**RAKBANK - DevOps Position Assessment**

Deploying a Java Application to AWS EKS using GitHub Actions and IAC with Terraform Objective: Candidates are required to demonstrate their proficiency in DevOps practices by deploying a Java application to Amazon Elastic Kubernetes Service (EKS) using GitHub Actions for CI/CD and Infrastructure as Code (IaC) with Terraform.

Submitted By

Hina Ikram

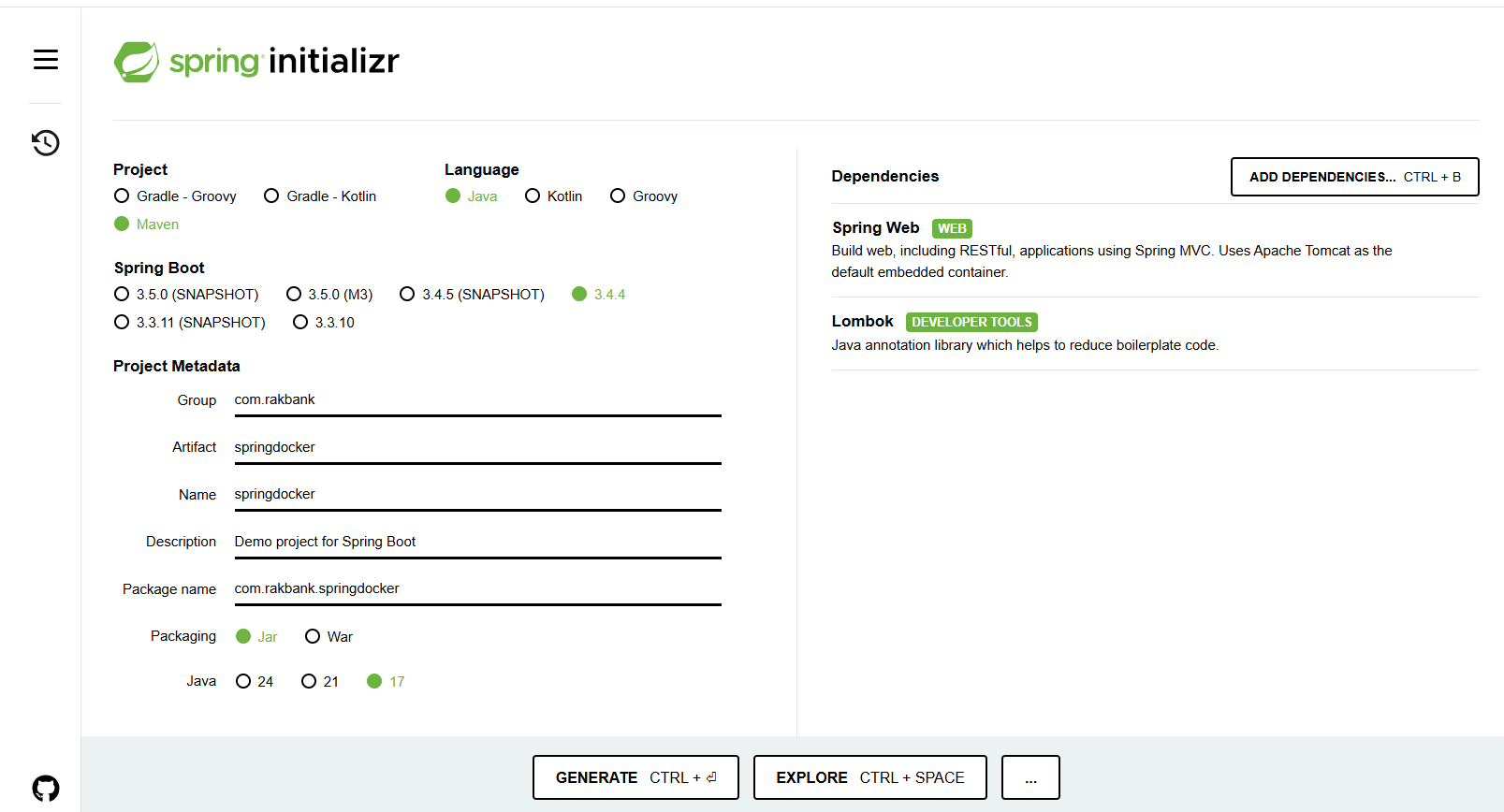
Solution:

GitHub repo Link: <https://github.com/Hinaikram/RBSpringBoot-K8.git>

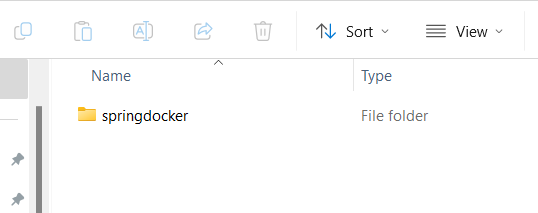
Github repo for Terraform: <https://github.com/Hinaikram/RBTerraform-EKS.git>

