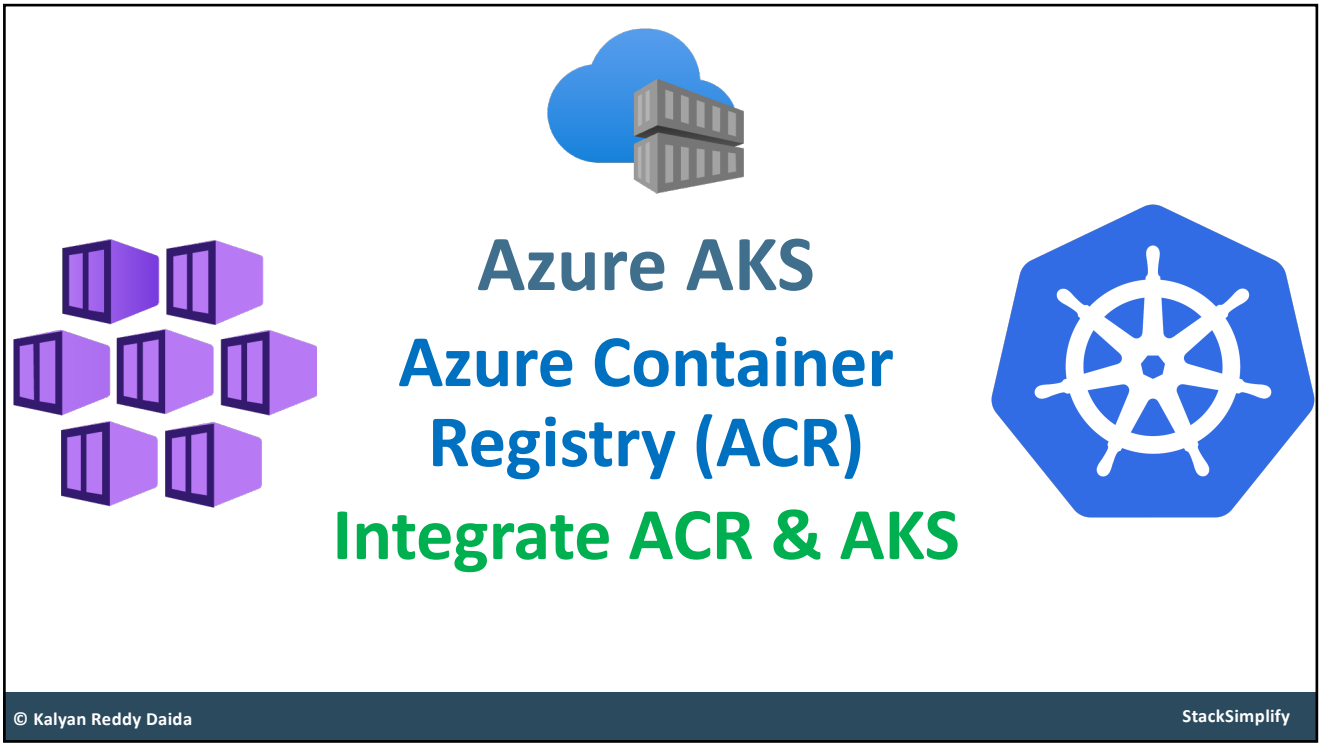
133

Azure Container Registry – Pricing Tiers

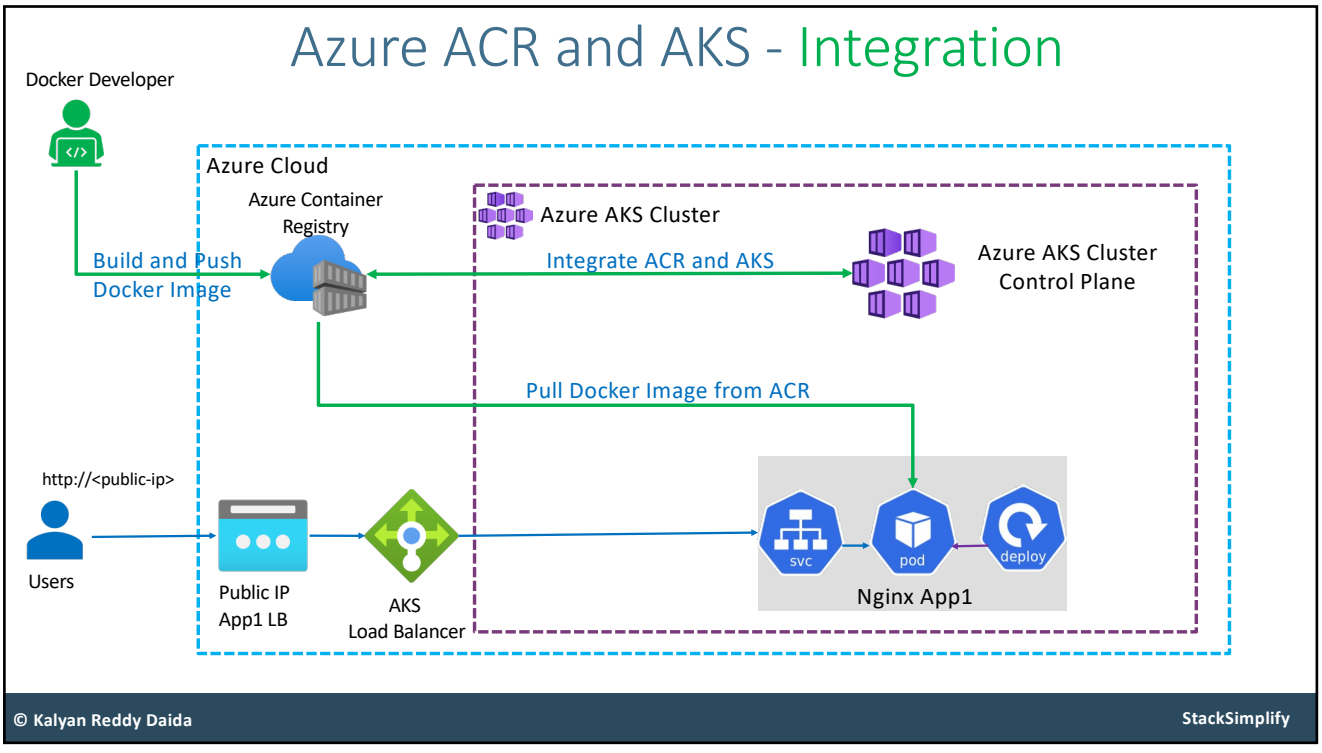
Basic	Standard	Premium
Enabled as Public Endpoint (Public Docker Registry)	Enabled as Public Endpoint (Public Docker Registry)	Enabled as Public & Private Endpoint (Private Docker Registry)
No Customer Managed Key for Encryption	No Customer Managed Key for Encryption	Can use Customer Managed Key for 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

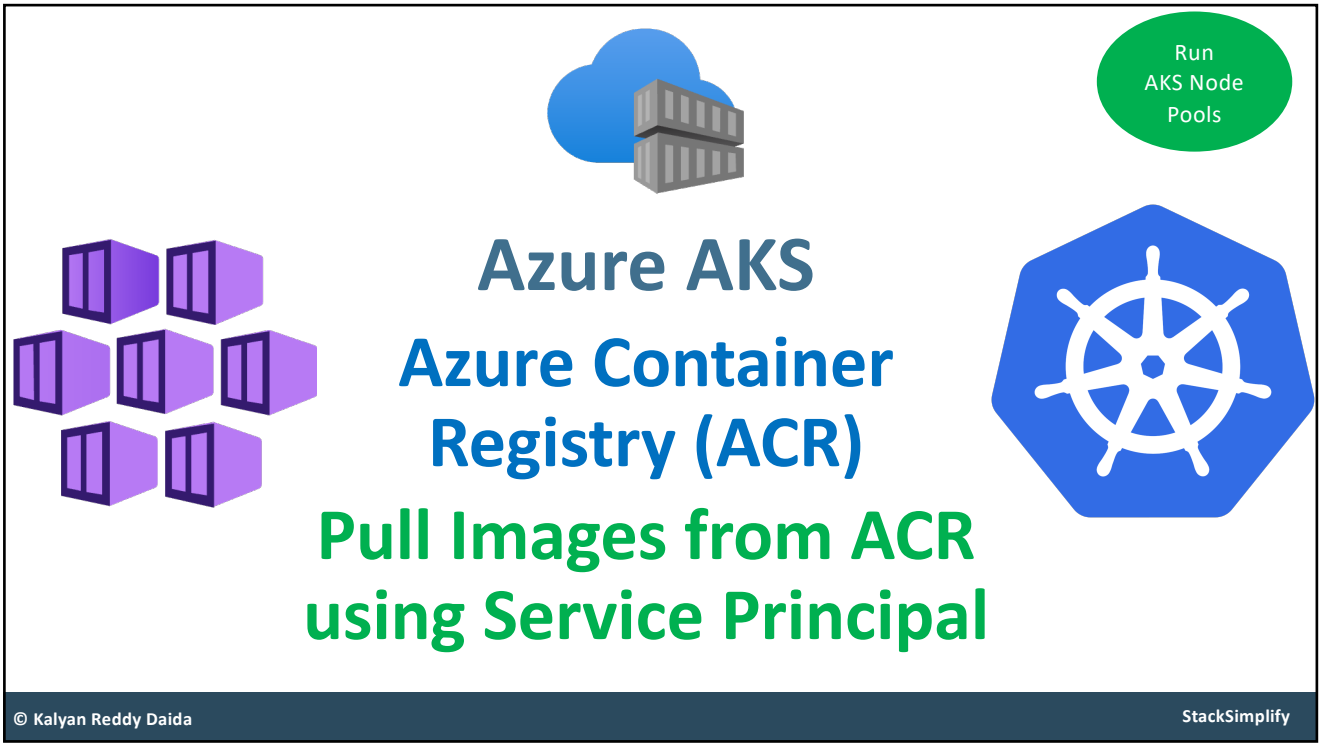
134


Azure AKS
Azure Container Registry (ACR)
Integrate ACR & AKS
© Kalyan Reddy Daida
StackSimplify

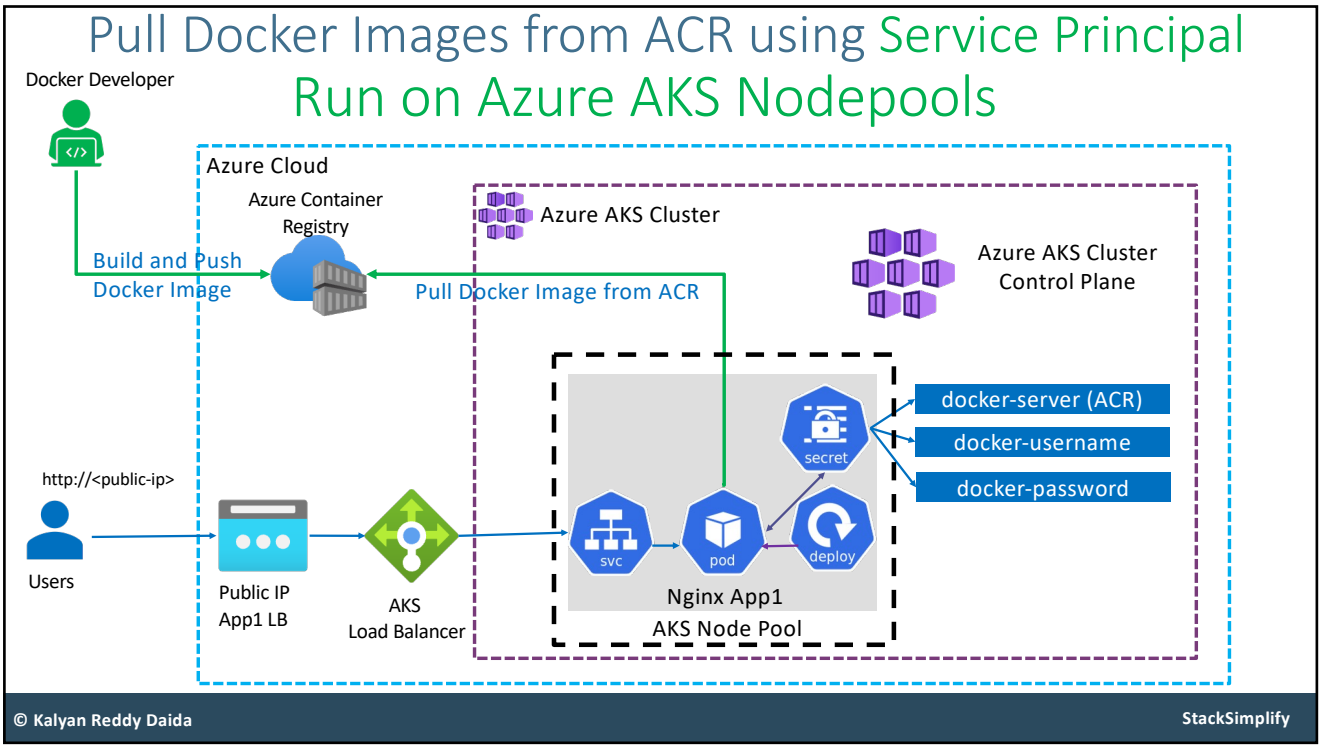
135


Azure ACR and AKS - Integration
Docker Developer
Azure Cloud
Azure Container Registry
Azure AKS Cluster
Build and Push Docker Image
Integrate ACR and AKS
Azure AKS Cluster Control Plane
Pull Docker Image from ACR
http://<public-ip>
Users
Public IP App1 LB
AKS Load Balancer
svc
pod
deploy
Nginx App1
© Kalyan Reddy Daida
StackSimplify

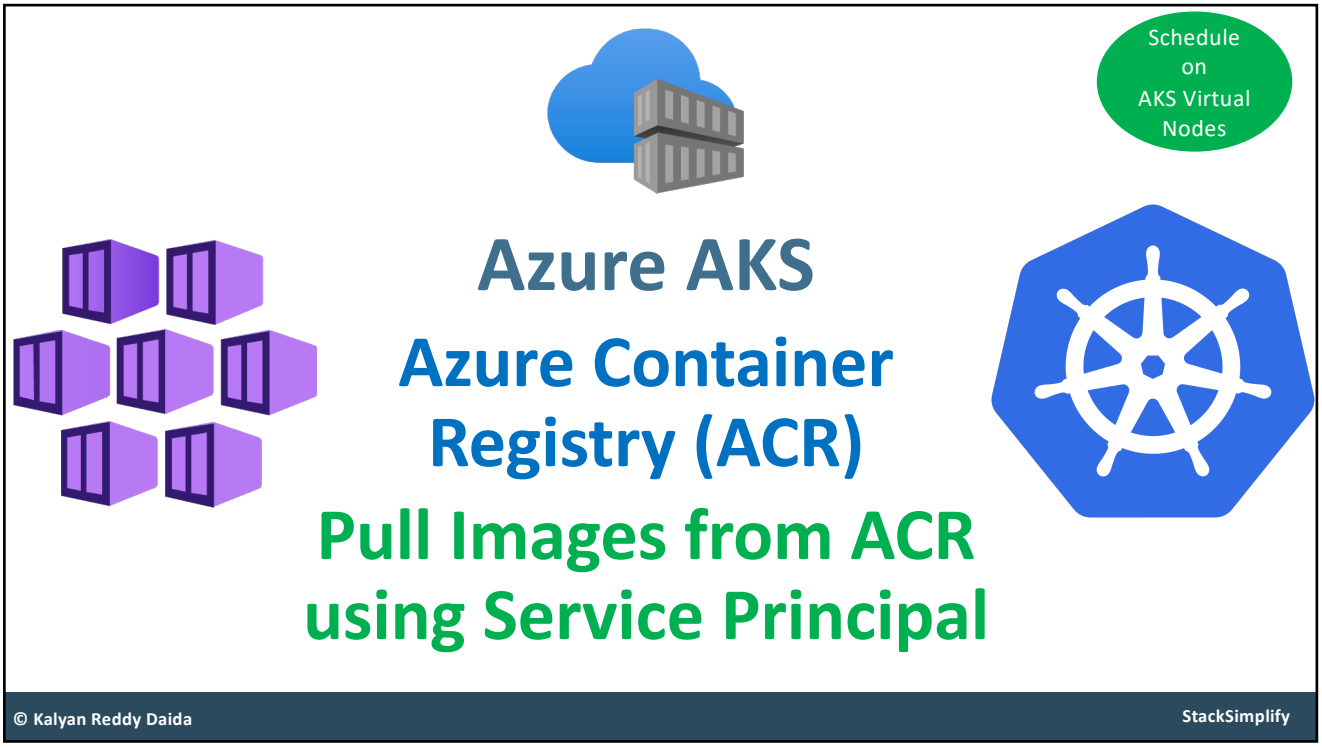
136


Run
AKS Node
Pools
Azure AKS
Azure Container
Registry (ACR)
Pull Images from ACR
using Service Principal
© Kalyan Reddy Daida
StackSimplify

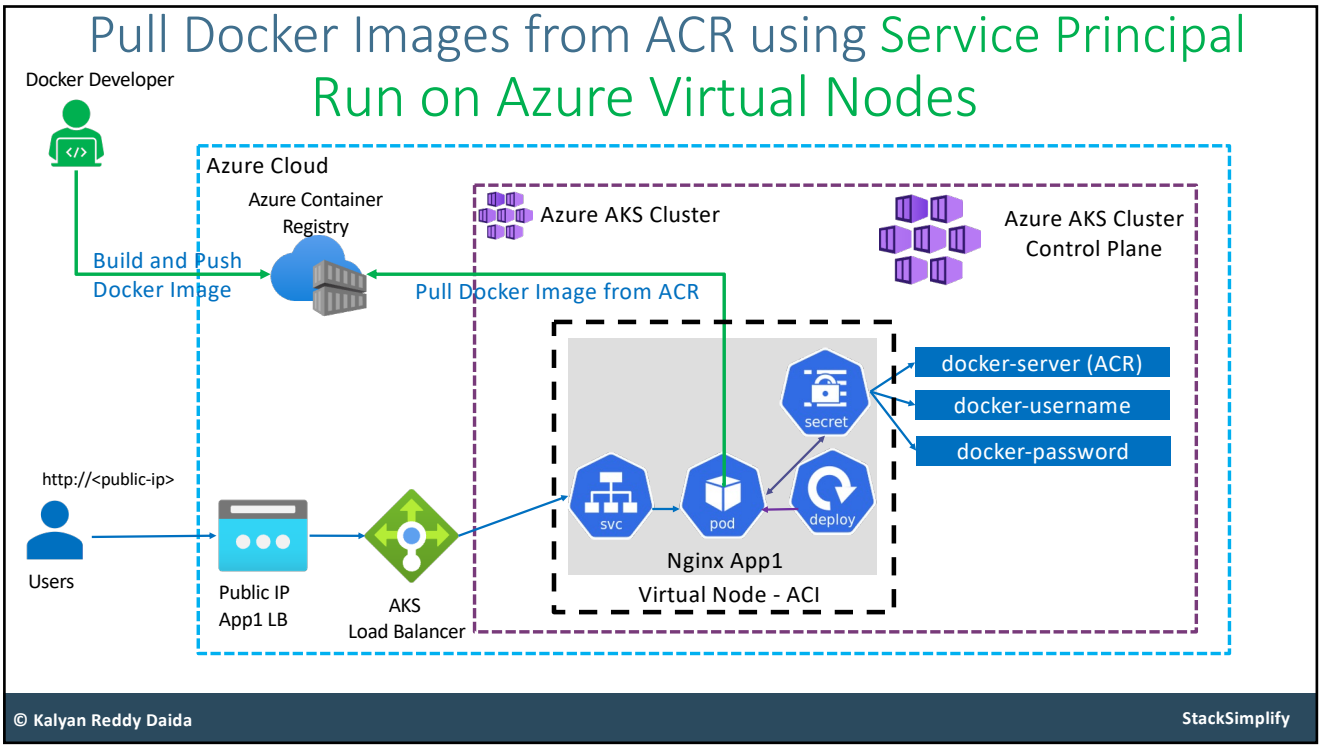
137


Pull Docker Images from ACR using Service Principal
Run on Azure AKS Nodepools
Docker Developer
Azure Cloud
Azure Container
Registry
Azure AKS Cluster
Build and Push
Docker Image
Pull Docker Image from ACR
Azure AKS Cluster
Control Plane
http://<public-ip>
Users
Public IP
App1 LB
AKS
Load Balancer
secret
svc
pod
deploy
Nginx App1
AKS Node Pool
docker-server (ACR)
docker-username
docker-password
© Kalyan Reddy Daida
StackSimplify

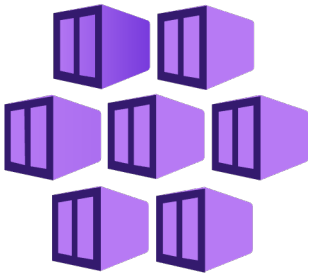
138


Schedule on AKS Virtual Nodes
Azure AKS
Azure Container Registry (ACR)
Pull Images from ACR using Service Principal
© Kalyan Reddy Daida
StackSimplify

139


Pull Docker Images from ACR using Service Principal
Run on Azure Virtual Nodes
Docker Developer
Azure Cloud
Azure Container Registry
Build and Push Docker Image
Azure AKS Cluster
Azure AKS Cluster Control Plane
Pull Docker Image from ACR
secret
docker-server (ACR)
docker-username
docker-password
http://<public-ip>
svc
pod
deploy
Users
Public IP App1 LB
AKS Load Balancer
Nginx App1
Virtual Node - ACI
© Kalyan Reddy Daida
StackSimplify

