**in summary,**

**Deployments are great for stateless applications that can be easily scaled horizontally,**

**StatefulSets are great for applications that require persistent storage and have state that needs to be maintained**

**DaemonSets are great for running an application on every node in the cluster,**

**Deployments** are a great choice for applications that don’t require any state to be maintained. These are applications that can easily scale horizontally by adding more instances of the application. Deployments allow you to manage the rollout and rollback of your application, as well as scaling and updating the application.

Think of it like baking cookies. If you want to bake more cookies, you just need to make more batches of the same recipe.

Deployments allow you to make more instances of the same application, just like making more batches of cookies.

**A good example of an application that could use a Deployment is a web server or a microservice.**

**StatefulSets** are great for applications that require persistent storage and have state that needs to be maintained. These are applications like databases, where each instance of the application needs to have a unique network identity and persistent storage.

StatefulSets ensure that each replica of the application has a stable, unique network identity, and that persistent storage is managed correctly.

**A good example of an application that could use a StatefulSet is a database.**

**DaemonSets** are great for running a single instance of an application on every node in the cluster. These are applications that need to be run on every node in the cluster.

This could be things like logging or monitoring agents.

For example, if you have a logging agent that you want to run on every node in your cluster, you could use a DaemonSet to make sure that there is an instance of the agent running on each node. This is useful because it ensures that the same instance of the application is running on each node, which can be important for tasks that require node-level access or coordination.

<https://blog.devgenius.io/deployments-daemonsets-and-statefulsets-in-kubernetes-which-one-to-use-for-your-application-5d837f738ff9>

Kubernetes version—1.26 version

|  |  |  |  |
| --- | --- | --- | --- |
| 1.23 | 7 December 2021 | 28 February 2023 | <https://kubernetes.io/blog/2021/12/07/kubernetes-1-23-release-announcement/> |
| 1.24 | 3 May 2022 | 28 July 2023 | <https://kubernetes.io/blog/2022/05/03/kubernetes-1-24-release-announcement/> |
| 1.25 | 23 August 2022 | 27 October 2023 | <https://kubernetes.io/blog/2022/08/23/kubernetes-v1-25-release/> |
| **1.26** | 9 December 2022 | 24 February 2024 | <https://kubernetes.io/blog/2022/12/09/kubernetes-v1-26-release/> |
| 1.27 | 11 April 2023 |  | <https://github.com/kubernetes/sig-release/tree/master/releases/release-1.27> |

1. What is Kubernetes?

2.Architecture & Components of Kubernetes

34.What is a pod ? what is a deployment? What is a replica set in Kubernetes?

5.What is Cluster Ip? What is NodePort? What is loadBalancer? Type pf services in Kubernetes?

6.What is Kubernetes ingress? Kubernetes ingress controller | NGINX ingress controller

7. Kubernetes ConfigMaps &Secrets

Troubleshooting Common Kubernetes Errors

* [CreateContainerConfigError](https://komodor.com/learn/how-to-fix-createcontainerconfigerror-and-createcontainer-errors/)
* [ImagePullBackOff or ErrImagePull](https://komodor.com/learn/how-to-fix-errimagepull-and-imagepullbackoff/)
* [CrashLoopBackOff](https://komodor.com/learn/how-to-fix-crashloopbackoff-kubernetes-error/)
* [Kubernetes Node Not Ready](https://komodor.com/learn/how-to-fix-kubernetes-node-not-ready-error/)

CreateContainerConfigError

This error is usually the result of a missing Secret or ConfigMap. Secrets are Kubernetes objects used to store sensitive information like database credentials. ConfigMaps store data as key-value pairs and are typically used to hold configuration information used by multiple pods.

How to identify the issue

Run kubectl get pods

How to resolve ?

Command:

kubectl describe [name] and look for a message indicating which ConfigMap is missing:

kubectl get configmap configmap-3.

If the result is null, the ConfigMap is missing, and you need to create it

2. ImagePullBackOff and ErrImagePull

This status means that a pod could not run because it attempted to pull a container image from a registry, and failed. The pod refuses to start because it cannot create one or more containers defined in its manifest.