# Procedure to deploy Spring Io application

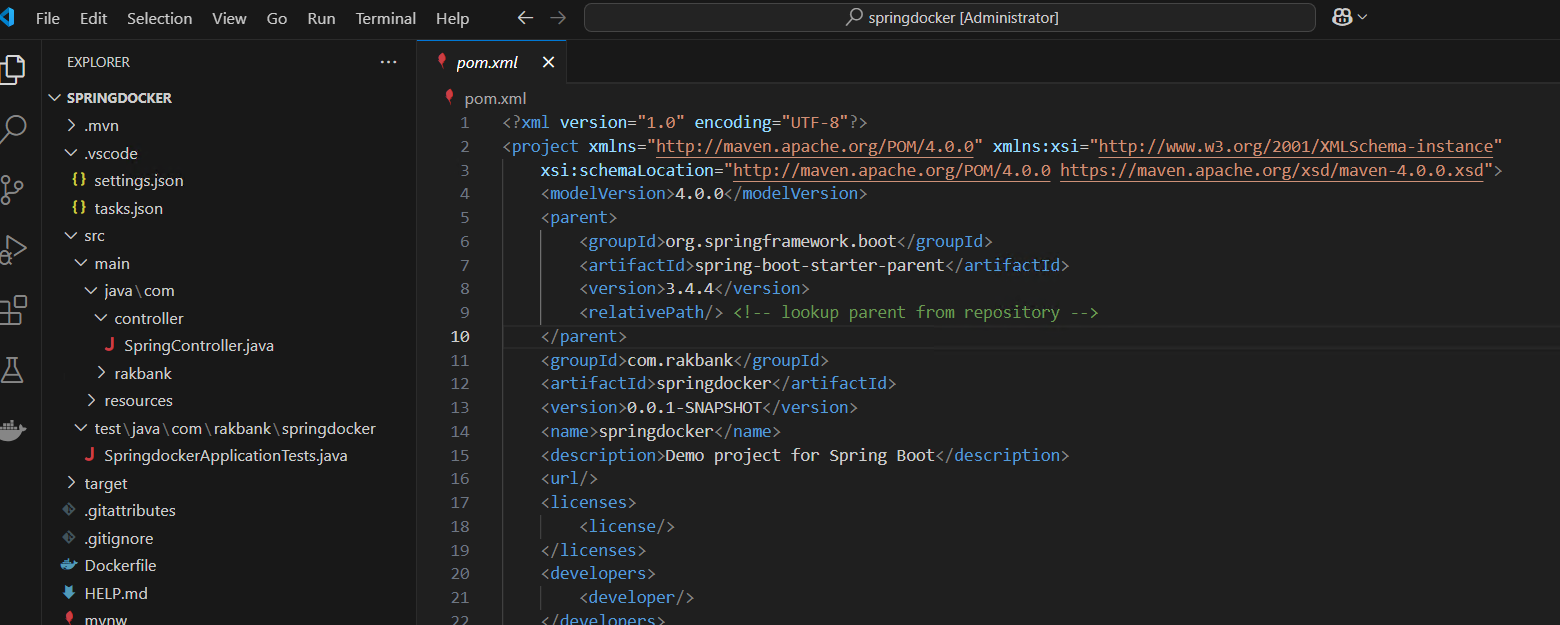
1. Open the link to download the Jave web base application following the link.
2. <https://start.spring.io/>
3. Choose the following option:
   1. Maven
   2. Nnn
4. Add web dependencies
   1. Spring web
   2. lombok



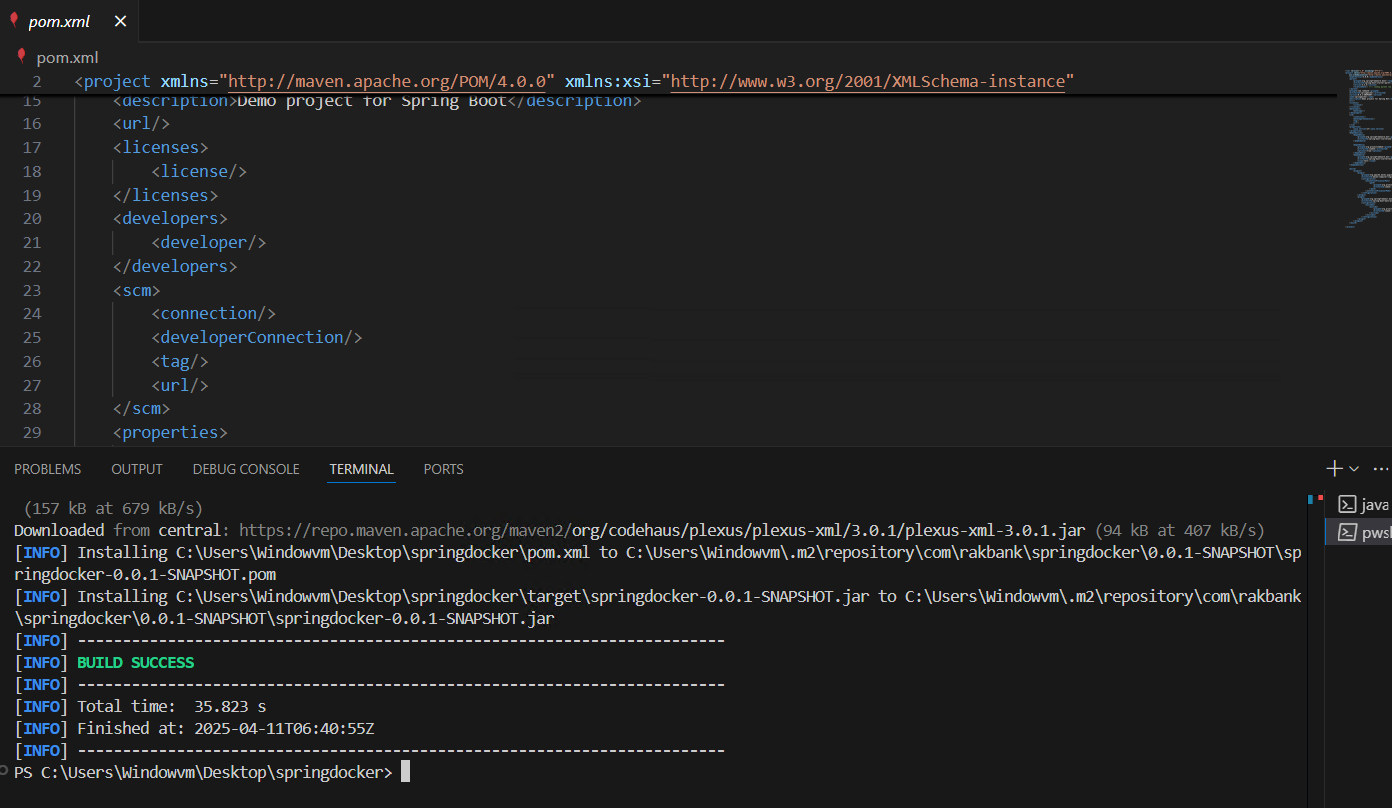
1. Generate the zip package, which will be downloaded.