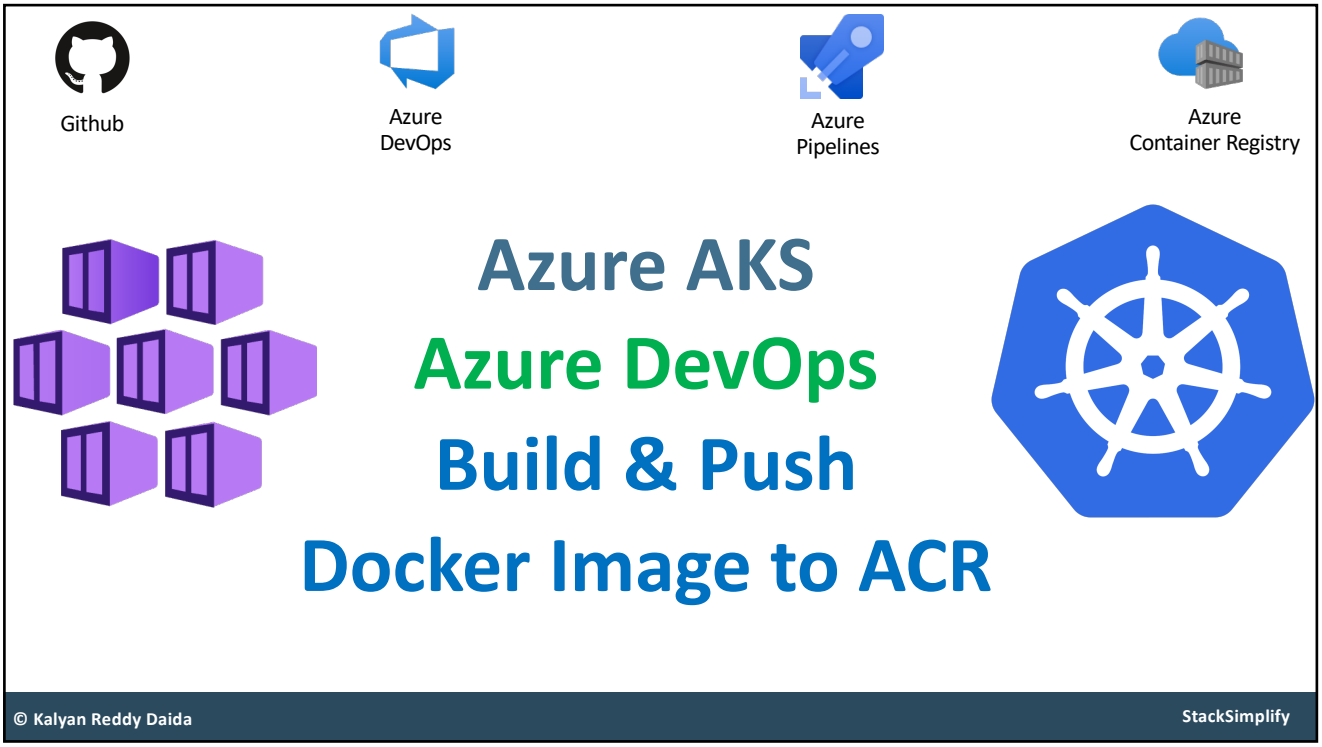
140



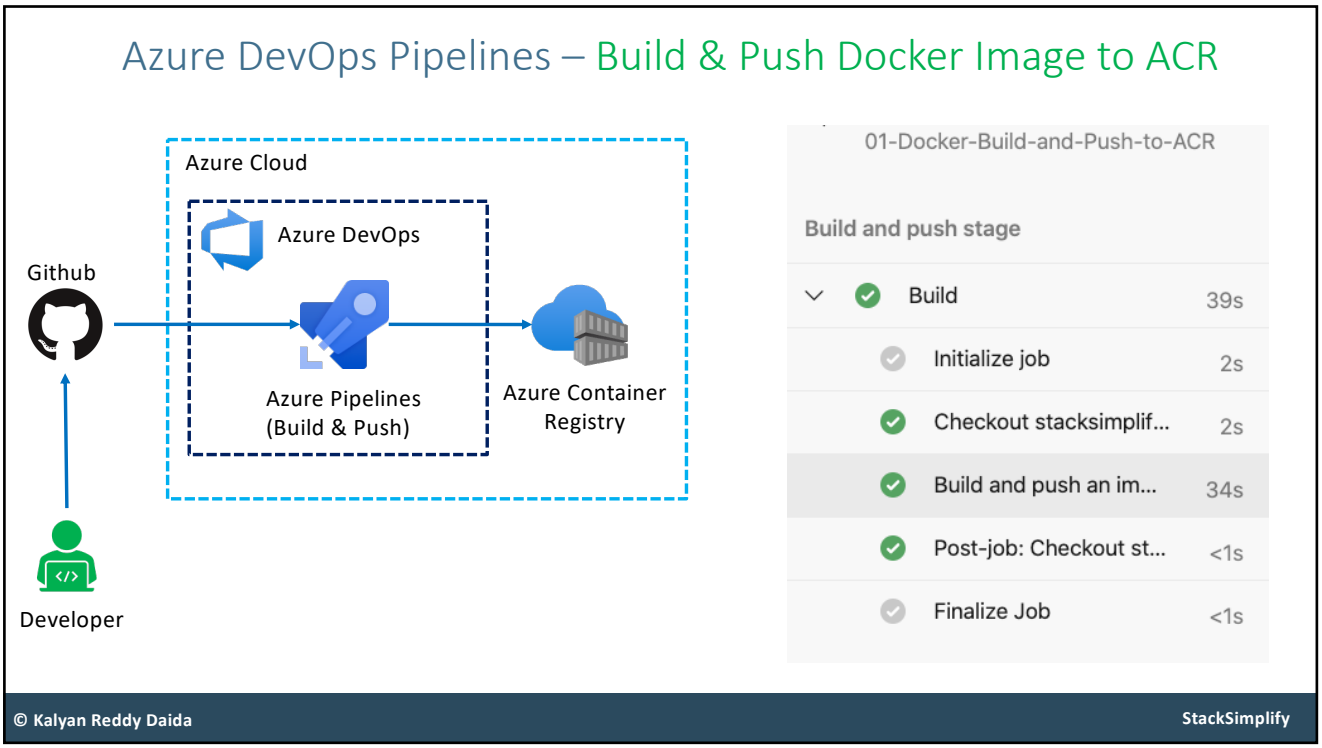
Azure AKS
Azure DevOps
Introduction

141

142

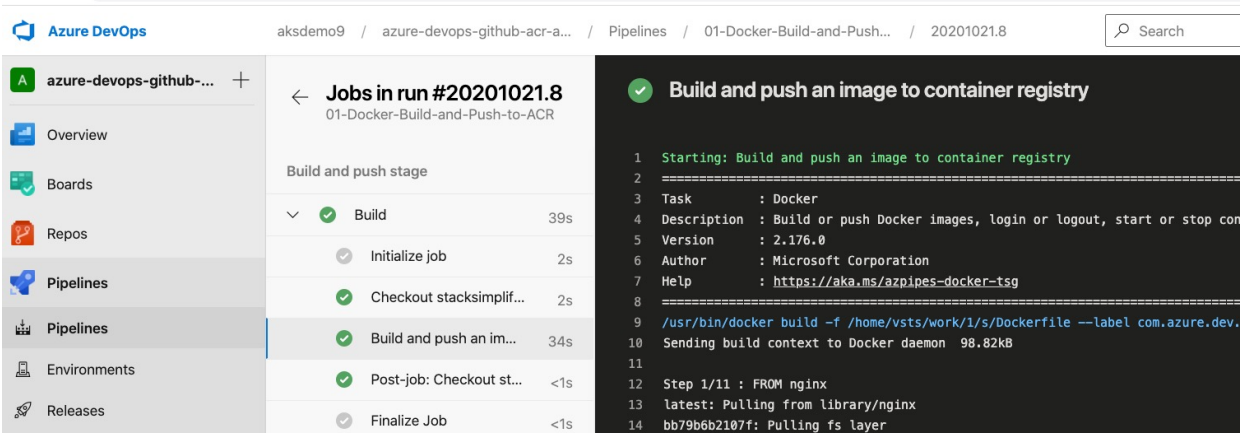

Github
Azure DevOps
Azure Pipelines
Azure Container Registry
Azure AKS
Azure DevOps
Build & Push
Docker Image to ACR
© Kalyan Reddy Daida
StackSimplify

143

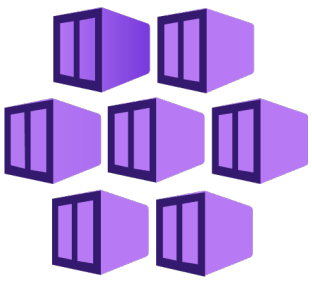

Azure DevOps Pipelines – Build & Push Docker Image to ACR
Azure Cloud
Azure DevOps
Github
Azure Pipelines
(Build & Push)
Azure Container
Registry
Developer
01-Docker-Build-and-Push-to-ACR
Build and push stage
Build 39s
Initialize job 2s
Checkout stacksimplif... 2s
Build and push an im... 34s
Post-job: Checkout st... <1s
Finalize Job <1s
© Kalyan Reddy Daida
StackSimplify

144

Azure DevOps Pipelines – Build & Push Docker Image to ACR

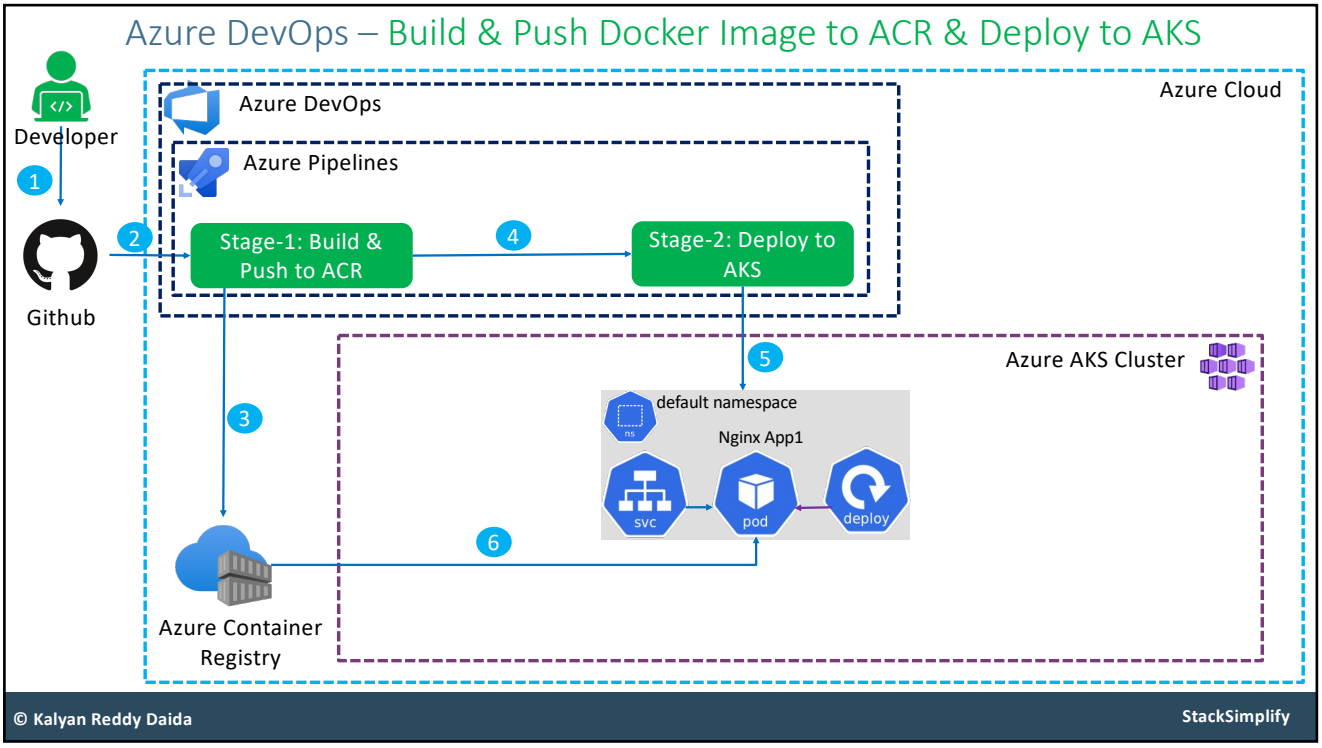


145

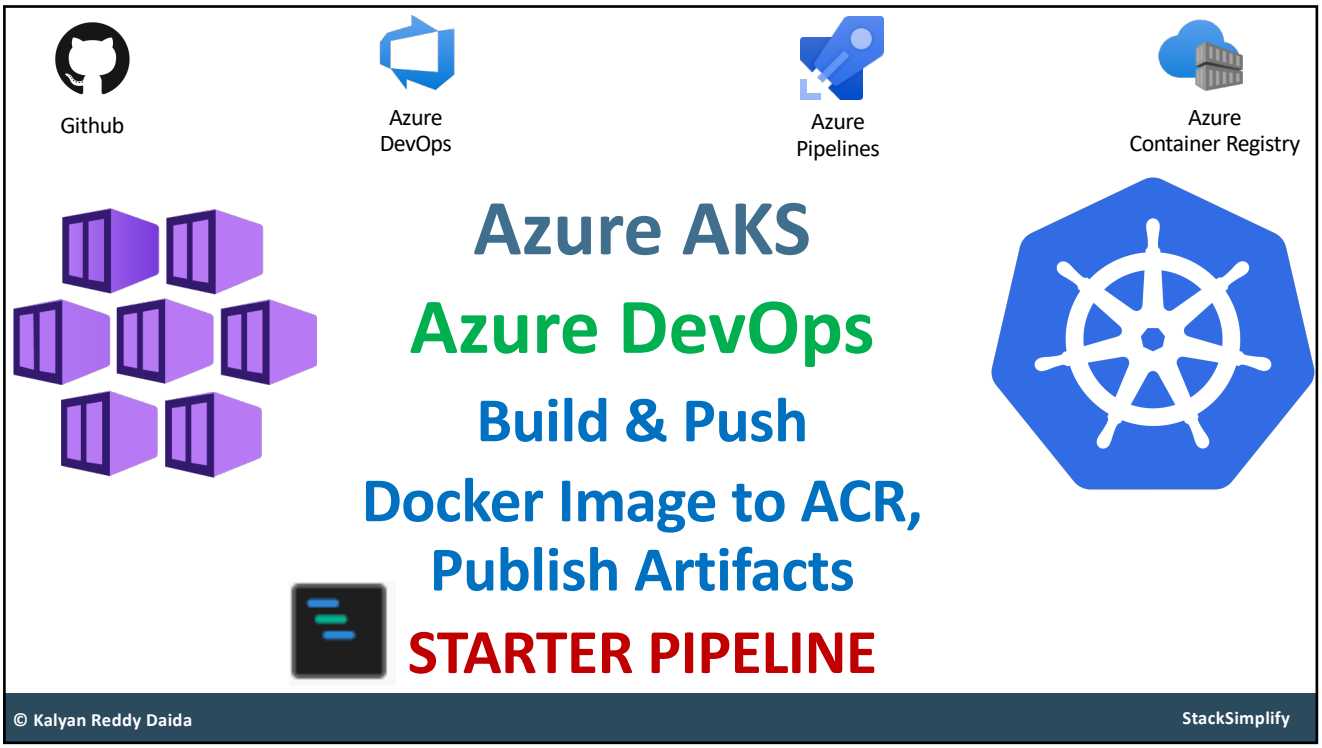


Azure AKS
Azure DevOps
Build & Push
Docker Image to ACR
Deploy to AKS

146

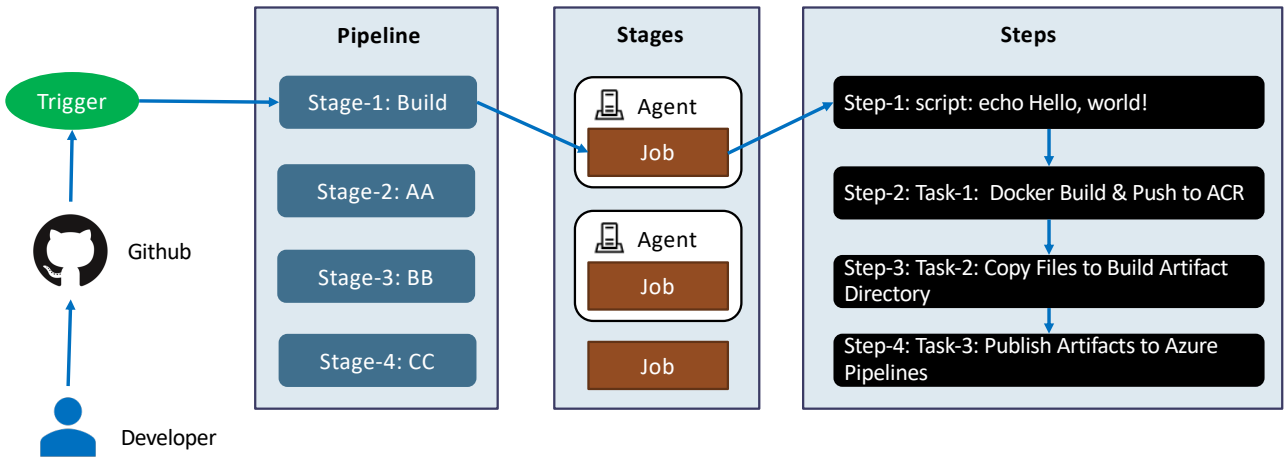

Azure DevOps – Build & Push Docker Image to ACR & Deploy to AKS
Developer
Azure Cloud
Azure DevOps
Azure Pipelines
1
2
Github
Stage-1: Build & Push to ACR
4
Stage-2: Deploy to AKS
5
Azure AKS Cluster
default namespace
Nginx App1
svc
pod
deploy
3
6
Azure Container Registry
© Kalyan Reddy Daida
StackSimplify

147


Github
Azure DevOps
Azure Pipelines
Azure Container Registry
Azure AKS
Azure DevOps
Build & Push
Docker Image to ACR, Publish Artifacts
STARTER PIPELINE
© Kalyan Reddy Daida
StackSimplify

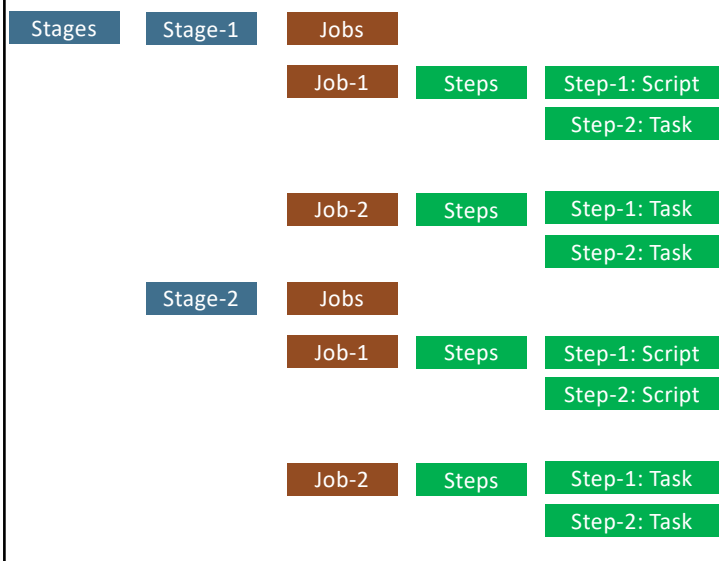
148

Azure Pipelines – Key Concepts



149

Azure Pipelines – Key Concepts




```yaml
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
```


150

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

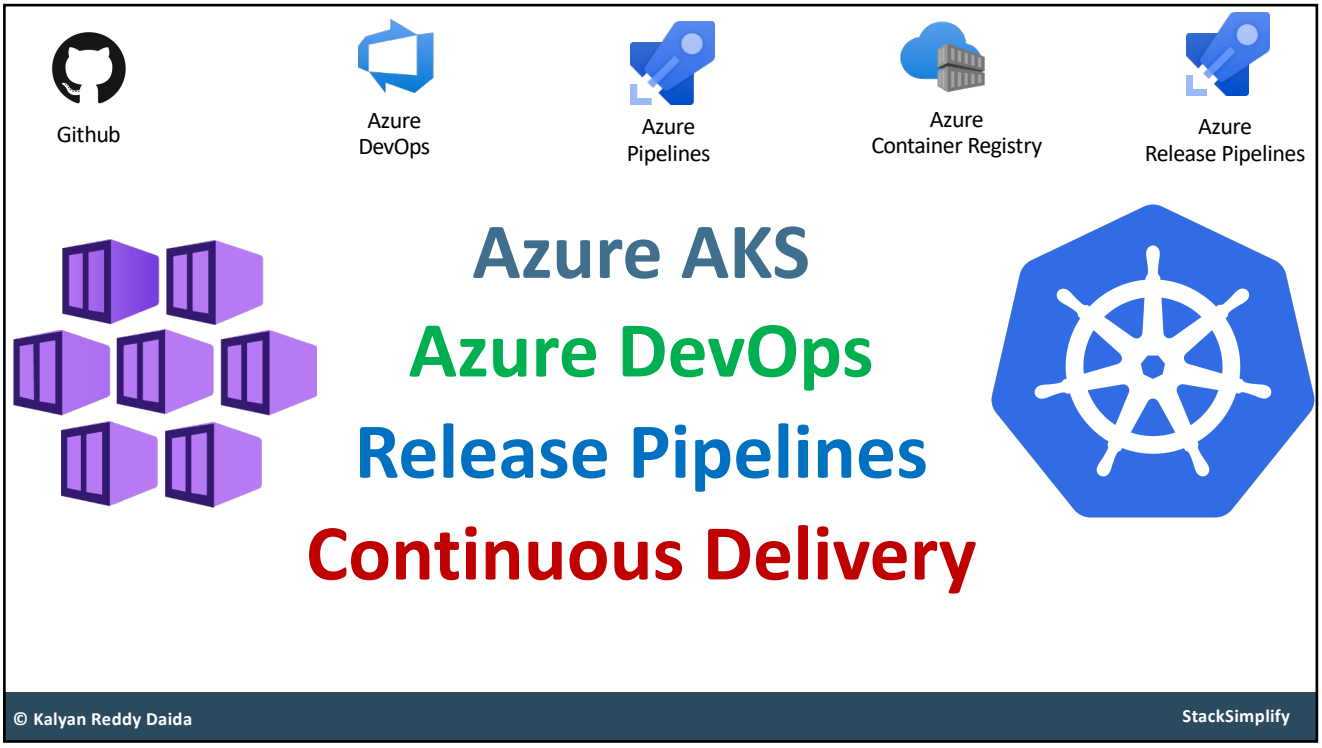
Part-1	Semi Customized	Part-2	Fully Customized using Starter Pipeline
Task-1	Use pre-defined Docker Build & Push Pipeline	Task-1	Start using Starter pipeline and use Docker Build or Push Docker Images Task
Task-2	Customize Pipeline to Use Copy Files Task	Task-2	Customize Pipeline to Use Copy Files Task
Task-3	Customize Pipeline to Use Publish Build Artifacts Task	Task-3	Customize Pipeline to Use Publish Build Artifacts Task

151

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

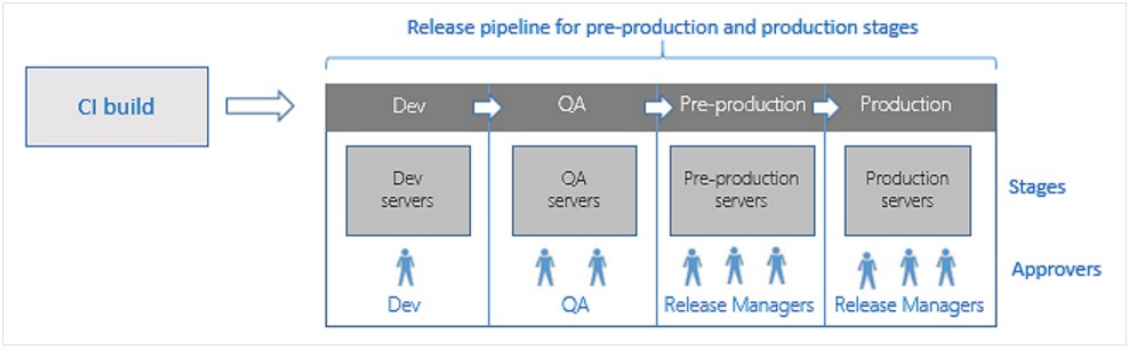
152



153

Azure DevOps – Release Pipelines

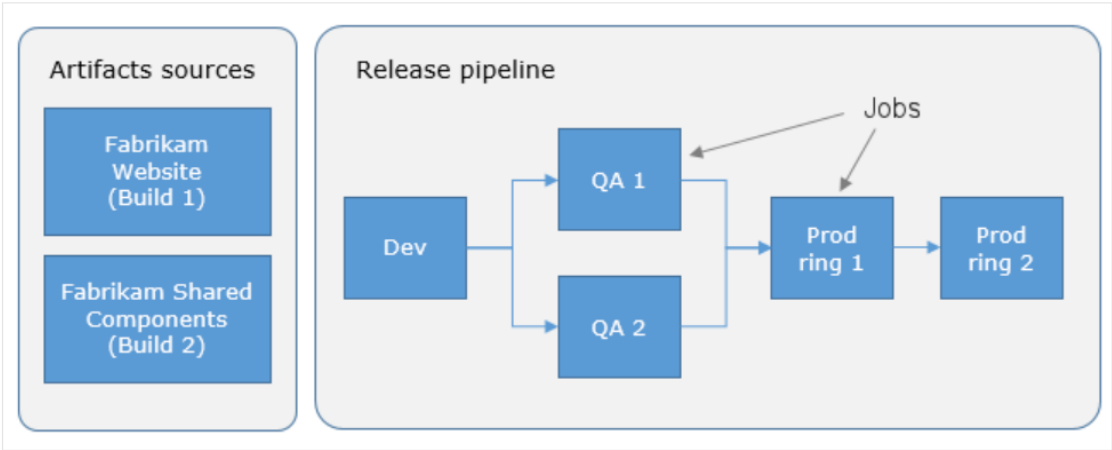
To achieve Continuous Delivery we use Release Pipelines



