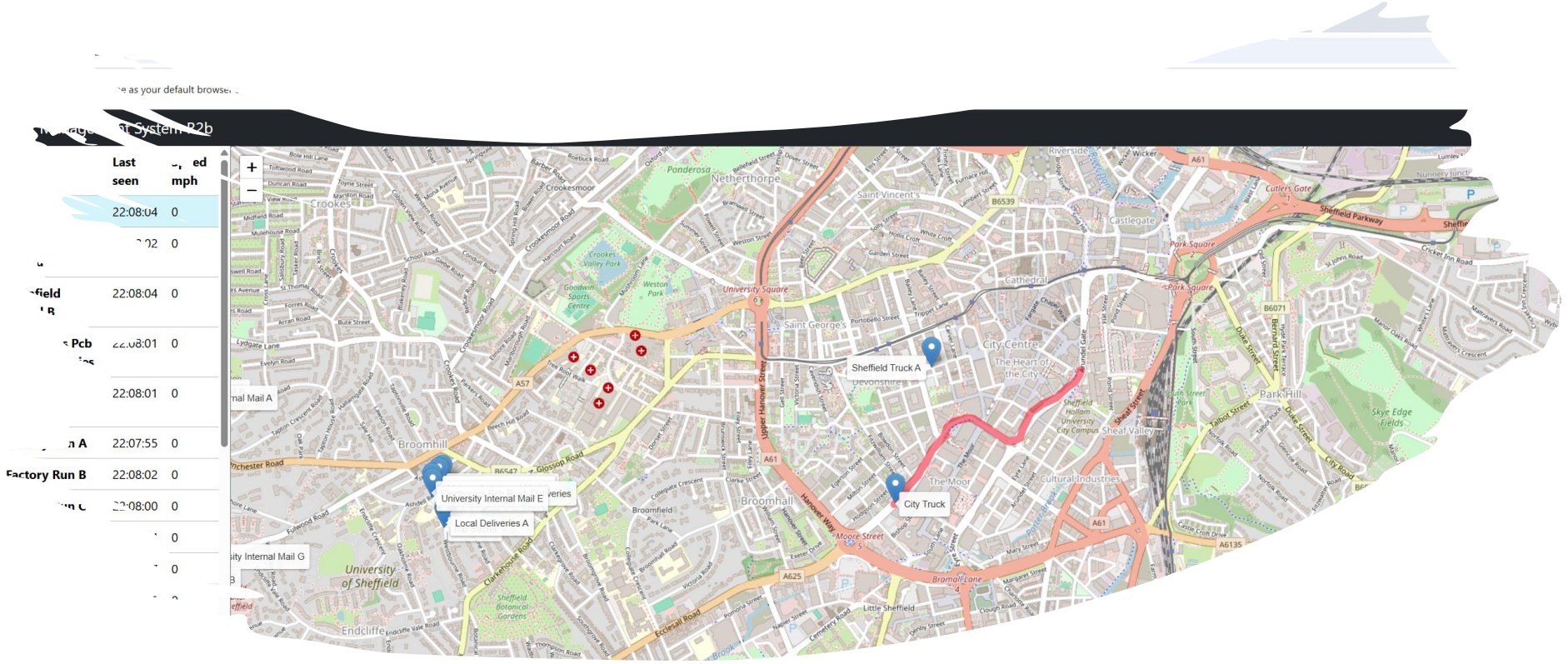


Microservices Tracking Application Deployment on AWS EKS using Vault, IAM, and Docker, Angular, Maven

Objective

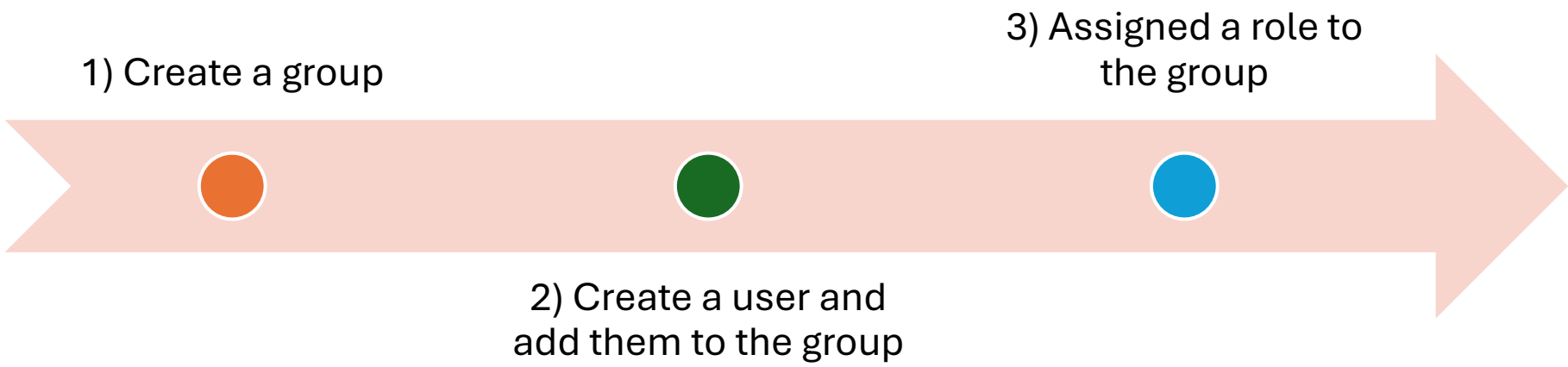
To design and deploy a microservices-based tracking application on AWS Elastic Kubernetes Service (EKS) utilizing Docker for containerization, Vault for secrets management, and IAM for access control, achieving a secure, scalable, and efficient deployment pipeline.

Tools & Technologies Used

- Cloud Platform: AWS (EKS, IAM, EC2)
- Containerization: Docker
- Orchestration: Kubernetes (EKS)
- Secret Management: HashiCorp Vault
- Version Control: Git & GitHub
- Operating System: Linux (RedHat)
- Build & Package Management: Maven
- Frontend Framework: Angular

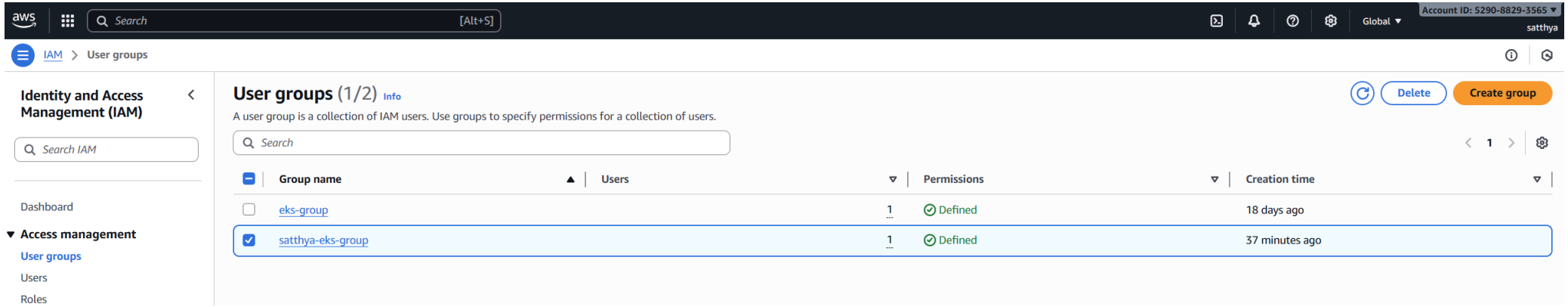
IAM

- Implemented AWS IAM roles and policies to manage access between EKS, Vault, and other AWS services, enabling secure deployment and operation of the microservices tracking application