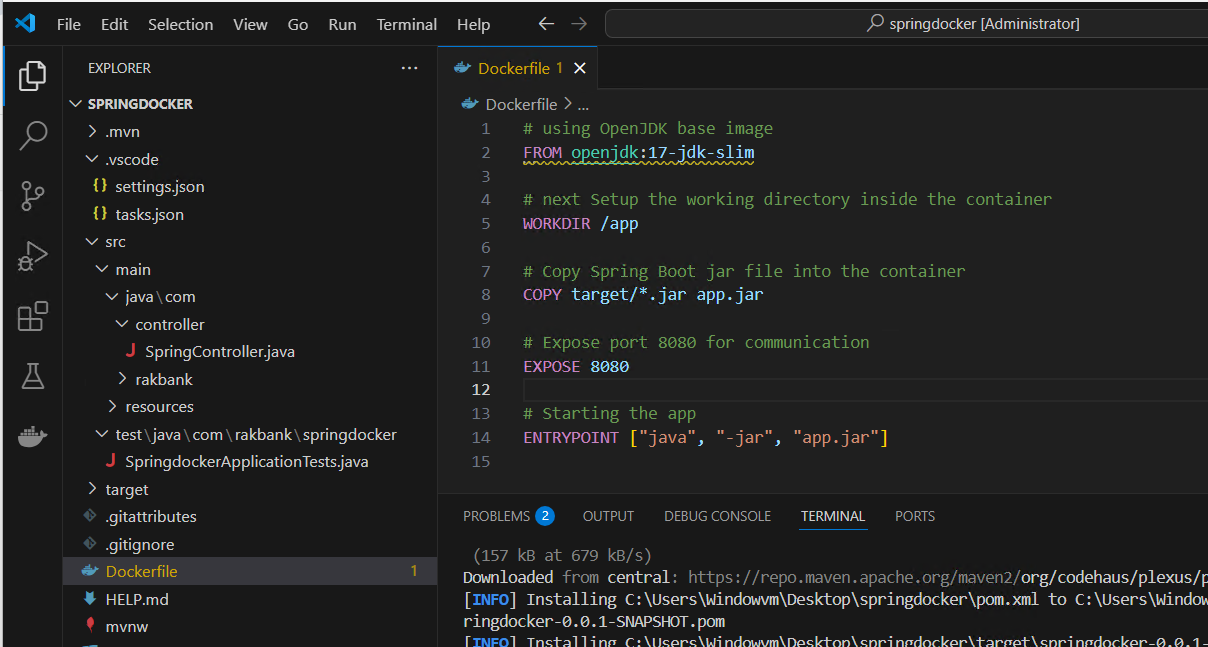
1. Open the project in any version control, Visual Studio. Add the class and controller in the project.



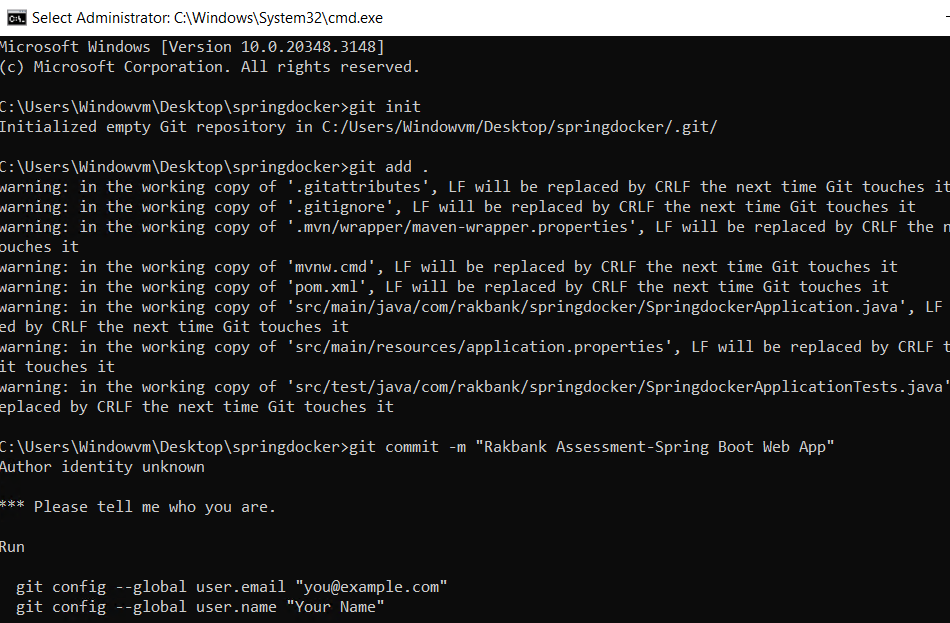
1. Build the project, using the pom file.The build is success full.