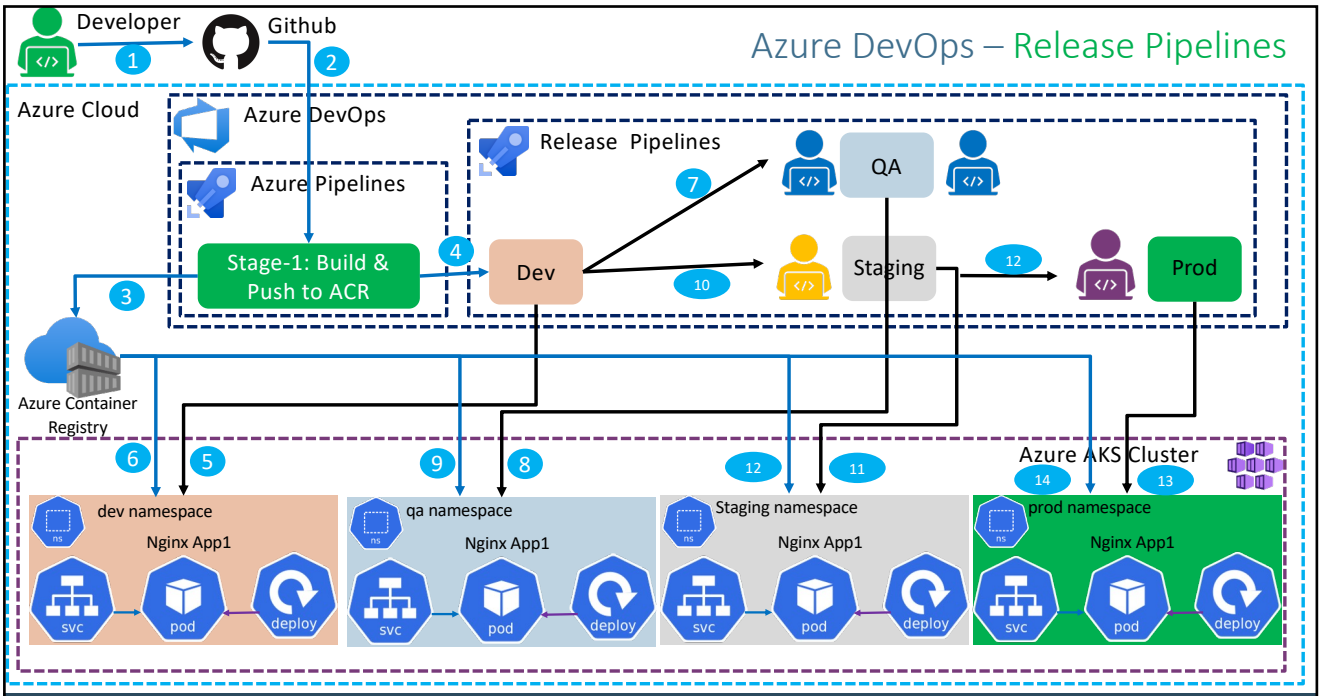
© Kalyan Reddy Daida
StackSimplify

154

Azure DevOps – Release Pipelines

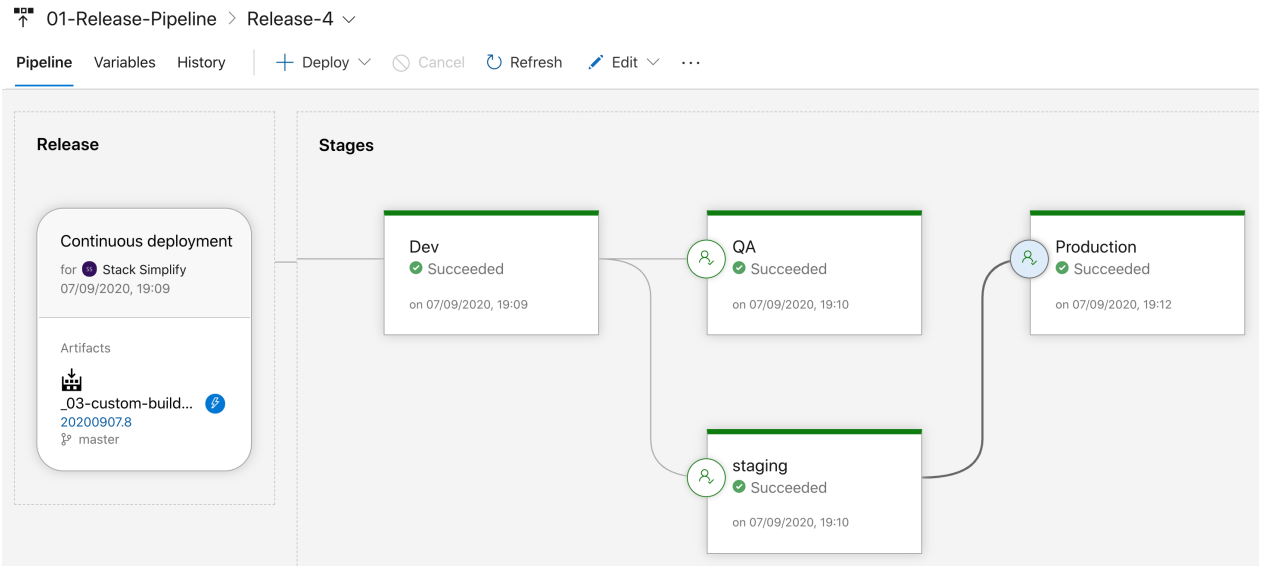


155



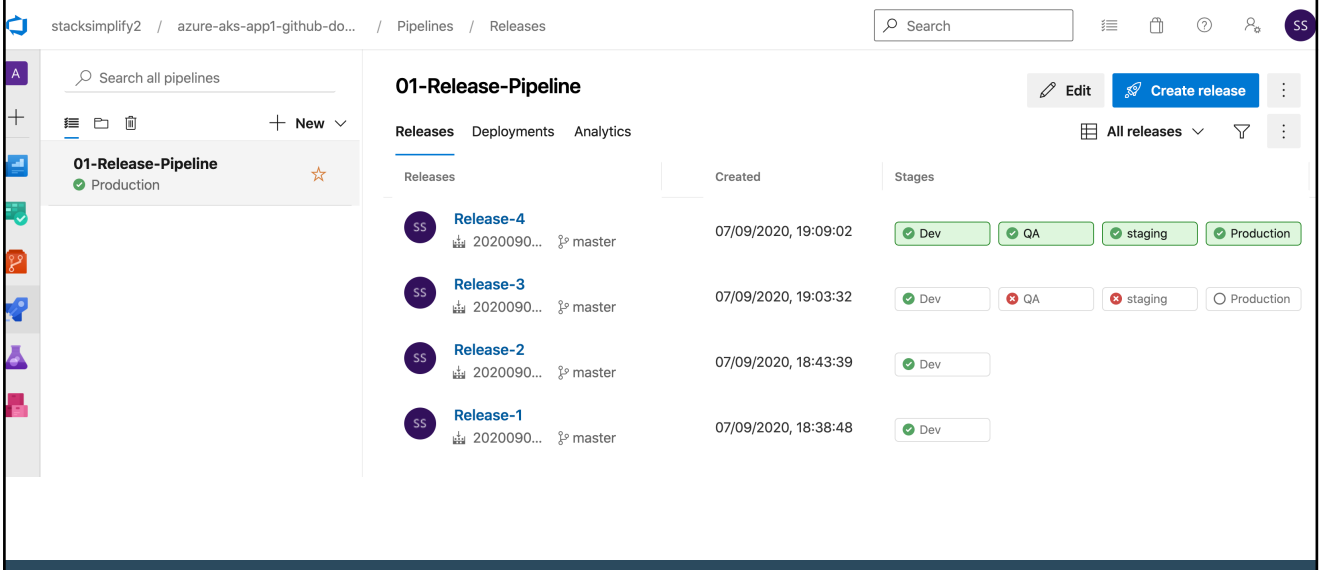
156

Azure Release Pipelines

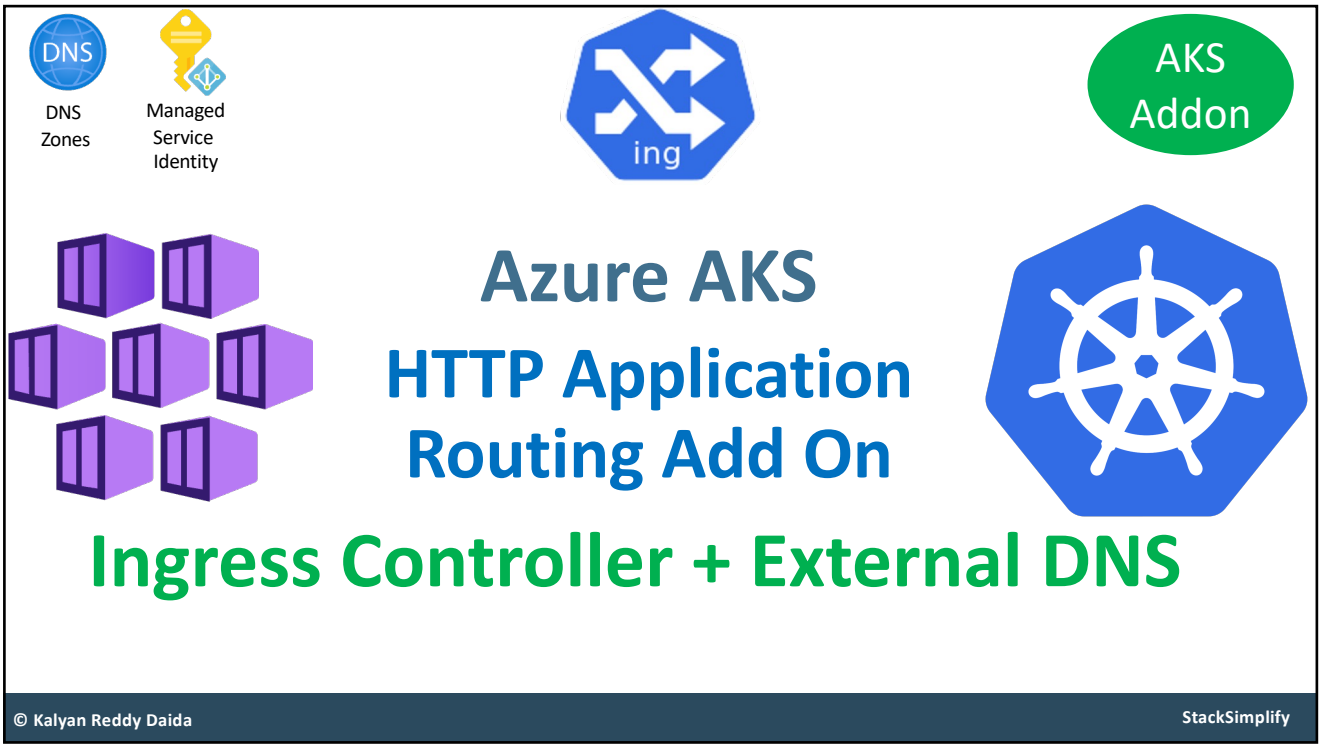


157

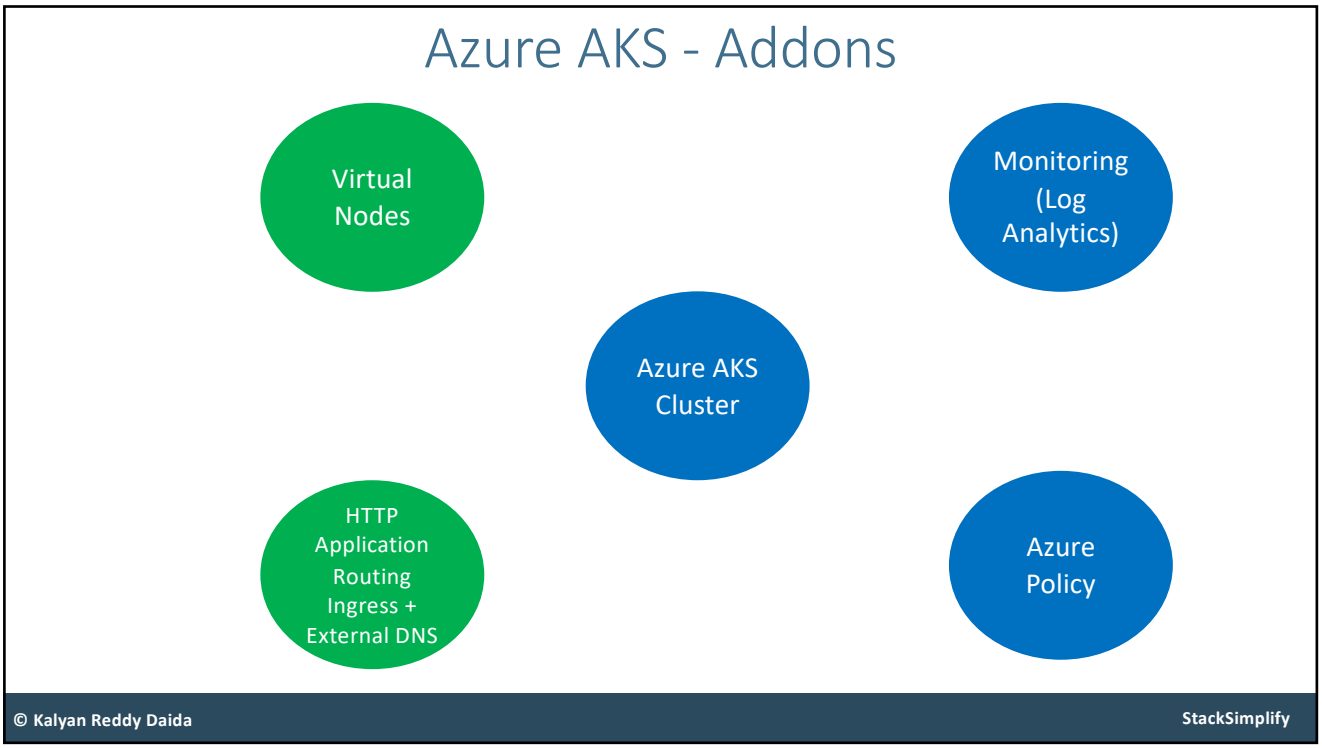
Azure Release Pipelines - Releases



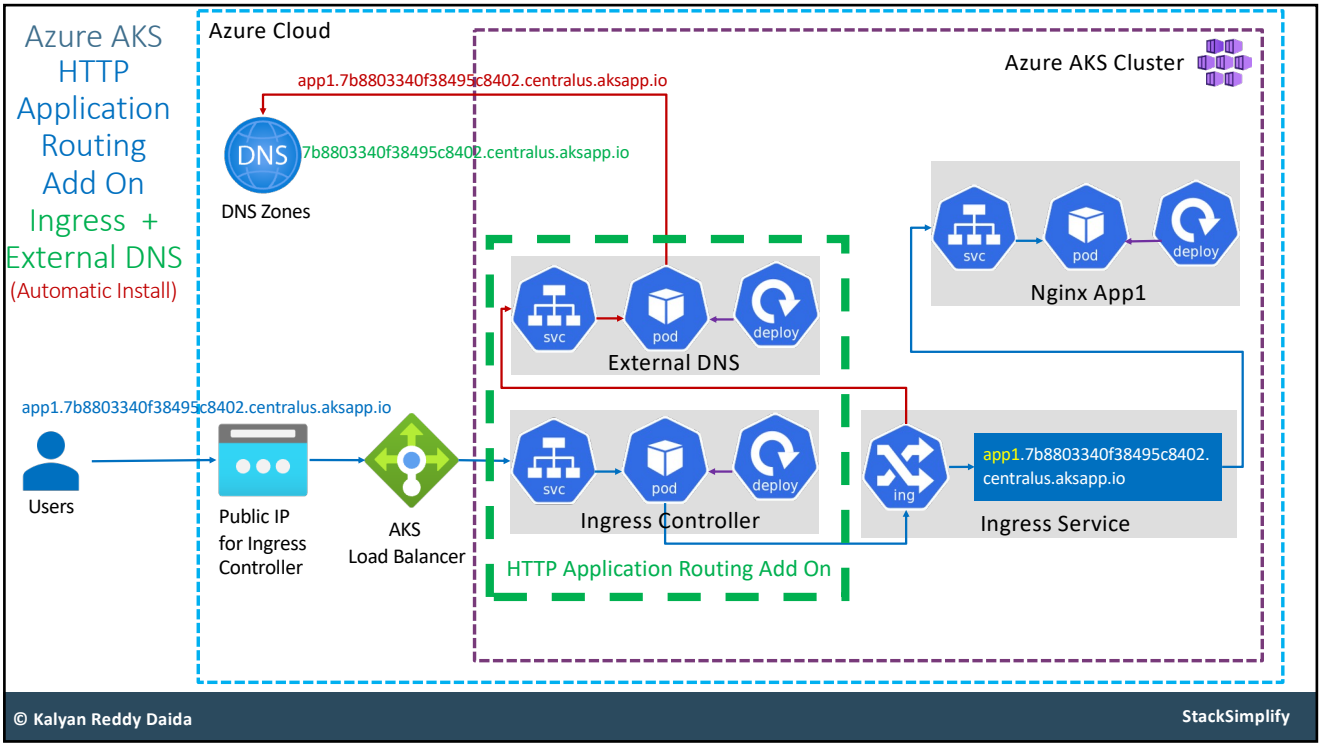
158


DNS
DNS Zones
Managed Service Identity
ing
AKS Addon
Azure AKS
HTTP Application Routing Add On
Ingress Controller + External DNS
© Kalyan Reddy Daida
StackSimplify

159


Azure AKS - Addons
Virtual Nodes
Monitoring (Log Analytics)
Azure AKS Cluster
HTTP Application Routing Ingress + External DNS
Azure Policy
© Kalyan Reddy Daida
StackSimplify

160



161

Ingress with External DNS

This annotation will ensure the ingress service we are creating will be part 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

```yaml
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
```


162

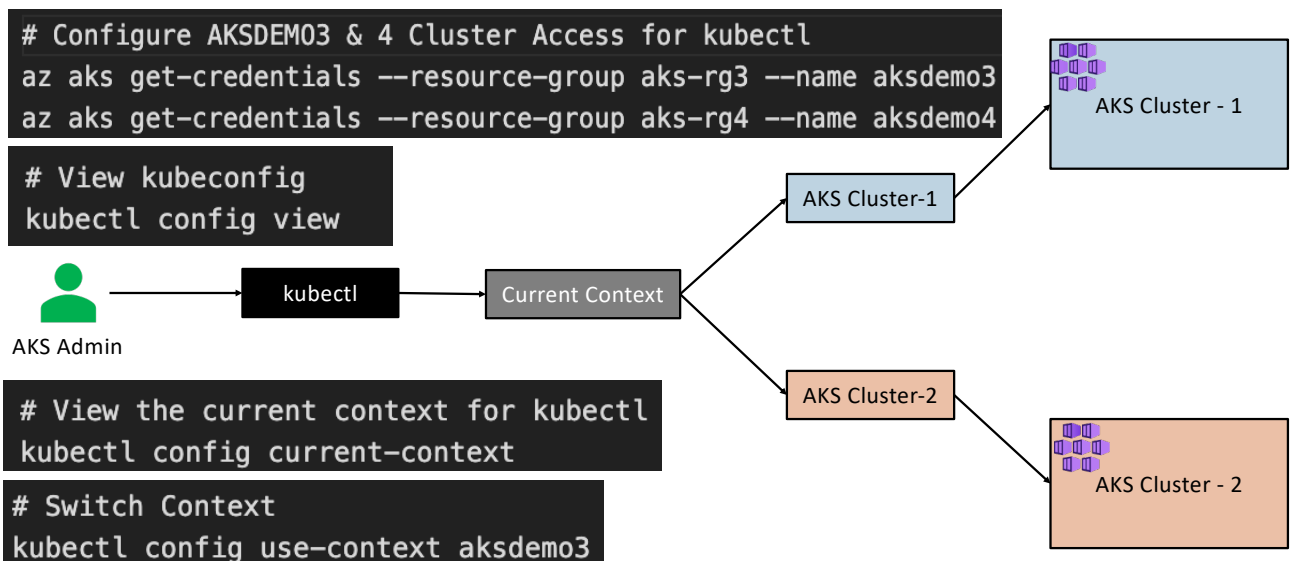
Azure AKS
Configure Access
To
Multiple AKS Clusters

163

Azure AKS Cluster Access

```
# Configure AKSDEMO3 & 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
```

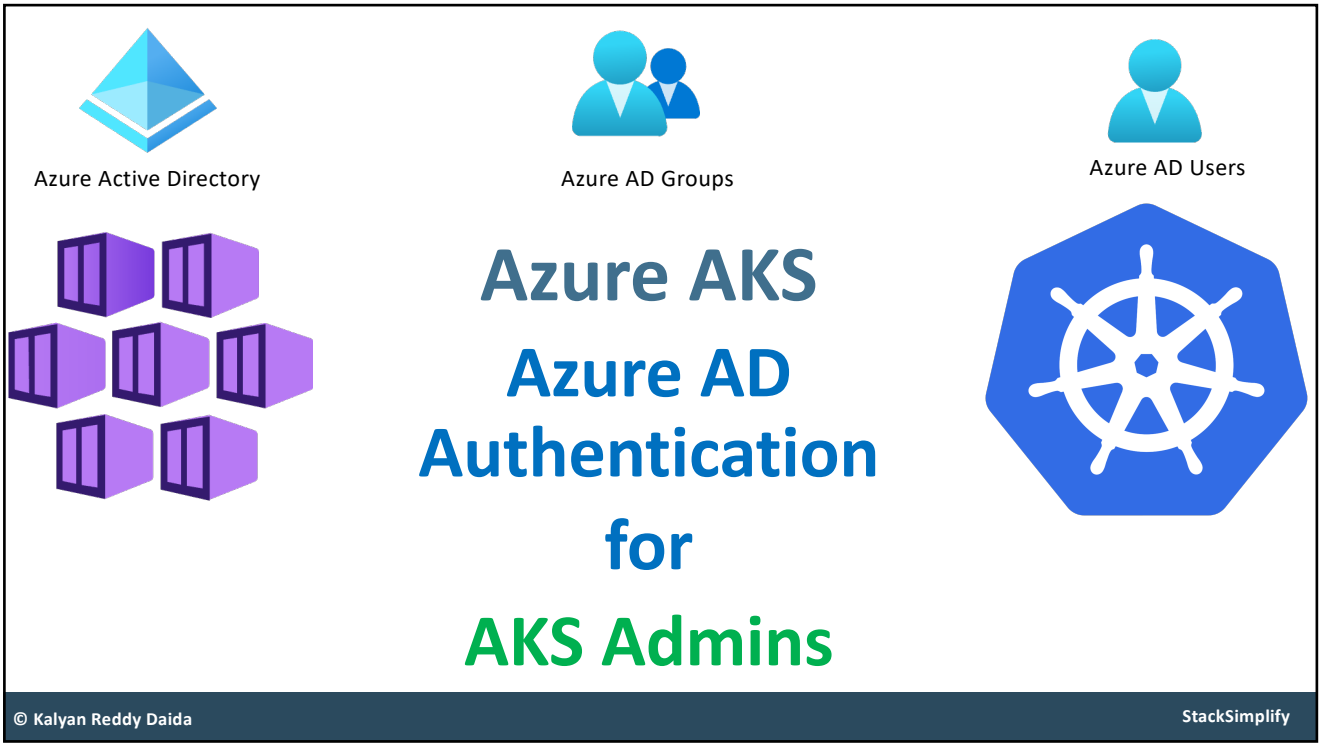
```
# View kubeconfig
kubectl config view
```



