# AWS EKS Two-Tier Flask Application Deployment Guide

# Table of Contents

# Project Overview

# Infrastructure Setup

# Application Deployment

# Security Analysis

# Verification & Testing

# Appendix: Screenshots

# This project demonstrates the deployment of a ****two-tier web application**** on ****Amazon Elastic Kubernetes Service (EKS)**** using:

**Frontend -** A **Flask-based web application** (serves as the presentation layer).

**Backend-**  A **MySQL database** (handles data storage).

**Infrastructure -** Managed Kubernetes cluster on AWS with **auto-scaling worker nodes**.

# The application allows users to: ✔ Submit messages via a web interface. ✔ Store messages in a MySQL database. ✔ Retrieve and display stored message dynamically.

# 

# 

# Architecture:

# [Internet] → [ELB:80] → [Flask Pods] → [MySQL Service] → [MySQL Pod]

# ↑ ↑

# │ └── Persistent Volume

# └── EKS Worker NodesApplication Deployment

# Repository Structure

# two-tier-flask-eks/

# ├── README.md

# ├── mysql-configmap.yml

# ├── mysql-deployment.yml

# ├── mysql-secrets.yml

# ├── mysql-svc.yml

# ├── two-tier-app-deployment.yml

# └── two-tier-app-svc.yml

# STEP-BY-STEP Guide:-

# 1. Access Key Creation

# The user created AWS access keys despite AWS recommending alternatives for better security

# The use case was for an "Application running on an AWS compute service"

# Best practices were shown but bypassed, including:

# Using temporary credentials instead of long-term access keys

# Enabling least-privilege permissions

# Regular key rotation.

# 

# 

# Created Access Key -

# 2. EC2 Instance Setup

# Launched an Amazon Linux 2023 AMI (Free tier eligible)

# Instance type: t2.micro (later changed to t2.medium for EKS)

# Created a new security group

# 8GB storage volume

# 

# 3. Software Installation

# Installed Git on the EC2 instance

# Configured AWS CLI with the access keys created earlier

# : The secret access key is visible in plain text in the screenshot

# 

# 4. EKS Cluster Creation

# Created cluster named "test-cluster" in ap-south-1 region

# Used eksctl version 0.207.0

# Cluster specifications:

# Node type: t2.medium

# Number of nodes: 2

# Kubernetes version: 1.32

# Standard support until March 21, 2026

# Subnets were automatically configured across two availability zones.

# 

# 5. Application Deployment

# The user appears to be working with a two-tier Flask application

# Kubernetes manifests include:

# MySQL configuration (configmaps, secrets, deployment, service)

# Flask app deployment and service

# The Docker image used is "harshj2003/flaskapp"

# Security Concerns

# **Access Key Exposure**: The secret access key is visible in plain text in 11.JPG

# **Long-term Credentials**: Used access keys instead of recommended temporary credentials or IAM roles

# **Key Management**: No evidence of key rotation or strict permission policies

# 

# 

# 

# Commands:

# Deployment Commands

# # 1. Clone repo

# git clone https://github.com/harshwardhanjadhav/two-tier-flask-eks.git

# # 2. Configure secrets

# echo -n "yourpassword" | base64 > mysql-secrets.yml

# # 3. Deploy MySQL

# kubectl apply -f mysql-\*.yml

# # 4. Deploy Flask

# kubectl apply -f two-tier-app-\*.yml

# 4. Security Analysis

# Critical Findings

# - Exposed AWS\_ACCESS\_KEY in terminal history

# - MySQL password "harsh" in base64 (not encrypted)

# 5. Verification

# Check Services

# kubectl get svc

# Output:

# NAME TYPE CLUSTER-IP EXTERNAL-IP

# two-tier-app-service LoadBalancer 10.100.60.131 a4f3d...amazonaws.com

# mysql ClusterIP 10.100.198.16 <none>

# Test Database

# sql

# mysql> SELECT \* FROM messages;

# +----+------------------------+

# | id | message |

# +----+------------------------+

# | 1 | Hello from Harsh! |

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

# 6. Appendix: Cheat Sheet

# Essential Commands

# # Cluster management

# eksctl get cluster

# kubectl get nodes -o wide

# # Troubleshooting

# kubectl logs -f <pod-name>

# kubectl describe pod <pod-name>

# YAML Samples

# mysql-secrets.yml:

# apiVersion: v1

# kind: Secret

# metadata:

# name: mysql-secret

# type: Opaque

# data:

# password: aGZyc2g= # "harsh" in base64

# ****Key Takeaways****

# ✅ ****How to deploy a scalable web app on AWS EKS.**** ✅ ****Best practices for managing secrets & configurations.**** ✅ ****Security risks & mitigation strategies in cloud deployments.**** ✅ ****End-to-end workflow from cluster setup to app deployment.****

# The workflow shows a complete setup from infrastructure provisioning to application deployment, but security practices could be improved.