How to identify the issue

Kubectl get pods

Resolving the issue:

Kubectl describe pod [name]

1. **Wrong image name or tag—docker pull**
2. **Authentication issue in Container registry --**Ensure the pod and node have the appropriate permissions and Secrets, then try the operation manually using docker pull.

3.CrashLoopBackOff

This issue indicates a pod cannot be scheduled on a node. This could happen because the node does not have sufficient resources to run the pod, or because the pod did not succeed in mounting the requested volumes

How to identify the issue:

Kubectl get pods

Resolving the issue:

Kubectl describe pod [pod name]

1. **Insufficient resources**
2. **Volume mounting**
3. **Use of hostPort**
4. Kubernetes Node Not Ready

When a worker node shuts down or crashes, all stateful pods that reside on it become unavailable, and the node status appears as NotReady.

If a node has a NotReady status for over five minutes (by default), Kubernetes changes the status of pods scheduled on it to Unknown, and attempts to schedule it on another node, with status ContainerCreating.

Kubectl get nodes

Resolving the issue

If the failed node can recover or is rebooted by the user, the issue will resolve itself. Once the failed node recovers and joins the cluster

**Namespaces**

In Kubernetes, *namespaces* provide a mechanism for isolating groups of resources within a single cluster.

Names of resources need to be unique within a namespace, but not across namespaces

**default**

Kubernetes includes this namespace so that you can start using your new cluster without first creating a namespace.

**kube-node-lease**

This namespace holds [Lease](https://kubernetes.io/docs/concepts/architecture/leases/) objects associated with each node. Node leases allow the kubelet to send [heartbeats](https://kubernetes.io/docs/concepts/architecture/nodes/#heartbeats) so that the control plane can detect node failure.

**kube-public**

This namespace is readable by all clients (including those not authenticated). This namespace is mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster. The public aspect of this namespace is only a convention, not a requirement.

**kube-system**

The namespace for objects created by the Kubernetes system

21-06-2023:

What is Kubernetes?

Explain Kubernetes architecture

Components of Kubernetes

Ingress controller using nginx

<https://kubernetes.io/docs/concepts/overview/components/>

kube-apiserver

The API server is a component of the Kubernetes [control plane](https://kubernetes.io/docs/reference/glossary/?all=true#term-control-plane) that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

### etcd

Consistent and highly available key value store used as Kubernetes' backing store for all cluster data.

kube-scheduler

Control plane component that watches for newly created [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) with no assigned [node](https://kubernetes.io/docs/concepts/architecture/nodes/), and selects a node for them to run on.

kube-controller-manager

Control plane component that runs [controller](https://kubernetes.io/docs/concepts/architecture/controller/) processes.

Logically, each [controller](https://kubernetes.io/docs/concepts/architecture/controller/) is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

Node, Job and EndpointSlice controller

### kubelet

An agent that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in the cluster. It makes sure that [containers](https://kubernetes.io/docs/concepts/containers/) are running in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/).

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

### kube-proxy

kube-proxy is a network proxy that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in your cluster, implementing part of the Kubernetes [Service](https://kubernetes.io/docs/concepts/services-networking/service/) concept.

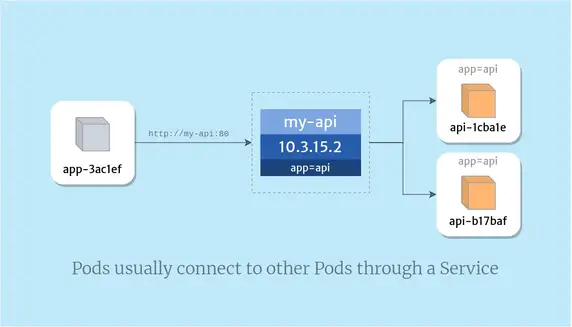
[kube-proxy](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/) maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself



Fronted pod communicate to backend pod

App -1------------------



App-xyz1000------http://my-api:80--------------🡪



Explain about Kubernetes architecture

What is services and how many services?