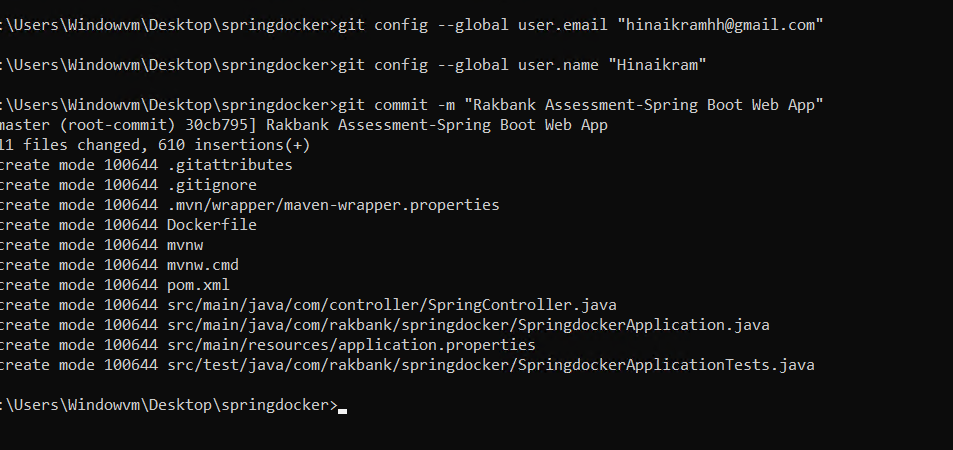


1. Create a Dockerfile at the root folder. Port 8080 is being used.

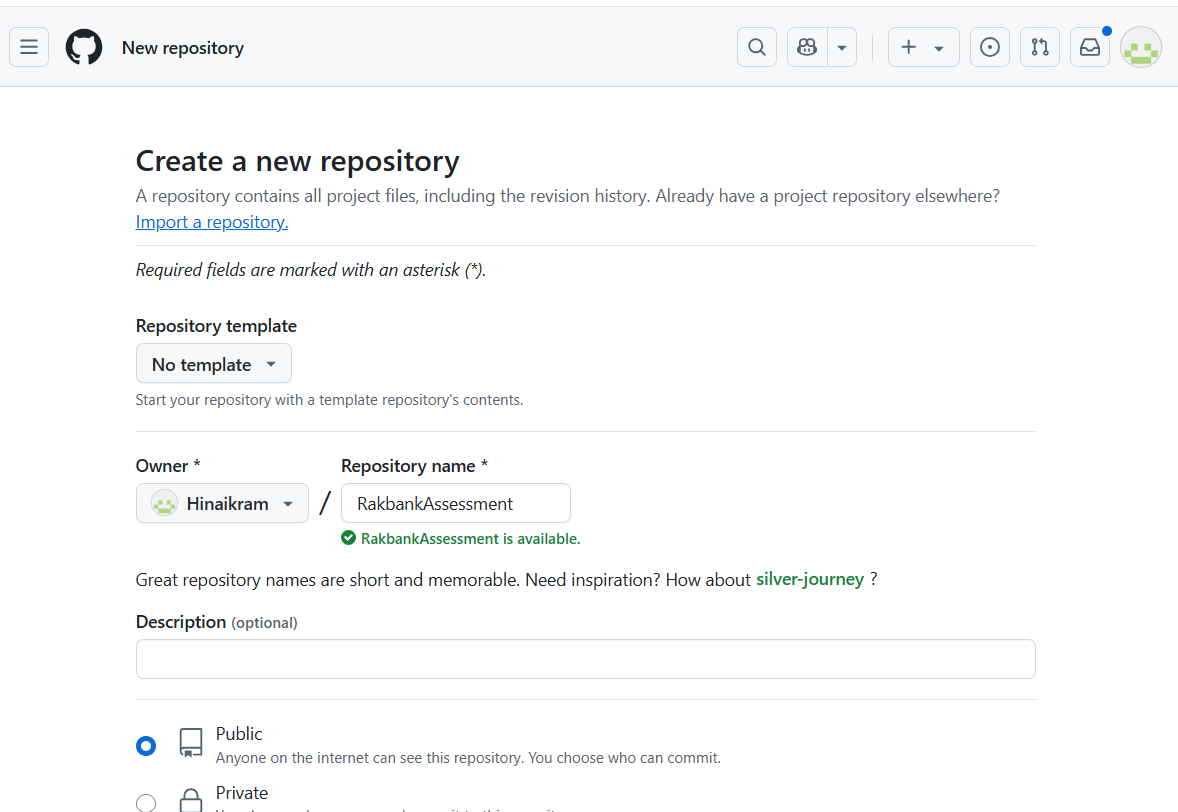


1. Initialize the Git repo for the project, using your credentials
   1. Git init
   2. Git add .
   3. Git commit -m “ Rakban Assessmet”

