

# AWS EKS Two-Tier Flask Application Deployment Guide

## - Table of Contents

1. Project Overview
2. Infrastructure Setup
3. Application Deployment
4. Security Analysis
5. Verification & Testing
6. Appendix: Screenshots

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

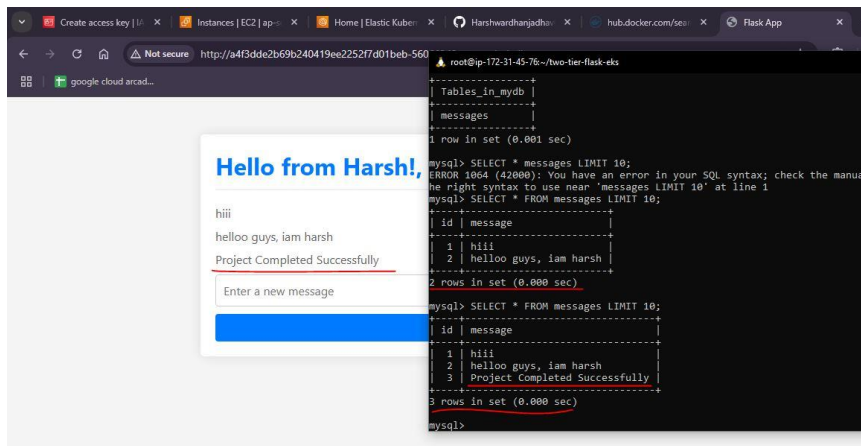
**Frontend** - A Flask-based web application (serves as the presentation layer).

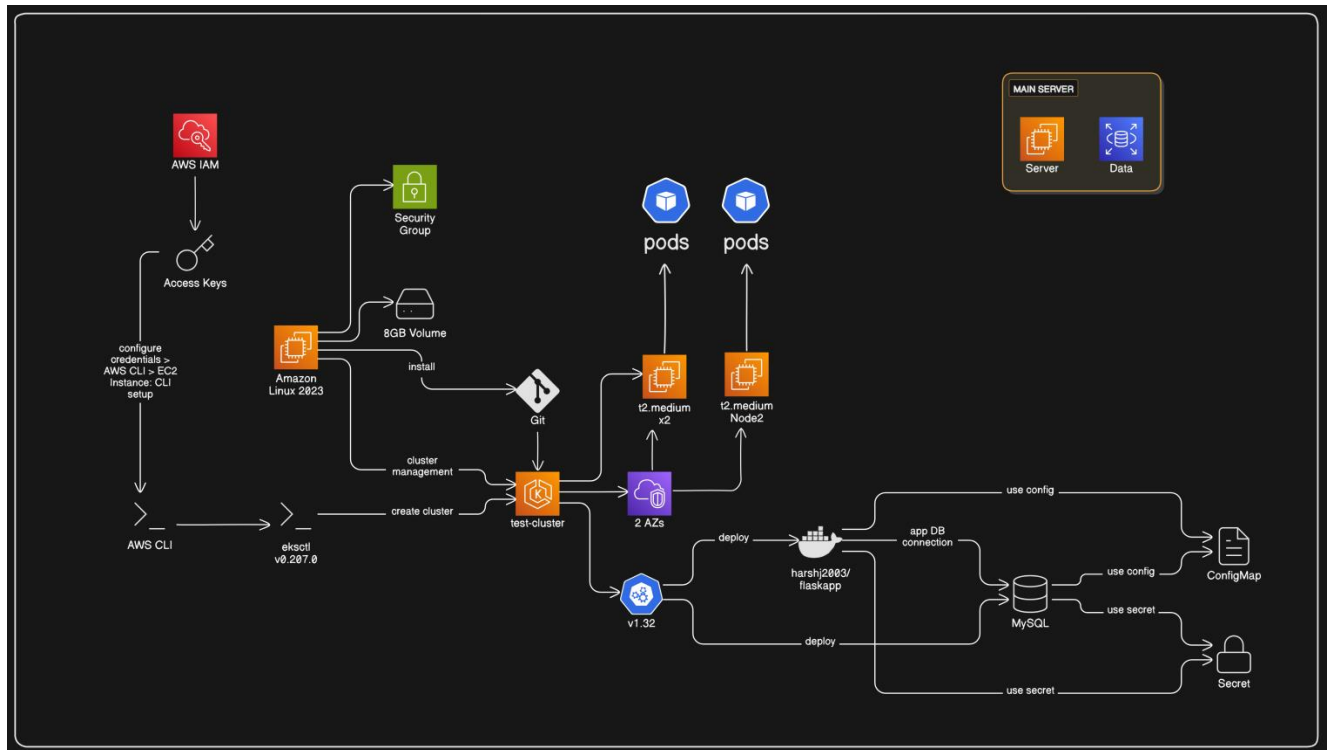
**Backend** - A MySQL database (handles data storage).