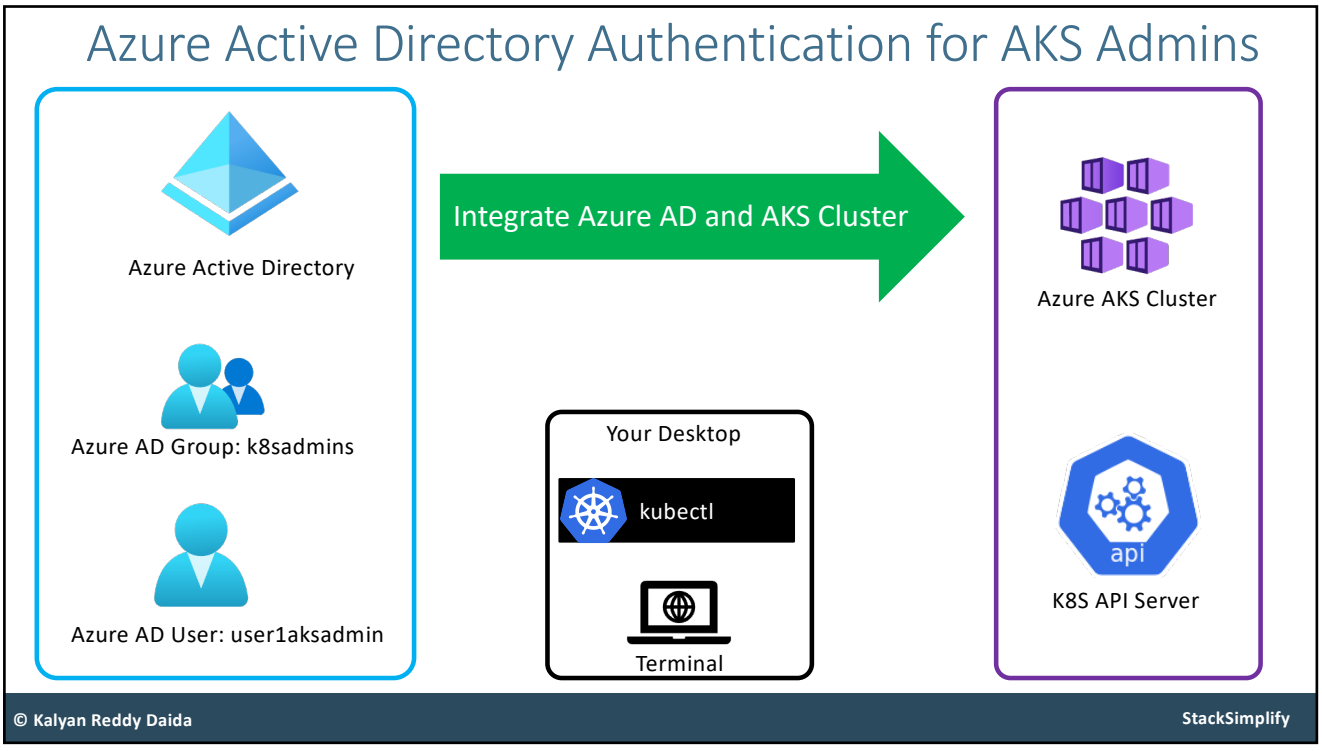

```
# View the current context for kubectl
kubectl config current-context
```

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

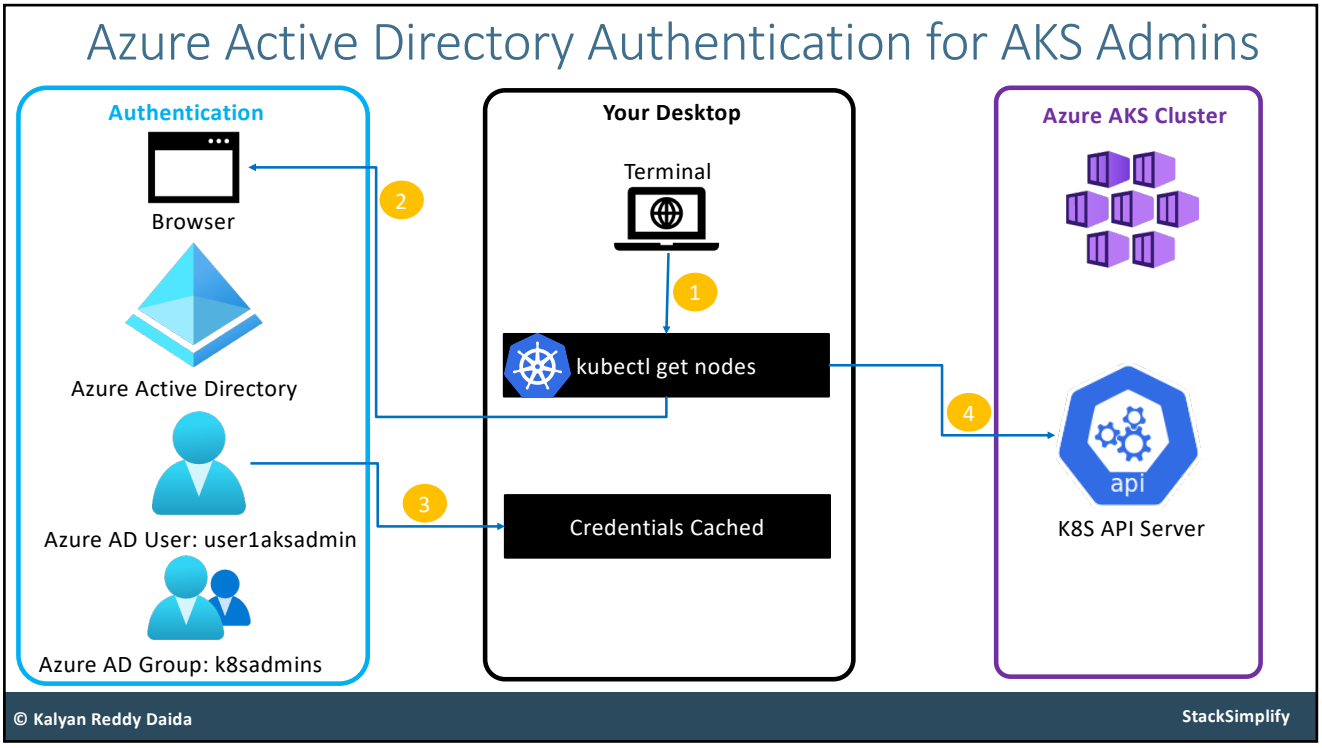

164


Azure Active Directory
Azure AD Groups
Azure AD Users
Azure AKS
Azure AD Authentication for
AKS Admins
© Kalyan Reddy Daida
StackSimplify

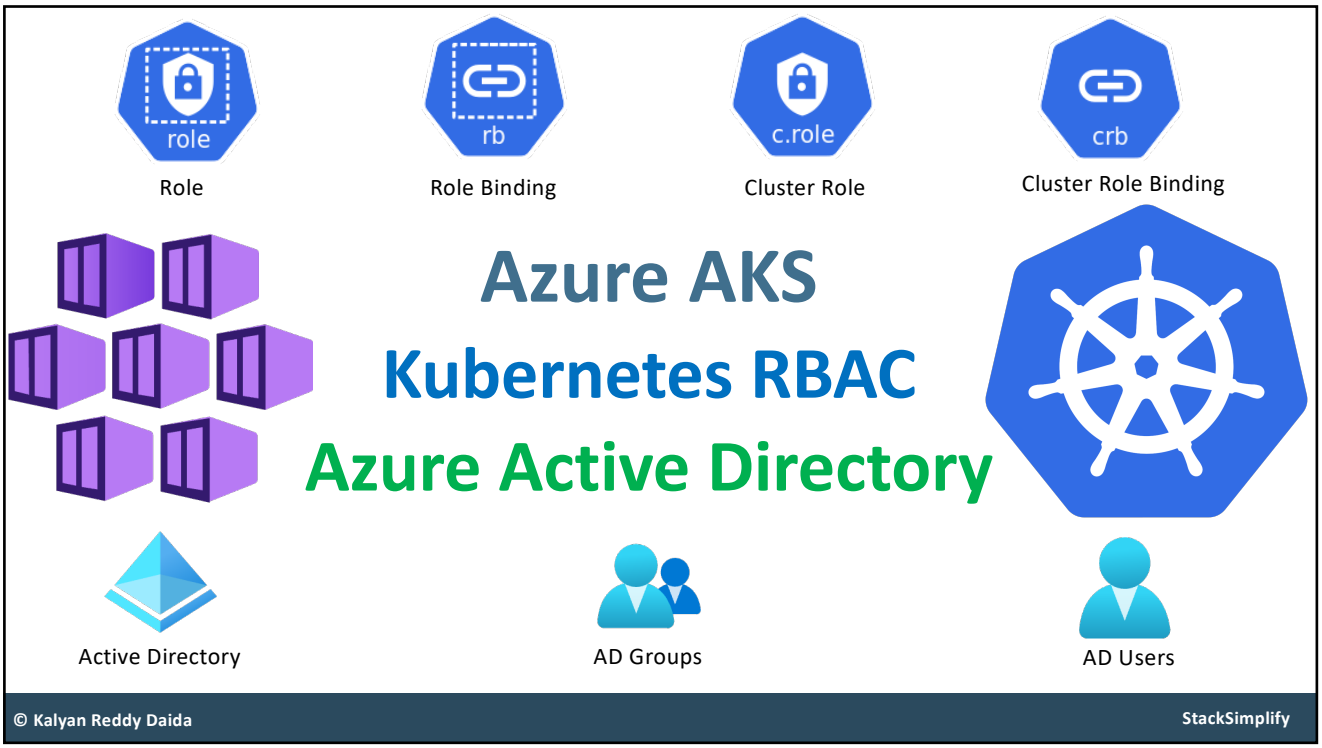
165


Azure Active Directory Authentication for AKS Admins
Azure Active Directory
Azure AD Group: k8sadmins
Azure AD User: user1aksadmin
Integrate Azure AD and AKS Cluster
Your Desktop
kubectl
Terminal
Azure AKS Cluster
api
K8S API Server
© Kalyan Reddy Daida
StackSimplify

166


Azure Active Directory Authentication for AKS Admins
Authentication
Browser
Azure Active Directory
Azure AD User: user1aksadmin
Azure AD Group: k8sadmins
Your Desktop
Terminal
kubectl get nodes
Credentials Cached
Azure AKS Cluster
api
K8S API Server
1
2
3
4
© Kalyan Reddy Daida
StackSimplify

167


role
Role
rb
Role Binding
c.role
Cluster Role
crb
Cluster Role Binding
Azure AKS
Kubernetes RBAC
Azure Active Directory
Active Directory
AD Groups
AD Users
© Kalyan Reddy Daida
StackSimplify

168

Kubernetes RBAC

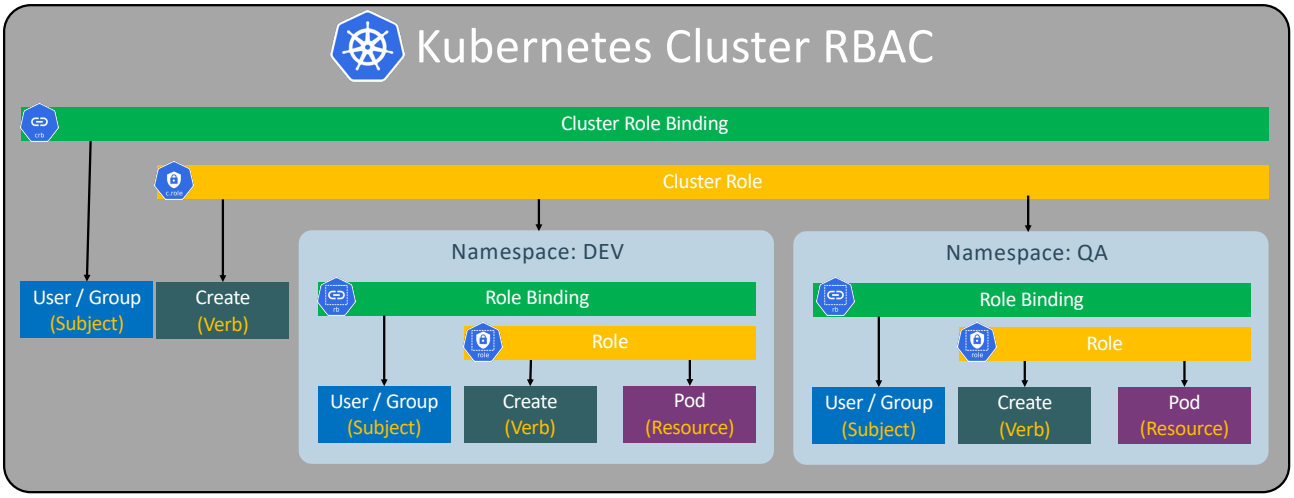
Kubernetes RBAC - Fundamentals

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

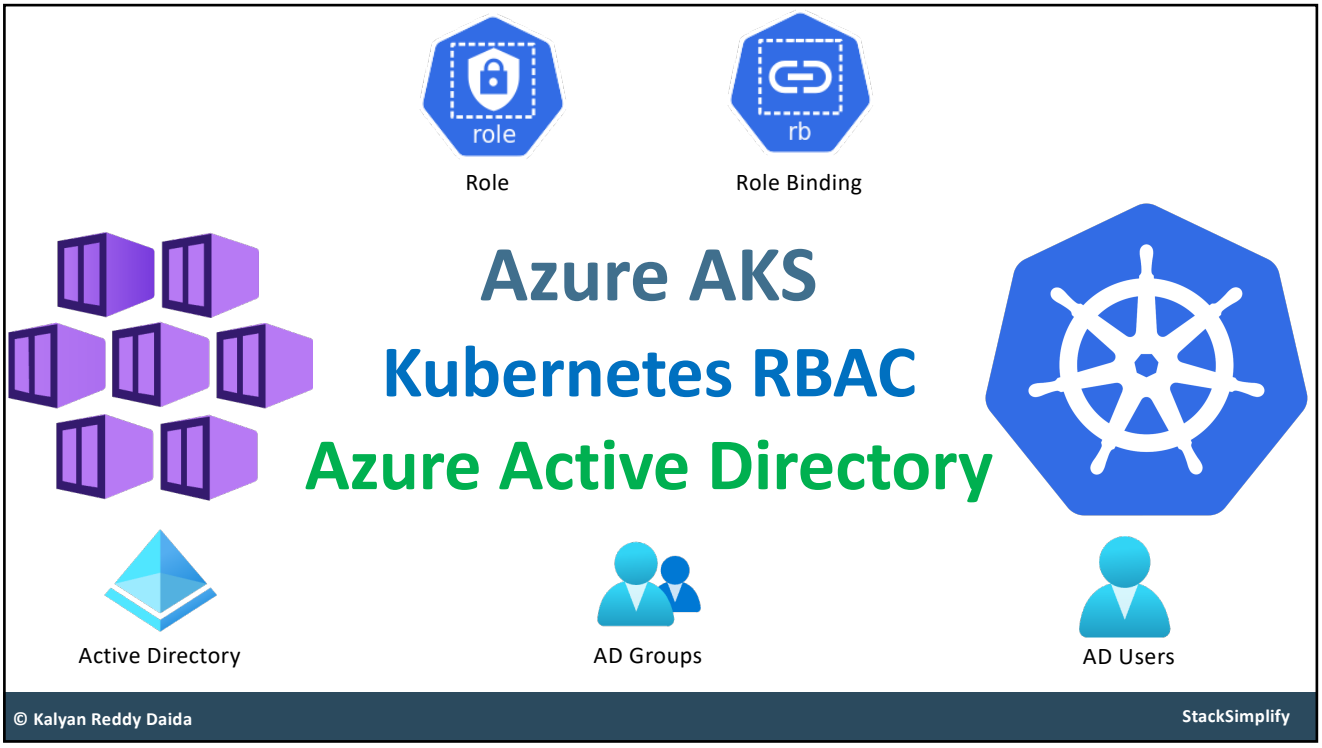
Reference: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19

169

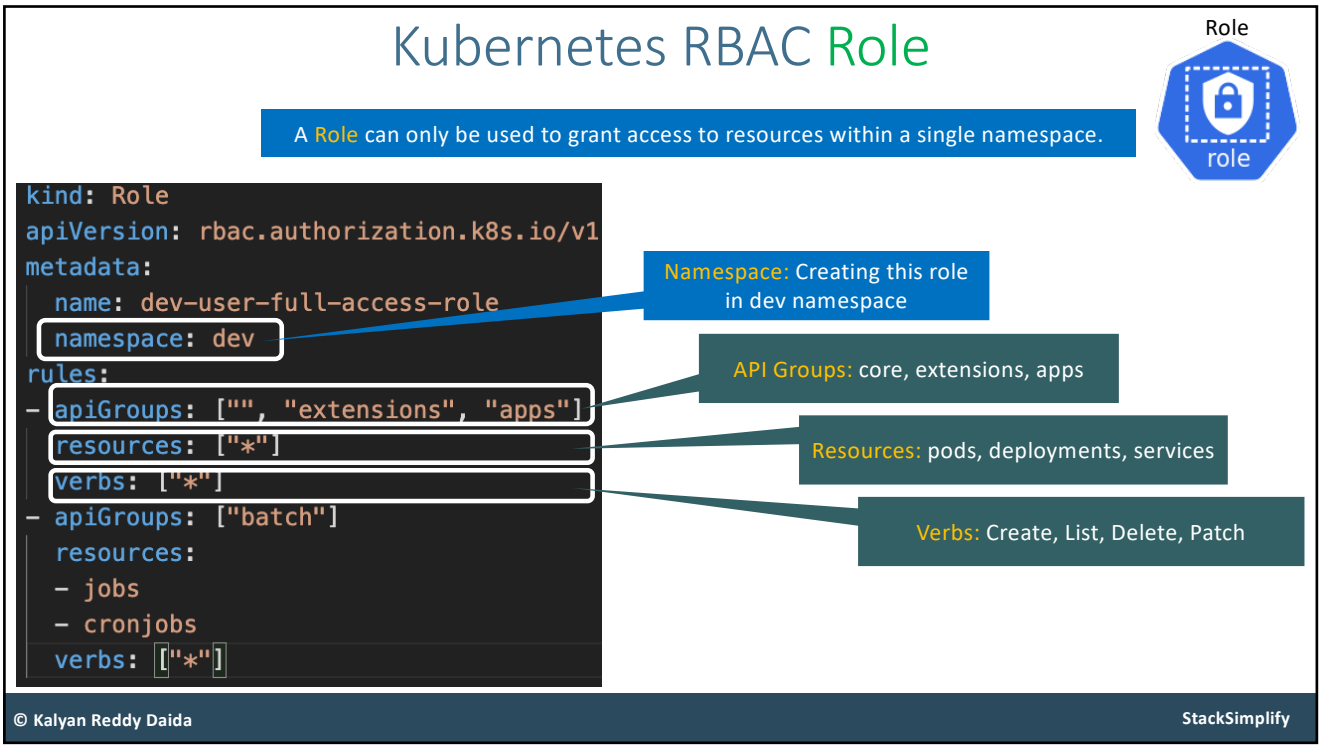
Kubernetes RBAC



170


role
Role
rb
Role Binding
Azure AKS
Kubernetes RBAC
Azure Active Directory
Active Directory
AD Groups
AD Users
© Kalyan Reddy Daida
StackSimplify

171

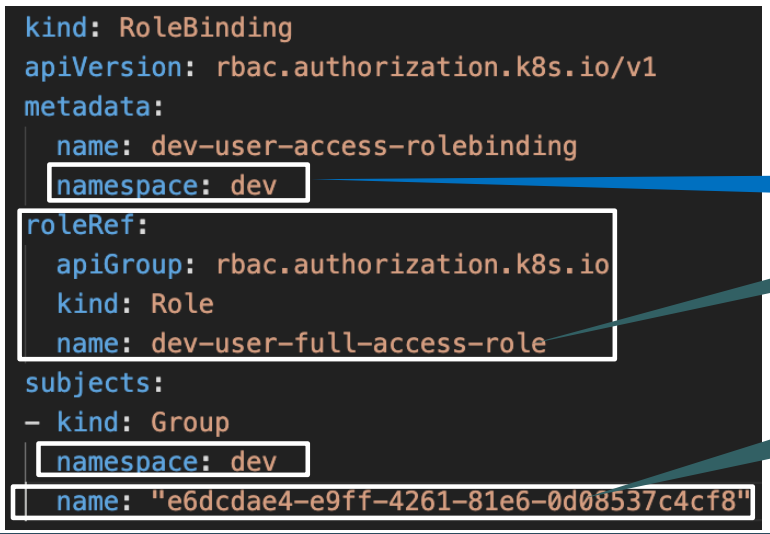

Kubernetes RBAC Role
Role
role
A Role can only be used to grant access to resources within a single namespace.
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: dev-user-full-access-role
namespace: dev
rules:
- apiGroups: ["", "extensions", "apps"]
resources: ["*"]
verbs: ["*"]
- apiGroups: ["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
© Kalyan Reddy Daida
StackSimplify

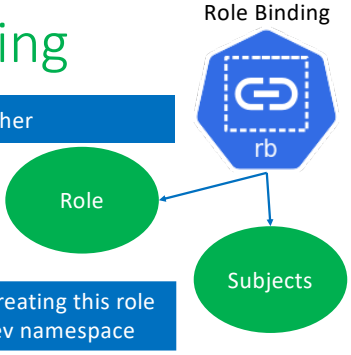
172

Kubernetes RBAC Role Binding

Role Binding

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





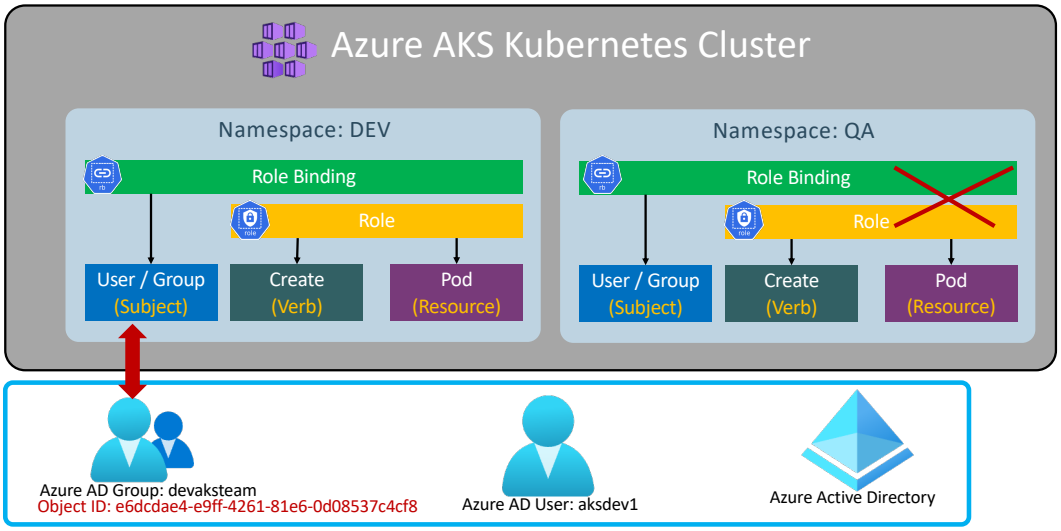
Namespace: Creating this role binding in dev namespace

