# NGINX on Kubernetes with CI/CD

## Project Overview

This project demonstrates a full CI/CD pipeline for deploying an NGINX web application to a self-hosted Kubernetes cluster, using GitHub Actions, Docker, Ingress, autoscaling, and readiness/liveness probes.

## ✅ Completed Features

| Step | Task | Status |
| --- | --- | --- |
| 1 | Create GitHub repository | ✅ Done |
| 2 | Register GitHub Actions self-hosted runner | ✅ Done |
| 3 | Deploy self-hosted Kubernetes cluster with Minikube | ✅ Done |
| 4 | Create CI/CD GitHub Actions workflow for Docker build & push | ✅ Done |
| 5 | Deploy NGINX Ingress controller | ✅ Done |
| 6 | Deploy NGINX image via CI/CD + Expose with Ingress (TLS/SSL) | ✅ Done |
| 7 | Configure readinessProbe | ✅ Done |
| 8 | Configure livenessProbe | ✅ Done |
| 9 | Configure HPA (Horizontal Pod Autoscaler) based on CPU | ✅ Done |

## 📁 Directory Structure

nginx-k8s-lab/  
├── .github/workflows/docker-build.yml # CI workflow to build & push image  
├── Dockerfile # Dockerfile with static index.html  
├── index.html # Hello World HTML  
├── k8s/  
│ ├── nginx-deployment.yaml # NGINX Deployment + HPA + Probes  
│ ├── nginx-service.yaml # ClusterIP Service  
│ └── ingress.yaml # Ingress with SSL (uses correct ingressClassName)  
├── tls/ # Self-signed TLS cert & key  
│ ├── tls.crt  
│ └── tls.key  
└── README.md

## ⚙️ Prerequisites

* GitHub account
* DockerHub account
* Self-hosted Linux machine (used Minikube on openSUSE SLES 15 SP6)
* Kubernetes CLI (kubectl)
* Helm
* Docker

### 🛠️ 1. Clone Repo

git clone git@github.com:vank1chaa/nginx-k8s-lab.git  
cd nginx-k8s-lab

### 🐳 2. Configure Docker & GitHub Secrets

Create GitHub repo secrets:

* DOCKER\_USERNAME: your Docker Hub username
* DOCKER\_PASSWORD: your Docker Hub token/password

These are used in the GitHub Actions workflow.

### ⚙️ 3. Register GitHub Runner

Download and configure self-hosted runner:

# From your GitHub repo > Settings > Actions > Runners  
./config.sh --url https://github.com/vank1chaa/nginx-k8s-lab --token <TOKEN>  
./run.sh

### 🧱 4. Deploy Minikube Cluster

minikube start --driver=docker

### 🔁 5. Enable Ingress Addon (Minikube built-in)

**Important:** Do **not** use both Helm and the Minikube addon. This project uses the **Minikube ingress addon**, which avoids Helm-related TLS issues.

minikube addons enable ingress

### 🧪 6. Deploy NGINX Application

kubectl apply -f k8s/

### 🔐 7. Create TLS Certificate and Secret

mkdir -p tls  
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \  
 -keyout tls/tls.key \  
 -out tls/tls.crt \  
 -subj "/CN=nginx.test" \  
 -addext "subjectAltName=DNS:nginx.test"  
  
kubectl delete secret nginx-tls --ignore-not-found  
kubectl create secret tls nginx-tls \  
 --cert=./tls/tls.crt \  
 --key=./tls/tls.key

### 📥 8. Update /etc/hosts to Access Ingress

**Reminder:** Ensure Minikube is running before executing this step, as the IP address depends on the active Minikube instance.

sudo bash -c "echo \"$(minikube ip) nginx.test\" >> /etc/hosts"

### 📈 9. Enable Metrics Server

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml  
  
kubectl patch deployment metrics-server -n kube-system \  
 --type=json \  
 -p='[{"op": "add", "path": "/spec/template/spec/containers/0/args/-", "value": "--kubelet-insecure-tls"}]'  
  
# Wait a few seconds, then verify:  
kubectl top nodes  
kubectl top pods

### 📊 10. Apply HPA

kubectl autoscale deployment nginx-deployment \  
 --cpu-percent=50 \  
 --min=2 \  
 --max=5  
  
kubectl get hpa

## 🧪 Testing

* Access service via curl -k https://nginx.test
* Apply load with hey or ab to trigger HPA
* hey -n 100 -c 10 <https://nginx.test>
* for i in {1..10}; do curl -s -k https://nginx.test | grep "Hello"; done

## 📝 Notes

* You can optionally integrate SonarCloud in .github/workflows/ as step 10
* TLS is self-signed, acceptable for demo purposes
* Uses GitHub Actions + DockerHub + Minikube + Kubernetes
* Ingress must be enabled using minikube addons enable ingress (not Helm)
* Ingress support requires the correct ingressClassName and networking.k8s.io/v1 schema.