**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.**

VPC

EC2

Elastic Kubernetes Service

IAM

Users

Create user

Step 1

Specify user details

Step 2

Set permissions

Step 3

Review and create

Specify user details

User details

User name

eks-adminj

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

☐ Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel

Next

CloudShell

Feedback

VPC

EC2

Elastic Kubernetes Service

IAM

Users

Create user

Step 1

Specify user details

Step 2

Set permissions

Step 3

Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1346)

Choose one or more policies to attach to your new user.

Search

Filter by Type

All types

< 1 2 3 4 5 6 7 ... 68 >

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	0
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	0
<input type="checkbox"/>	AdministratorAccess-AWS-Elastic-Beanshell	AWS managed	0

CloudShell

Feedback

VPC

EC2

Elastic Kubernetes Service

IAM

Users

Create user

Step 1

Specify user details

Step 2

Set permissions

Step 3

Review and create

VPC

EC2

Elastic Kubernetes Service

IAM

Users

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

Access reports

Users (1)

Info

Delete

Create user

Search

User name

Path

Group

Last activity

MFA

Password age

Console last sign-in

Access key ID

eks-admin

/

0

CloudShell

Feedback

VPC

EC2

Elastic Kubernetes Service

IAM

Users

eks-admin

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

Access reports

eks-admin

Info

Delete

Summary

ARN

arn:aws:iam::197693987718:user/eks-admin

Console access

Disabled

Access key 1

Create access key

Created

May 01, 2025, 11:16 (UTC+05:30)

Last console sign-in

-

Permissions

Groups

Tags

Security credentials

Last Accessed

Console sign-in

Console sign-in link

https://197693987718.signin.aws.amazon.com/console

Console password

Not enabled

Enable console access

Multi-factor authentication (MFA) (0)

Remove

Resync

Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type

Identifier

Certifications

Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment.

CloudShell

Feedback

VPC

EC2

Elastic Kubernetes Service

IAM

Users

eks-admin

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

Access reports

Console sign-in

Console sign-in link

https://197693987718.signin.aws.amazon.com/console

Console password

Not enabled

Enable console access

Multi-factor authentication (MFA) (0)

Remove

Resync

Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type

Identifier

Certifications

Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment.

Access keys (0)

Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

Create access key

SSH public keys for AWS CodeCommit (0)

Actions

Upload SSH public key

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

The image displays two screenshots of the AWS Management Console's 'Launch an instance' wizard for an EC2 instance.

**Top Screenshot:** Shows the initial setup. The 'Name' field is 'eks-controller'. Under 'Application and OS Images (Amazon Machine Image)', 'Amazon Linux 2023 AMI' is selected. The 'Virtual server type (instance type)' is 't2.micro'. The 'Firewall (security group)' is 'New security group'. The 'Storage (volumes)' section shows '1 volume(s) - 8 GiB'. The 'Launch instance' button is visible.

**Bottom Screenshot:** Shows the 'Instance type' section where 't2.medium' is selected. The 'Key pair (login)' section shows 'Create new key pair' with a red box around the button. The 'Summary' section on the right shows the instance type changed to 't2.medium' and a 'Free tier' notification: 'Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when...'. The 'Launch instance' button is visible.