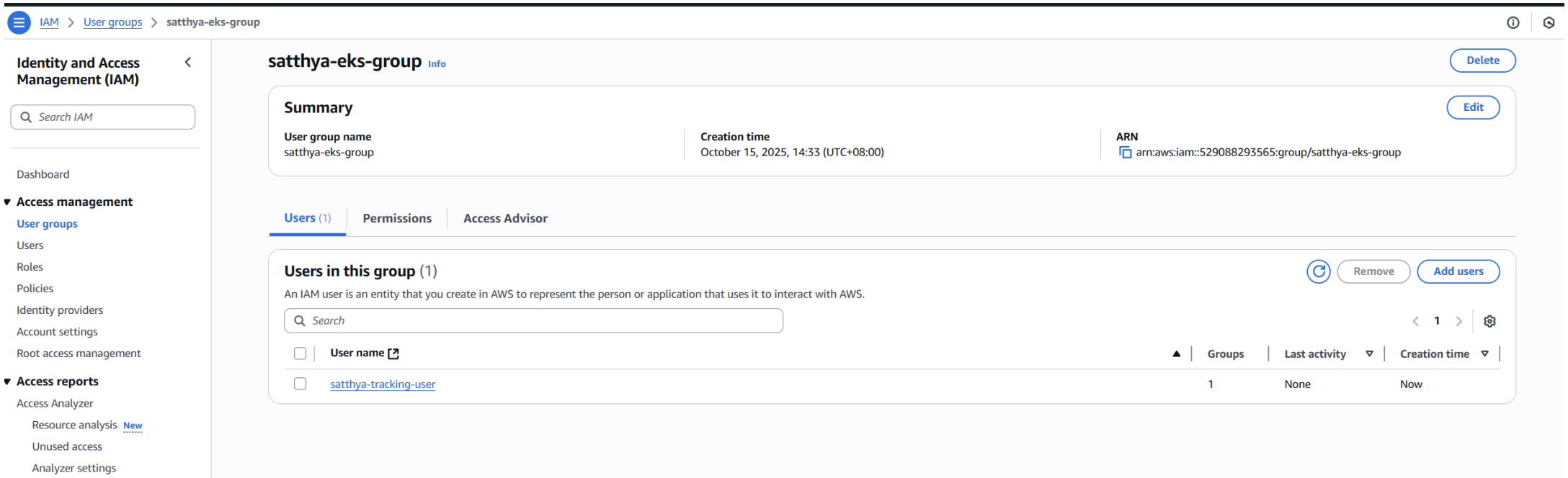
IAM

- Created a group.



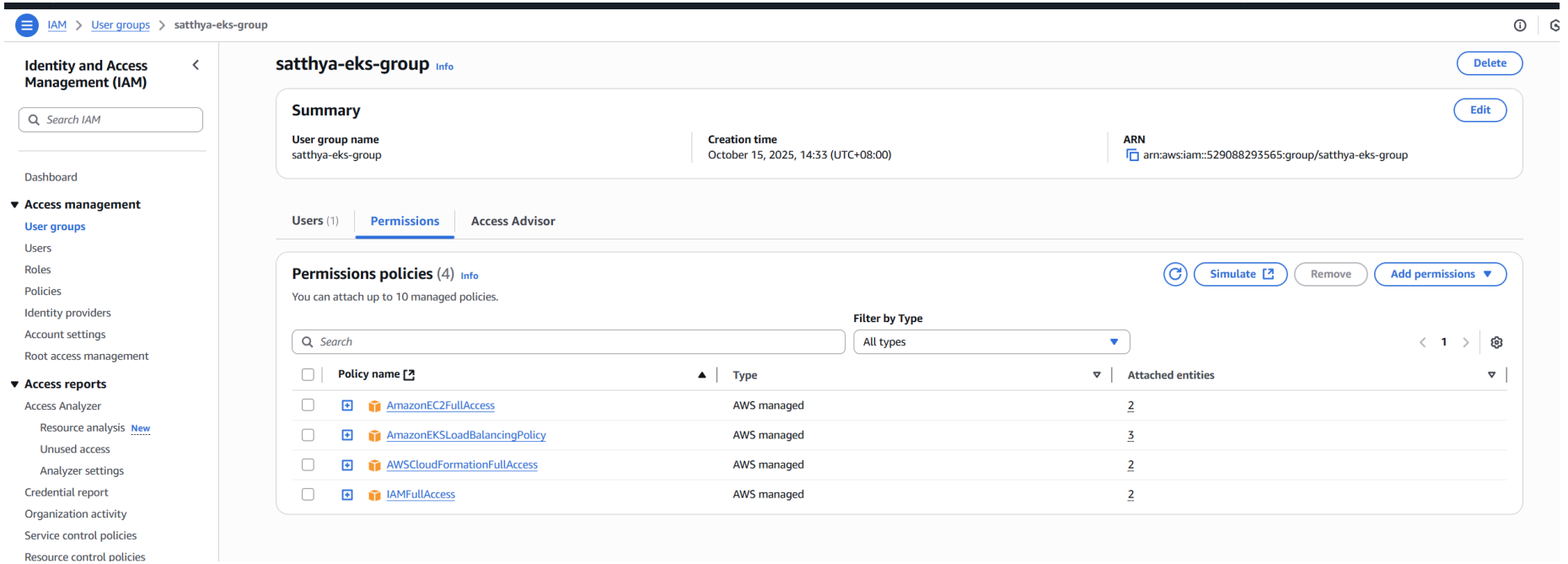
IAM

- Created a user and added them to the group.



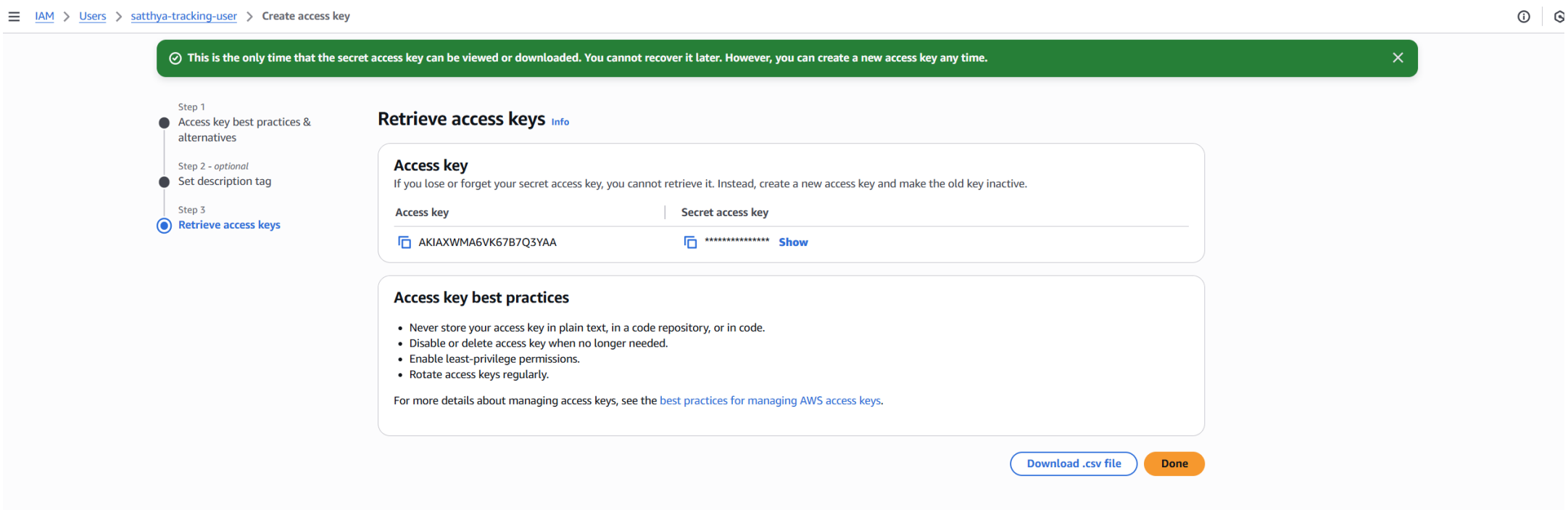
IAM

- Added a policy to the group. Any user in the group gets the same access.