Explain about ingress controller and nginx

Explain difference load balance and api gateway?

Difference between an API gateway and a load balancer?

An API gateway is focused on 𝗿𝗼𝘂𝘁𝗶𝗻𝗴 requests to the appropriate microservice,

while a load balancer is focused on 𝗱𝗶𝘀𝘁𝗿𝗶𝗯𝘂𝘁𝗶𝗻𝗴 requests evenly across a group of backend servers

Explain Namespace?

How to troubleshoot in Kubernetes

What is HPA in k8s

AKS:

1) Why is load balancer needed?

A load balancer is needed because it gives a standard way to distribute network traffic among different services, which runs in the backend.

2) Define Ingress Network

Ingress network is defined as a collection of rules which allow permission for connections into the Kubernetes cluster.

3) Explain Replica set

A Replica set is used to keep replica pods stable.

It enables us to specify the available number of identical pods.

This can be considered a replacement for the replication. Controller

4) What do you mean by persistent volume?

A persistent volume is a storage unit that is controlled by the administrator. It is used to manage an individual pod in a cluster.

5) Explain PVC?

The full form of PVC stands for Persistent Volume Claim. It is storage requested by Kubernetes for pods. The user does not require to know the underlying provisioning. This claim should be created in the same namespace where the pod is created.

1.What is differebce b/w configmap and Secrets?

The big difference between Secrets and ConfigMaps are that Secrets are obfuscated with a Base64 encoding. but it is good practice to use Secrets for confidential data (like API keys) and ConfigMaps for non-confidential data (like port numbers).

kubectl create secret generic apikey --from-literal=API\_KEY=123–456

kubectl create configmap language --from-literal=LANGUAGE=English

2.What is difference b/w a replication controller and replica set?

The major difference between a replication controller and replica set is that the rolling-update command works with Replication Controllers,

but won't work with a Replica Set. This is because Replica Sets are meant to be used as the backend for Deployments.

3.What difference b/w Deployments, statefulset and daemonsetes?

Deployments are great for stateless applications that can be easily **scaled horizontally**.

StatefulSets are great for applications that require persistent storage and have state that needs to be maintained

DaemonSets are great for running an application on every node in the cluster.

4.What is auto scaling ?

Kubernetes enables autoscaling at the cluster/node level as well as at the pod level, two different but fundamentally connected layers of Kubernetes architecture.

Horizontal Pod Autoscaling

In Kubernetes, a HorizontalPodAutoscaler automatically updates a workload resource (such as a Deployment or StatefulSet), with the aim of automatically scaling the workload to match demand.

Horizontal scaling means that the response to increased load is to deploy more Pods. This is different from vertical scaling, which for Kubernetes would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload.

If the load decreases, and the number of Pods is above the configured minimum, the HorizontalPodAutoscaler instructs the workload resource (the Deployment, StatefulSet, or other similar resource) to scale back down

kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10

What is difference b/w liveness & readiness?

Both liveness & readiness probes are used to **control the health of an application**.

Failing liveness probe will restart the container,

whereas failing readiness probe will stop our application from serving traffic.

5.How do you scale Kubernetes pods automatically?

Kubernetes Automatic Scaling with the Horizontal Pod Autoscaler (HPA) Object. The Kubernetes HorizontalPodAutoscaler object updates pods, deployments and statefulsets to match demands by automatically scaling workloads. The HPA's response to increased load is to dynamically provision additional pods

What is Ingress Controller?

Ingress Controller is an intelligent Load Balancer. Ingress is a high-level abstraction responsible for allowing simple host or URL based HTTP routing. It is always implemented using a third-party proxy. These implementations are nothing but Ingress Controller. It is a Layer-7 load balancer.