Role: Mapped role here

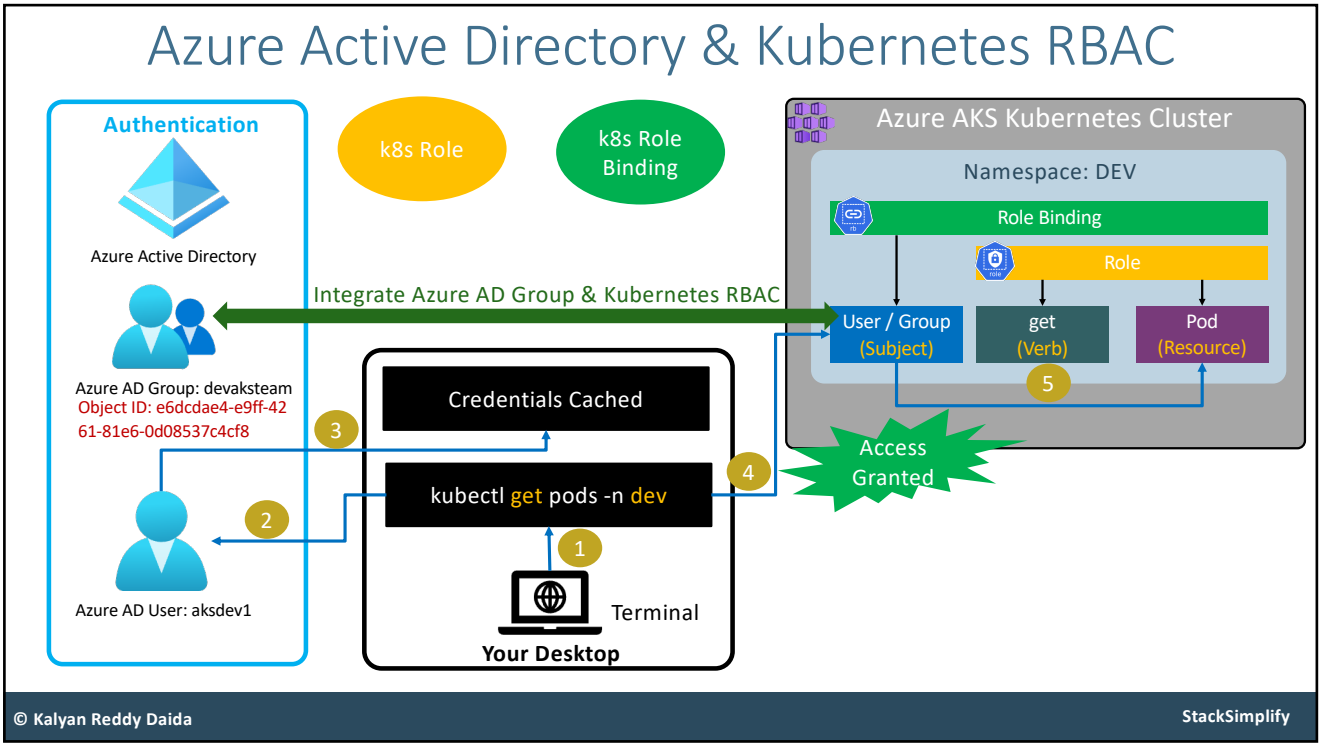
Subjects: Group: Azure AD Group with Object ID xxx has full access to namespace dev in AKS Cluster

173

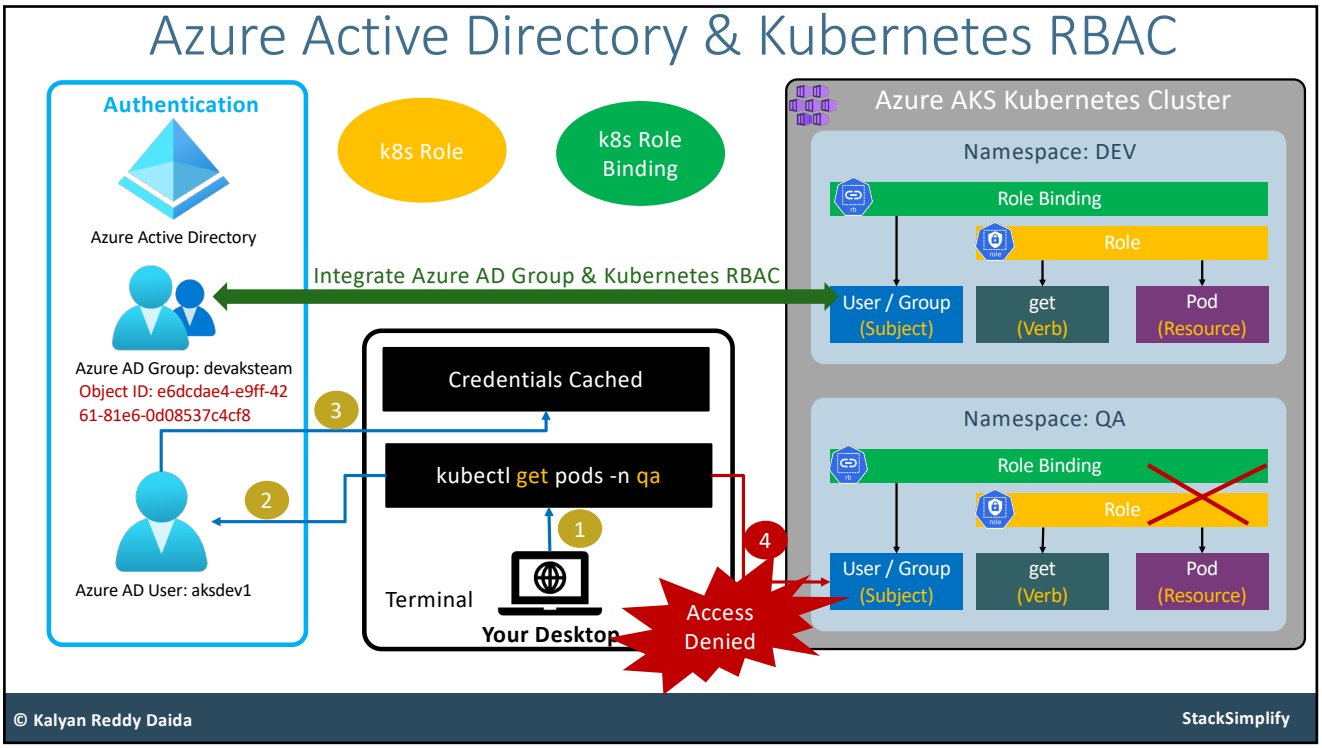
Azure Active Directory & Kubernetes RBAC



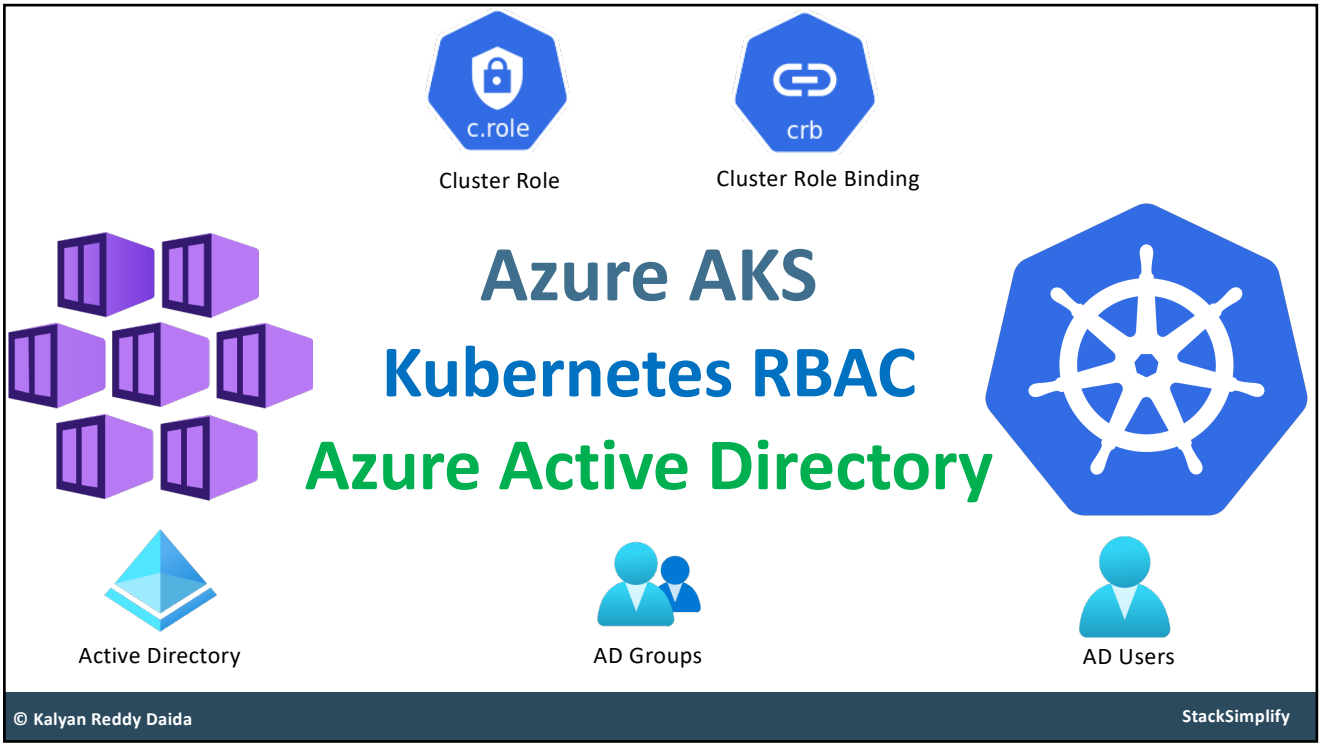
174


Azure Active Directory & Kubernetes RBAC
Authentication
Azure Active Directory
k8s Role
k8s Role Binding
Azure AKS Kubernetes Cluster
Namespace: DEV
Role Binding
Role
Integrate Azure AD Group & Kubernetes RBAC
User / Group (Subject)
get (Verb)
Pod (Resource)
5
Azure AD Group: devaksteam
Object ID: e6dcdae4-e9ff-42
61-81e6-0d08537c4cf8
3
Credentials Cached
4
Access Granted
kubectl get pods -n dev
2
1
Azure AD User: aksdev1
Terminal
Your Desktop
© Kalyan Reddy Daida
StackSimplify

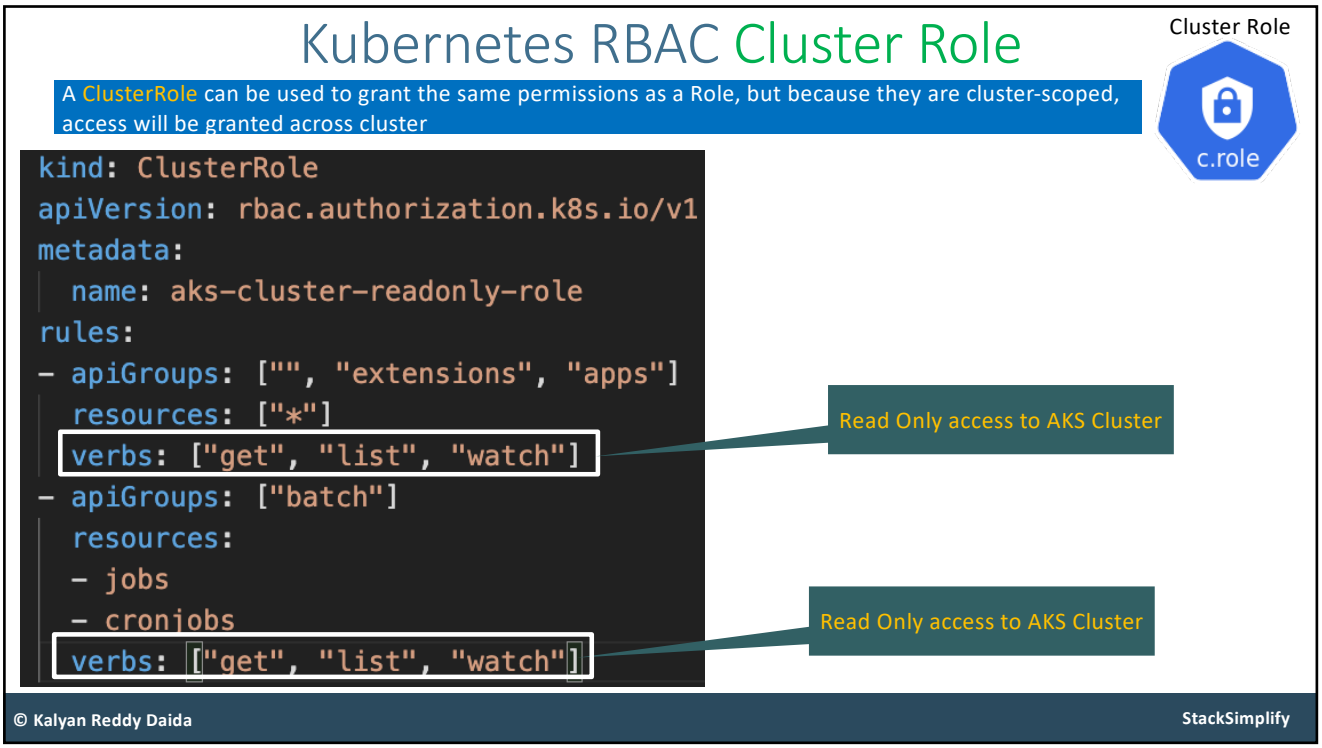
175


Azure Active Directory & Kubernetes RBAC
Authentication
Azure Active Directory
k8s Role
k8s Role Binding
Azure AKS Kubernetes Cluster
Namespace: DEV
Role Binding
Role
Integrate Azure AD Group & Kubernetes RBAC
User / Group (Subject)
get (Verb)
Pod (Resource)
Azure AD Group: devaksteam
Object ID: e6dcdae4-e9ff-42
61-81e6-0d08537c4cf8
3
Credentials Cached
Namespace: QA
Role Binding
Role
kubectl get pods -n qa
2
1
4
Azure AD User: aksdev1
Terminal
Your Desktop
Access Denied
User / Group (Subject)
get (Verb)
Pod (Resource)
© Kalyan Reddy Daida
StackSimplify

176


c.role
Cluster Role
crb
Cluster Role Binding
Azure AKS
Kubernetes RBAC
Azure Active Directory
Active Directory
AD Groups
AD Users
© Kalyan Reddy Daida
StackSimplify

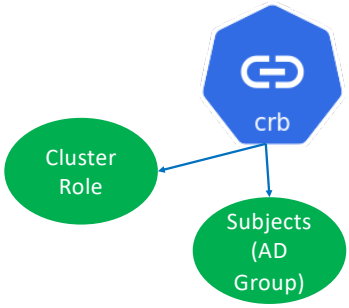
177


Kubernetes RBAC Cluster Role
Cluster Role
c.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
© Kalyan Reddy Daida
StackSimplify

178

Kubernetes RBAC Cluster Role Binding

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



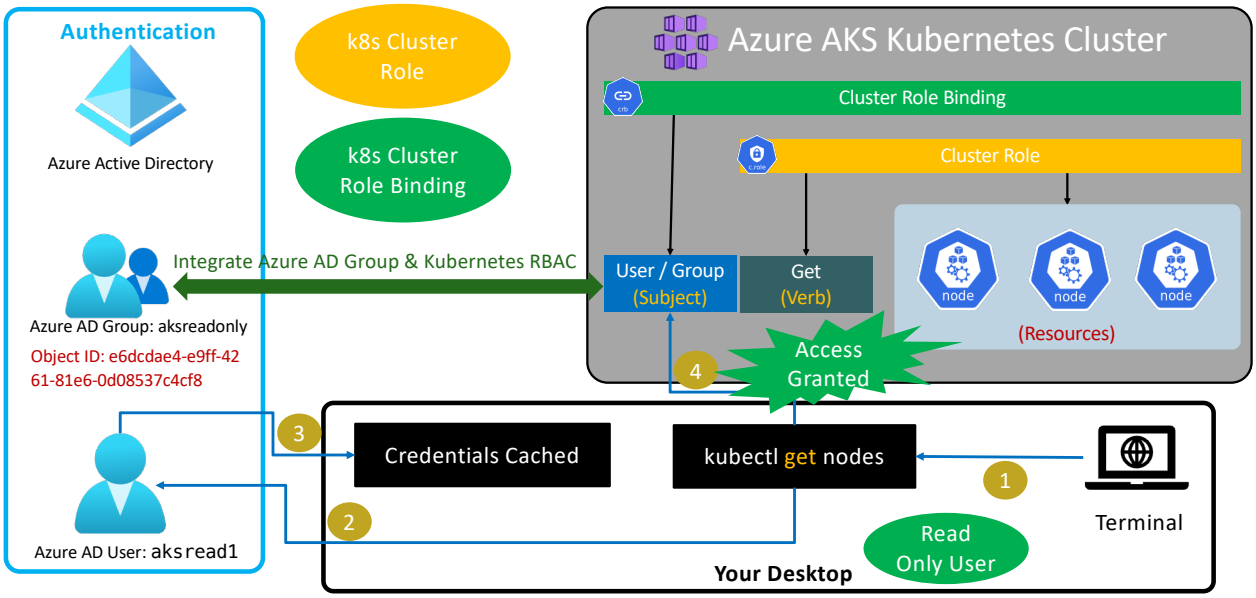

```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: aks-cluster-readonly-rolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: aks-cluster-readonly-role
subjects:
- kind: Group
  name: "e808215d-d159-49ba-8bb6-9661ba478842"
```

Role: Mapped role here

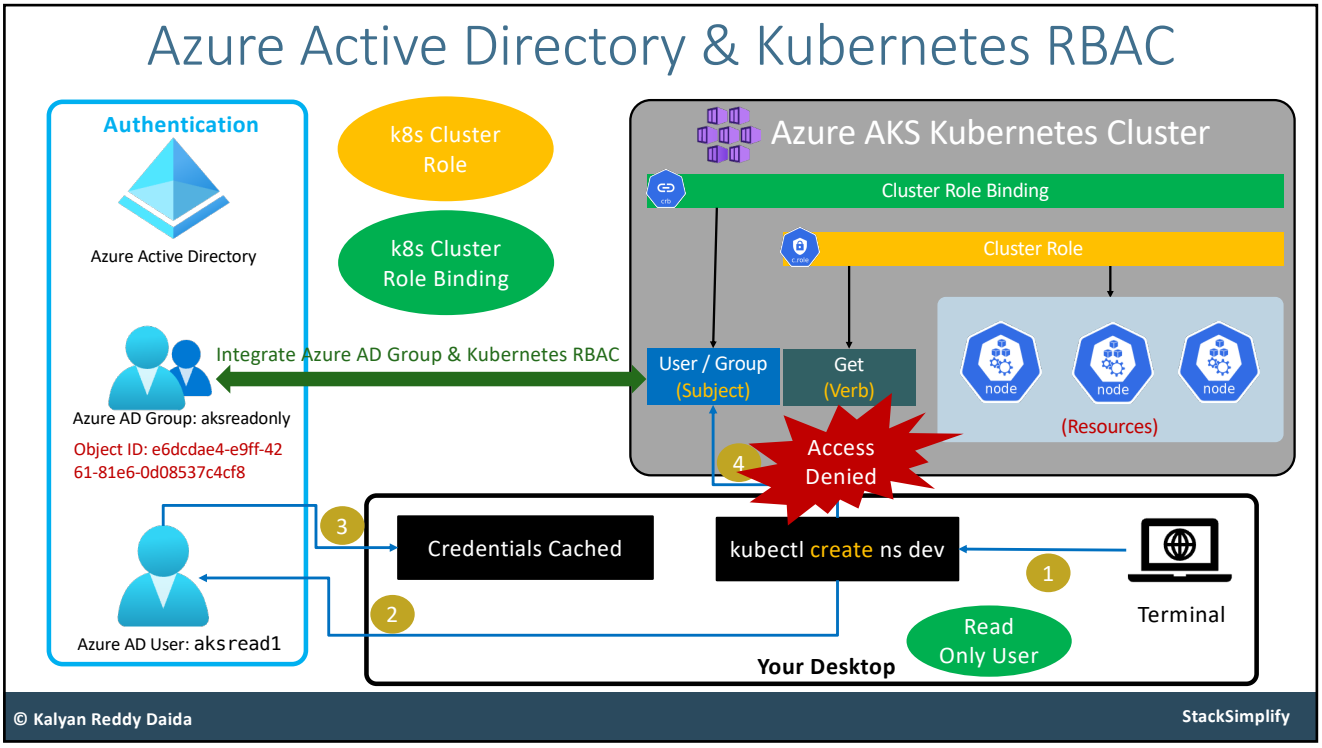
Subjects: Group: Azure AD Group with Object ID xxx has read only access to AKS Cluster

179

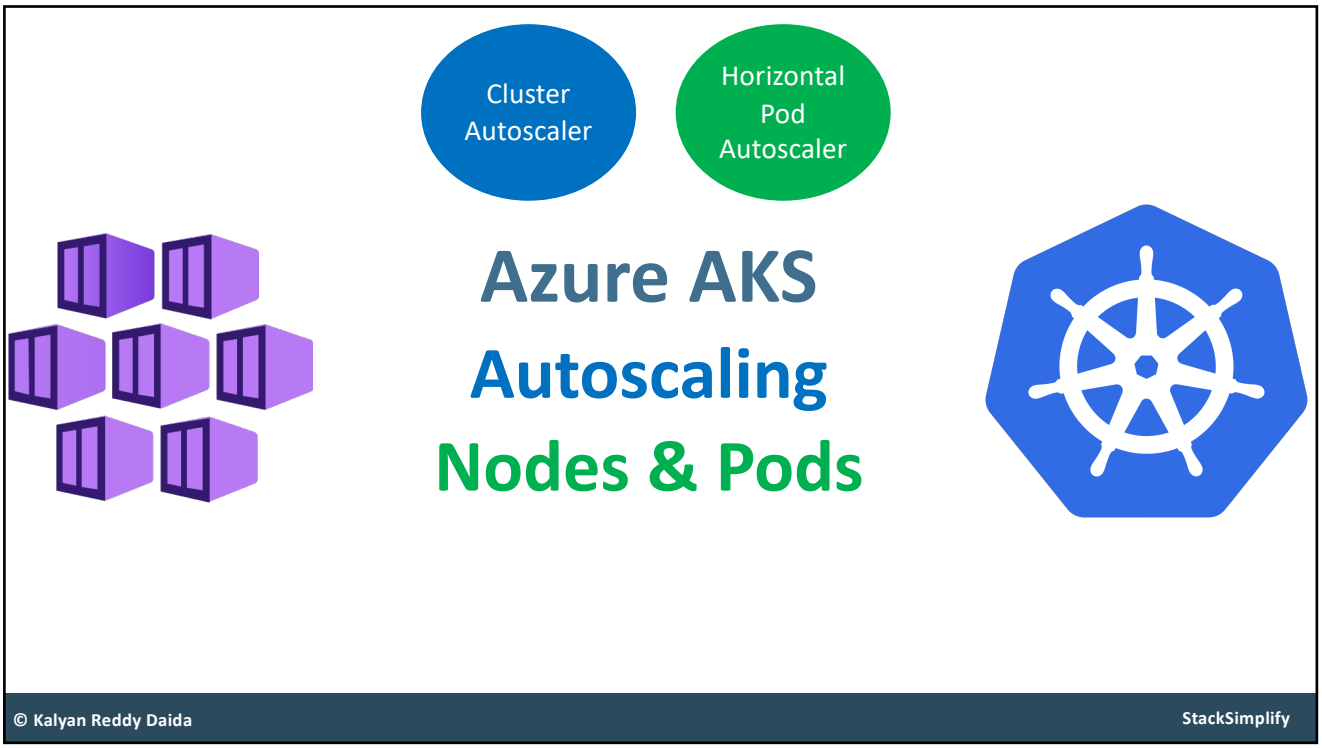
Azure Active Directory & Kubernetes RBAC



180

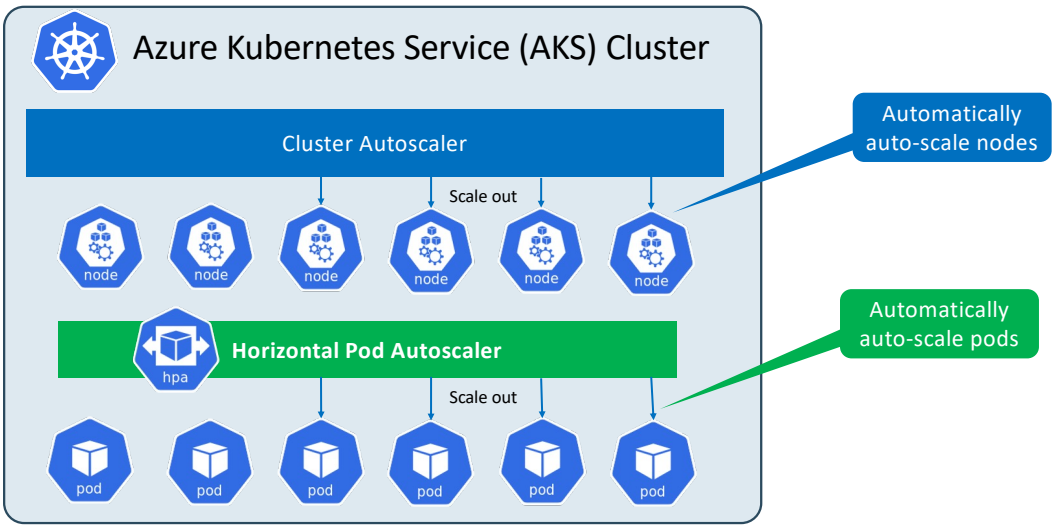

Azure Active Directory & Kubernetes RBAC
Authentication
Azure Active Directory
k8s Cluster Role
k8s Cluster Role Binding
Azure AKS Kubernetes Cluster
Cluster Role Binding
Cluster Role
Integrate Azure AD Group & Kubernetes RBAC
User / Group (Subject)
Get (Verb)
node
node
node
(Resources)
Azure AD Group: aksreadonly
Object ID: e6dcdae4-e9ff-42 61-81e6-0d08537c4cf8
4
Access Denied
3
Credentials Cached
kubectl create ns dev
1
Terminal
2
Azure AD User: aksread1
Read Only User
Your Desktop
© Kalyan Reddy Daida
StackSimplify

181

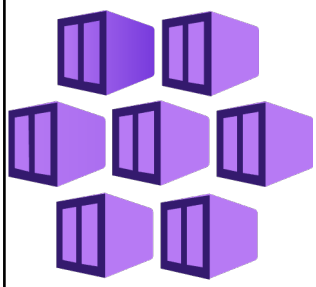

Cluster Autoscaler
Horizontal Pod Autoscaler
Azure AKS
Autoscaling
Nodes & Pods
© Kalyan Reddy Daida
StackSimplify

182

Azure AKS – Autoscaling Nodes & Pods



183



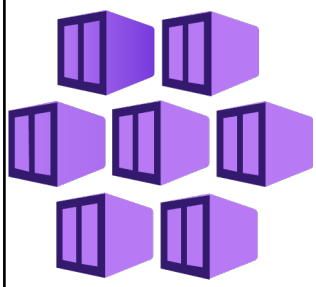
Azure AKS
Autoscaling
Cluster Autoscaler

184

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.