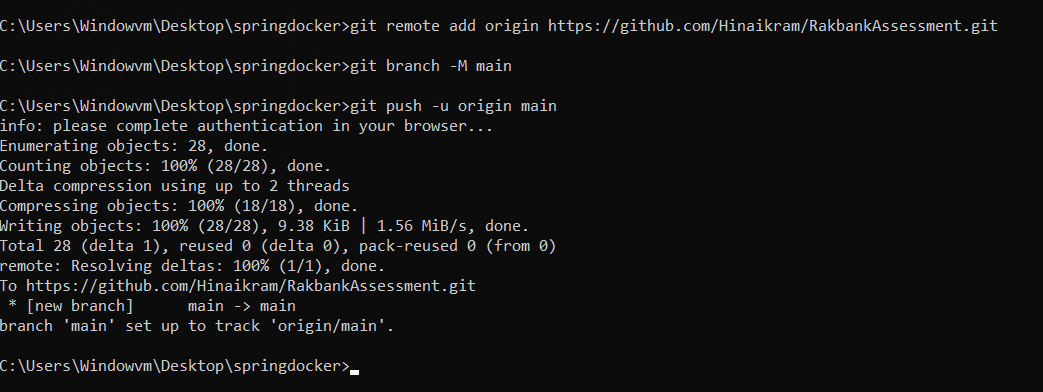




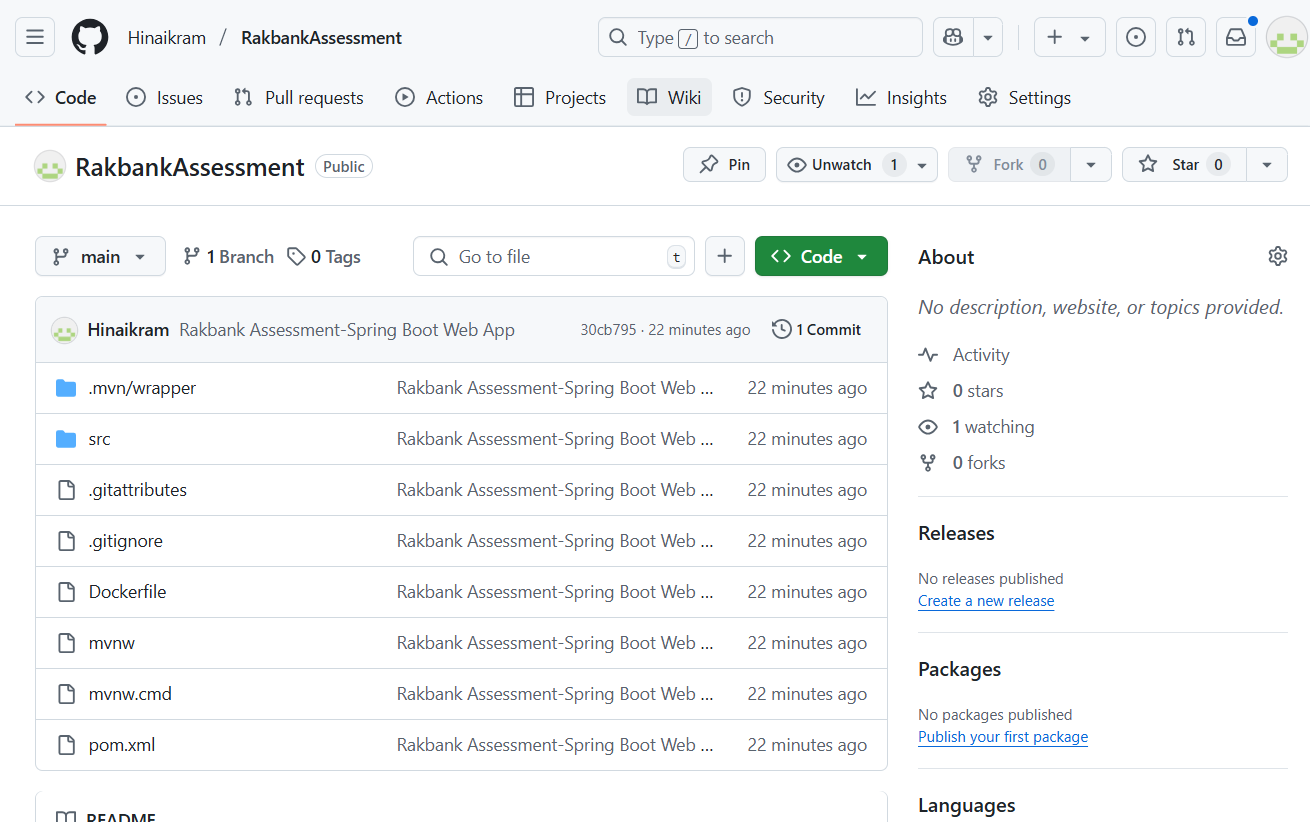
1. Create a repository in the Github account without the readme file as we have already initialized it in the project folder.



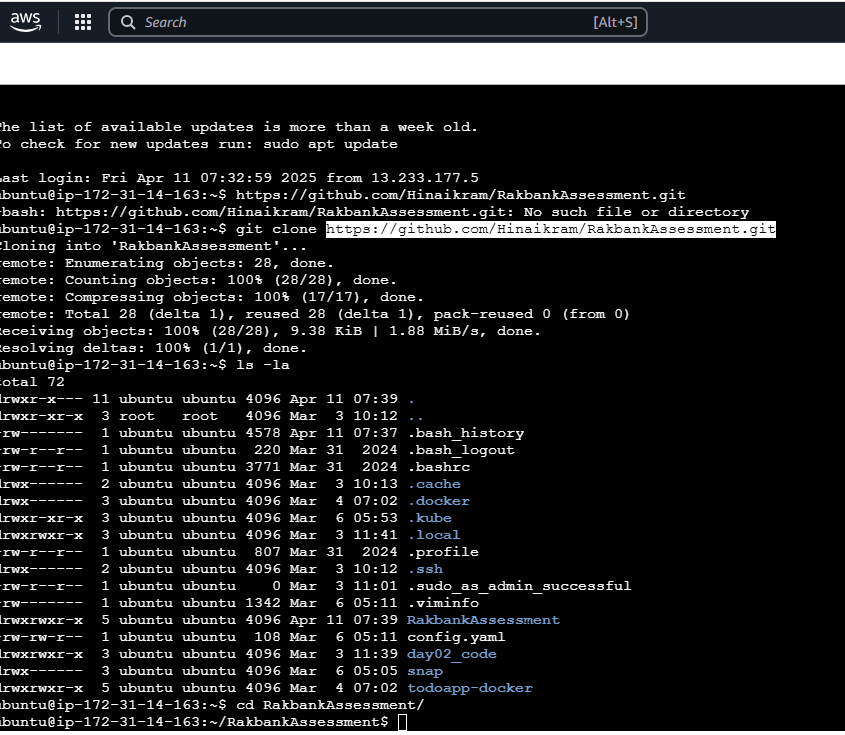
1. Now push the code in repository.
   1. git remote add origin <https://github.com/Hinaikram/RakbankAssessment.git>
   2. git branch -M main
   3. git push -u origin main