1.NGINX-Based Ingress Controllers

This is the most popular and only open-source Ingress Controller maintained by the K8s team, built on top of NGINX reverse proxy. It is a popular option for simple HTTP/S routing and SSL termination use case. Hence of the popularity, there is comprehensive documentation and tutorials available for common ingress tasks and related tools

2.Kubernetes Ingress Controller with a LoadBalancer

Whatever is your ingress strategy, you presumably will need to start with an external load balancer. This will route traffic to a K8s service on the cluster that will perform service-specific routing. In this setup, your load balancer provides a stable endpoint which is nothing but an IP address for external traffic to access.

Both ingress controllers and K8s services require an external load balancer. So, this concludes that NodePort is not designed to be directly used for production

![image](https://github.com/krishnamsdpl/krishnamsdpl.github.io/assets/30367367/049efc8b-8e76-4b7a-a67d-1a659d9c0cf2)

3. Cloud-Based Ingress Controller

4. Opensource Ingress Controller

5. HAProxy-Based Ingress Controllers

6. F5 Container Ingress

08-06-2023:

What is a Kubernetes Manifest File?

A Kubernetes manifest file is your personal guide through a Kubernetes cluster: A configuration file written in a format called YAML or JSON,

that describes the resources you want in your cluster.

Manifest files:

pods (that run your applications),

services (that help your applications communicate),

deployments (that manage your applications).

What is Helm?

Helm is a tool that automates the creation, packaging, configuration, and deployment of Kubernetes applications by combining your configuration files into a single reusable package.

In a microservice architecture, you create more microservices as the application grows, making it increasingly difficult to manage.

Helm provides one of the most accessible solutions to this problem, making deployments more consistent, repeatable, and reliable

Helm Charts:

A Helm chart is a package that contains all the necessary resources to deploy an application to a Kubernetes cluster. This includes YAML configuration files for deployments, services, secrets, and config maps that define the desired state of your application

20-06-2023:

Troubleshooting Common Kubernetes Errors?

1.CreateContainerConfigError

2.ImagePullBackOff or ErrImagePull

3.CrashLoopBackOff

kubectl describe pod [name]

4.Kubernetes Node Not Ready

https://komodor.com/learn/kubernetes-troubleshooting-the-complete-guide/#:~:text=Kubernetes%20troubleshooting%20is%20the%20process,prevent%20issues%20in%20Kubernetes%20components.

What is a Service in k8s?

A service is a k8s object that exposes an application running in one or many pods as a “network service”

What is an Endpoint in k8s

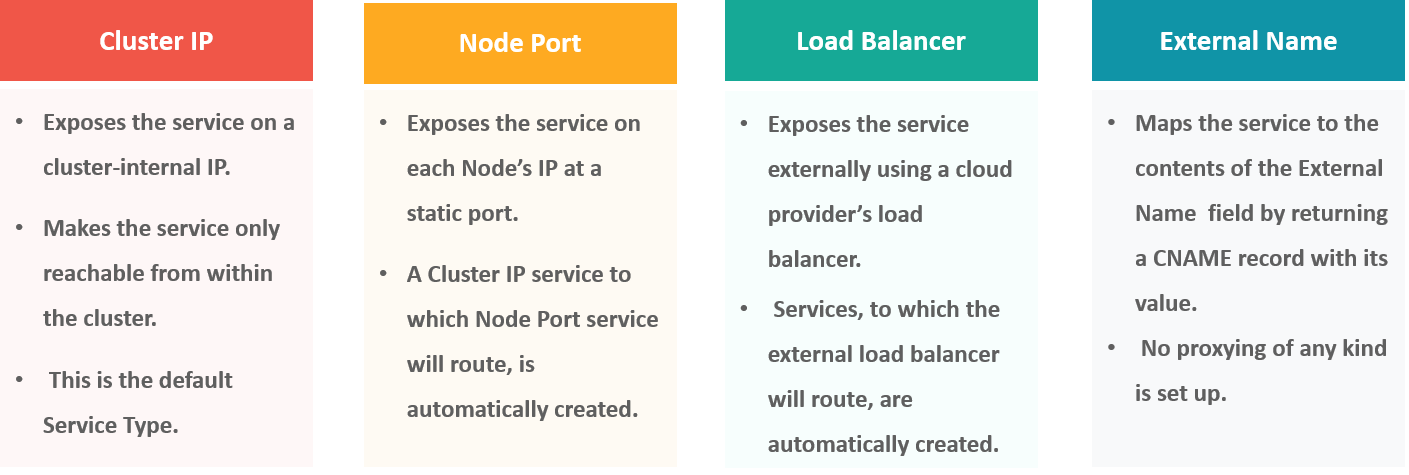
An Endpoint is a k8s object that lists all the addresses (IP addresses and Ports) that are used by a service