185



Azure AKS

Autoscaling

Horizontal Pod Autoscaling

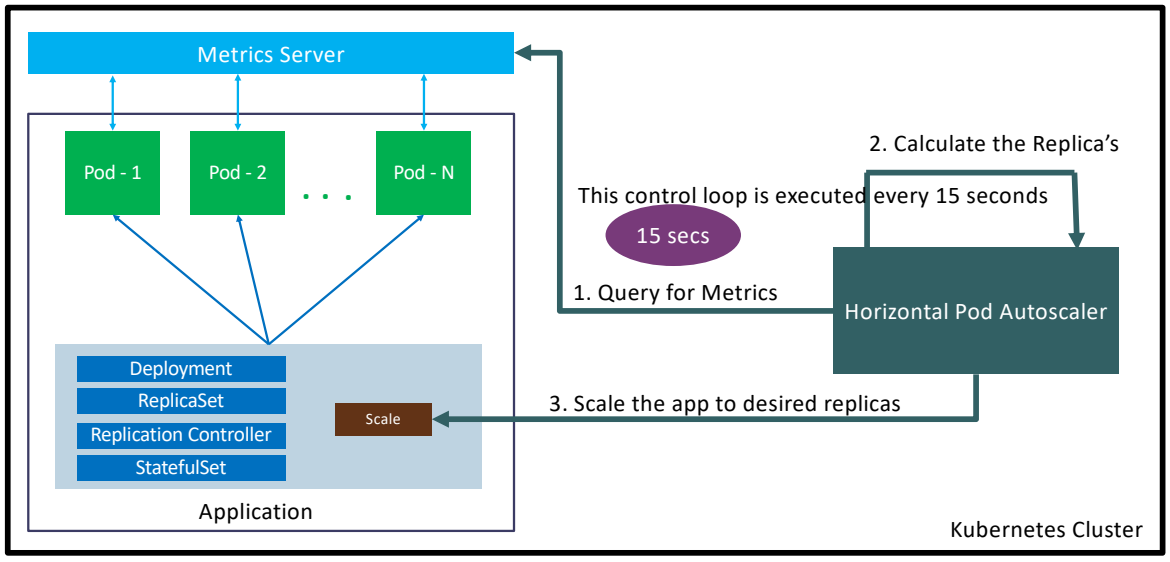
186

Horizontal Pod Autoscaler – HPA - Intoduction

- 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.

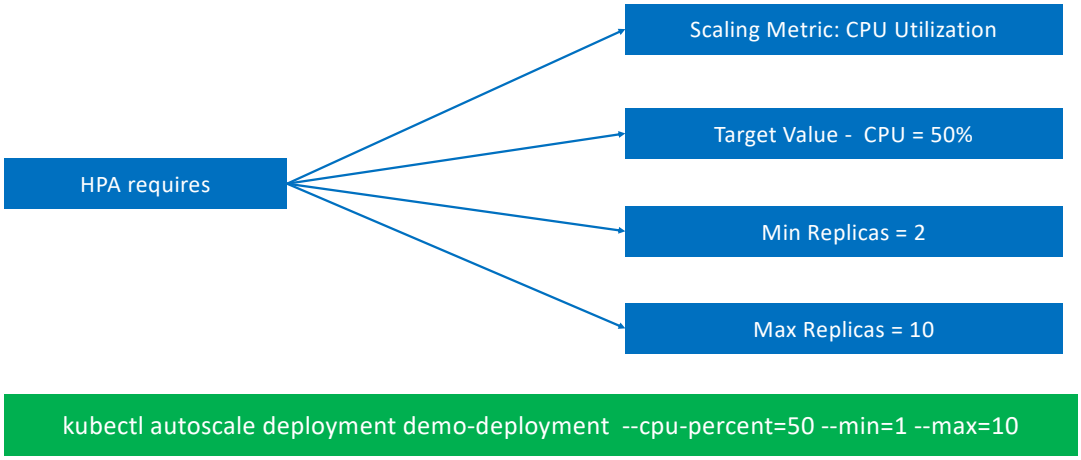
187

How HPA works?



188

How is HPA configured?

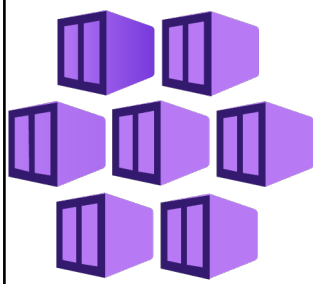


189


AKS Linux NodePools


AKS Windows NodePools

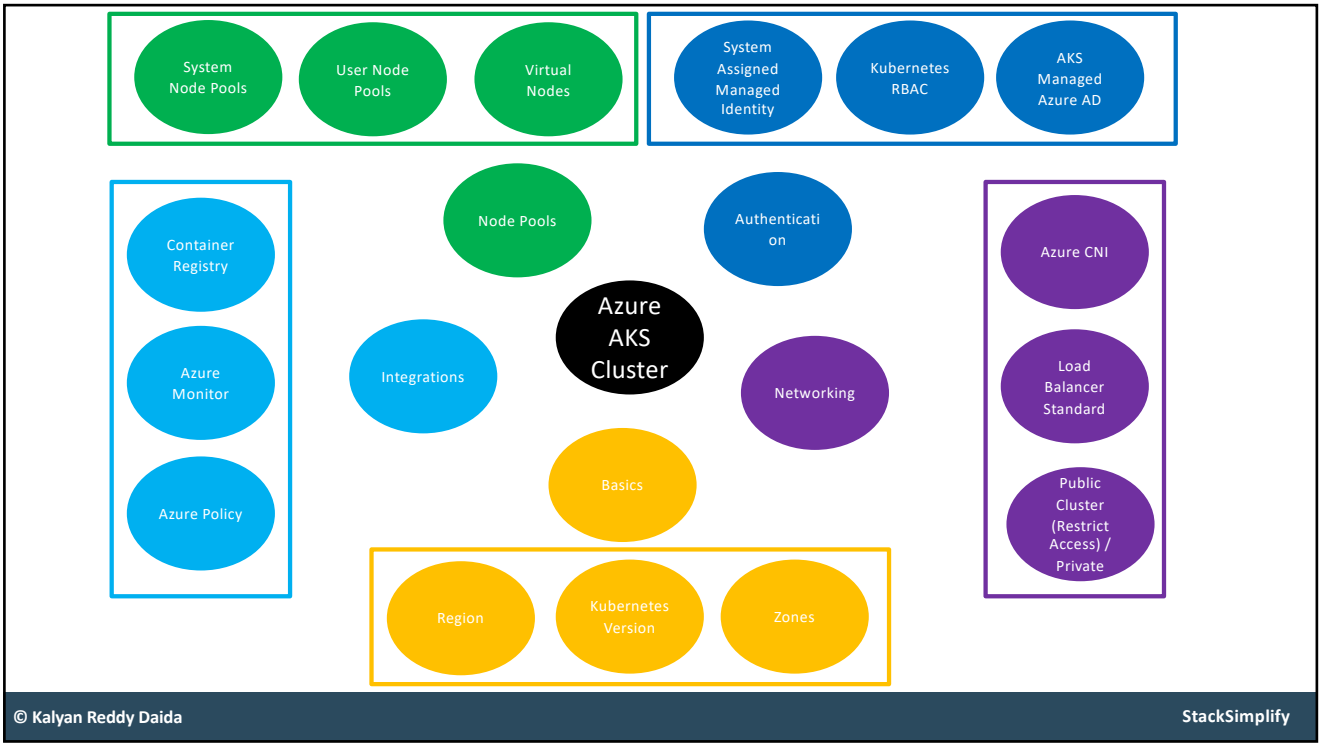

AKS Virtual Nodes



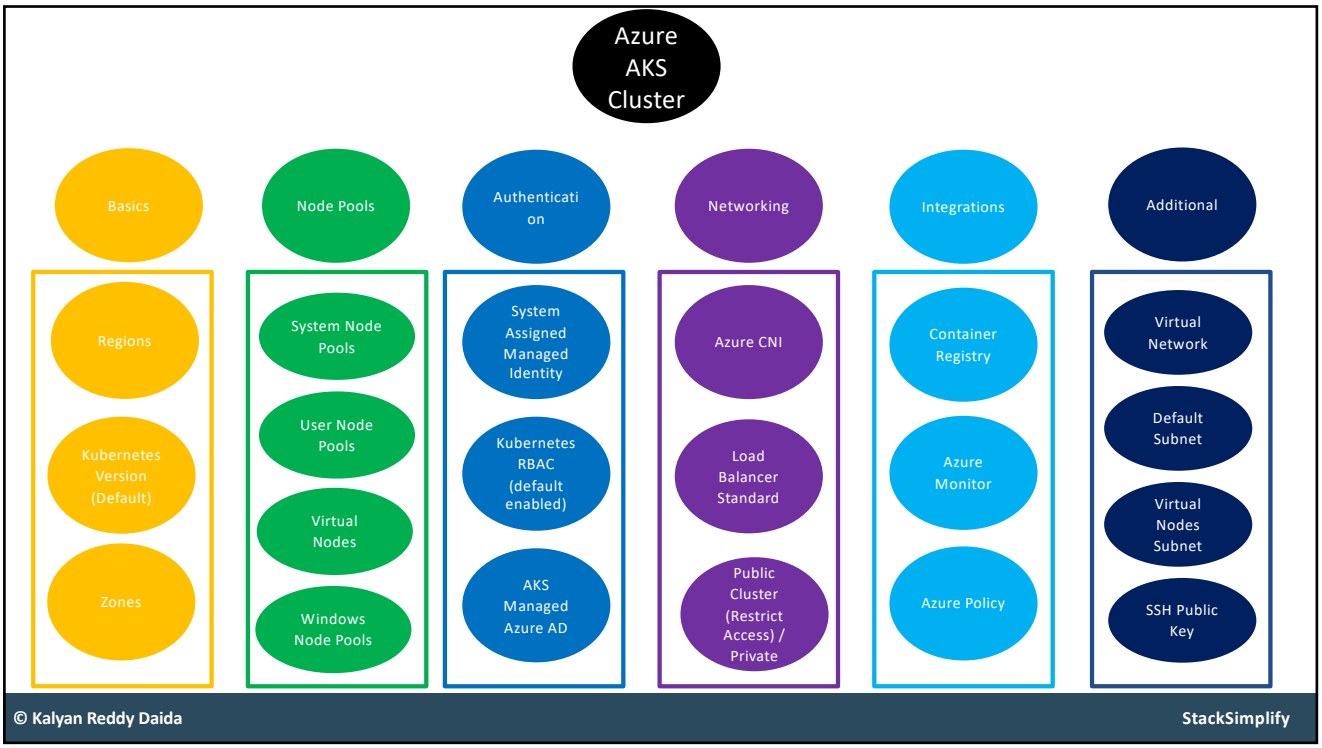
Azure AKS Design Production Grade AKS Cluster

az aks cli

190


System Node Pools
User Node Pools
Virtual Nodes
System Assigned Managed Identity
Kubernetes RBAC
AKS Managed Azure AD
Container Registry
Azure Monitor
Azure Policy
Node Pools
Authentication
Azure AKS Cluster
Integrations
Networking
Basics
Region
Kubernetes Version
Zones
Azure CNI
Load Balancer Standard
Public Cluster (Restrict Access) / Private
© Kalyan Reddy Daida
StackSimplify

191


Azure AKS Cluster
Basics
Node Pools
Authentication
Networking
Integrations
Additional
Regions
Kubernetes Version (Default)
Zones
System Node Pools
User Node Pools
Virtual Nodes
Windows Node Pools
System Assigned Managed Identity
Kubernetes RBAC (default enabled)
AKS Managed Azure AD
Azure CNI
Load Balancer Standard
Public Cluster (Restrict Access) / Private
Container Registry
Azure Monitor
Azure Policy
Virtual Network
Default Subnet
Virtual Nodes Subnet
SSH Public Key
© Kalyan Reddy Daida
StackSimplify

192

Section-1
Create AKS Cluster

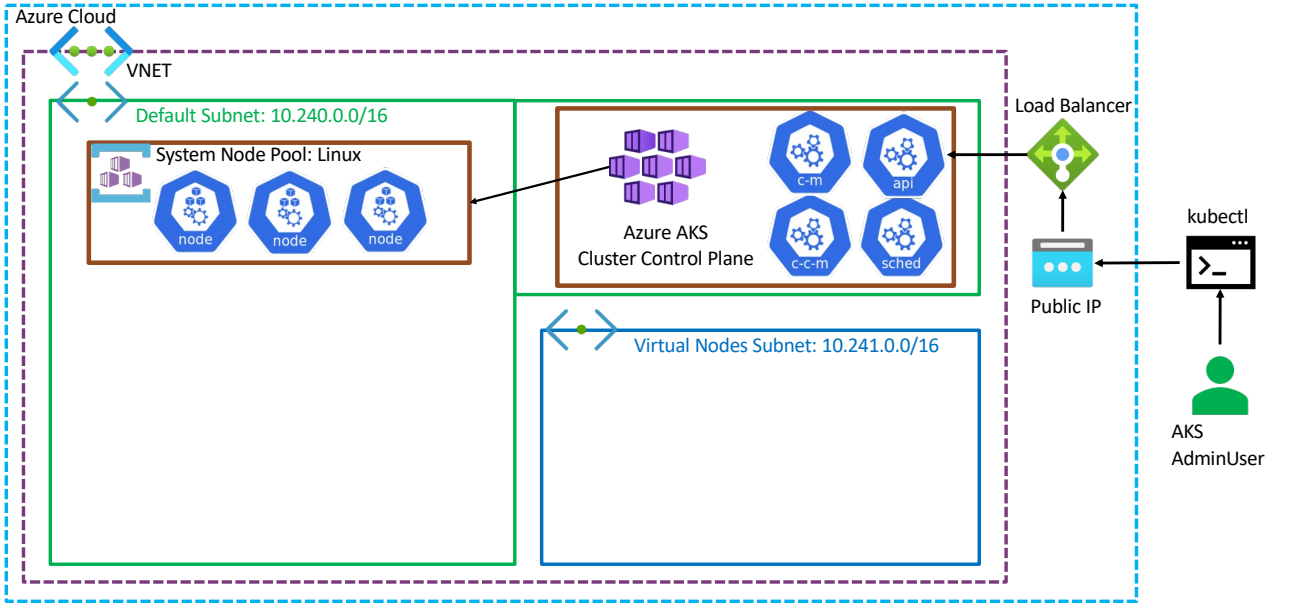
Section-2
Create Windows, Linux User Node Pools and Virtual Nodes

Section-3
Deploy Apps to all 4 node pools using Kubernetes Node Selector concept

 StackSimplify

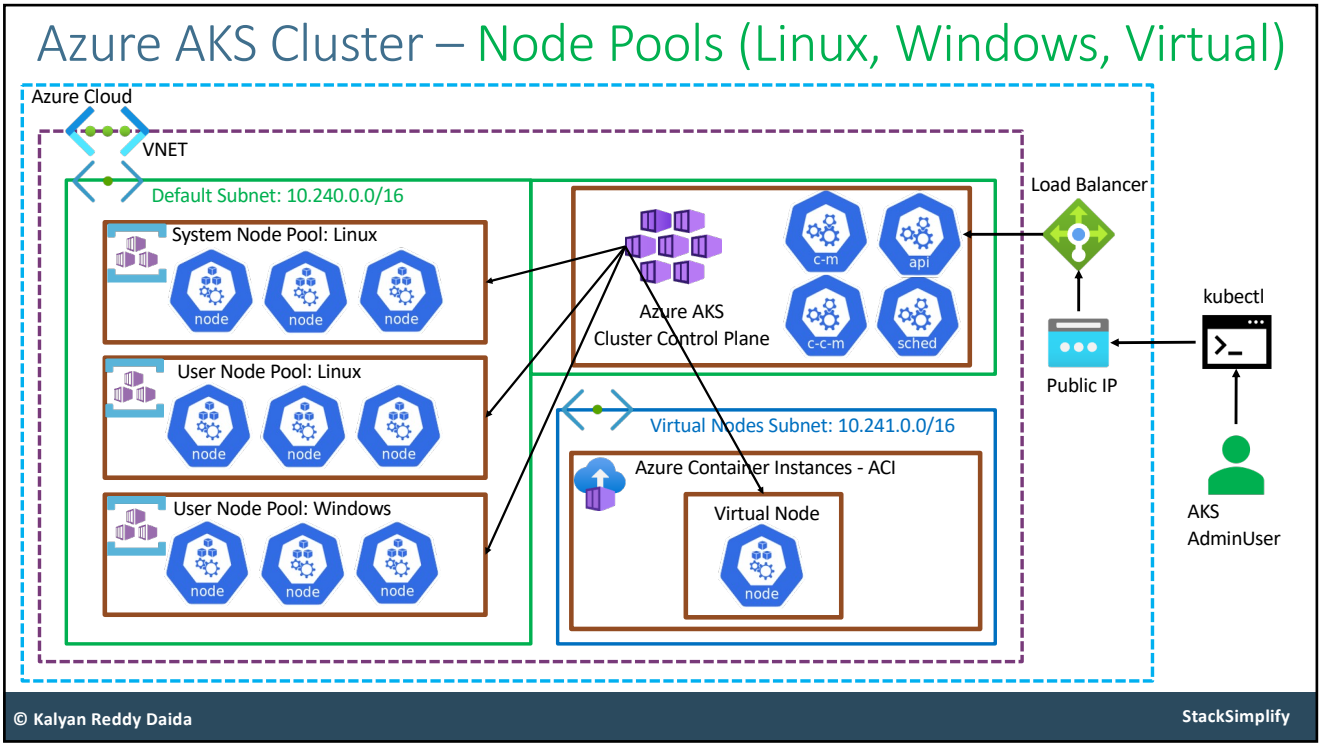
193

Azure AKS Cluster – Create AKS Cluster using CLI

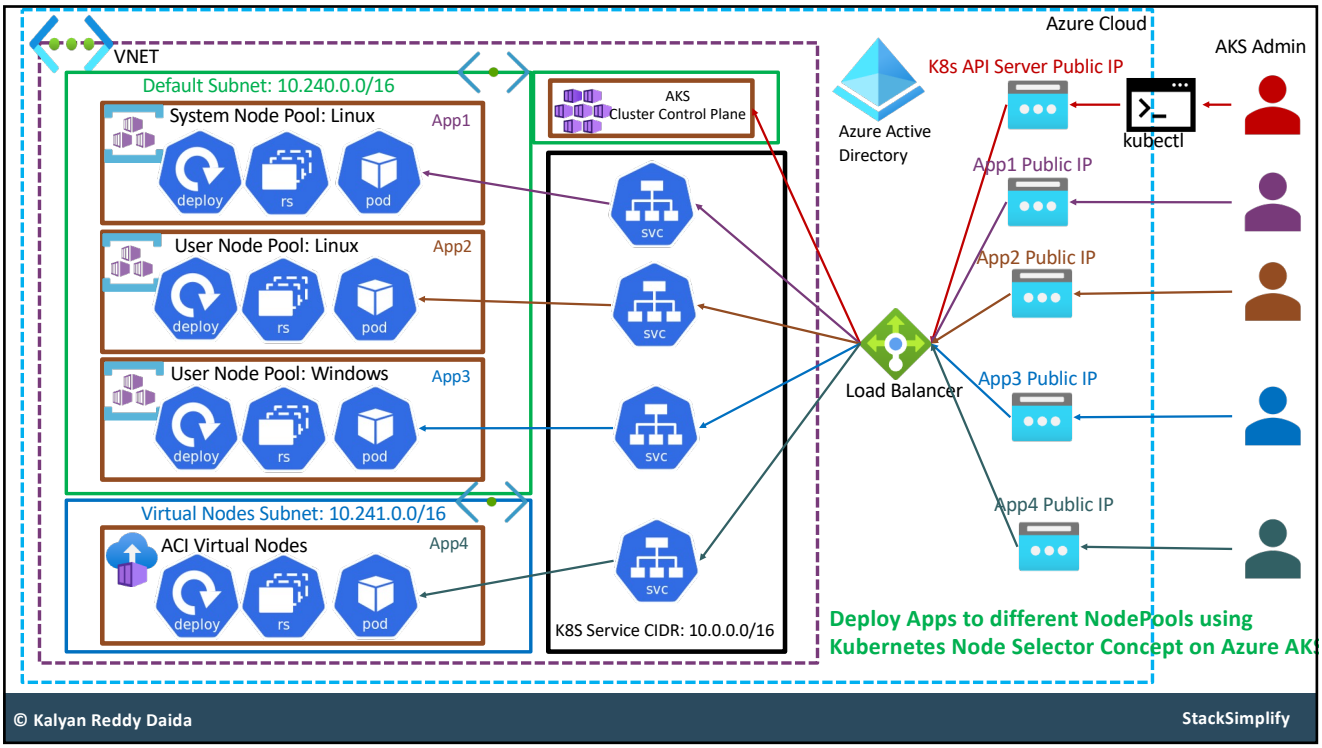


 StackSimplify

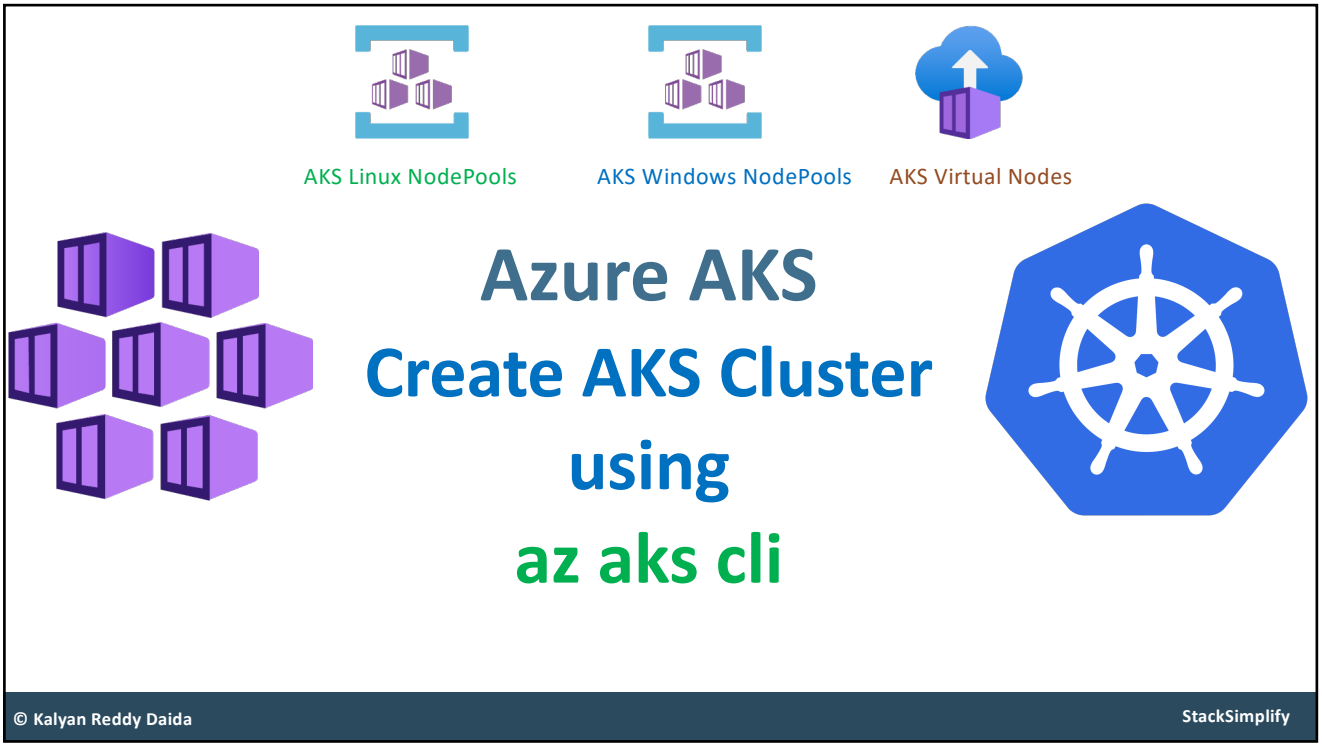
194


Azure AKS Cluster – Node Pools (Linux, Windows, Virtual)
Azure Cloud
VNET
Default Subnet: 10.240.0.0/16
System Node Pool: Linux
User Node Pool: Linux
User Node Pool: Windows
node
Azure AKS
Cluster Control Plane
c-m
api
c-c-m
sched
Load Balancer
Public IP
kubectl
AKS
AdminUser
Virtual Nodes Subnet: 10.241.0.0/16
Azure Container Instances - ACI
Virtual Node
© Kalyan Reddy Daida
StackSimplify

