**1. What is Ingress?**

* In Kubernetes, a **Service** exposes your app **inside the cluster**.
* To expose apps to the **outside world** in a controlled way, we use **Ingress**.
* Ingress is like a **smart router** sitting at the cluster’s edge, deciding where external traffic goes (to which service).

💡 Example:

* /nginx → goes to **nginx-service** (port 80)
* /todo → goes to **todoapp** (port 5000)

**2. Ingress Controller vs Ingress Resource**

* **Ingress Resource** → The YAML object you wrote (kind: Ingress), just rules.
* **Ingress Controller** → The brain that implements the rules. Popular options:
  + **NGINX Ingress Controller** (most widely used)
  + **Traefik**
  + **HAProxy**
  + **AWS/GCP/Azure Load Balancer Ingress**

⚠️ Without a controller, Ingress YAML does nothing.

**3. Ingress Annotations & Features**

Annotations allow customization:

* nginx.ingress.kubernetes.io/rewrite-target: "/" → rewrites path.
* nginx.ingress.kubernetes.io/ssl-redirect: "true" → force HTTPS.
* nginx.ingress.kubernetes.io/limit-rps: "5" → rate limiting.
* nginx.ingress.kubernetes.io/auth-type: basic → basic authentication.
* nginx.ingress.kubernetes.io/proxy-body-size: "10m" → allow bigger uploads.

**4. TLS/HTTPS with Ingress**

Secure traffic with certificates:

spec:

tls:

- hosts:

- example.com

secretName: tls-secret

* tls-secret is a Kubernetes secret containing cert + key.
* You can automate certs using **cert-manager** + Let’s Encrypt.

**5. Automation (Deployment Approaches)**

* **Helm Chart**: Bundle ingress + service + deployment into a Helm chart.
* **Kustomize**: Keep overlays for dev/stage/prod ingress.
* **GitOps (ArgoCD / Flux)**: Keep ingress definitions in Git, auto-sync.
* **Scripts (kubectl + bash)**: Automate rollout + updates.

Example automation via Helm values:

ingress:

enabled: true

className: nginx

hosts:

- host: myapp.local

paths:

- path: /nginx

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

**6. Best Practices**

* Always use **IngressClass** (instead of depending on default).
* Use **wildcard hosts** (\*.example.com) for multi-tenant apps.
* Centralize TLS with **cert-manager**.
* Monitor with **Prometheus** + NGINX ingress metrics.
* Use **path-based** for microservices, **host-based** for multiple domains.

**What is Ingress?**

Ingress is a Kubernetes resource that manages external access to services inside the cluster. It provides HTTP/HTTPS routing, load balancing, SSL termination, and name-based virtual hosting.

**Components**

* **Ingress Resource**: Defines routing rules (YAML file).
* **Ingress Controller**: Executes rules (NGINX, Traefik, HAProxy, cloud LB).

**Example**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-ingress

namespace: nginx

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /nginx

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

- path: /todo

pathType: Prefix

backend:

service:

name: todoapp

port:

number: 5000

**Annotations (NGINX Ingress)**

* nginx.ingress.kubernetes.io/rewrite-target: "/" → Rewrites request path.
* nginx.ingress.kubernetes.io/ssl-redirect: "true" → Enforces HTTPS.
* nginx.ingress.kubernetes.io/limit-rps: "5" → Rate limiting.
* nginx.ingress.kubernetes.io/proxy-body-size: "10m" → Max upload size.
* nginx.ingress.kubernetes.io/auth-type: basic → Basic auth support.

**TLS/HTTPS**

Ingress supports TLS termination.

spec:

tls:

- hosts:

- example.com

secretName: tls-secret

* tls-secret contains SSL certificate + key.
* Automate with **cert-manager** for Let’s Encrypt.

**Automation**

* **Helm**: Template ingress configuration.
* **Kustomize**: Environment overlays for ingress.
* **GitOps**: Manage ingress in Git, auto-deploy via ArgoCD/Flux.
* **Bash/Pipelines**: Automate rollout and updates.

**Best Practices**

1. Use IngressClass for clarity.
2. Use host-based rules for multiple domains.
3. Automate TLS with cert-manager.
4. Apply RBAC to control ingress creation.
5. Monitor ingress traffic with Prometheus/Grafana.
6. Centralize ingress definitions for maintainability.

**Conclusion**

Ingress provides a powerful way to route traffic into Kubernetes. With proper automation, annotations, TLS, and monitoring, it becomes a secure, scalable, and production-ready edge solution.