Endpoints in Kubernetes is a resource to track the IP addresses of the objects or pods which are dynamically assigned to it and which works as a service selector which matches a pod label by adding the IP addresses to the endpoints and these points can be viewed using software

kubectl get endpoints.

**Q7. What are the different types of services in Kubernetes?**

The following are the different types of services used:



Master Node:

<https://kubernetes.io/docs/concepts/overview/components/>

Etcd: Distubuted key-value store. Which is used in store cluster wide secrets , its only access by Kubernetes API server

API Server: The API server is a component of the Kubernetes [control plane](https://kubernetes.io/docs/reference/glossary/?all=true#term-control-plane) that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

The main implementation of a Kubernetes API server is [kube-apiserver](https://kubernetes.io/docs/reference/generated/kube-apiserver/). kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances

### kube-scheduler:

Control plane component that watches for newly created [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) with no assigned [node](https://kubernetes.io/docs/concepts/architecture/nodes/), and selects a node for them to run on.

Factors taken into account for scheduling decisions include individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

### kube-controller-manager:

Control plane component that runs [controller](https://kubernetes.io/docs/concepts/architecture/controller/) processes.

Logically, each [controller](https://kubernetes.io/docs/concepts/architecture/controller/) is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process..

Slave node /worker Node:

kubelet

An agent that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in the cluster. It makes sure that [containers](https://kubernetes.io/docs/concepts/containers/) are running in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/).

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

kube-proxy

kube-proxy is a network proxy that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in your cluster, implementing part of the Kubernetes [Service](https://kubernetes.io/docs/concepts/services-networking/service/) concept.

[kube-proxy](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/) maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself

What is Pods:

Pods can I have one or more containers coupled together They are basic unit of Kubernetes to increase high availability we always prefer pods to be replicas

What is service?

A service is basically round -robin load balancer for all pods , which matches with name and selector.

Its constantly monitoring pods, in case any unhealthy the service will start deploying to other health pods.

Cluster -Node-pod -container

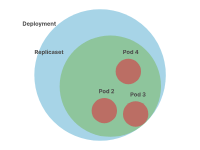
Deployments

Services

Endpoints

Ingress

Nginx



**kubectl rollout undo deployments/kubernetes-bootcamp**

<https://kubernetes.io/docs/concepts/overview/components/>

kubectl rollout history deployment/app

kubectl rollout undo deployment/app --to-revision=2

Kubernetes concepts:

1. Azure Kubernetes service overview
2. Managed vs Self-managed Kubernetes
3. AKS Capacity Analysis
4. AKS Network – Architecture
5. AKS Network -Egress traffic
6. ASK Network -Ingress traffic
7. Create AKS Cluster
8. AKS access and identity
9. Integrate AKS with Azure active directory
10. Kubernetes users
11. AKS Volume
12. AKS monitoring and logging

**Roles and Responsibilities of Kubernetes Administrator**

* To Create the container infrastructure platform
* Designing and implementing solutions
* Enhancing delivery systems with Continuous Integration
* Coordinate with other internal teams to meet goals
* Deploying, monitoring the applications in a Kubernetes environment
* Configuring hardware, services, managing settings and storage
* Improve the monitoring and alert system
* Troubleshoot issues and solve problems where needed
* Documentation and support