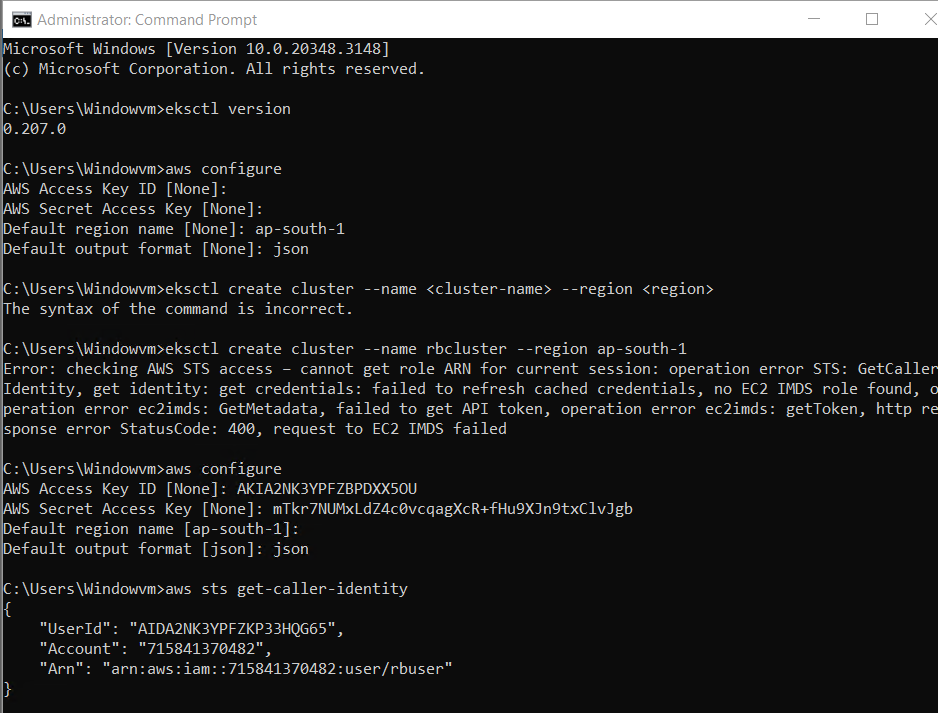
1. Refresh the browser page and repo will be updated.

