AWS EKS Two-Tier Flask Application Deployment Guide

- Table of Contents

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

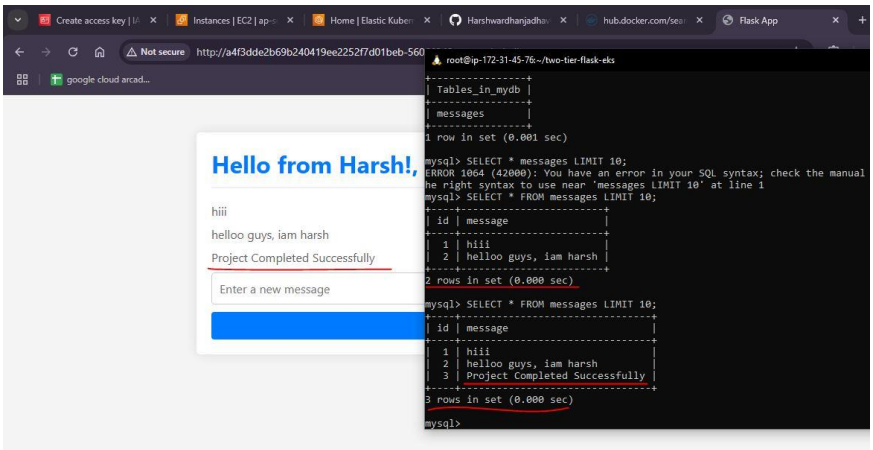
Frontend - A **Flask-based web application** (serves as the presentation layer).

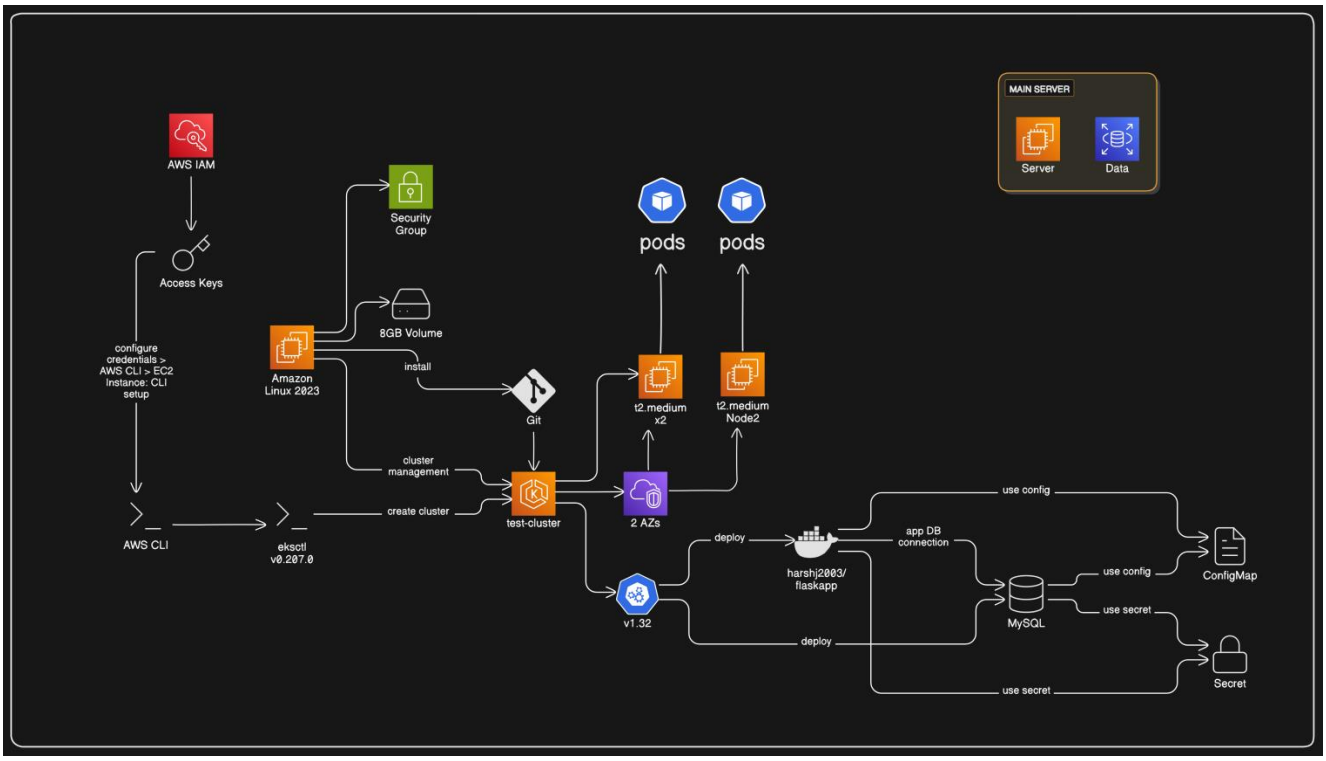
Backend- A **MySQL database** (handles data storage).