Application gateway

Authentication and autherization

User –RBAC

**Creating a private cluster with limited access to the public endpoint**

cloudshell container clusters create-auto private-cluster-1 \  
    --create-subnetwork name=my-subnet-1 \  
**--enable-master-authorized-networks \**    --enable-private-nodes

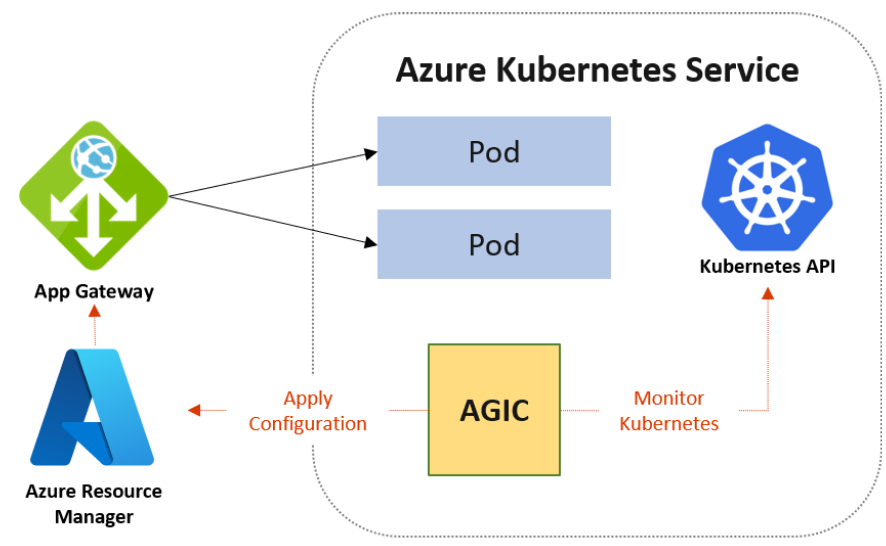
gcloud container clusters create private-cluster-1 \  
    --create-subnetwork name=my-subnet-1 \  
    --enable-master-authorized-networks \  
    --enable-ip-alias \  
**--enable-private-nodes \**    --master-ipv4-cidr 172.16.0.0/28

* --create-subnetwork name=my-subnet-1 causes GKE to automatically create a subnet named my-subnet-1.
* **--enable-master-authorized-networks specifies that access to the public endpoint is restricted to IP address ranges that you authorize.**
* --enable-ip-alias makes the cluster VPC-native (not required for Autopilot).
* --enable-private-nodes indicates that the cluster's nodes do not have external IP addresses.
* --master-ipv4-cidr 172.16.0.0/28 specifies an internal IP address range for the control plane (optional for Autopilot). This setting is permanent for this cluster and must be unique within the VPC. The [use of non RFC 1918 internal IP addresses](https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips#internal_ip_addresses) is supported.

## Creating a private cluster with unrestricted access to the public endpoint

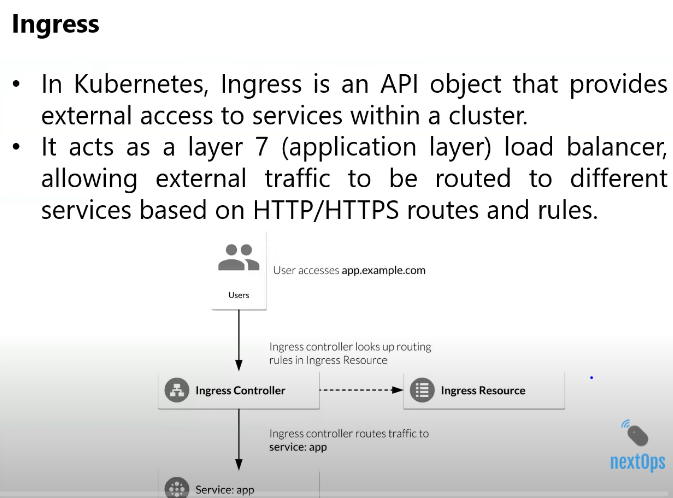
gcloud container clusters create private-cluster-3 \  
    --create-subnetwork name=my-subnet-3 \  
    **--no-enable-master-authorized-networks \**    --enable-ip-alias \  
    --enable-private-nodes \  
    --master-ipv4-cidr 172.16.0.32/28

