**Setting up a Kubernetes environment for a web application deployment. The web application is a simple application (e.g., Node.js or Python Flask) that serves web content. The application needs to be deployed in a highly available and scalable manner, with the ability to handle varying levels of traffic and ensure reliable operation.**

1. Launch Ubuntu EC2 instance
2. Host a domain name in R53
3. Install required tools (awscli, kubectl, kops, terraform, docker)

|  |
| --- |
| # Update system |
| sudo apt update && sudo apt upgrade -y |
| # Install AWS CLI |
| sudo apt install awscli -y |
| # Install kubectl |
| curl -LO "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" |
| chmod +x kubectl |
| sudo mv kubectl /usr/local/bin/ |
| # Install kops |
| curl -LO https://github.com/kubernetes/kops/releases/download/v1.28.0/kops-linux-amd64 |
| chmod +x kops-linux-amd64 |
| sudo mv kops-linux-amd64 /usr/local/bin/kops |
| # Install Terraform |
| sudo apt install unzip -y |
| wget https://releases.hashicorp.com/terraform/1.8.0/terraform\_1.8.0\_linux\_amd64.zip |
| unzip terraform\_1.8.0\_linux\_amd64.zip |
| sudo mv terraform /usr/local/bin/  #Install docker  sudo apt install docker.io -y |

1. Verify whether installation is successful

|  |
| --- |
| aws --version |
| kubectl version --client |
| kops version |
| terraform version  docker --version |

5. Folder structure and terraform scripts

kops-cluster/

├── main.tf

├── variables.tf

├── outputs.tf

✏️ variables.tf

|  |
| --- |
| variable "region" { default = "us-east-1" } |
| variable "bucket\_name" { default = "kops-state-miyasa-shop-123456" } |

✏️ main.tf

|  |
| --- |
| provider "aws" { |
| region = var.region |
| } |
|  |
| resource "aws\_s3\_bucket" "kops" { |
| bucket = var.bucket\_name |
| acl = "private" |
| } |
|  |
| resource "aws\_s3\_bucket\_versioning" "versioning" { |
| bucket = aws\_s3\_bucket.kops.id |
| versioning\_configuration { |
| status = "Enabled" |
| } |
| } |

✏️ outputs.tf

|  |
| --- |
| output "bucket\_name" { |
| value = aws\_s3\_bucket.kops.bucket |
| } |

🔨 Run Terraform

1. Terraform init

2. Terraform apply -auto-approve

1. Set Env Vars and Create Cluster with Kops

|  |
| --- |
| export KOPS\_CLUSTER\_NAME=cluster.miyasa.shop |
| export KOPS\_STATE\_STORE=s3://kops-state-miyasa-shop-123456 |
| Create the cluster: |
| kops create cluster \ |
| --cloud=aws \ |
| --zones=us-east-1a,us-east-1b \ |
| --name=${KOPS\_CLUSTER\_NAME} \ |
| --state=${KOPS\_STATE\_STORE} \ |
| --node-count=2 \ |
| --node-size=t3.medium \ |
| --master-size=t3.medium \ |
| --dns-zone=miyasa.shop \ |
| --yes |

1. Cluster creation will take 5 to 10 mins then

kops validate cluster

1. 🐍 Python Flask App with Docker

📁 Flask App Structure

flask-app/

├── app.py

├── Dockerfile

✏️ app.py

|  |
| --- |
| from flask import Flask |
| app = Flask(\_\_name\_\_) |
| @app.route('/') |
| def hello(): |
| return "Hello from Kubernetes!" |
| if \_\_name\_\_ == '\_\_main\_\_': |
| app.run(host='0.0.0.0', port=80) |

✏️ Dockerfile

|  |
| --- |
| FROM python:3.9-slim |
| WORKDIR /app |
| COPY app.py . |
| RUN pip install flask |
| EXPOSE 80 |
| CMD ["python", "app.py"] |

9.🐳 Build and Push Docker Image

# Log in to Docker

docker login

# Build and push image

docker build -t yourdockerhub/flask-k8s-app:latest .

docker push yourdockerhub/flask-k8s-app:latest

10.🚀 Deploy Flask App on Kubernetes

📁 flask-k8s-deploy.yaml

|  |
| --- |
| apiVersion: v1 |
| kind: Service |
| metadata: |
| name: flask-service |
| spec: |
| selector: |
| app: flask-app |
| ports: |
| - port: 80 |
| targetPort: 80 |
| type: LoadBalancer |
| --- |
| apiVersion: apps/v1 |
| kind: Deployment |
| metadata: |
| name: flask-app |
| spec: |
| replicas: 2 |
| selector: |
| matchLabels: |
| app: flask-app |
| template: |
| metadata: |
| labels: |
| app: flask-app |
| spec: |
| containers: |
| - name: flask-container |
| image: yourdockerhub/flask-k8s-app:latest |
| ports: |
| - containerPort: 80 |

11. 🔧 Apply it

kubectl apply -f flask-k8s-deploy.yaml

12. 🌍 Get LoadBalancer URL

kubectl get svc

Look for EXTERNAL-IP of flask-service. Open in your browser.

13. Configure in R53

Create a A record in a Hosted zone mark alias on and add load balancer id.

This links loadbalancer to DNS so while hitting with dns we can see the webpage.