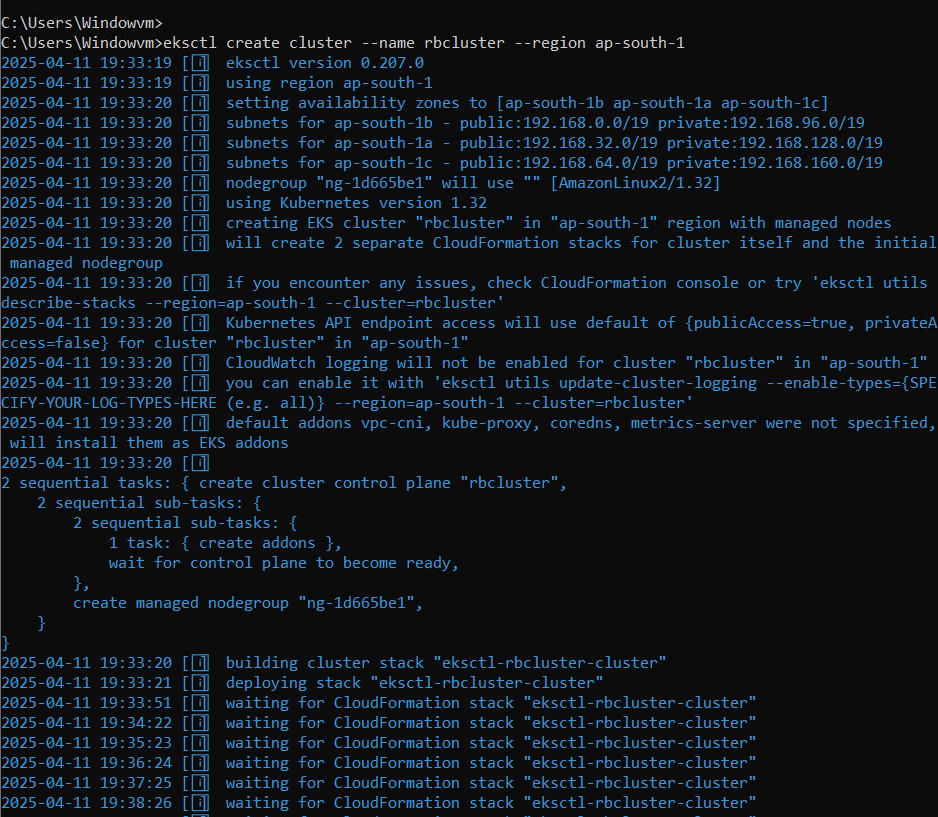


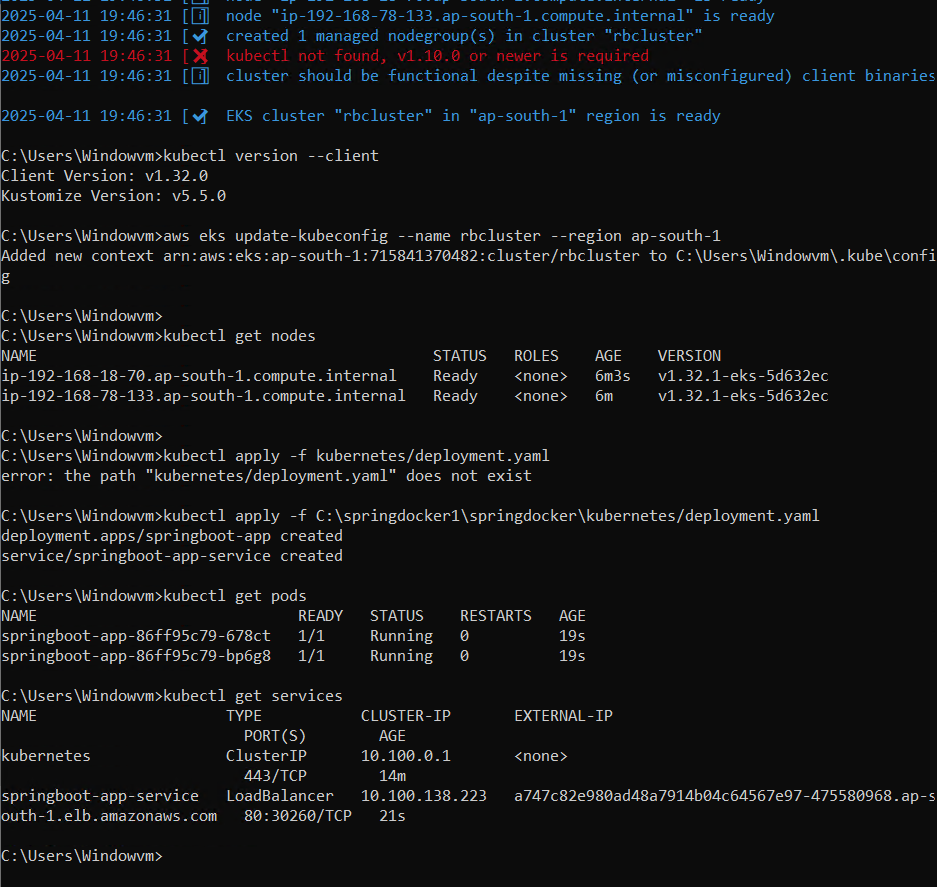
1. Now build a docker image using the Git repo



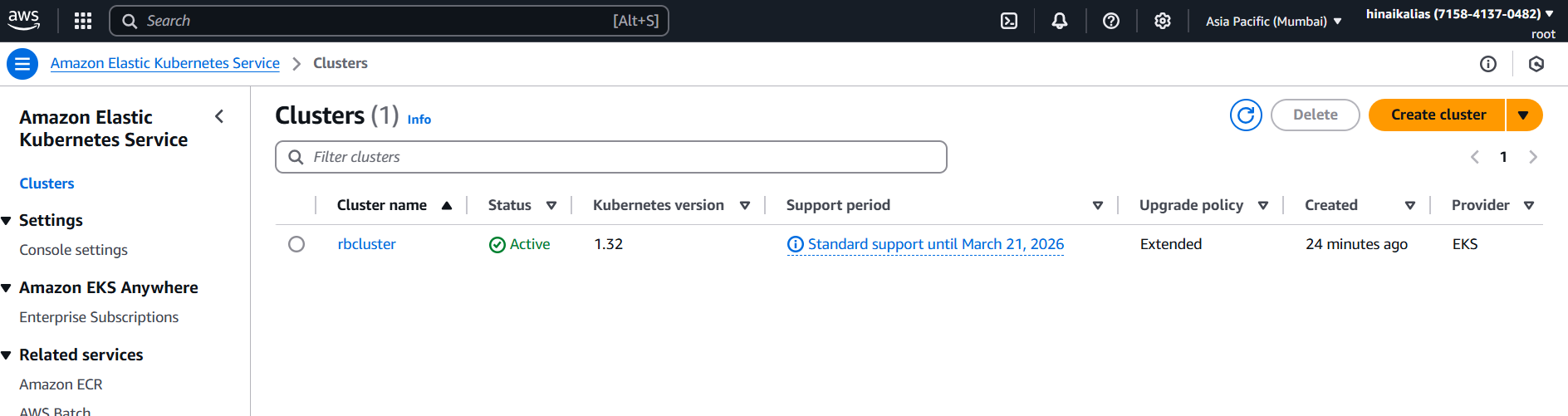
1. Run the docker image
   1. docker build -t springboot-app .
   2. docker run -d -p 8080:8080 springboot-app
2. Crete a cluster
   1. eksctl create cluster --name rbcluster --region ap-south-1
   2. kubectl apply -f C:\springdocker1\springdocker\kubernetes/deployment.yaml