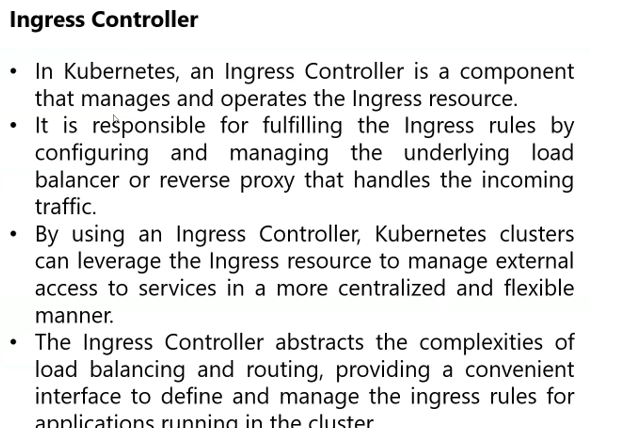
* --no-enable-master-authorized-networks disables authorized networks for the cluster.



 What is Azure Active Directory?  
Ans:

*Microsoft's Azure Active Directory (Azure AD) is a cloud-based identity and access management (IAM) solution for businesses. The backbone of the Office 365 system is Azure Active Directory, which can sync with on-premises Active Directory and offer OAuth authentication to cloud-based applications.*