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-admin

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 © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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.

Create policy

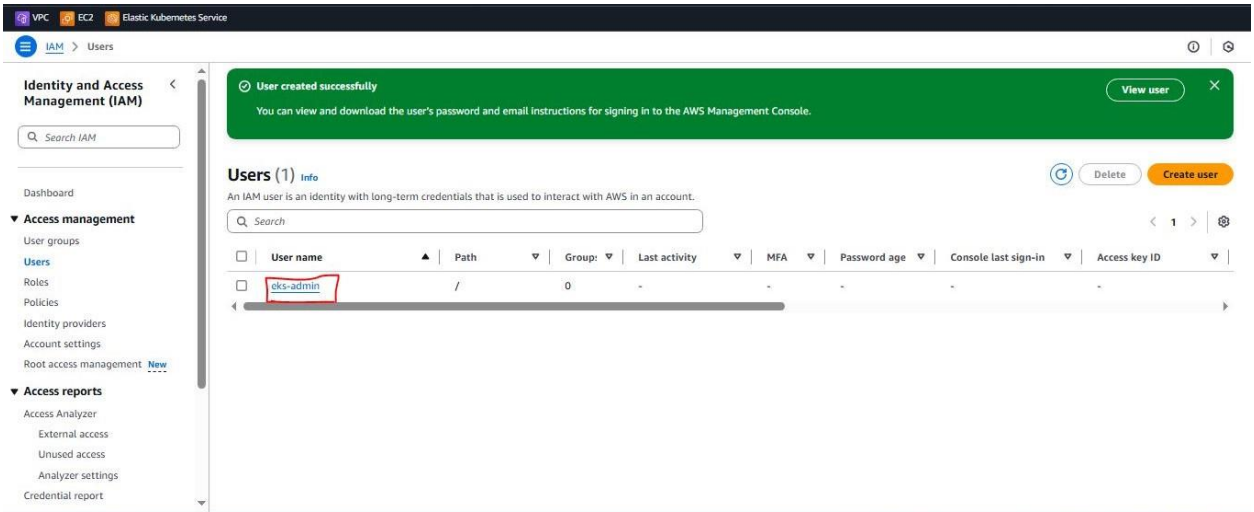
Filter by Type

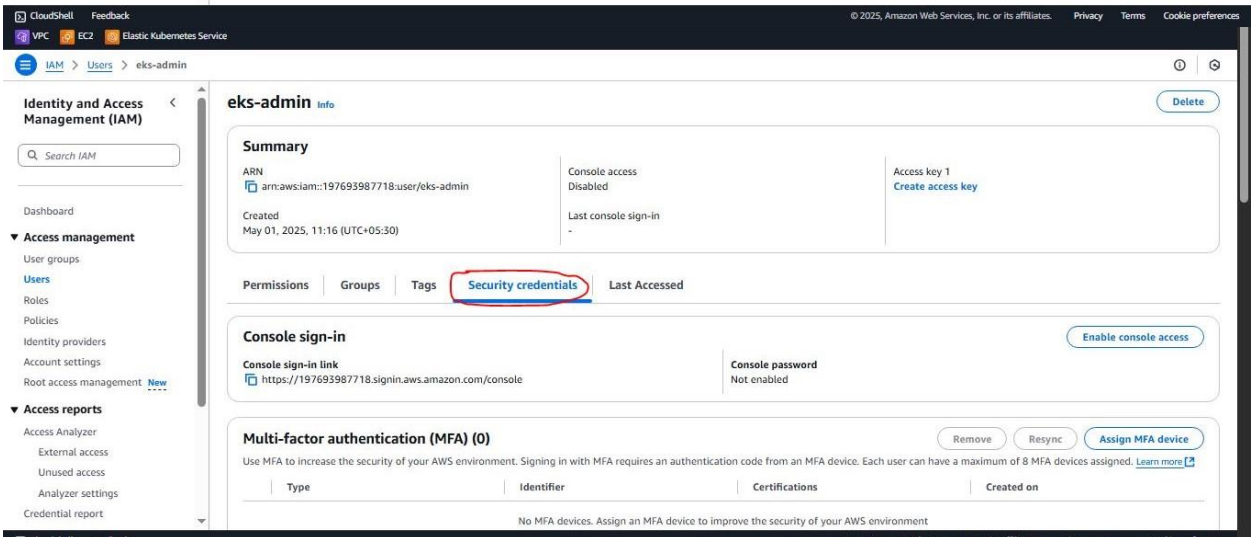