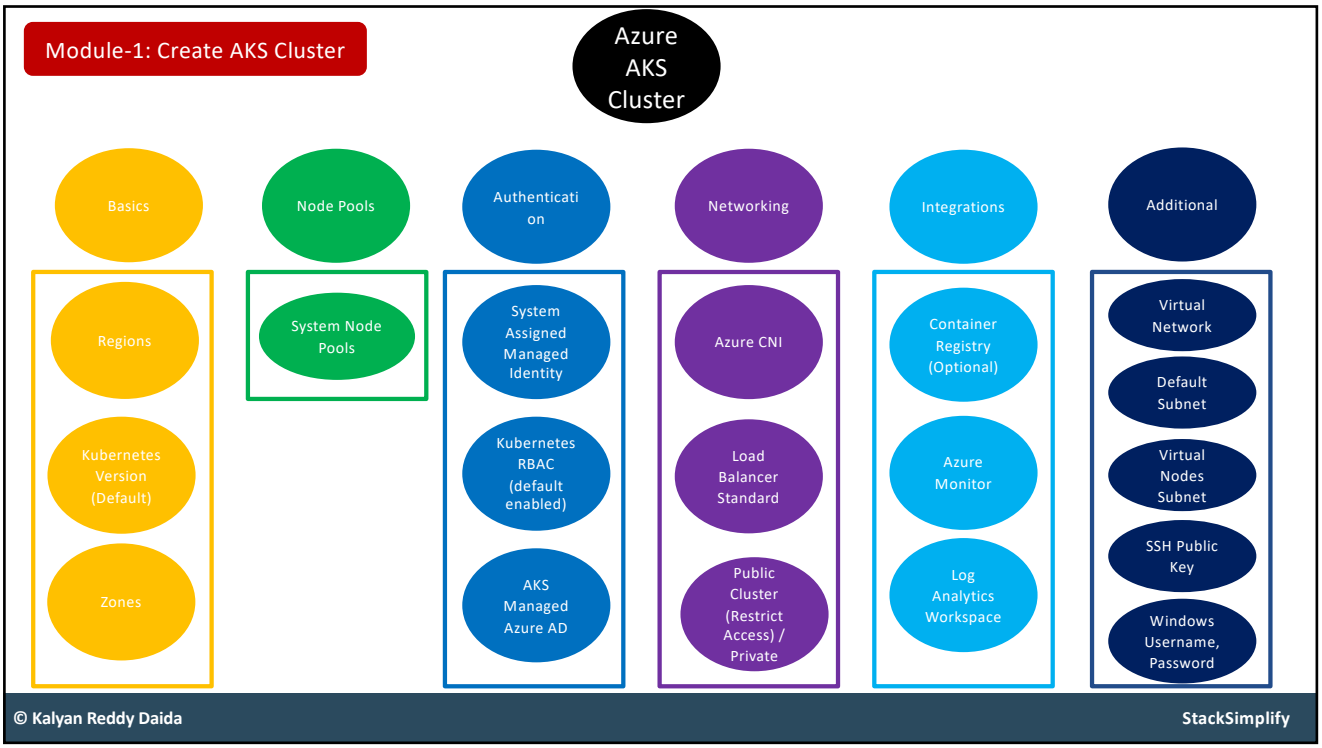
195


Azure Cloud
VNET
Default Subnet: 10.240.0.0/16
System Node Pool: Linux
App1
User Node Pool: Linux
App2
User Node Pool: Windows
App3
deploy
rs
pod
Virtual Nodes Subnet: 10.241.0.0/16
ACI Virtual Nodes
App4
AKS
Cluster Control Plane
svc
K8S Service CIDR: 10.0.0.0/16
Azure Active
Directory
K8s API Server Public IP
kubectl
AKS Admin
App1 Public IP
App2 Public IP
App3 Public IP
App4 Public IP
Load Balancer
Deploy Apps to different NodePools using
Kubernetes Node Selector Concept on Azure AKS
© Kalyan Reddy Daida
StackSimplify

196

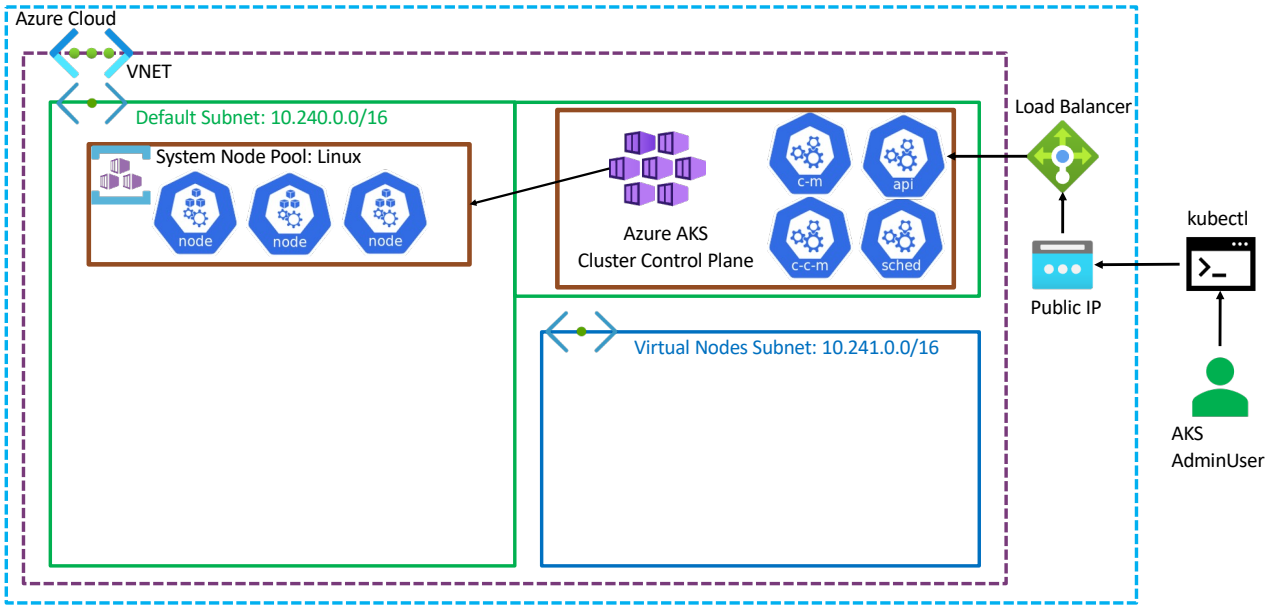

AKS Linux NodePools
AKS Windows NodePools
AKS Virtual Nodes
Azure AKS
Create AKS Cluster
using
az aks cli
© Kalyan Reddy Daida
StackSimplify

197

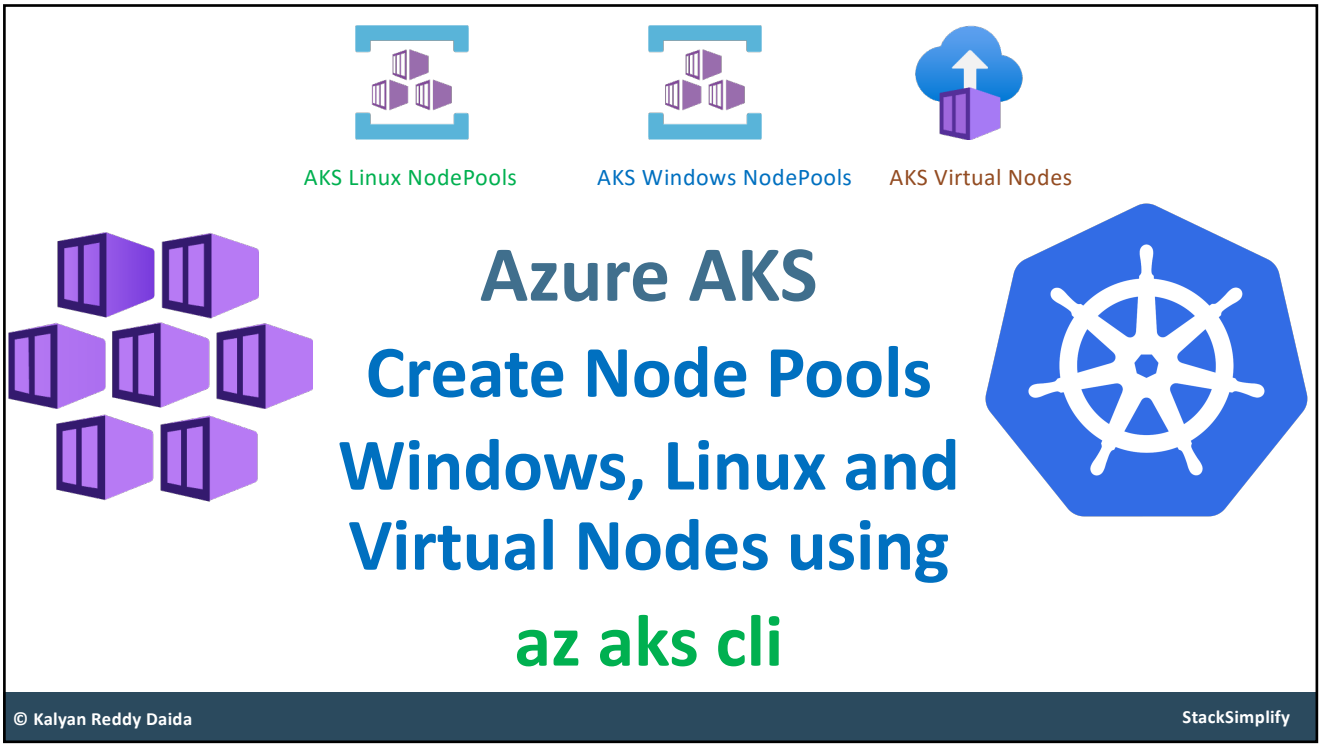

Module-1: Create AKS Cluster
Azure AKS Cluster
Basics
Regions
Kubernetes Version (Default)
Zones
Node Pools
System Node Pools
Authentication
System Assigned Managed Identity
Kubernetes RBAC (default enabled)
AKS Managed Azure AD
Networking
Azure CNI
Load Balancer Standard
Public Cluster (Restrict Access) / Private
Integrations
Container Registry (Optional)
Azure Monitor
Log Analytics Workspace
Additional
Virtual Network
Default Subnet
Virtual Nodes Subnet
SSH Public Key
Windows Username, Password
© Kalyan Reddy Daida
StackSimplify

198

Azure AKS Cluster – Create AKS Cluster using CLI

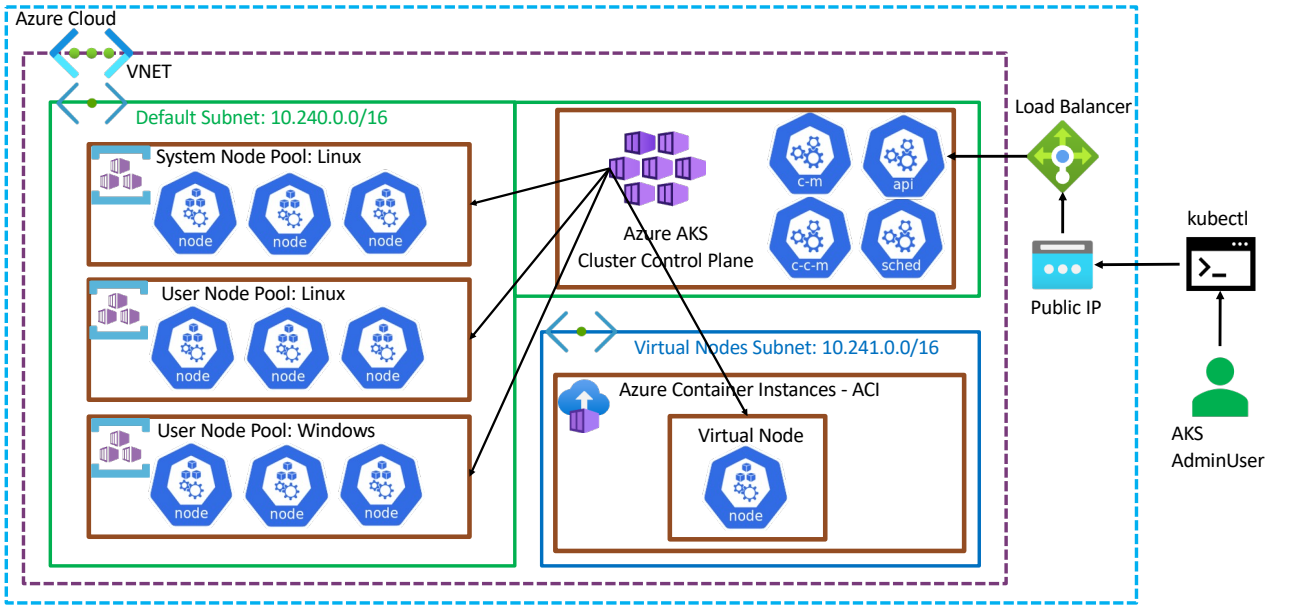


199



200

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

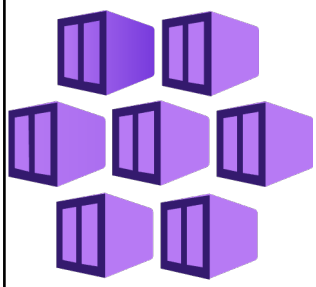


201

AKS Linux NodePools

AKS Windows NodePools

AKS Virtual Nodes

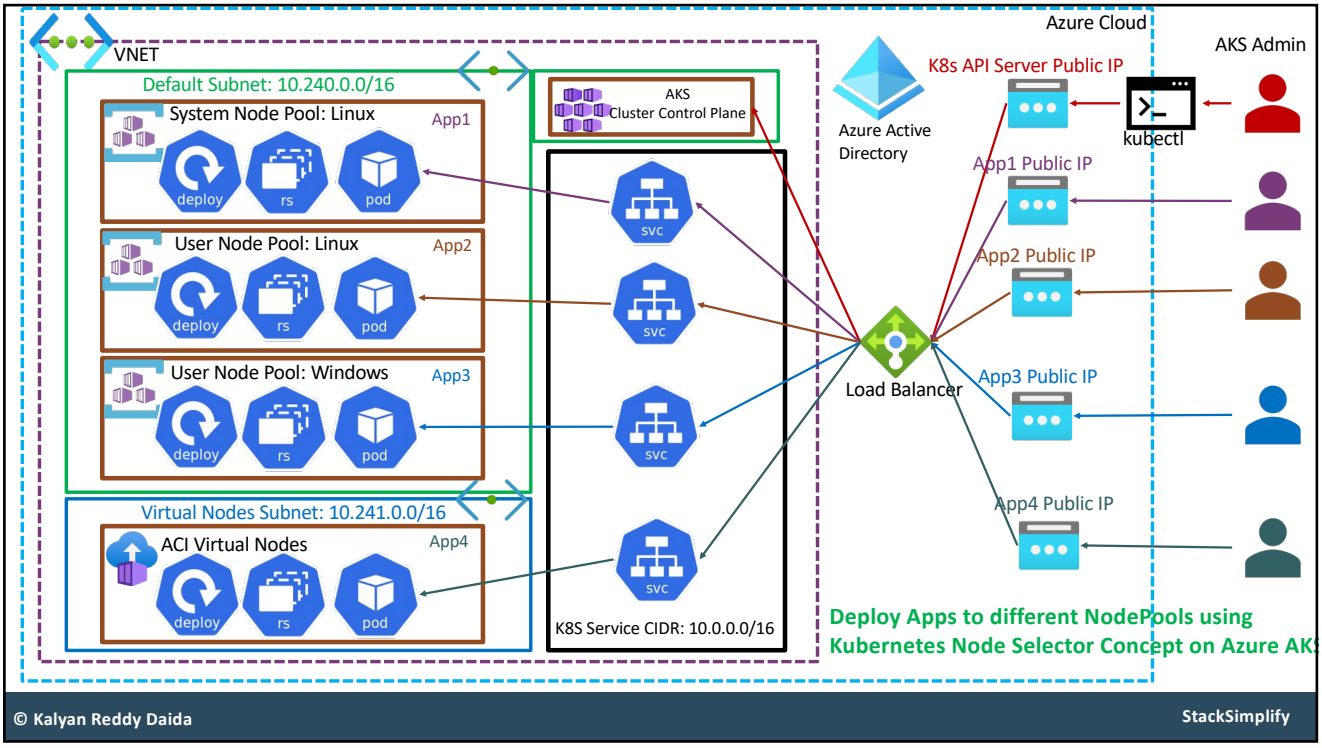


Azure AKS

Deploy Apps to different NodePools using Node Selectors

az aks cli

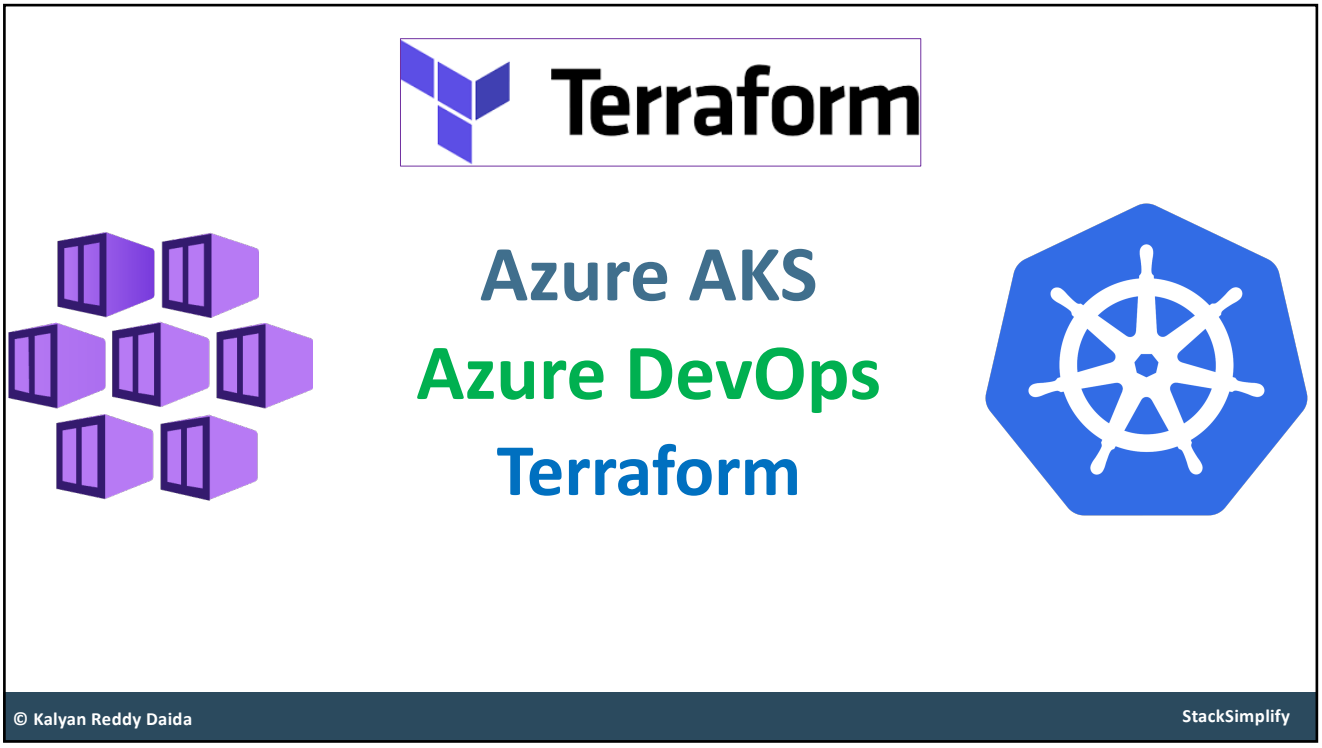
202


Azure Cloud
AKS Admin
VNET
Default Subnet: 10.240.0.0/16
System Node Pool: Linux
App1
deploy
rs
pod
User Node Pool: Linux
App2
User Node Pool: Windows
App3
Virtual Nodes Subnet: 10.241.0.0/16
ACI Virtual Nodes
App4
AKS
Cluster Control Plane
svc
K8S Service CIDR: 10.0.0.0/16
Azure Active
Directory
K8s API Server Public IP
kubectl
App1 Public IP
App2 Public IP
App3 Public IP
App4 Public IP
Load Balancer
Deploy Apps to different NodePools using
Kubernetes Node Selector Concept on Azure AKS
© Kalyan Reddy Daida
StackSimplify

