🔶 1. What are Ingress Controllers and how do they work?

Your app is running inside a **Kubernetes Pod** in EKS. But:
❓ **How will a user from the internet access that app through a browser (via DNS like myapp.com)?**
You need a mechanism to:

- Accept **external HTTP/HTTPS traffic**
- **Route** it to the right **service and pod**
- Optionally **use paths (/api, /web)** or **hostnames (app.example.com)**

✅ **Here's where Ingress comes in:**

- **Ingress** is like a **traffic router** inside Kubernetes.
- It contains **rules** like:
  - If the URL is /api, send it to the API service.
  - If the URL is /web, send it to the frontend service.

🔁 But Ingress is **just a configuration (rules)**. It doesn't do the actual routing.

✅ **Here's where Ingress Controller comes in:**

- The **Ingress Controller** is the **real traffic manager** (the driver).
- It watches the **Ingress rules** and sets up a **load balancer or reverse proxy**.

Common Ingress Controllers in EKS:

Controller Name	Based On	Used For
aws-load-balancer-controller	AWS ALB (Application Load Balancer)	Managed, scalable load balancing
nginx-ingress-controller	NGINX	Lightweight, customizable

Analogy:
Ingress => traffic rules
Ingress Controller => driver follows the rules.(**Ingress)**

🔶 80. How do you configure ALB Ingress Controller in EKS?

Goal:
We want ALB to send internet traffic to your Kubernetes apps running in EKS.

Steps:

✅ **1. Enable IAM OIDC Provider (one-time setup)**
OIDC stands for OpenID Connect.

It is an authentication layer built on top of the OAuth 2.0 framework, providing a **standardized way for clients to verify the identity of an end-user based on the authentication performed by an authorization server.** In the context of Amazon EKS (Elastic Kubernetes Service) and IAM (Identity and Access Management), OIDC allows you to integrate external identity providers to authenticate users and grant them access to your Kubernetes cluster.

This allows AWS to trust your Kubernetes cluster's service accounts securely.
Why?
So EKS can talk securely to other AWS services (like ALB) using IAM roles.
How?
Run this command (replace the region and cluster name):
bash
---
aws eks --region <your-region> update-cluster-config \
  --name <your-cluster-name> \
  --region <your-region> \
  --output json

✅ 2. Download ALB Controller Policy (Permissions)
Why?
ALB controller needs permissions to create and manage ALBs.
How?
bash
---
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam_policy.json

---

✅ 3. Create IAM Role + EKS Service Account
Why?
You create a role with the above permissions and link it to your EKS cluster.
How?
Use this helper tool:
bash
---
eksctl create iamserviceaccount \
  --cluster <your-cluster-name> \
  --namespace kube-system \
  --name aws-load-balancer-controller \
  --attach-policy-arn arn:aws:iam::<your-account-id>:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve

✅ 4. Install ALB Controller using Helm
Why?
The controller watches your cluster and creates ALBs when needed.
How?
bash
---
helm repo add eks https://aws.github.io/eks-charts

helm upgrade -i aws-load-balancer-controller eks/aws-load-balancer-controller \
  --set clusterName=<your-cluster-name> \

```
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  -n kube-system
```

✅ 5. Create Ingress Resource with Annotations

Why?

You define *how* traffic should reach your app (like which path, domain, port, etc.)

How?

Create a Kubernetes Ingress YAML like this:

yaml

```
------
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: my-service
                port:
                  number: 80
```

📌 Summary for Beginners

Step	What You're Doing	Why?
1	Enable OIDC in EKS	So AWS can trust your cluster
2	Download IAM policy	Permissions for ALB controller
3	Create IAM role + EKS service account	Link EKS with right AWS permissions
4	Install ALB Controller with Helm	Add the brain that creates ALBs
5	Create Ingress resource	Define how ALB sends traffic to your service

📌 Use **Ingress** when:

- You want **smart routing**
- You want **HTTPS termination**
- You want **one ALB for multiple services** (cost saving)

🔷 82. How do you implement SSL/TLS termination with ALB Ingress?

Goal:

You want your app to be securely accessed by users using HTTPS, like:

```
------
https://myapp.com
```

Instead of plain HTTP (http://myapp.com), which is not secure.

Steps:

✅ 1. Create an SSL Certificate in AWS ACM

What to do?

Go to the **AWS ACM (Certificate Manager)** in the AWS Console.

- Request a public certificate for your domain (e.g., myapp.com).
- Verify the domain (via DNS or email).
- Once it's "Issued", note down the **ARN** (Amazon Resource Name).

Example:

---

arn:aws:acm:us-east-1:123456789012:certificate/abc123...

✅ 2. Annotate your Ingress resource with HTTPS settings

Update your Kubernetes Ingress YAML with these annotations:

yaml

```
metadata:
  annotations:
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:<region>:<account-
id>:certificate/<your-cert-id>
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80, "HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: "443"
```

What do these mean?