Search | All types

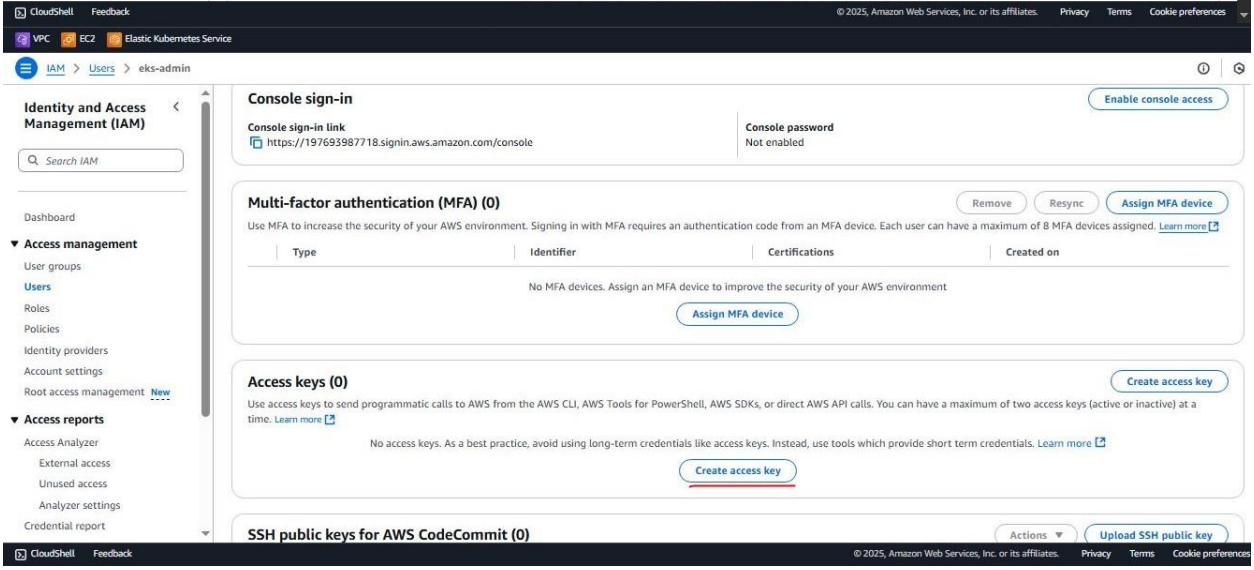
1 2 3 4 5 6 7 ... 68

Policy name	Type	Attached entities
AccessAnalyzerServiceRolePolicy	AWS managed	0
AdministratorAccess	AWS managed - job function	0
AdministratorAccess-Amplify	AWS managed	0
AdministratorAccess-AWSElasticBeanst	AWS managed	0