IAM

- Access keys are created for the user to enable secure programmatic access, which is used for automated deployment of the application



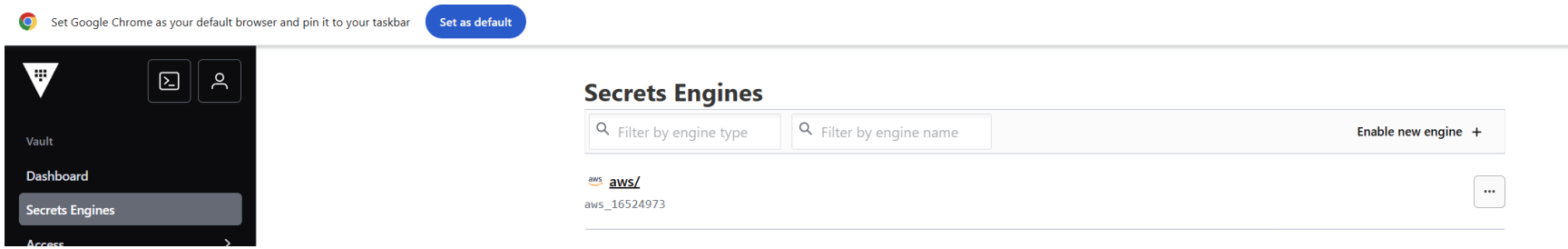
VAULT CONFIGURATION

- Store the AWS Access Key ID and Secret Access Key in Vault

  ➢ Enable the AWS secrets engine in Vault by running the command below

    ❑ vault secrets enable aws

```
ec2-user@ip-172-31-31-39:~
[ec2-user@ip-172-31-31-39 ~]$ vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
[ec2-user@ip-172-31-31-39 ~]$ vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
GitHub/       kv           kv_5dbe8771           n/a
aws/          aws          aws_b5335c4e          n/a
cubbyhole/    cubbyhole    cubbyhole_728bab33    per-token private secret storage
identity/     identity     identity_0b4377d8     identity store
secrets/      kv           kv_aa499a1c           n/a
sys/          system       system_27514caa       system endpoints used for control, policy and debugging
[ec2-user@ip-172-31-31-39 ~]$
```

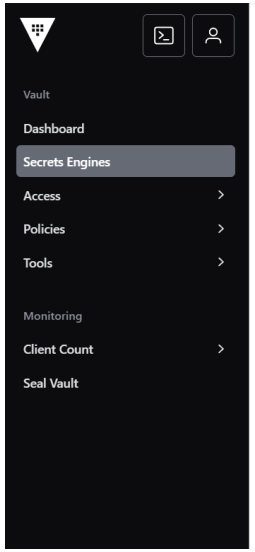


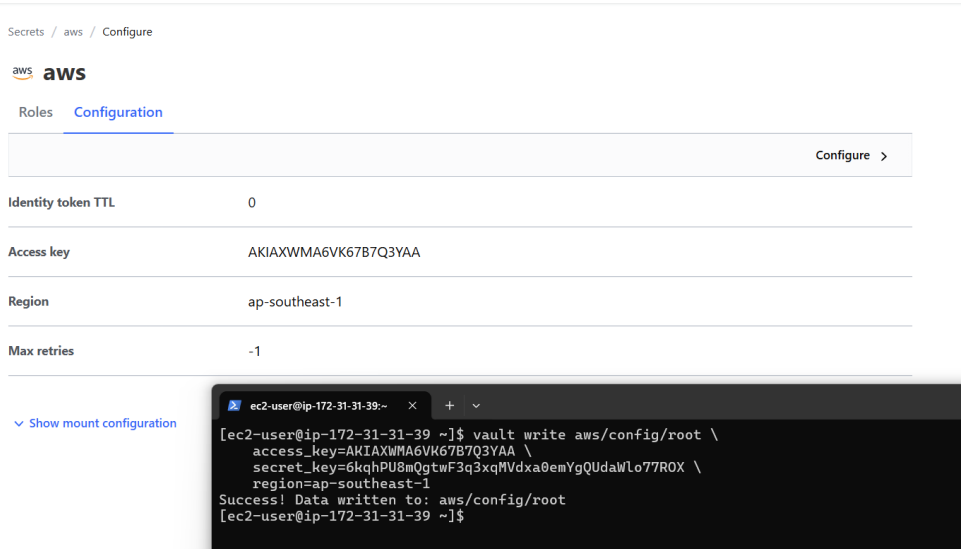
VAULT CONFIGURATION

- Store the AWS Access Key ID and Secret Access Key in Vault

  ➢ Configure the AWS secrets engine in Vault.

```
vault write aws/config/root \
    access_key=NLNA \
    secret_key=R4nm063hg \
    region=ap-southeast-1
```



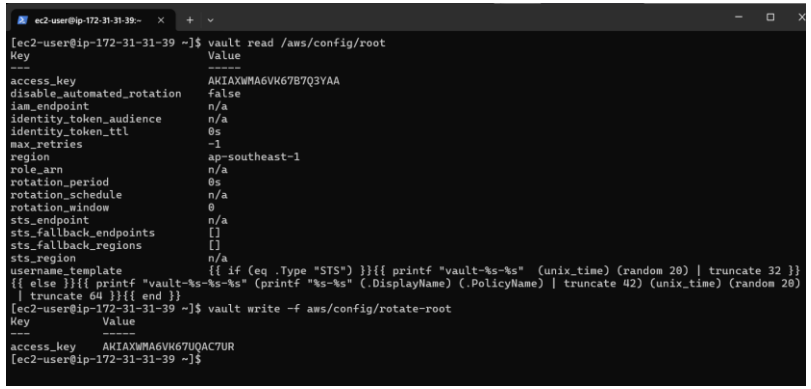


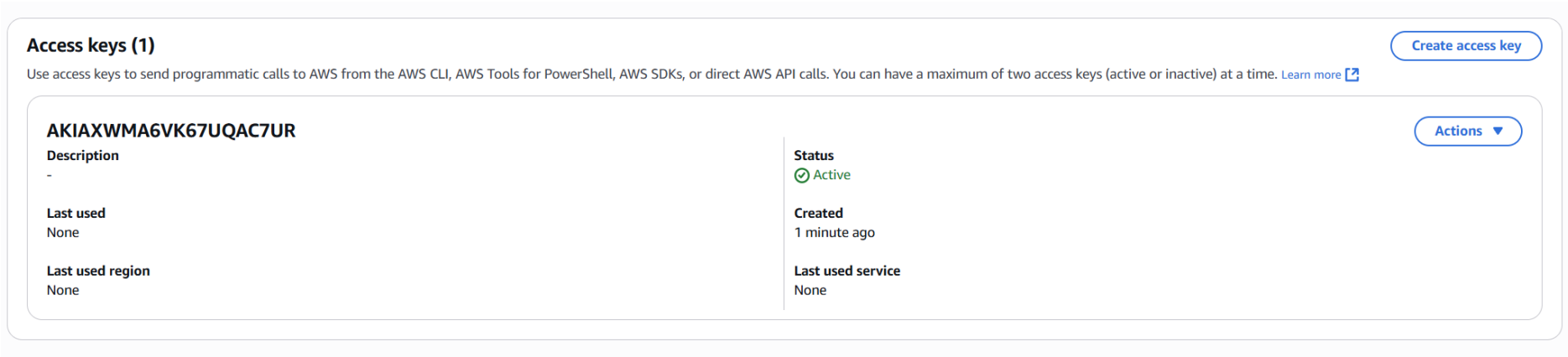
VAULT CONFIGURATION

- Store the AWS Access Key ID and Secret Access Key in Vault

  - ➢ Rotate AWS Root Credentials in Vault.

    - ❑ vault write –f aws/config/rotate-root



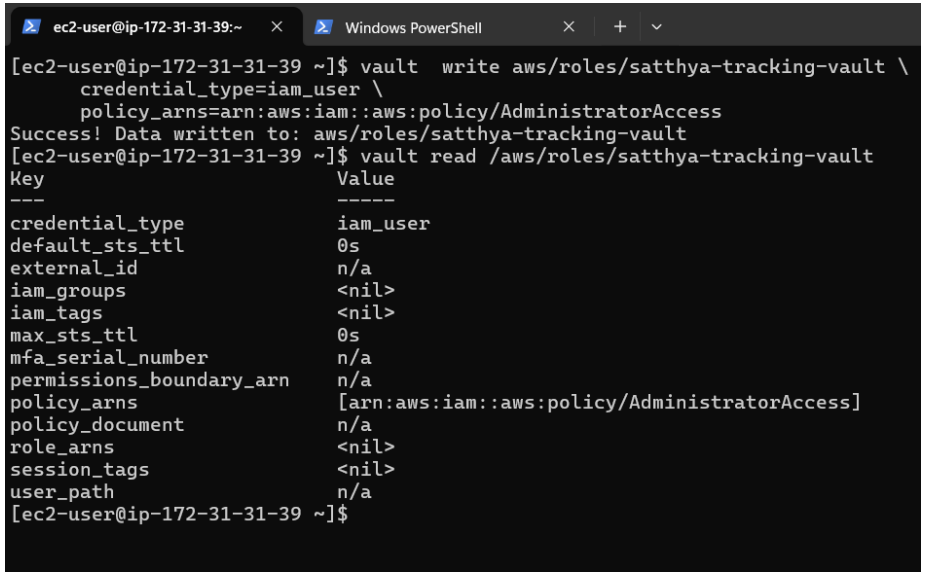


VAULT CONFIGURATION

- Store the AWS Access Key ID and Secret Access Key in Vault

  - Vault Role Setup for AWS Access.

```
vault  write aws/roles/satthya-tracking-vault \
    credential_type=iam_user \
    policy_arns=arn:aws:iam::aws:policy/AdministratorAccess
```