### 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

```
root@ip-172-31-45-76:~  
Installing      : git-2.47.1-1.amzn2023.0.2.x86_64  
Running scriptlet: git-2.47.1-1.amzn2023.0.2.x86_64  
Verifying      : git-2.47.1-1.amzn2023.0.2.x86_64  
Verifying      : git-core-2.47.1-1.amzn2023.0.2.x86_64  
Verifying      : git-core-doc-2.47.1-1.amzn2023.0.2.noarch  
Verifying      : perl-Error-1:0.17029-5.amzn2023.0.2.noarch  
Verifying      : perl-File-Find-1.37-477.amzn2023.0.6.noarch  
Verifying      : perl-Git-2.47.1-1.amzn2023.0.2.noarch  
Verifying      : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64  
Verifying      : perl-lib-0.65-477.amzn2023.0.6.x86_64  
  
Installed:  
git-2.47.1-1.amzn2023.0.2.x86_64          git-core-2.47.1-1.amzn2023.0.2.x86_64  
git-core-doc-2.47.1-1.amzn2023.0.2.noarch perl-Error-1:0.17029-5.amzn2023.0.2.noarch  
perl-File-Find-1.37-477.amzn2023.0.6.noarch perl-Git-2.47.1-1.amzn2023.0.2.noarch  
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 perl-lib-0.65-477.amzn2023.0.6.x86_64  
  
Complete!  
[root@ip-172-31-45-76 ~]#  
[root@ip-172-31-45-76 ~]# ls  
aws  awscli2.zip  
[root@ip-172-31-45-76 ~]# aws configure  
AWS Access Key ID [None]: AKIAS4B323ODF6WQVT3Z  
AWS Secret Access Key [None]: AKIAS4B323ODF6WQVT3Z  
Default region name [None]: ap-south-1  
Default output format [None]:  
[root@ip-172-31-45-76 ~]# eksctl  
The official CLI for Amazon EKS  
  
Usage: eksctl [command] [flags]
```

### 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.



```
root@ip-172-31-45-76:~  
[root@ip-172-31-45-76 ~]# eksctl create cluster --name test-cluster --region ap-south-1 --node-type t2.medium --nodes-max 2  
2025-05-01 09:21:56 [i] eksctl version 0.207.0  
2025-05-01 09:21:56 [i] using region ap-south-1  
2025-05-01 09:21:56 [i] skipping ap-south-1c from selection because it doesn't support the following instance type(s)  
t2.medium  
2025-05-01 09:21:56 [i] setting availability zones to [ap-south-1a ap-south-1b]  
2025-05-01 09:21:56 [i] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.64.0/19  
2025-05-01 09:21:56 [i] subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.96.0/19  
2025-05-01 09:21:56 [i] nodegroup "ng-cf468eb2" will use "" [AmazonLinux2/1.32]  
2025-05-01 09:21:56 [i] using Kubernetes version 1.32  
2025-05-01 09:21:56 [i] creating EKS cluster "test-cluster" in "ap-south-1" region with managed nodes  
2025-05-01 09:21:56 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup  
2025-05-01 09:21:56 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stack --region=ap-south-1 --cluster=test-cluster'  
2025-05-01 09:21:56 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "test-cluster" in "ap-south-1"  
2025-05-01 09:21:56 [i] CloudWatch logging will not be enabled for cluster "test-cluster" in "ap-south-1"  
2025-05-01 09:21:56 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=test-cluster'  
2025-05-01 09:21:56 [i] default addons kube-proxy, coredns, metrics-server, vpc-cni were not specified, will install them as EKS addons  
2025-05-01 09:21:56 [i]  
2 sequential tasks: { create cluster control plane "test-cluster",  
  2 sequential sub-tasks: {  
    2 sequential sub-tasks: {  
      1 task: { create addons },  
      wait for control plane to become ready,  
    }  
  }  
}
```

Amazon Elastic  
Kubernetes Service

Clusters

▼ Settings  
Console settings




▼ Amazon EKS Anywhere  
Enterprise Subscriptions

▼ Related services  
Amazon ECR  
AWS Batch

Documentation

Clusters (1) Info

Filter clusters

Cluster name ▲	Status ▼	Kubernetes version ▼	Support period ▼	Upgrade policy ▼	Created ▼
 <u>test-cluster</u>	 Creating	1.32	 Standard support until March 21, 2026	Extended	<u>5 minutes ago</u>



```

root@ip-172-31-45-76:~
er `addon.PodIdentityAssociations`, and run `eksctl update addon`
2025-05-01 09:28:59 [D] creating addon: vpc-cni
2025-05-01 09:29:00 [D] successfully created addon: vpc-cni
2025-05-01 09:31:00 [D] building managed nodegroup stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:31:00 [D] deploying stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:31:00 [D] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:31:30 [D] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:32:18 [D] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:33:32 [D] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:35:09 [D] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"
2025-05-01 09:35:09 [D] waiting for the control plane to become ready
2025-05-01 09:35:10 [D] saved kubeconfig as "/root/.kube/config"
2025-05-01 09:35:10 [D] no tasks
2025-05-01 09:35:10 [D] all EKS cluster resources for "test-cluster" have been created
2025-05-01 09:35:10 [D] nodegroup "ng-cf468eb2" has 2 node(s)
2025-05-01 09:35:10 [D] node "ip-192-168-13-247.ap-south-1.compute.internal" is ready
2025-05-01 09:35:10 [D] node "ip-192-168-51-180.ap-south-1.compute.internal" is ready
2025-05-01 09:35:10 [D] waiting for at least 2 node(s) to become ready in "ng-cf468eb2"
2025-05-01 09:35:10 [D] nodegroup "ng-cf468eb2" has 2 node(s)
2025-05-01 09:35:10 [D] node "ip-192-168-13-247.ap-south-1.compute.internal" is ready
2025-05-01 09:35:10 [D] node "ip-192-168-51-180.ap-south-1.compute.internal" is ready
2025-05-01 09:35:10 [D] created 1 managed nodegroup(s) in cluster "test-cluster"
2025-05-01 09:35:11 [D] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2025-05-01 09:35:11 [D] EKS cluster "test-cluster" in "ap-south-1" region is ready
[root@ip-172-31-45-76 ~]#
[root@ip-172-31-45-76 ~]# eksctl get cluster
NAME          REGION    EKSCTL CREATED
test-cluster  ap-south-1  True
[root@ip-172-31-45-76 ~]#