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



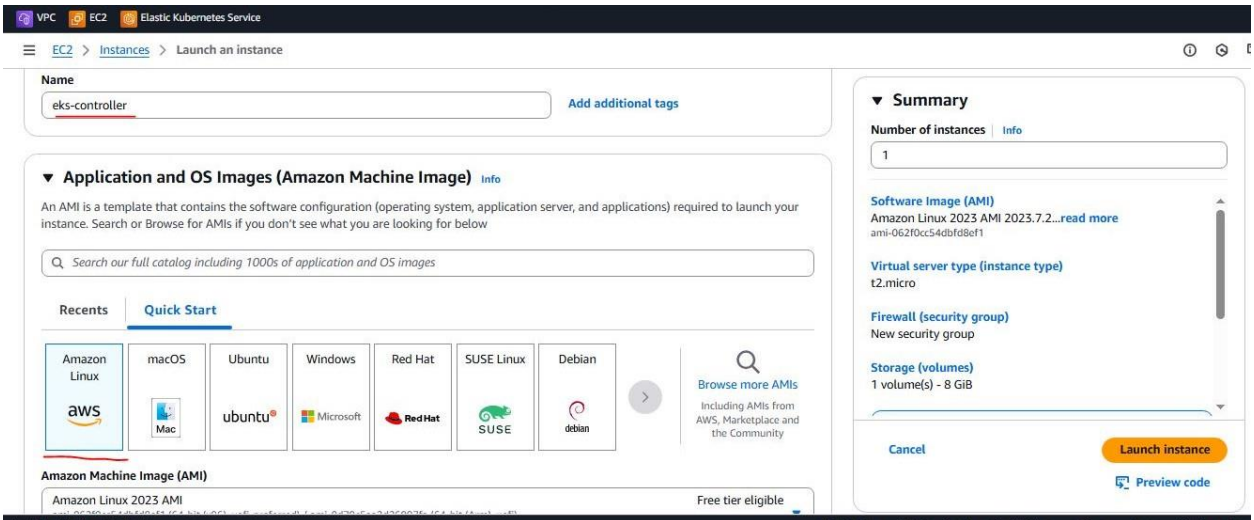


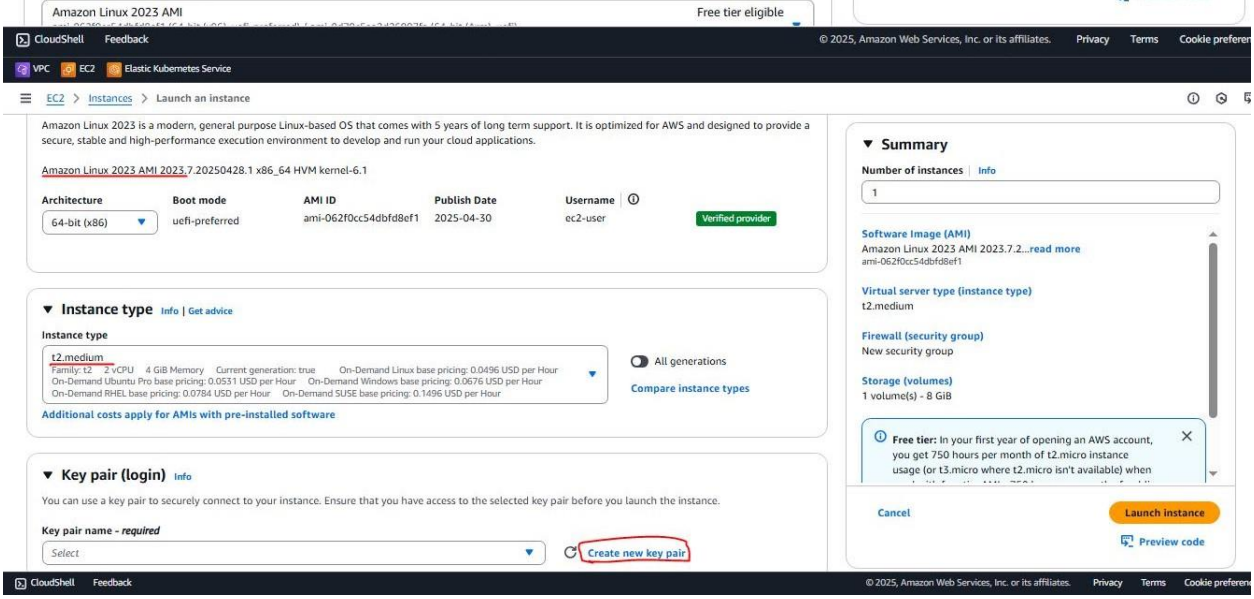


Created Access Key -

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**

```
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  awscliv2.zip
[root@ip-172-31-45-76 ~]# aws configure
AWS Access Key ID [None]: AKIAS4B323ODF6WQVT3Z
AWS Secret Access Key [None]: [redacted]
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-m
n 2 --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 nodegr
up
2025-05-01 09:21:56 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks
--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} fo
 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-T
PES-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 t
em 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,
```