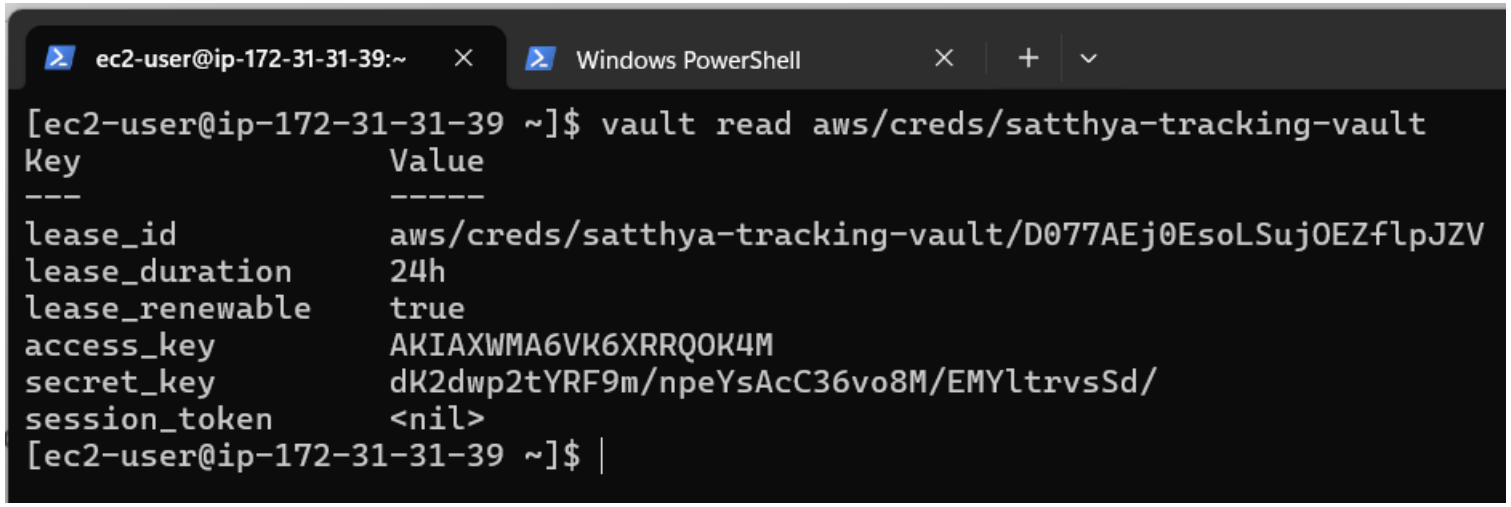


VAULT CONFIGURATION

- Store the AWS Access Key ID and Secret Access Key in Vault

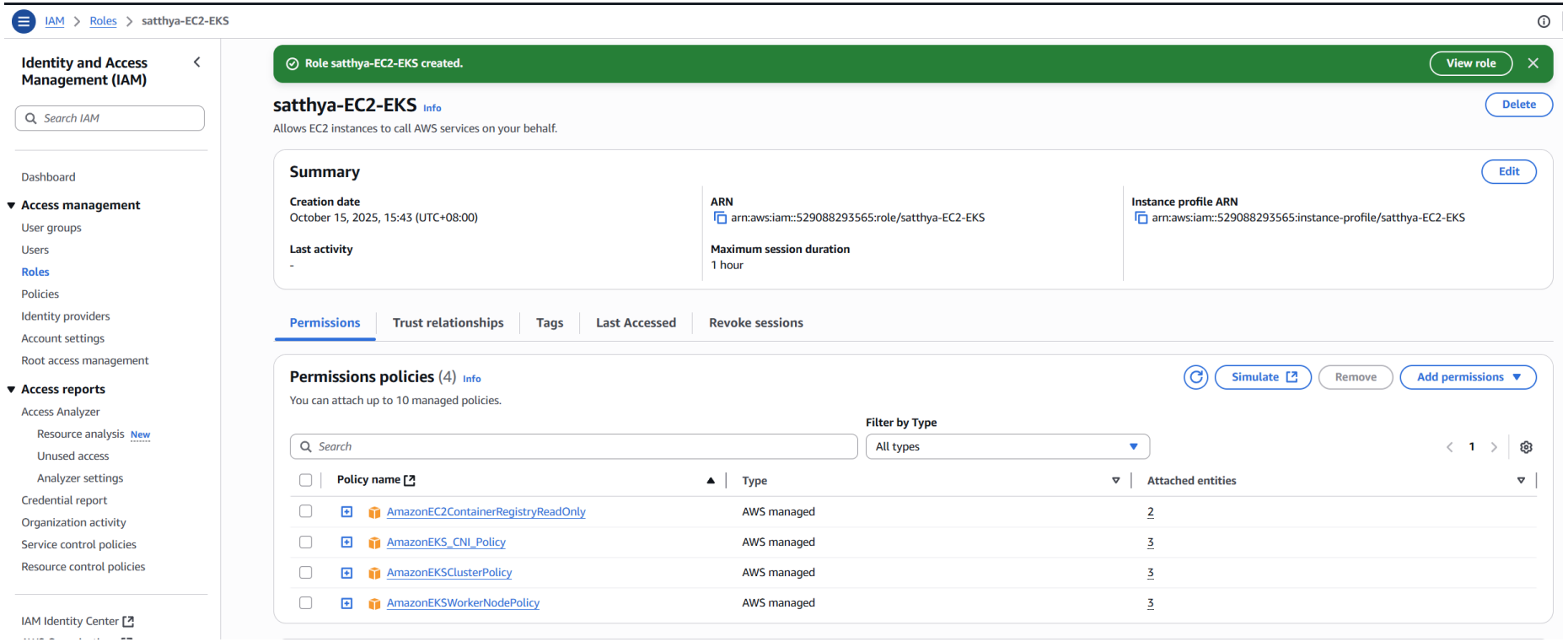
  - Read AWS Credentials via Vault.

