Here’s your refined content, maintaining the original structure and details:

**Kubernetes Ingress - Theory & Practical Guide (Day 38 DevOps Course)**

**Instructor:** Abhishek

**Key Topics Covered**

1. **Why Ingress is Needed**
2. **Challenges with Kubernetes Services**
3. **Ingress vs. LoadBalancer Service**
4. **Ingress Controllers**
5. **Practical Demo: Host-Based Routing**

**1. Why Ingress?**

**Challenges with Kubernetes Services:**

1. **Limited Load Balancing Capabilities:**
   * Kubernetes services provide basic **round-robin** load balancing.
   * Lacks enterprise features such as:
     + Sticky sessions
     + Path-based routing
     + Host-based routing
     + TLS termination (HTTPS)
     + Rate limiting, Web Application Firewall (WAF), etc.
2. **Expensive External Exposure:**
   * Each LoadBalancer service creates a **new cloud load balancer and static IP**, leading to high costs.
   * Example: 1000 services → 1000 load balancers → Increased expenses.

**Solution:** **Ingress** (introduced in Kubernetes v1.1).

**2. What is Ingress?**

* **Ingress Resource:** A Kubernetes object that defines routing rules (e.g., host-based or path-based routing).
* **Ingress Controller:** A load balancer (e.g., Nginx, HAProxy, F5) that implements these rules and manages traffic flow.

**How It Works:**

1. Users define **Ingress rules** in a YAML configuration.
2. The **Ingress Controller** monitors these rules and dynamically configures routing.
3. This setup provides a **single entry point** for multiple services, optimizing cost and management.

**3. Ingress vs. LoadBalancer Service**

| **Feature** | **Ingress** | **LoadBalancer Service** |
| --- | --- | --- |
| **Routing Rules** | Supports path/host-based routing | Only basic round-robin load balancing |
| **Cost** | Single load balancer for multiple services | Each service requires a separate load balancer (expensive) |
| **Enterprise Features** | TLS termination, WAF, rate limiting, etc. | Limited to cloud provider's load balancer capabilities |

**4. Practical Demo: Host-Based Routing**

**Step 1: Install Ingress Controller (Nginx)**

minikube addons enable ingress # Enable Ingress in Minikube

kubectl get pods -n ingress-nginx # Verify installation

**Step 2: Create an Ingress Resource**

# ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: example-ingress

spec:

rules:

- host: foo.bar.com

http:

paths:

- path: /bar

pathType: Prefix

backend:

service:

name: python-django-app-service # Match with your service name

port:

number: 80

Apply the configuration:

kubectl apply -f ingress.yaml

**Step 3: Test Host-Based Routing**

* Update the /etc/hosts file for local testing:
* echo "192.168.64.11 foo.bar.com" | sudo tee -a /etc/hosts
* Access the service:
* curl http://foo.bar.com/bar

**Key Takeaways**

1. **Ingress solves major service limitations:**
   * Enables advanced load balancing.
   * Reduces cloud infrastructure costs by minimizing LoadBalancer services.
2. **An Ingress Controller is required** (e.g., Nginx, Traefik).
3. **Ingress provides powerful features:**
   * Path-based and host-based routing.
   * TLS termination for HTTPS.
   * Rate limiting, security policies, and WAF support.

**Next Steps:**

* Implement **TLS/HTTPS Ingress** for secure access.
* Explore other **Ingress Controllers** like Traefik and Ambassador.

**Like & Subscribe for more DevOps content!** 🚀

### \*\*Kubernetes Ingress - Theory & Practical Guide (Day 38 DevOps Course)\*\*

\*\*Instructor:\*\* Abhishek

---

### \*\*Key Topics Covered\*\*

1. \*\*Why Ingress is Needed\*\*

2. \*\*Problems with Kubernetes Services\*\*

3. \*\*Ingress vs. LoadBalancer Service\*\*

4. \*\*Ingress Controllers\*\*

5. \*\*Practical Demo: Host-Based Routing\*\*

---

### \*\*1. Why Ingress?\*\*

\*\*Problems with Kubernetes Services:\*\*

1. \*\*Limited Load Balancing Capabilities:\*\*

- Services offer basic \*\*round-robin\*\* load balancing.

- Missing \*\*enterprise features\*\* like:

- Sticky sessions

- Path-based routing

- Host-based routing

- TLS termination (HTTPS)

- Rate limiting, WAF, etc.

2. \*\*Costly External Exposure:\*\*

- Each `LoadBalancer` service creates a \*\*new cloud LB + static IP\*\* (charged by cloud providers).

- Example: 1000 services → 1000 LBs → High cost!

\*\*Solution:\*\* \*\*Ingress\*\* (introduced in Kubernetes v1.1).

---

### \*\*2. What is Ingress?\*\*

- \*\*Ingress Resource:\*\* A Kubernetes manifest defining routing rules (e.g., host/path-based routing).

- \*\*Ingress Controller:\*\* A \*\*load balancer\*\* (e.g., Nginx, HAProxy, F5) that implements these rules.

\*\*How It Works:\*\*

1. User defines \*\*Ingress rules\*\* (YAML).

2. \*\*Ingress Controller\*\* watches these rules and configures the LB accordingly.

3. \*\*Single entry point\*\* for multiple services (saves cost).

---

### \*\*3. Ingress vs. LoadBalancer Service\*\*

| Feature | Ingress | LoadBalancer Service |

|-----------------------|----------------------------------|----------------------------------|

| \*\*Routing Rules\*\* | Supports path/host-based routing | Only basic round-robin LB |

| \*\*Cost\*\* | Single LB for multiple services | 1 LB per service (expensive) |

| \*\*Enterprise Features\*\*| TLS, WAF, rate limiting, etc. | Limited to cloud provider LB |

---

### \*\*4. Practical Demo: Host-Based Routing\*\*

#### \*\*Step 1: Install Ingress Controller (Nginx)\*\*

```sh

minikube addons enable ingress # For Minikube

kubectl get pods -n ingress-nginx # Verify

```

#### \*\*Step 2: Create Ingress Resource\*\*

```yaml

# ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: example-ingress

spec:

rules:

- host: foo.bar.com

http:

paths:

- path: /bar

pathType: Prefix

backend:

service:

name: python-django-app-service # Your service name

port:

number: 80

```

Apply:

```sh

kubectl apply -f ingress.yaml

```

#### \*\*Step 3: Test Host-Based Routing\*\*

- Update `/etc/hosts` (for local testing):

```sh

echo "192.168.64.11 foo.bar.com" | sudo tee -a /etc/hosts

```

- Access:

```sh

curl http://foo.bar.com/bar

```

---

### \*\*Key Takeaways\*\*

1. \*\*Ingress solves:\*\*

- Advanced LB features missing in Services.

- Costly cloud LB IPs.

2. \*\*Always deploy an Ingress Controller\*\* (e.g., Nginx, Traefik).

3. \*\*Ingress supports:\*\*

- Path/host-based routing

- TLS termination

- Rate limiting, WAF, etc.

\*\*Next Steps:\*\*

- Try \*\*TLS Ingress\*\* (HTTPS).

- Explore other \*\*Ingress Controllers\*\* (e.g., Traefik, Ambassador).

\*\*Like & Subscribe for More DevOps Content!\*\* 🚀