VPC EC2 Elastic Kubernetes Service

Amazon Elastic Kubernetes Service

Clusters

▼ Settings

Console settings

▼ Amazon EKS Anywhere

Enterprise Subscriptions

▼ Related services

Amazon ECR

AWS Batch

Documentation

Clusters (1) Info

Delete Create

Filter clusters

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

```
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          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-eks Public

Pin | Unwatch 1 | Fork 0 | Sta

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

Harshwardhanjadhav0 Update mysql-secrets.yml

README.md	first commit
mysql-configmap.yml	Add MySQL and Fla
mysql-deployment.yml	Add MySQL and Fla
mysql-secrets.yml	Update mysql-secre
mysql-svc.yml	Add MySQL and Fla
two-tier-app-deployment.yml	Add MySQL and Fla
two-tier-app-svc.yml	Add MySQL and Flask app Kubernetes manifests 14 hours ago

Local | Codespaces

Clone

HTTPS | SSH | GitHub CLI

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

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

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          EKSCTL CREATED
test-cluster    ap-south-1      True
[root@ip-172-31-45-76 ~]#
```

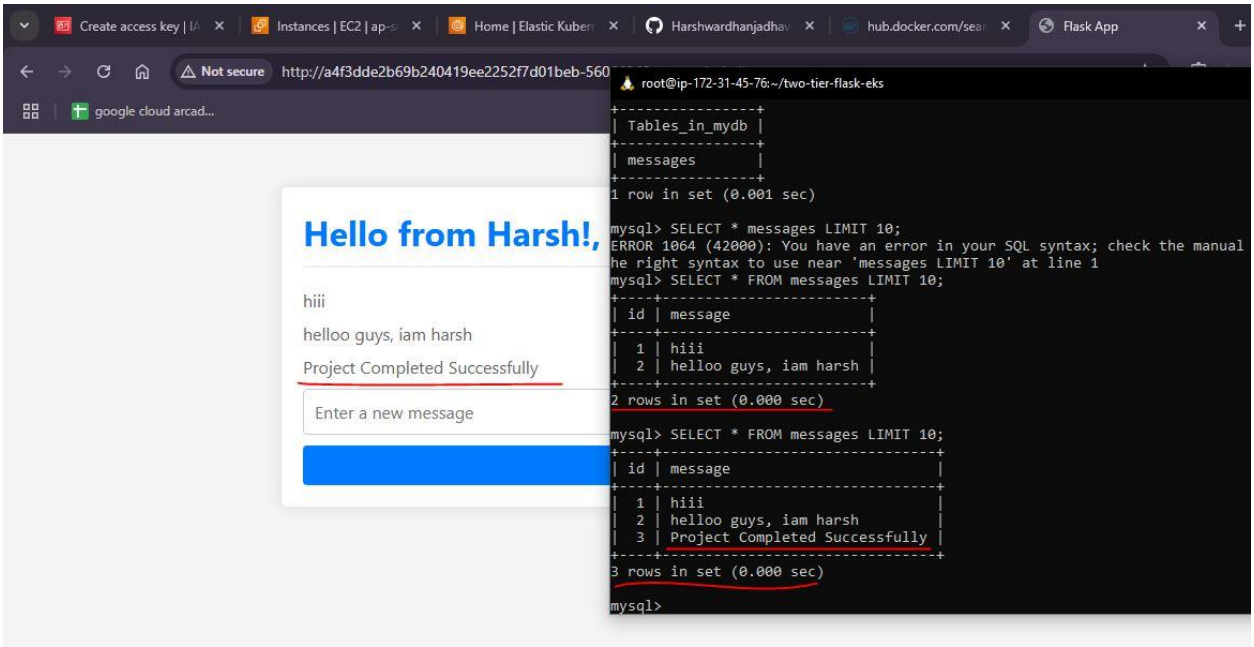
```
root@ip-172-31-45-76:~/two-tier-flask-eks
[root@ip-172-31-45-76 ~]# git clone https://github.com/Harshwardhanjadhav0/two-tier-flask-eks.git
Cloning into 'two-tier-flask-eks'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (11/11), 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  awscliv2.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
       PORT(S)        AGE
kubernetes            ClusterIP      10.100.0.1      <none>
       443/TCP        17m
mysql                 ClusterIP      10.100.198.16   <none>
       3306/TCP       105s
two-tier-app-service  LoadBalancer   10.100.60.131   a4f3dde2b69b240419ee2252f7d01beb-56066349.ap-south-1.elb.amazonaws
.com   80:30264/TCP   19s
[root@ip-172-31-45-76 two-tier-flask-eks]# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
mysql-56c8c6468b-4bzlm            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]#
```



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.