    - vault read aws/creds/satthya-tracking-vault



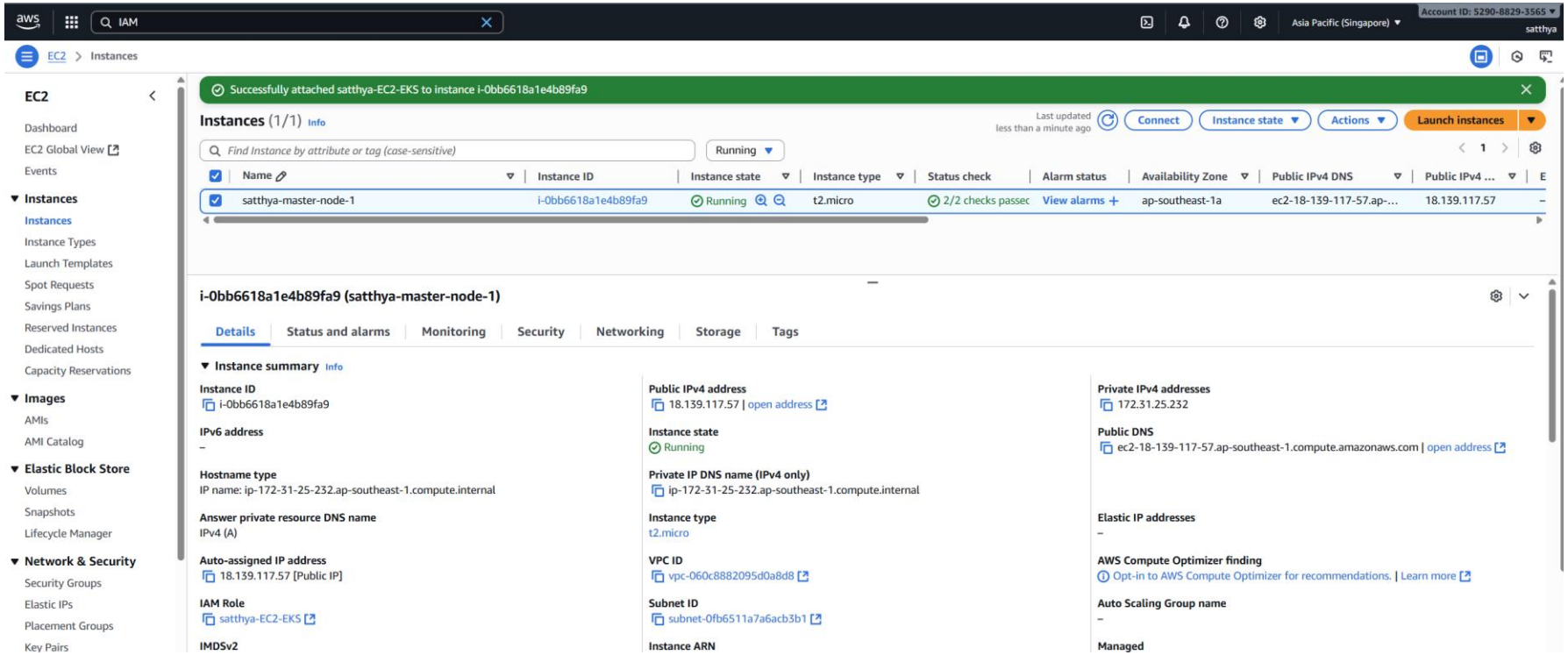
IAM

- Created an IAM role to be attached to an EC2 instance for serving as the master node in an EKS cluster.



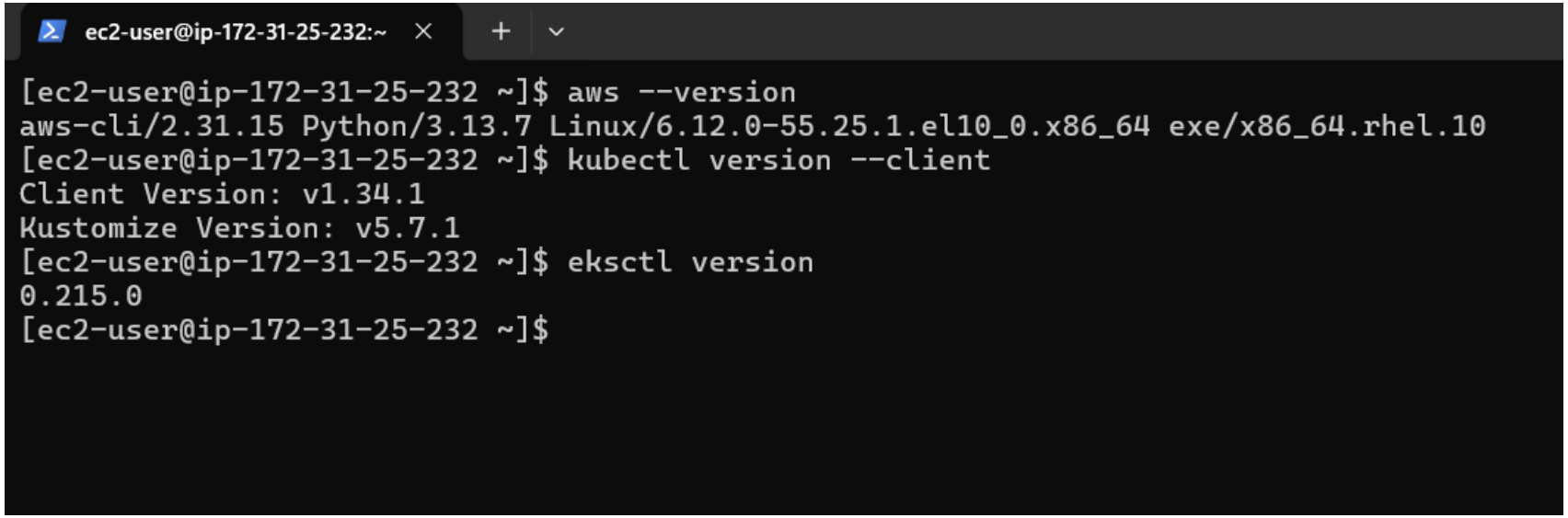
EKS Cluster Master Node on EC2

- Deployed an EC2 instance as the master node for an AWS EKS cluster and attached a custom IAM role to manage cluster resources and integrate with other AWS services.



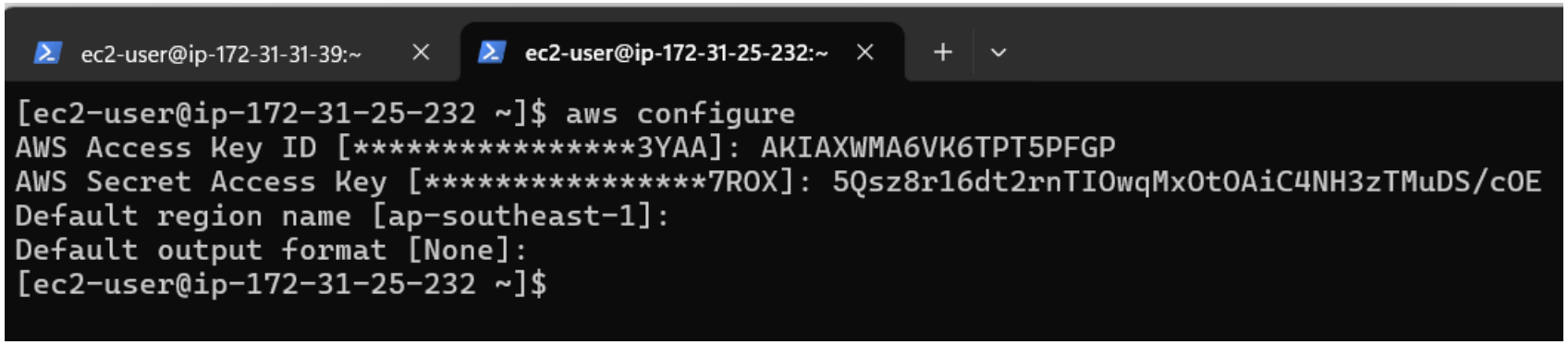
EKS Cluster Master Node on EC2

- Installed AWS CLI, eksctl, and kubectl on the master node.



EKS Cluster Master Node on EC2

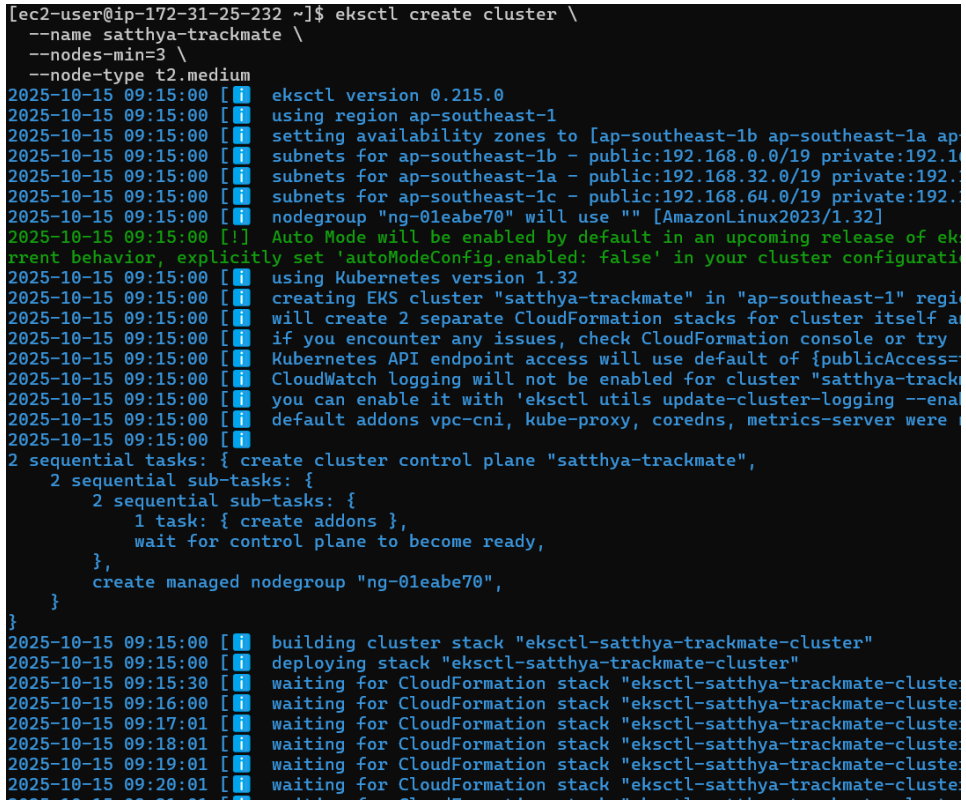
- Configured AWS CLI on an EC2 instance as a control node to manage AWS resources and Kubernetes clusters using dynamic AWS credentials retrieved from Vault.



