# Kubernetes Ingress Setup and Rate Limiting on Minikube

## Objective

This guide will walk you through setting up **Ingress routing** and implementing **rate limiting** in a Kubernetes environment using **Minikube**. You will deploy two sample web applications (app1 and app2), expose them via services, set up routing using NGINX Ingress, and configure custom rate limiting.

## 1. Setup Minikube Kubernetes Environment

### Step 1: Install Minikube

If you haven't installed **Minikube** yet, follow the installation instructions from the official documentation.

Once installed, start Minikube:

minikube start

This will set up a local Kubernetes cluster using Minikube. Once it's up, check the status:

kubectl get nodes

You should see a node with the Ready status.

## 2. Deploy Web Applications

### Step 2: Create Deployments for app1 and app2

Create two deployments for app1 and app2. Below are the YAML files for both applications.

#### app1-deployment.yaml

apiVersion: apps/v1

kind: Deploymentmetadata:

name: app1spec:

replicas: 1

selector:

matchLabels:

app: app1

template:

metadata:

labels:

app: app1

spec:

containers:

- name: app1

image: gcr.io/google-samples/hello-app:1.0

ports:

- containerPort: 8080

#### app2-deployment.yaml

apiVersion: apps/v1

kind: Deploymentmetadata:

name: app2spec:

replicas: 1

selector:

matchLabels:

app: app2

template:

metadata:

labels:

app: app2

spec:

containers:

- name: app2

image: gcr.io/google-samples/hello-app:2.0

ports:

- containerPort: 8080

**Apply these files to your Kubernetes cluster:**

kubectl apply -f app1-deployment.yaml

kubectl apply -f app2-deployment.yaml

### Step 3: Expose app1 and app2 via Services

Next, create services to expose these applications. These services will use port 8080 to expose the applications internally.

#### app1-service.yaml

apiVersion: v1

kind: Servicemetadata:

name: app1-servicespec:

selector:

app: app1

ports:

- protocol: TCP

port: 8080

targetPort: 8080

type: ClusterIP

#### app2-service.yaml

apiVersion: v1

kind: Servicemetadata:

name: app2-servicespec:

selector:

app: app2

ports:

- protocol: TCP

port: 8080

targetPort: 8080

type: ClusterIP

**Apply these services:**

kubectl apply -f app1-service.yaml

kubectl apply -f app2-service.yaml

## 3. Deploy NGINX Ingress Controller

### Step 4: Install NGINX Ingress Controller

Deploy the **NGINX Ingress Controller** in your Minikube cluster using the following command:

## minikube addons enable ingress

Note:- Ingress controll comes as addon on minikube

## 4. Create Ingress Resources

### Step 5: Define Ingress Rules

Create an Ingress resource to route requests to app1 and app2. It will also configure custom logging and rate limiting.

#### app-ingress.yaml

apiVersion: networking.k8s.io/v1kind: Ingressmetadata:

name: app-ingress

annotations:

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

nginx.ingress.kubernetes.io/limit-rps: "4"

nginx.ingress.kubernetes.io/limit-burst-multiplier: "1"

nginx.ingress.kubernetes.io/limit-rate-after: "0"

nginx.ingress.kubernetes.io/limit-connections: "5"

nginx.ingress.kubernetes.io/limit-client: "$remote\_addr+$http\_x\_client\_id"spec:

rules:

- host: example.local

http:

paths:

- path: /v1

pathType: Prefix

backend:

service:

name: app1-service

port:

number: 8080

- path: /v2

pathType: Prefix

backend:

service:

name: app2-service

port:

number: 8080

**Apply the Ingress resource:**

kubectl apply -f app-ingress.yaml

## Configure global rate limiting using nginx ingress:

kubectl -n ingress-nginx edit configmap ingress-nginx-controller

## data:

limit-rate: "10"

limit-rate-after: "0"

limit-req-key: $binary\_remote\_addr$http\_x\_client\_id

limit-req-status-code: "429"

limit-req-zone: req\_limit\_per\_ip\_client\_id

limit-req-zone-size: 10m

apply changes  
kubectl rollout restart deployment ingress-nginx-controller -n ingress-nginx

## 6. Testing the Setup

### Step 7: Verify Routing and Rate Limiting

1. **Test the** /v1 **route** (should go to app1):

curl http://example.local/v1

Expected response:

Hello, world!

Version: 1.0.0

Hostname: <hostname>

1. **Test the** /v2 **route** (should go to app2):

curl http://example.local/v2

Expected response:

Hello,

world!Version: 2.0.0

Hostname: <hostname>

1. **Test a random route** (should return a 404):

curl http://example.local/random

Expected response:

404 Not Found

1. **Test rate limiting**:
   1. Send 6 requests to /v1 or /v2 and you should receive a 429 Too Many Requests response after the 5th request.

curl http://example.local/v1

curl http://example.local/v1

curl http://example.local/v1

curl http://example.local/v1

curl http://example.local/v1

curl http://example.local/v1 # This should return 429

## 6. Logs and Custom Logging

To check logs for custom headers, use the following command:

kubectl logs -n ingress-nginx <nginx-ingress-pod-name>

This will show logs with the custom header X-Client-Id as part of the log entries.