Used Ingress controller will access multiple application will access

Will used

1.api gate way

2.Load balancer

Will used thirtd party using nginx



Ingress -controller

Ingress- resource

Deployment

Service

Pods

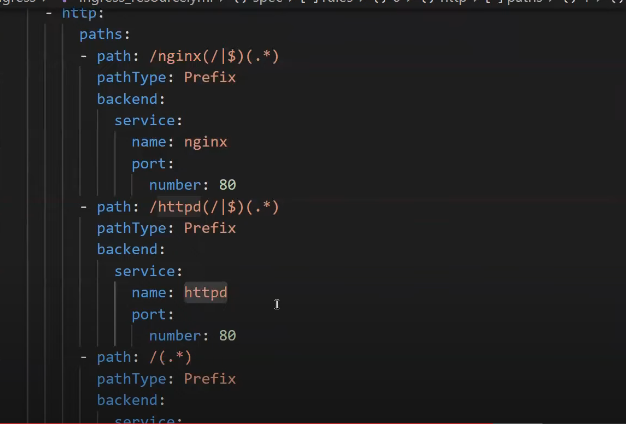
Create Https and nginx

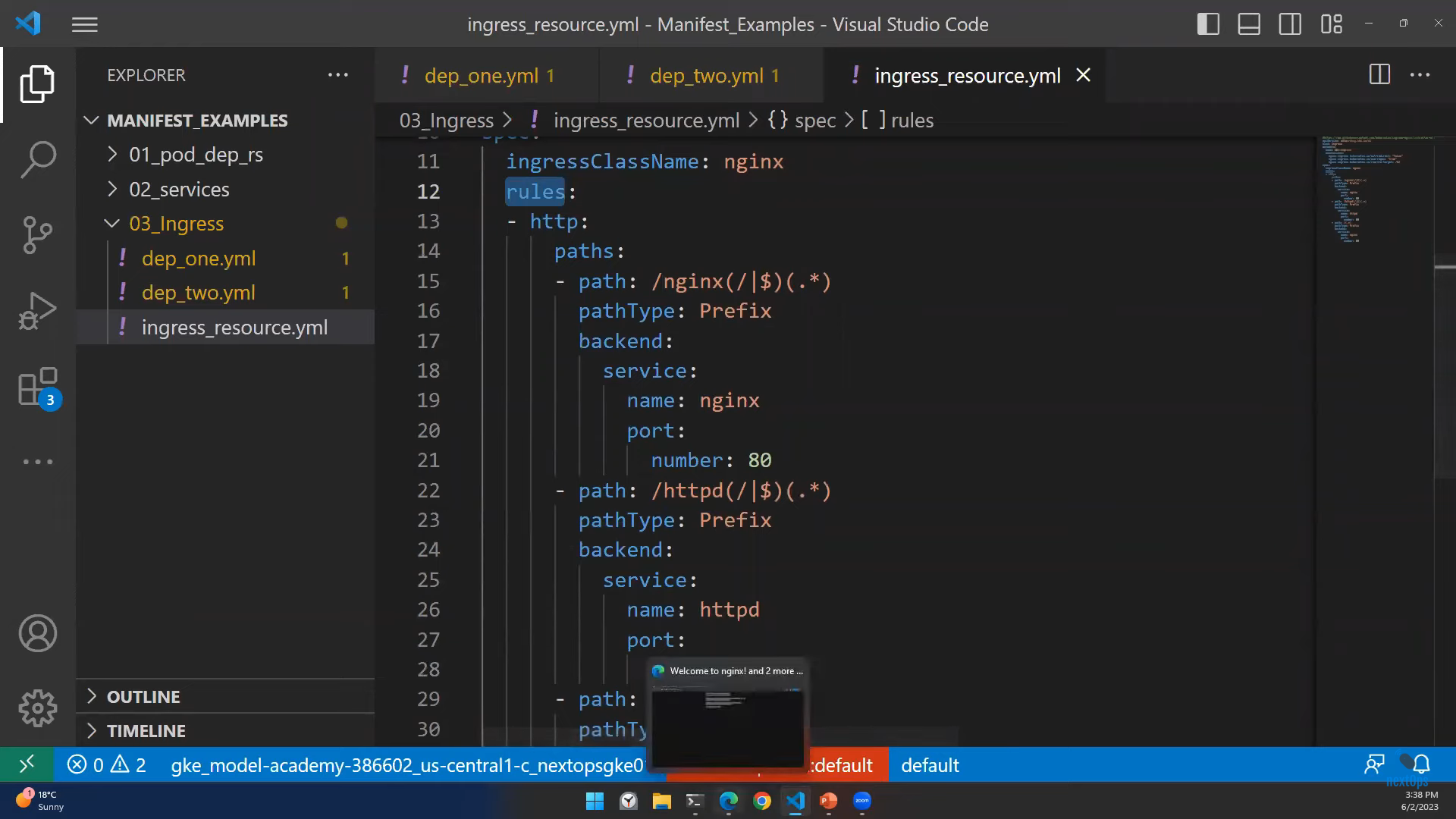
If want access out side will acesss ingress-resource

Resource definination rules:

Path based routing

Host based routing





**Understanding Kubernetes RBAC**

In Kubernetes, **ClusterRoles**and **Roles** define the actions a user can perform within a cluster or namespace, respectively. You can assign these roles to Kubernetes **subjects**(users, groups, or [service accounts](https://www.strongdm.com/blog/service-accounts)) with **role bindings and cluster role bindings**. Kubernetes allows you to configure custom roles or use default user-facing roles, including, but not limited to:

* **Cluster-admin**: This “superuser” can perform any action on any resource in a cluster. You can use this in a ClusterRoleBinding to grant full control over every resource in the cluster (and in all namespaces) or in a RoleBinding to grant full control over every resource in the respective namespace.
* **Admin**: This role permits unlimited read/write access to resources within a namespace. This role can create roles and role bindings within a particular namespace. It does not permit write access to the namespace itself.
* **Edit**: This role grants read/write access within a given Kubernetes namespace. It cannot view or modify roles or role bindings.
* **View**: This role allows read-only access within a given namespace. It does not allow viewing or modifying of roles or role bindings.

<https://www.strongdm.com/blog/kubernetes-rbac-role-based-access-control>

Role--read-only

RoleBinding--read-only-binding

Role--read-write

RoleBinding--read-write-binding

ClusterRole—superuser

ClusterRoleBinding--superuser-binding