EKS Cluster Setup with Worker Nodes

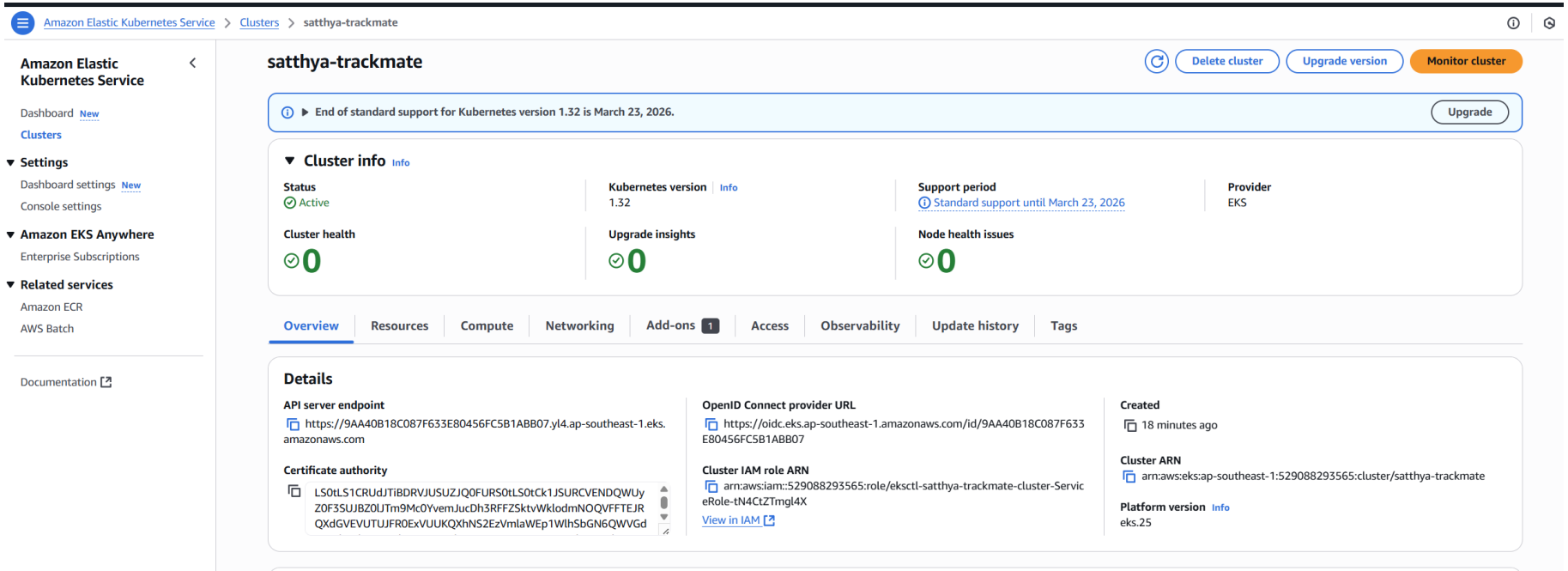
- Created an EKS cluster satthya-trackmate with 3 t2.medium worker nodes for running Kubernetes workloads.

➢ eksctl create cluster \
  --name satthya-trackmate \
  --nodes-min=3 \
  --node-type t2.medium



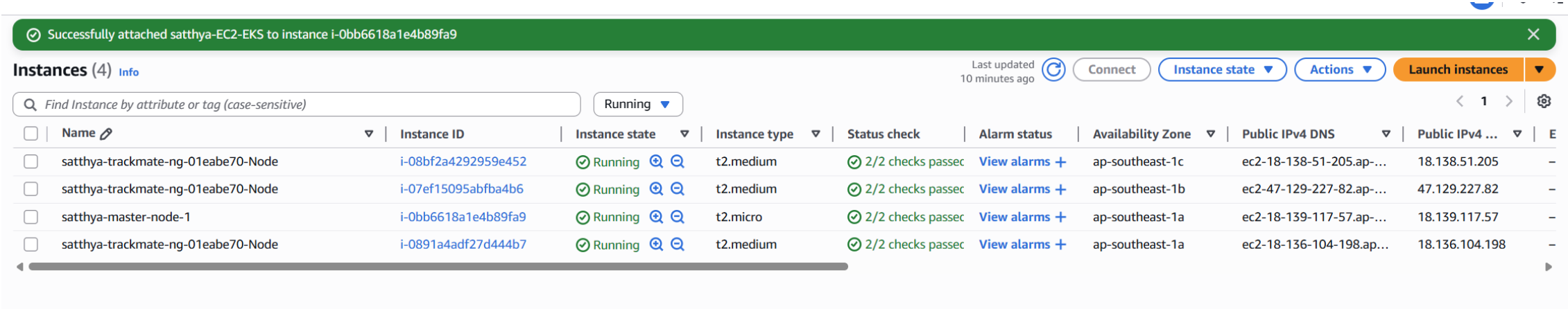
EKS Cluster Deployment on AWS

- Amazon EKS cluster satthya-trackmate created in ap-southeast-1, ready to manage Kubernetes workloads.



EKS Cluster Deployment on AWS

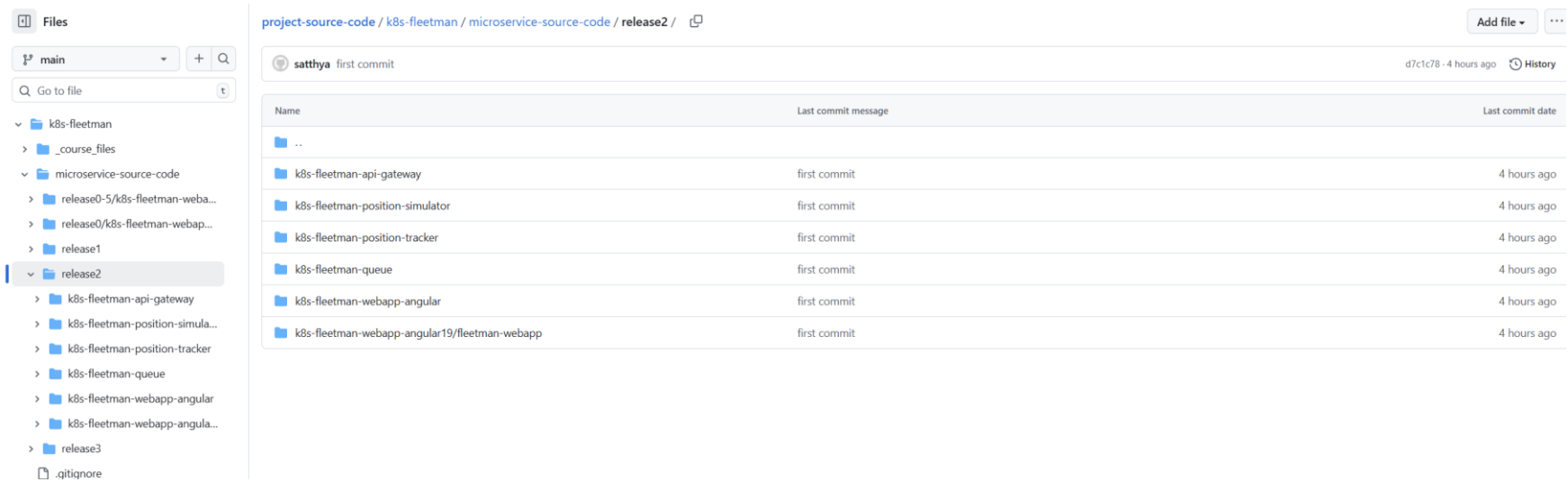
- Once the eksctl command ran, it provisioned 3 worker nodes in the cluster



Download Source Code

- The project is on GitHub. Use git clone to download a complete copy locally, including all files and history, ready for building and running.