```

## 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

Harshwardhanjadhav0 / two-tier-flask-eks

Type to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

two-tier-flask-eksPublic

PinUnwatch1Fork

main1 Branch0 Tags

Go to file

Add fileCode

Harshwardhanjadhav0 Update mysql-secrets.yml

LocalCodespaces

Clone

HTTPSSSHGitHub CLI

https://github.com/Harshwardhanjadhav0/two-tier

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

README.mdfirst commit

mysql-configmap.ymlAdd MySQL and Fla

mysql-deployment.ymlAdd MySQL and Fla

mysql-secrets.ymlUpdate mysql-secre

mysql-svc.ymlAdd MySQL and Fla

two-tier-app-deployment.ymlAdd MySQL and Fla

two-tier-app-svc.ymlAdd MySQL and Flask app Kubernetes manifests14 hours ago

About

No description, website, or topic

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

root@ip-172-31-45-76:~

er `addon.PodIdentityAssociations`, and run `eksctl update addon`

2025-05-01 09:28:59 [i] creating addon: vpc-cni

2025-05-01 09:29:00 [i] successfully created addon: vpc-cni

2025-05-01 09:31:00 [i] building managed nodegroup stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:31:00 [i] deploying stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:31:00 [i] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:31:30 [i] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:32:18 [i] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:33:32 [i] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:35:09 [i] waiting for CloudFormation stack "eksctl-test-cluster-nodegroup-ng-cf468eb2"

2025-05-01 09:35:09 [i] waiting for the control plane to become ready

2025-05-01 09:35:10 [i] saved kubeconfig as "/root/.kube/config"

2025-05-01 09:35:10 [i] no tasks

2025-05-01 09:35:10 [i] all EKS cluster resources for "test-cluster" have been created

2025-05-01 09:35:10 [i] nodegroup "ng-cf468eb2" has 2 node(s)

2025-05-01 09:35:10 [i] node "ip-192-168-13-247.ap-south-1.compute.internal" is ready

2025-05-01 09:35:10 [i] node "ip-192-168-51-180.ap-south-1.compute.internal" is ready

2025-05-01 09:35:10 [i] waiting for at least 2 node(s) to become ready in "ng-cf468eb2"

2025-05-01 09:35:10 [i] nodegroup "ng-cf468eb2" has 2 node(s)

2025-05-01 09:35:10 [i] node "ip-192-168-13-247.ap-south-1.compute.internal" is ready

2025-05-01 09:35:10 [i] node "ip-192-168-51-180.ap-south-1.compute.internal" is ready

2025-05-01 09:35:10 [i] created 1 managed nodegroup(s) in cluster "test-cluster"

2025-05-01 09:35:11 [i] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'

2025-05-01 09:35:11 [i] EKS cluster "test-cluster" in "ap-south-1" region is ready