- certificate-arn: Tells ALB to use your SSL certificate.
- listen-ports: Tells ALB to listen on both HTTP (80) and HTTPS (443).
- ssl-redirect: Forces users to always use **HTTPS**.

✅ 3. Add TLS Section in Your Ingress YAML

This tells Kubernetes that your app supports TLS.

Add this to the **bottom** of your Ingress YAML:

yaml

```
spec:
  tls:
    - hosts:
        - myapp.com
```

🧪 Example Ingress YAML (Full)

yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-
1:123456789012:certificate/abc123
```

```
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80, "HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: "443"
spec:
  ingressClassName: alb
  tls:
    - hosts:
        - myapp.com
  rules:
    - host: myapp.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: my-service
                port:
                  number: 80
```

🎯 **What Happens in the End:**

Task	Who Handles It
Handles HTTPS & SSL	🔐 ALB (outside of your cluster)
Sends traffic to your app	ALB → HTTP → Pod (inside VPC)

So your **users connect securely**, but inside your AWS environment, traffic stays fast and simple using HTTP.

🔷 83. How do you configure path-based and host-based routing?

s 🎯 **Scenario:**
You want to route traffic based on **domain name and path** like:

User Accesses	Goes To
app.example.com/api	→ api-service
app.example.com/web	→ web-service

Step-by-Step Guide:

✅ 1. Understand What You're Doing:

- **Host-based Routing**: Match the domain (app.example.com)
- **Path-based Routing**: Match the path after the domain (/api, /web)

The ALB Ingress Controller in EKS will **route traffic smartly** based on these.

✅ 2. Create the Ingress YAML

Here's the Ingress resource with both host and path-based routing:
yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress

```
metadata:
  name: my-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - host: app.example.com          # Host-based routing
      http:
        paths:
          - path: /api               # Path-based routing
            pathType: Prefix
            backend:
              service:
                name: api-service    # Your backend microservice
                port:
                  number: 80
          - path: /web
            pathType: Prefix
            backend:
              service:
                name: web-service    # Your frontend UI service
                port:
                  number: 80
```

💡 **What's Happening Behind the Scenes:**

- ALB listens to incoming requests on app.example.com
- It checks the **path**:
  - /api → forwards to api-service
  - /web → forwards to web-service

ALB does **layer 7 routing (smart routing)** using the Ingress rules.

🧪 **Example Flow:**

1. User goes to: https://app.example.com/api
2. ALB reads the host and path
3. ALB forwards it to your api-service pod

Same for /web.

📌 **Summary Table**

Feature	How it works
Host Routing	host: app.example.com in Ingress
Path Routing	path: /api and path: /web rules
Backend Target	api-service, web-service in EKS
Load Balancer	ALB handles routing outside the cluster

🔶 84. What are the annotations used with ALB Ingress Controller?

🛡️ What Are Annotations?

Annotations in Kubernetes Ingress are like **extra instructions** for the **ALB Ingress Controller**.
They **customize** how the Application Load Balancer (ALB) behaves.

In Kubernetes:

- **metadata** is a standard field for naming, labeling, and categorizing objects.
- **configuration** usually refers to the core specs (like spec:) that define how the service behaves.
- **annotations** are key-value pairs used to **attach non-identifying, extra information** to Kubernetes objects.

So, we use **annotations** when:

- The data is not used to identify or select the object.
- The information is mostly used by tools/controllers (like the **ALB Ingress Controller**) to **augment behavior**.
- They **don't affect how Kubernetes schedules or manages pods** directly — they're just extra instructions.

📋 Key ALB Annotations — One by One:

Annotation	🧠 What It Means (Beginner Explanation)
alb.ingress.kubernetes.io/scheme	Sets ALB type: 🔷 internet-facing (for public access) 🔷 internal (for private VPC-only access)
alb.ingress.kubernetes.io/certificate-arn	Use this to attach an **SSL certificate** from ACM for **HTTPS** traffic
alb.ingress.kubernetes.io/ssl-redirect	Forces users to use **HTTPS only** (redirects all HTTP traffic to HTTPS)
alb.ingress.kubernetes.io/listen-ports	Tells ALB to listen on specific ports like **80 (HTTP)** and **443 (HTTPS)**
alb.ingress.kubernetes.io/target-type	Set to ip so ALB can forward traffic **directly to EKS pods** (not EC2 instances)
alb.ingress.kubernetes.io/healthcheck-path	Path that ALB uses to check if your service is **healthy**, like /healthz
alb.ingress.kubernetes.io/group.name	Group multiple Ingresses into a **single ALB** (useful for microservices)
alb.ingress.kubernetes.io/load-balancer-attributes	Advanced settings like **access logs**, idle timeout, etc.

🧪 Example Ingress with Annotations:

yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress

```
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-
1:123456789012:certificate/abc123
    alb.ingress.kubernetes.io/ssl-redirect: "443"
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80, "HTTPS": 443}]'
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/healthcheck-path: /healthz
    alb.ingress.kubernetes.io/group.name: my-apps
    alb.ingress.kubernetes.io/load-balancer-attributes:
access_logs.s3.enabled=true,access_logs.s3.bucket=my-log-bucket
spec:
  ingressClassName: alb
  rules:
    - host: app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: my-service
                port:
                  number: 80
```

Summary Table for Reference:

Annotation	Used For
scheme	Public or private ALB
certificate-arn	Use HTTPS with SSL
ssl-redirect	Force secure traffic
listen-ports	Define HTTP/HTTPS ports
target-type	Send traffic to pods
healthcheck-path	ALB checks app health
group.name	Use one ALB for many Ingresses
load-balancer-attributes	Access logs and settings

Summary

Concept	Explanation
Ingress	Set of rules to route external traffic into the cluster
Ingress Controller	Load balancer or reverse proxy that follows those rules
AWS ALB Controller	Lets EKS use AWS ALB to route HTTP/S traffic
Service vs Ingress	Service exposes a pod, Ingress routes across services
TLS with ALB	Use ACM cert + annotations for HTTPS
Path/Host Routing	Ingress can map /api or web.example.com to correct service

Concept	Explanation
Annotations	Fine-tune how ALB behaves for your app

What is the primary function of an Ingress in Kubernetes?
A. Monitor pod memory usage
B. Route external HTTP(S) traffic to services
C. Create storage volumes for pods
D. Manage IAM roles for pods

✅ **Answer:** B
📝 *Ingress manages how users access your app from the internet over HTTP/HTTPS.*

What is the role of an Ingress Controller?
A. Stores application logs
B. Monitors containers for security issues
C. Implements and enforces Ingress rules
D. Connects EKS to S3 buckets

✅ **Answer:** C

Which of the following is a valid Ingress Controller for AWS EKS?
A. aws-s3-ingress-controller
B. aws-load-balancer-controller
C. eks-network-controller
D. alb-ingress-nginx

✅ **Answer:** B

Before installing the ALB Ingress Controller, which of the following must be set up?
A. ECR image for pods
B. OIDC provider and IAM role
C. RDS instance and subnet group
D. Fargate profiles

✅ **Answer:** B

Before you install the **AWS ALB Ingress Controller** in your EKS cluster, you need to set up:

1. **OIDC Provider (OpenID Connect)**
   - This lets your EKS **Service Accounts** talk securely with AWS services.
   - Think of it like giving your Kubernetes components an **AWS identity** so they can access AWS APIs (like to create/load ALBs).
2. **IAM Role and IAM Policy**
   - You create a **role with specific permissions** (IAM policy).
   - You link this role to a **Kubernetes Service Account** used by the ALB Controller.

Without these two:

- The controller cannot **create ALBs**, attach **listeners**, or **update DNS records** — all actions require AWS permissions.

Which tool is commonly used to install the ALB Ingress Controller in EKS?

A. eksctl
B. kubectl apply
C. Helm
D. AWS CloudShell

✅ **Answer:** C

Which Kubernetes object allows advanced HTTP routing using paths or hostnames?

A. ConfigMap
B. Service
C. Ingress
D. PVC
✅ **Answer:** C

What is true about Kubernetes Services?

A. Can create one shared load balancer for multiple apps
B. Can terminate SSL
C. Expose pods using IP/DNS
D. Provide built-in path-based routing
✅ **Answer:** C

Which of the following features is provided only by Ingress, not by Service?

A. Internal DNS for pods
B. Path-based routing
C. Kubernetes metrics
D. Container logs
✅ **Answer:** B

Where should the SSL certificate be created for ALB TLS termination?

A. Inside the pod using self-signed certs
B. AWS Certificate Manager (ACM)
C. Amazon Route 53
D. Kubernetes Secret
✅ **Answer:** B

Which Ingress annotation is used to specify the SSL certificate ARN?

A. alb.ingress.kubernetes.io/ssl-cert
B. tls.cert.aws.com
C. alb.ingress.kubernetes.io/certificate-arn
D. kubernetes.io/ssl-arn
✅ **Answer:** C

What does ALB do after SSL/TLS termination?
A. Encrypt traffic to pods again
B. Drops unencrypted traffic
C. Forwards traffic as HTTP to backend pods
D. Shuts down the Ingress
✅ **Answer:** C

What does host-based routing allow in Ingress rules?
A. Route traffic based on CPU load
B. Route based on file size
C. Route based on the incoming domain (e.g., app.example.com)
D. Route based on IP address
✅ **Answer:** C

What should alb.ingress.kubernetes.io/target-type be set to in EKS for pod-level routing?
A. instance
B. pod
C. ip
D. internal
✅ **Answer:** C
📝 *EKS uses pod IPs, so the correct value is ip.*