  ➢ git clone https://github.com/satthya/project-source-code.git



Tracking App Build

- Build the Maven project, build the Angular front-end application, and create a Docker image for each microservice.

➢ Backend (Maven) :

mvn clean package

Run this in each microservice directory if a pom.xml file exists.

This will create war/jar artifact

```
[satthya@localhost k8s-fleetman-api-gateway]$ ll
total 32
-rw-r--r--. 1 satthya satthya  251 Oct 19 11:31 Dockerfile
-rw-r--r--. 1 satthya satthya 1083 Oct 19 11:31 LICENSE
-rwxr-xr-x. 1 satthya satthya 7058 Oct 19 11:31 mvnw
-rw-r--r--. 1 satthya satthya 5006 Oct 19 11:31 mvnw.cmd
-rw-r--r--. 1 satthya satthya 2277 Oct 19 11:31 pom.xml
-rw-r--r--. 1 satthya satthya  247 Oct 19 11:31 README.md
drwxr-xr-x. 4 satthya satthya   30 Oct 19 11:31 src
[satthya@localhost k8s-fleetman-api-gateway]$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< com.virtualpairprogrammers:fleetman >-----------------
[INFO] Building fleetman 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.2.0:clean (default-clean) @ fleetman ---
[INFO]
[INFO] --- maven-resources-plugin:3.3.0:resources (default-resources) @ fleetman ---
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ fleetman ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 6 source files to /home/satthya/fleetman-project/project-source-code/k8s-fleetma
ode/release2/k8s-fleetman-api-gateway/target/classes
```

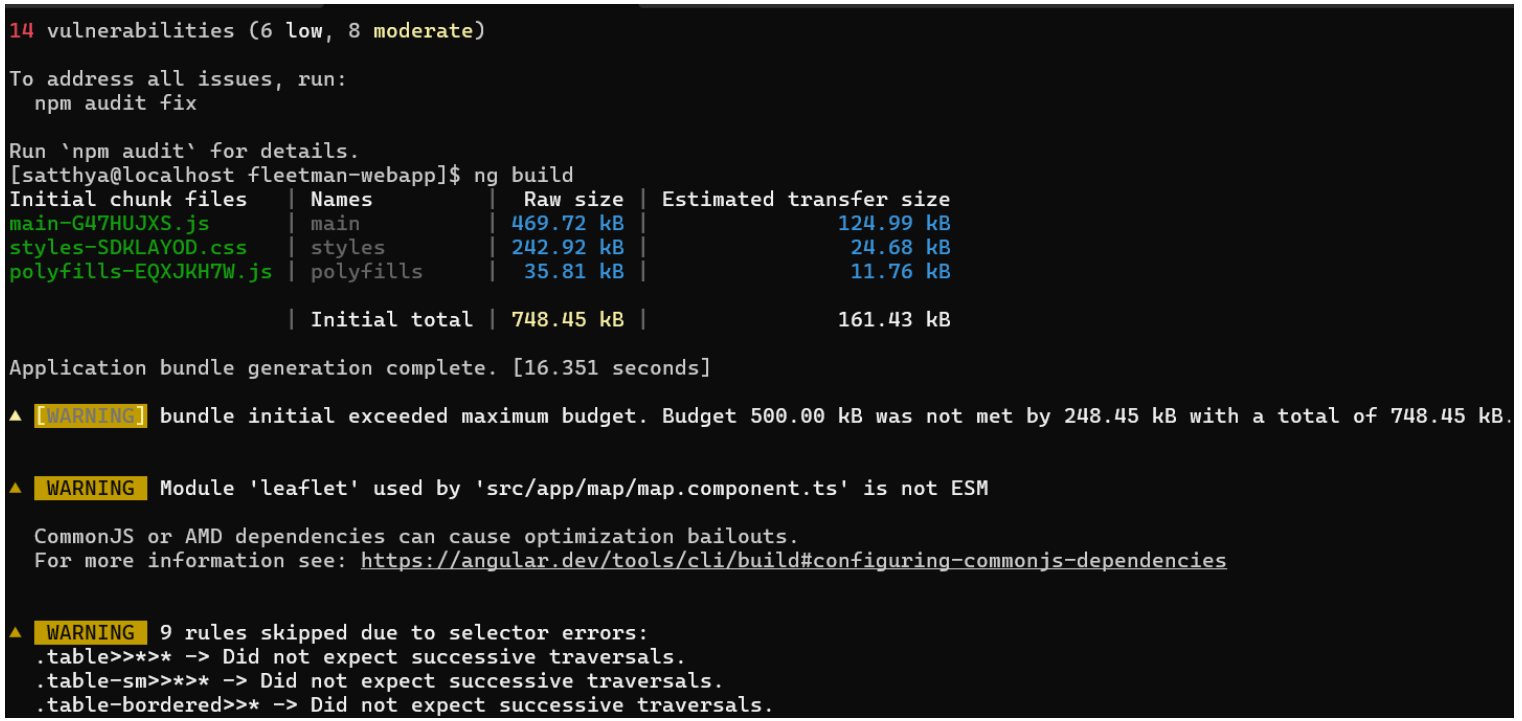
Tracking App Build

- Build the Maven project, build the Angular front-end application, and create a Docker image for each microservice.

➢ Frontend (Angular) :
ng build

Run this in the front-end web directory.

Turns your Angular code into a deployable front-end application



Tracking App Build

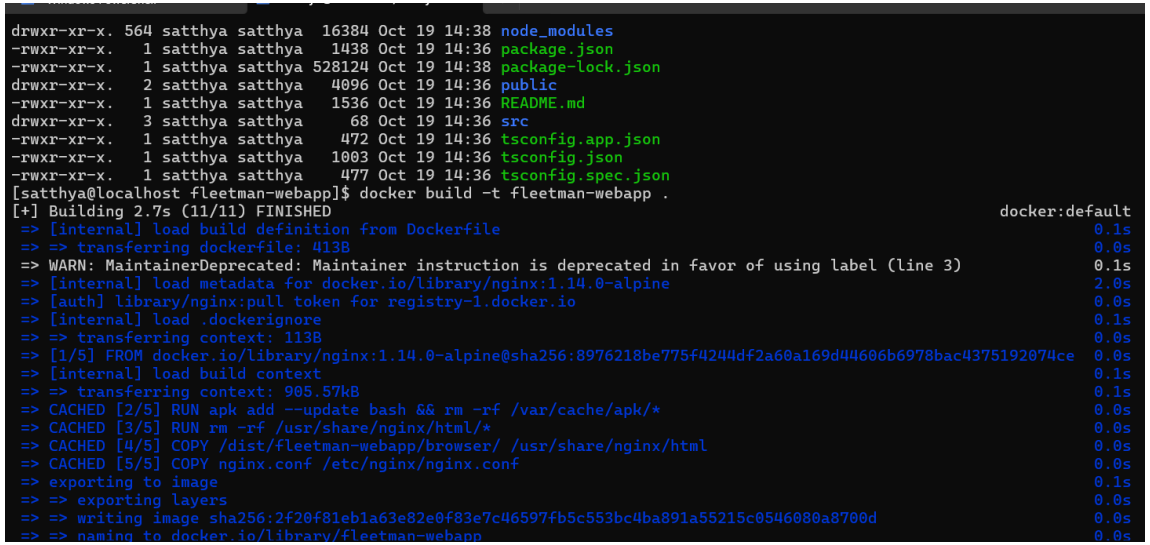
- Build the Maven project, build the Angular front-end application, and create a Docker image for each microservice.

  ➢ Docker image :

  docker build –t “name”

  docker push satthya04/fleetman-webapp .