[root@ip-172-31-45-76 ~]#

[root@ip-172-31-45-76 ~]# eksctl get cluster

NAME REGION EKSTL CREATED

test-cluster ap-south-1 True

[root@ip-172-31-45-76 ~]#



```

root@ip-172-31-45-76:~/two-tier-flask-eks
[root@ip-172-31-45-76 ~]# git clone https://github.com/Harshwardhanjadhav0/two-tier-flask-eks.git
Cloning into 'two-tier-flask-eks'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 14 (delta 4), reused 9 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (4/4), done.
[root@ip-172-31-45-76 ~]# ls
aws awscli v2.zip two-tier-flask-eks
[root@ip-172-31-45-76 ~]# cd two-tier-flask-eks/
[root@ip-172-31-45-76 two-tier-flask-eks]# ls
README.md      mysql-deployment.yml  mysql-svc.yml      two-tier-app-svc.yml
mysql-configmap.yml  mysql-secrets.yml    two-tier-app-deployment.yml
[root@ip-172-31-45-76 two-tier-flask-eks]#

root@ip-172-31-45-76:~/two-tier-flask-eks
[root@ip-172-31-45-76 two-tier-flask-eks]# ls
README.md      mysql-deployment.yml  mysql-svc.yml      two-tier-app-svc.yml
mysql-configmap.yml  mysql-secrets.yml    two-tier-app-deployment.yml
[root@ip-172-31-45-76 two-tier-flask-eks]# cat mysql-secrets.yml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: Opaque
data:
  p: <password in base64>
[root@ip-172-31-45-76 two-tier-flask-eks]# echo -n "harsh" | base64
aGFyc2g=
[root@ip-172-31-45-76 two-tier-flask-eks]#

[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl apply -f mysql-secrets.yml
secret/mysql-secret unchanged
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get cm
NAME          DATA      AGE
kube-root-ca.crt  1         13m
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl apply -f mysql-configmap.yml
configmap/mysql-init-container-db-config created
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get cm
NAME          DATA      AGE
kube-root-ca.crt  1         14m
mysql-init-container-db-config  1         8s
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get secrets
NAME          TYPE      DATA      AGE
mysql-secret  Opaque    1          57s
[root@ip-172-31-45-76 two-tier-flask-eks]#

[root@ip-172-31-45-76 two-tier-flask-eks]# ls
README.md      mysql-deployment.yml  mysql-svc.yml      two-tier-app-svc.yml
mysql-configmap.yml  mysql-secrets.yml    two-tier-app-deployment.yml
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl apply -f mysql-deployment.yml -f mysql-svc.yml
deployment.apps/mysql created
service/mysql created
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
mysql-56c8c6468b-4bzlm  0/1     ContainerCreating   0          9s
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get svc
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
kubernetes    ClusterIP  10.100.0.1     <none>       443/TCP    15m
mysql         ClusterIP  10.100.198.16  <none>       3306/TCP   15s
[root@ip-172-31-45-76 two-tier-flask-eks]#
[root@ip-172-31-45-76 two-tier-flask-eks]#

```

```
root@ip-172-31-45-76: ~/two-tier-flask-eks
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl apply -f two-tier-app-deployment.yml -f two-tier-app-svc.yml
deployment.apps/two-tier-app created
service/two-tier-app-service created
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get svc
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
kubernetes                          ClusterIP           10.100.0.1       <none>
mysql                               ClusterIP           10.100.198.16    <none>
two-tier-app-service                LoadBalancer       10.100.60.131    a4f3dde2b69b240419ee2252f7d01beb-56066349.ap-south-1.elb.amazonaws.com
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-56c8c6468b-4bz1m             1/1     Running   0           2m4s
two-tier-app-667fcc5d55-rz2jj      1/1     Running   0           38s
[root@ip-172-31-45-76 two-tier-flask-eks]#
```

Create access key | IA

Instances | EC2 | ap-s

Home | Elastic Kuber

Harshwardhanjadhav

hub.docker.com/sear

Flask App

Not secure http://a4f3dde2b69b240419ee2252f7d01beb-560

google cloud arcad...

Hello from Harsh!

hiiii

helloo guys, iam harsh

Project Completed Successfully

Enter a new message

```
root@ip-172-31-45-76: ~/two-tier-flask-eks
mysql> SELECT * FROM messages LIMIT 10;
+-----+
| Tables_in_mydb |
+-----+
| messages        |
+-----+
1 row in set (0.001 sec)

mysql> SELECT * FROM messages LIMIT 10;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
he right syntax to use near 'messages LIMIT 10' at line 1
mysql> SELECT * FROM messages LIMIT 10;
+-----+
| id | message        |
+-----+
| 1 | hiiii          |
| 2 | helloo guys, iam harsh |
+-----+
2 rows in set (0.000 sec)

mysql> SELECT * FROM messages LIMIT 10;
+-----+
| id | message        |
+-----+
| 1 | hiiii          |
| 2 | helloo guys, iam harsh |
| 3 | Project Completed Successfully |
+-----+
3 rows in set (0.000 sec)

mysql>
```

**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.**