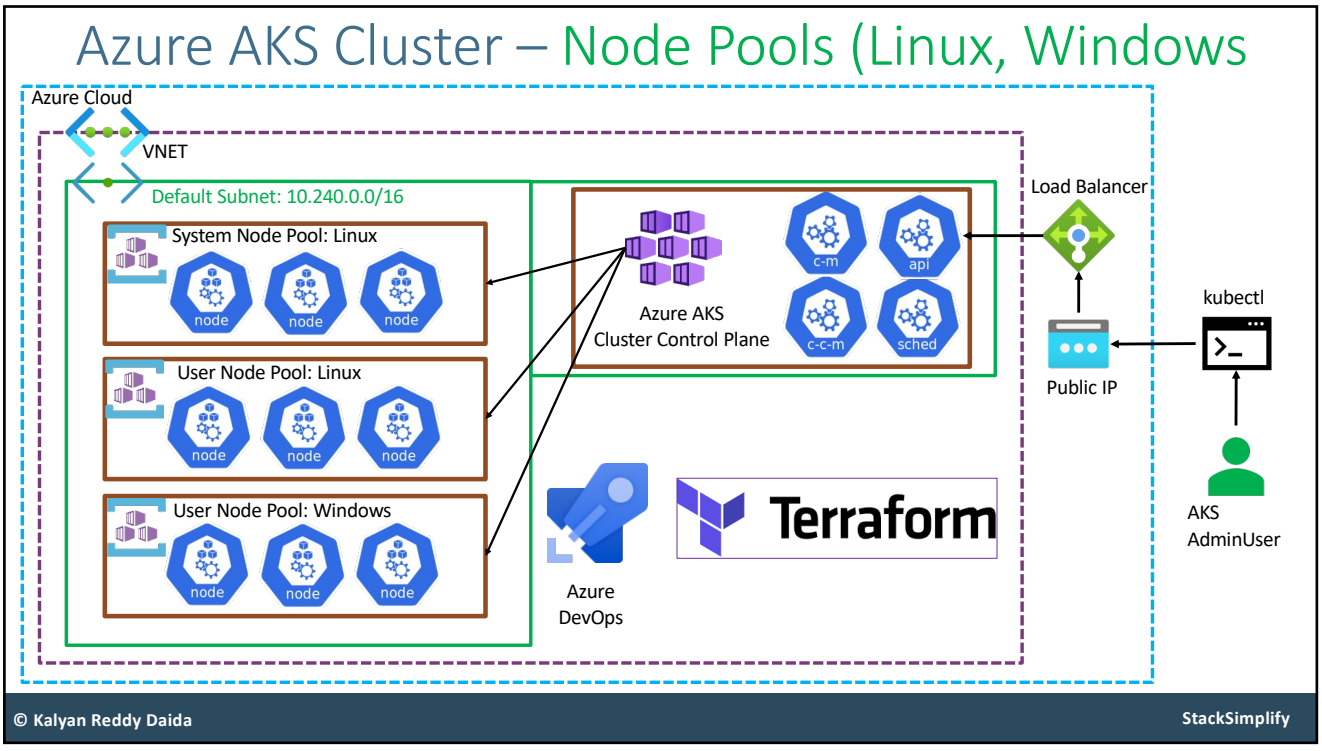
203


Azure AKS
Terraform Basics
Terraform
© Kalyan Reddy Daida
StackSimplify

204


Terraform
Azure AKS
Azure DevOps
Terraform
© Kalyan Reddy Daida
StackSimplify

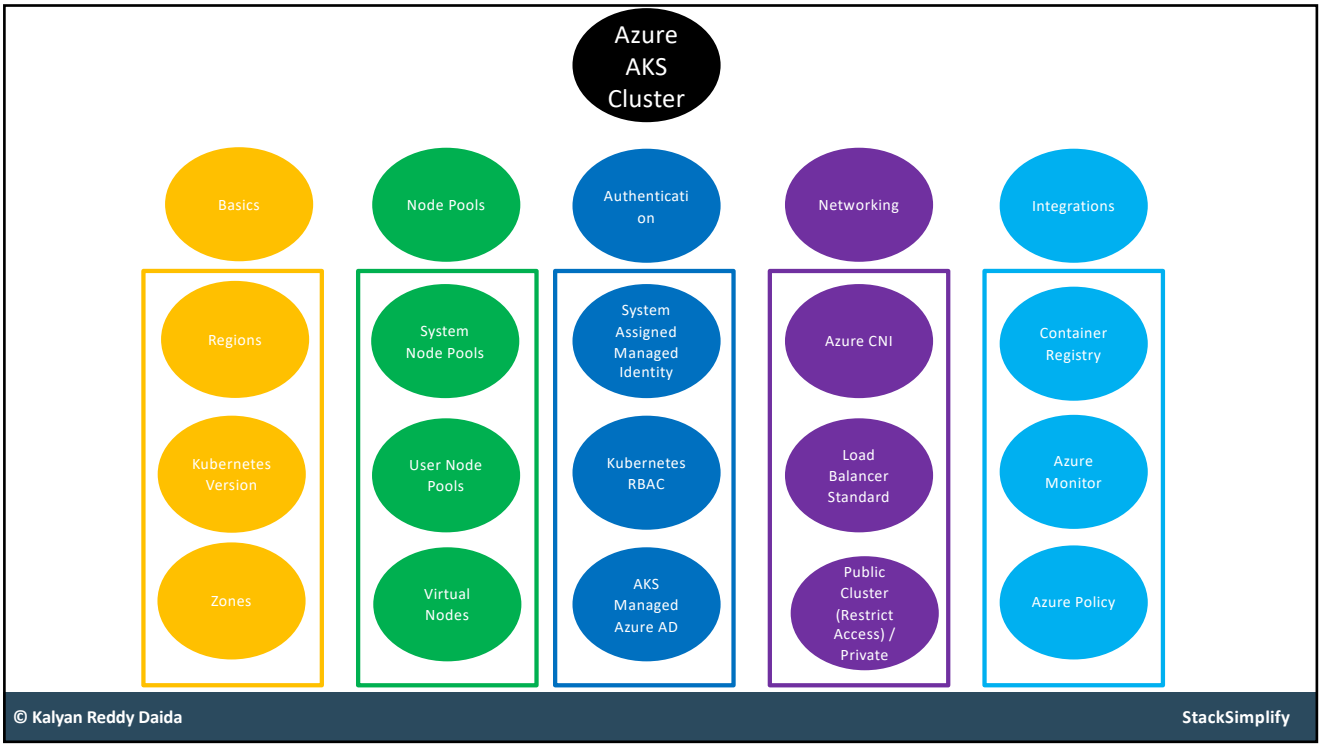
205


Azure AKS Cluster – Node Pools (Linux, Windows
Azure Cloud
VNET
Default Subnet: 10.240.0.0/16
System Node Pool: Linux
User Node Pool: Linux
User Node Pool: Windows
node
Azure AKS
Cluster Control Plane
c-m
api
c-c-m
sched
Load Balancer
Public IP
kubectl
AKS
AdminUser
Azure
DevOps
Terraform
© Kalyan Reddy Daida
StackSimplify

206

BEFORE THIS ARE LIVE SLIDES

207



208

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

209

AKS - Addons

HTTP Application Routing Ingress + External DNS

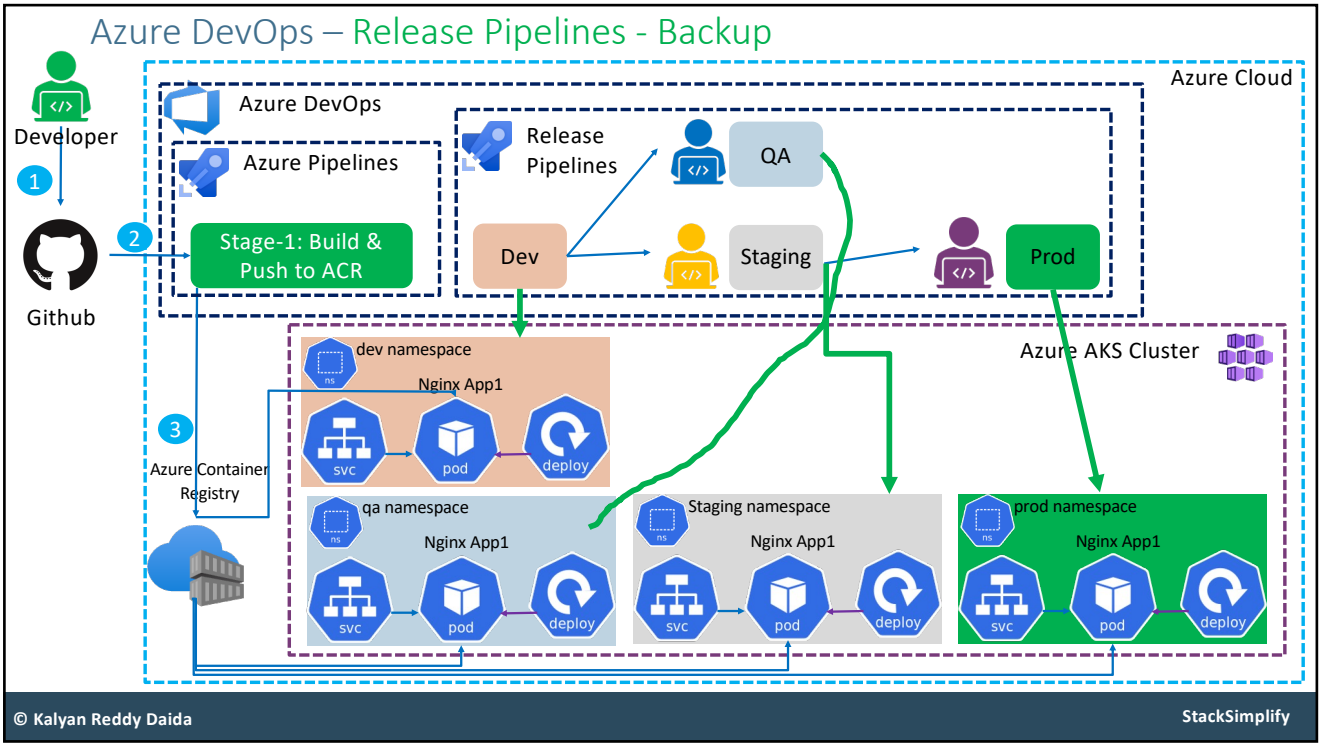
Azure AKS Cluster

Virtual Nodes

Monitoring (Log Analytics)

Azure Policy

210

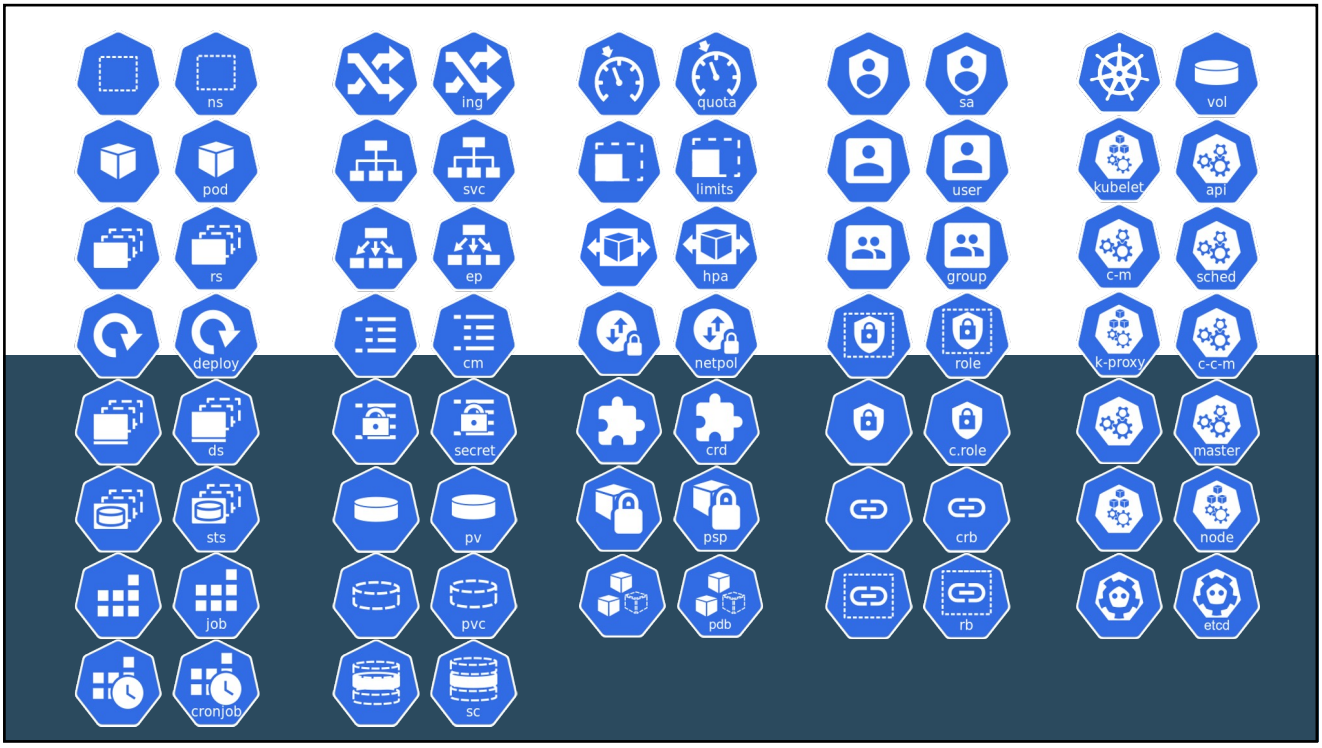


211

Which Ingress Controller to Use?

Nginx Ingress Controller (Open Source)	Azure Application Gateway Ingress Controller (AGIC)
Free of cost, only uses AKS memory and cpu resources	Very costly and still uses AKS memory and cpu resources
Stable and very very good open source community support.	Recently launched and most of the features are in preview mode.
Good for smaller workloads and smaller deployment foot print	Good for large scale workloads, at the same time who can afford the price

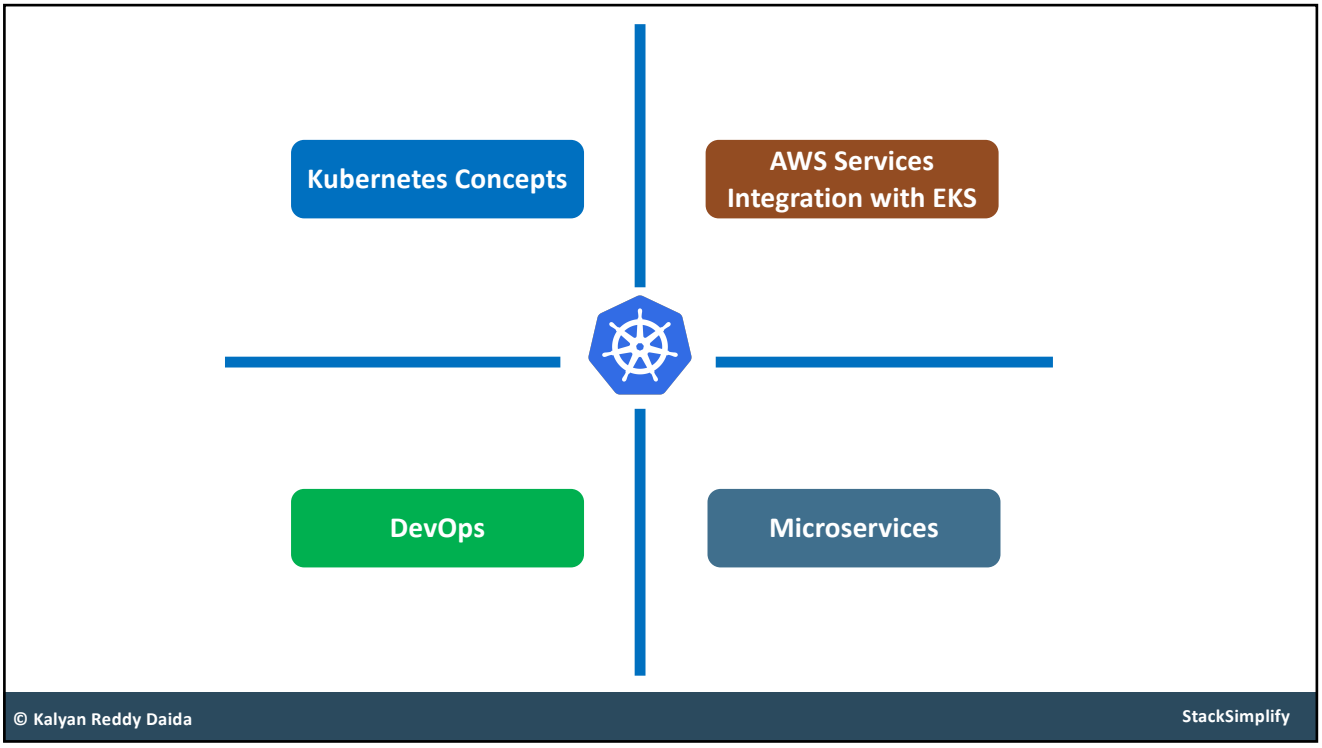
212



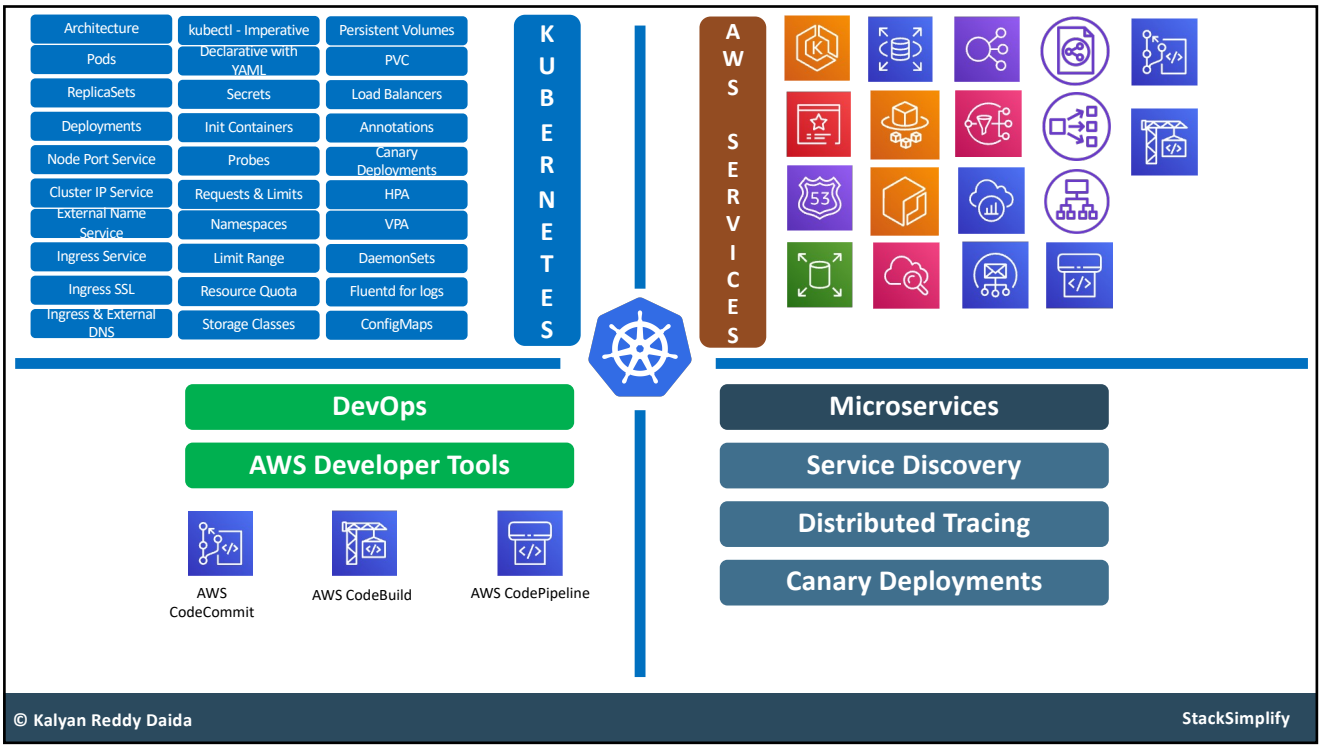
213

Backup Slides

214

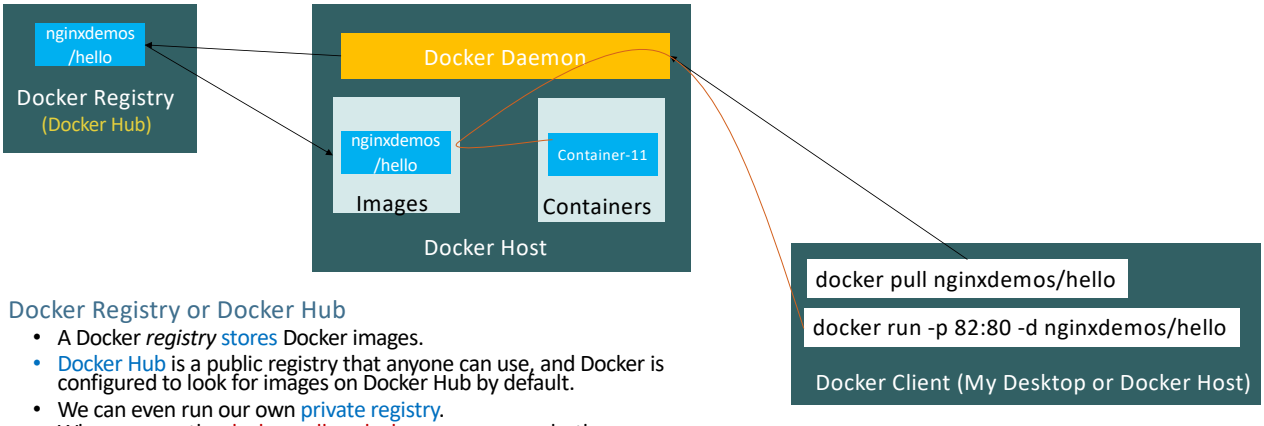

Kubernetes Concepts
AWS Services Integration with EKS
DevOps
Microservices
© Kalyan Reddy Daida
StackSimplify

215


Architecture
Pods
ReplicaSets
Deployments
Node Port Service
Cluster IP Service
External Name Service
Ingress Service
Ingress SSL
Ingress & External DNS
kubectl - Imperative
Declarative with YAML
Secrets
Init Containers
Probes
Requests & Limits
Namespaces
Limit Range
Resource Quota
Storage Classes
Persistent Volumes
PVC
Load Balancers
Annotations
Canary Deployments
HPA
VPA
DaemonSets
Fluentd for logs
ConfigMaps
KUBERNETES
AWS SERVICES
DevOps
AWS Developer Tools
AWS CodeCommit
AWS CodeBuild
AWS CodePipeline
Microservices
Service Discovery
Distributed Tracing
Canary Deployments
© Kalyan Reddy Daida
StackSimplify

216

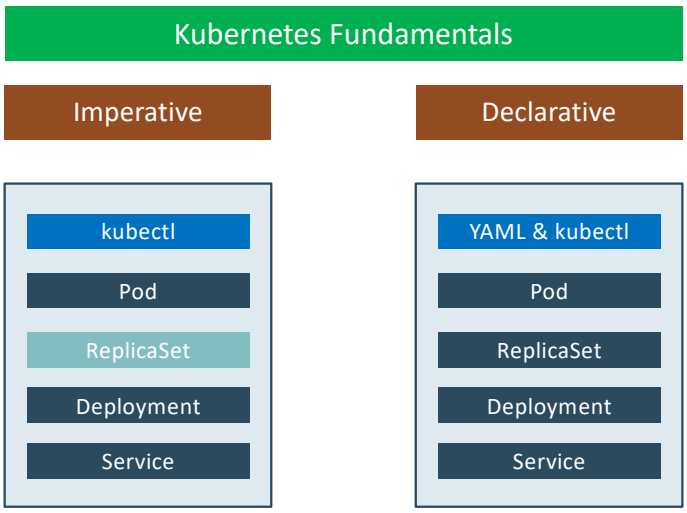
Docker - Fundamentals



- 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.

217

Kubernetes - Imperative & Declarative



218