Run this in each directory containing a Dockerfile, replace name with your image name, and push each image to Docker Hub



Tracking App Deployment

- Deployed a microservices tracking app with Docker & Kubernetes using YAML files and exposed the frontend via LoadBalancer.

```
[ec2-user@ip-172-31-25-232 tracking-app]$ cat deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fleetman-webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: satthya04/fleetman-webapp:release2
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: production-microservice

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: fleetman-api-gateway
spec:
  replicas: 1
  selector:
    matchLabels:
      app: api-gateway
  template:
    metadata:
      labels:
        app: api-gateway
    spec:
      containers:
      - name: api-gateway
        image: satthya04/fleetman-api-gateway:latest
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: production-microservice
```

```
---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: fleetman-position-simulator
spec:
  replicas: 1
  selector:
    matchLabels:
      app: position-simulator
  template:
    metadata:
      labels:
        app: position-simulator
    spec:
      containers:
      - name: position-simulator
        image: satthya04/fleetman-position-simulator:latest
        env:
        - name: SPRING_PROFILES_ACTIVE
          value: production-microservice

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: fleetman-queue
spec:
  replicas: 1
  selector:
    matchLabels:
      app: queue
  template:
    metadata:
      labels:
        app: queue
    spec:
      containers:
      - name: queue
        image: satthya04/fleetman-queue:latest
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fleetman-position-tracker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: position-tracker
  template:
    metadata:
      labels:
        app: position-tracker
    spec:
      containers:
      - name: position-tracker
        image: satthya04/fleetman-position-tracker:latest

        env:
        - name: SPRING_PROFILES_ACTIVE
          value: production-microservice

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo:3.6.5-jessie
          volumeMounts:
            - name: mongo-persistent-storage
              mountPath: /data/db
      volumes:
        - name: mongo-persistent-storage
          persistentVolumeClaim:
             claimName: mongo-pvc
```

Tracking App Deployment

- Created a Service YAML file in Kubernetes to expose the microservices and route traffic to the Pods through a stable endpoint.

ec2-user@ip-172-31-25-232:~

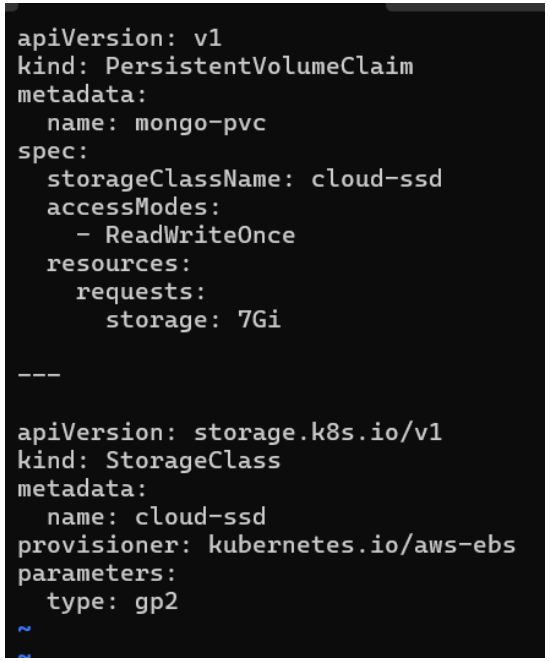
```yaml
kind: Service
apiVersion: v1
metadata:
  name: fleetman-webapp
spec:
  ports:
    - name: webapp
      port: 80

  selector:
    app: webapp

  type: LoadBalancer

---

kind: Service
apiVersion: v1
metadata:
  name: fleetman-api-gateway
spec:
  ports:
    - name: api-gateway
      port: 8080

  selector:
    app: api-gateway

  type: ClusterIP
```

```yaml
---

kind: Service
apiVersion: v1
metadata:
  name: fleetman-queue
spec:
  ports:
    - name: queue
      port: 8161

    - name: endpoint
      port: 61616
  selector:
    app: queue

  type: ClusterIP

---

kind: Service
apiVersion: v1
metadata:
  name: fleetman-position-tracker
spec:
  ports:
    - name: position-tracker
      port: 8080


  selector:
    app: position-tracker

  type: ClusterIP
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: fleetman-mongodb
spec:
  selector:
    app: mongodb
  ports:
    - name: mongoport
      port: 27017

  type: ClusterIP
```

Tracking App Deployment

- Created a Storage YAML file in Kubernetes to provide persistent storage for the database, ensuring data remains intact even if the database Pod restarts or is replaced.



```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  storageClassName: cloud-ssd
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 7Gi

---

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cloud-ssd
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
~
~
```

Tracking App Deployment

- Deploy the YAML file using the command below:

  - kubectl apply –f deployment.yml
  - kubectl apply –f service.yml
  - kubectl apply –f storage.yml

```
[ec2-user@ip-172-31-25-232 tracking-app]$ ll
total 12
-rw-r--r--. 1 ec2-user ec2-user 2479 Oct 24 13:21 deployment.yml
-rw-r--r--. 1 ec2-user ec2-user  957 Oct 24 13:25 service.yml
-rw-r--r--. 1 ec2-user ec2-user  330 Oct 24 13:29 storage-aws.yml
[ec2-user@ip-172-31-25-232 tracking-app]$ kubectl apply -f deployment.yml
deployment.apps/fleetman-webapp created
deployment.apps/fleetman-api-gateway created
deployment.apps/fleetman-position-simulator created
deployment.apps/fleetman-queue created
deployment.apps/fleetman-position-tracker created
error: error parsing deployment.yml: error converting YAML to JSON: yaml: line 19: found a tab character that violates indentation
[ec2-user@ip-172-31-25-232 tracking-app]$ kubectl apply -f service.yml
service/fleetman-webapp created
service/fleetman-api-gateway created
service/fleetman-queue created
service/fleetman-position-tracker created
service/fleetman-mongodb created
[ec2-user@ip-172-31-25-232 tracking-app]$ kubectl apply -f storage-aws.yml
persistentvolumeclaim/mongo-pvc created
storageclass.storage.k8s.io/cloud-ssd created
[ec2-user@ip-172-31-25-232 tracking-app]$
```

Tracking App Deployment

- View all Kubernetes resources in the current namespace, including Pods, Services, Deployments, and ReplicaSets.

```
[ec2-user@ip-172-31-25-232 tracking-app]$ kubectl get all
NAME                                          READY   STATUS    RESTARTS   AGE
pod/fleetman-api-gateway-9d595b467-7npkg      1/1     Running   0          18m
pod/fleetman-position-simulator-7bcf87d74b-hhktm   1/1     Running   0          18m
pod/fleetman-position-tracker-5b5dc8bd88-6298z     1/1     Running   0          18m
pod/fleetman-queue-69c5d887cc-n6llq           1/1     Running   0          18m
pod/fleetman-webapp-77c9f67b8f-x7jjx          1/1     Running   0          18m
pod/mongodb-5979dc7894-fcv7w                  1/1     Running   0          21s

NAME                                TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)             AGE
service/fleetman-api-gateway        ClusterIP      10.100.224.210   <none>                                                                     8080/TCP            18m
service/fleetman-mongodb            ClusterIP      10.100.115.51    <none>                                                                     27017/TCP           18m
service/fleetman-position-tracker   ClusterIP      10.100.241.238   <none>                                                                     8080/TCP            18m
service/fleetman-queue              ClusterIP      10.100.121.129   <none>                                                                     8161/TCP,61616/TCP  18m
service/fleetman-webapp             LoadBalancer   10.100.144.201   a105c8f15bd324f13b06f85e6e7802ee-361513463.ap-southeast-1.elb.amazonaws.com   80:32054/TCP        18m
service/kubernetes                  ClusterIP      10.100.0.1       <none>                                                                     443/TCP             28m

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/fleetman-api-gateway       1/1     1            1           18m
deployment.apps/fleetman-position-simulator   1/1     1            1           18m
deployment.apps/fleetman-position-tracker     1/1     1            1           18m
deployment.apps/fleetman-queue             1/1     1            1           18m
deployment.apps/fleetman-webapp            1/1     1            1           18m
deployment.apps/mongodb                    1/1     1            1           21s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/fleetman-api-gateway-9d595b467      1         1         1       18m
replicaset.apps/fleetman-position-simulator-7bcf87d74b   1         1         1       18m
replicaset.apps/fleetman-position-tracker-5b5dc8bd88     1         1         1       18m
replicaset.apps/fleetman-queue-69c5d887cc           1         1         1       18m
replicaset.apps/fleetman-webapp-77c9f67b8f          1         1         1       18m
replicaset.apps/mongodb-5979dc7894                  1         1         1       21s
[ec2-user@ip-172-31-25-232 tracking-app]$
```

Tracking App UI