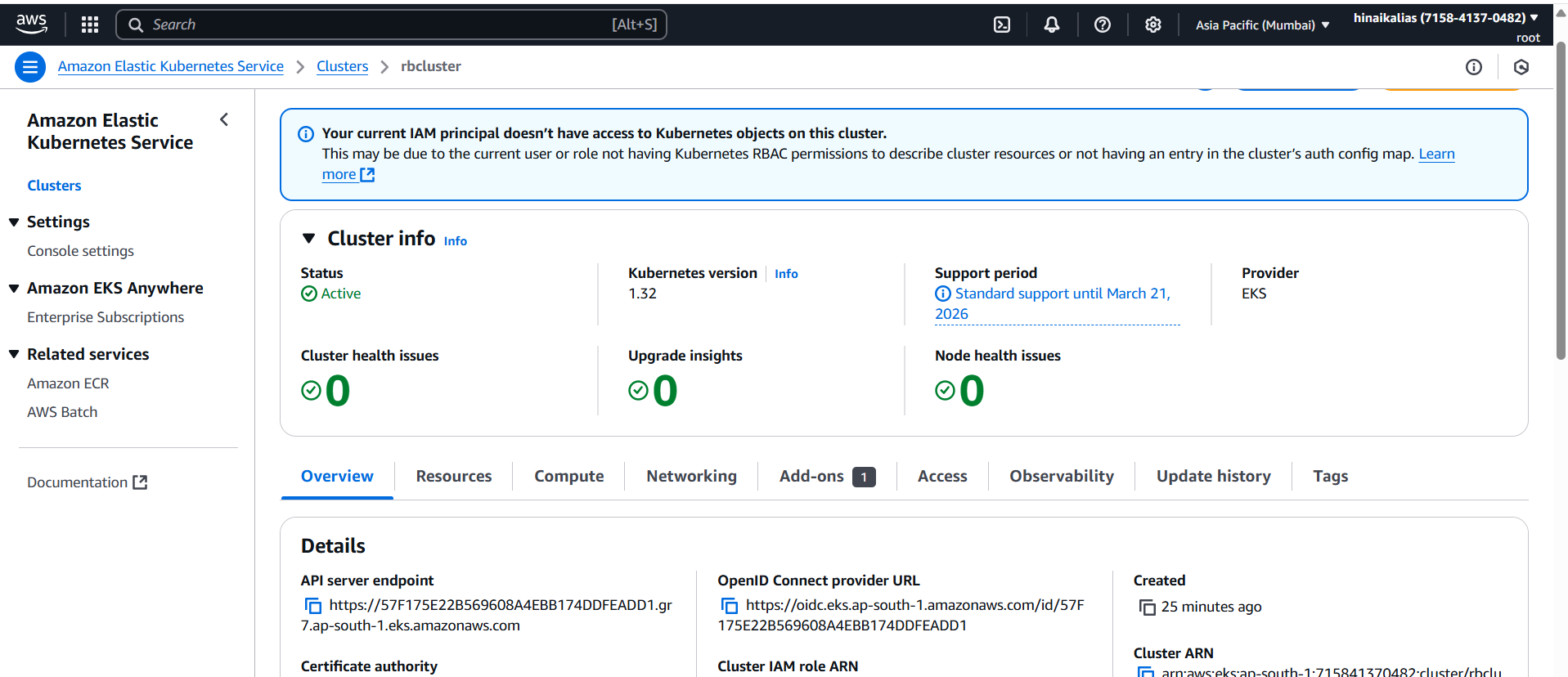


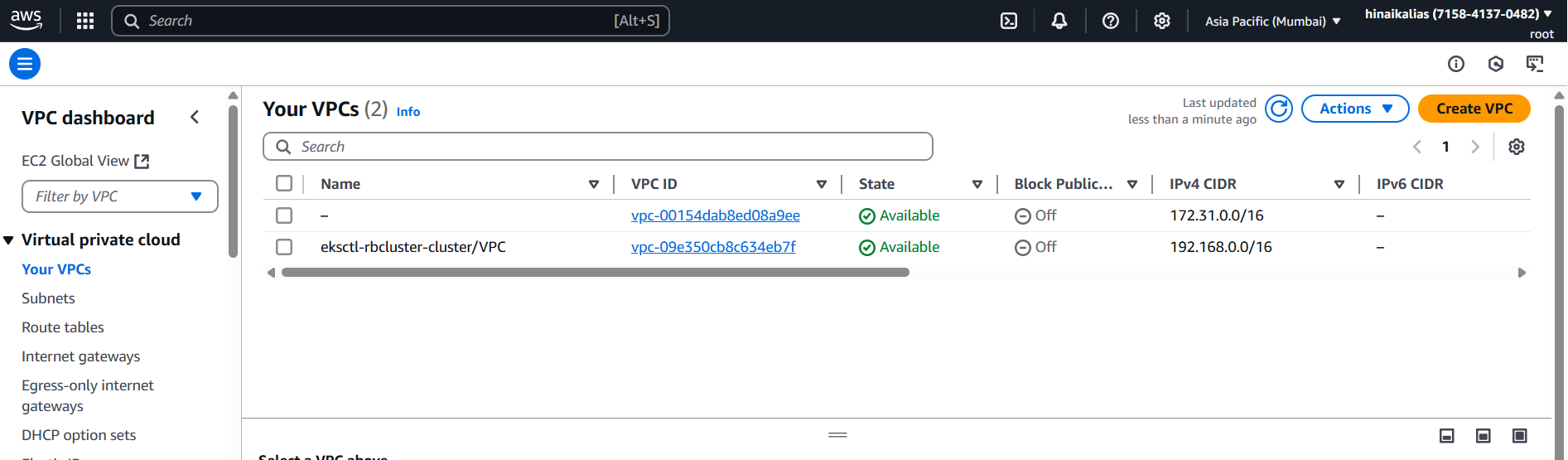


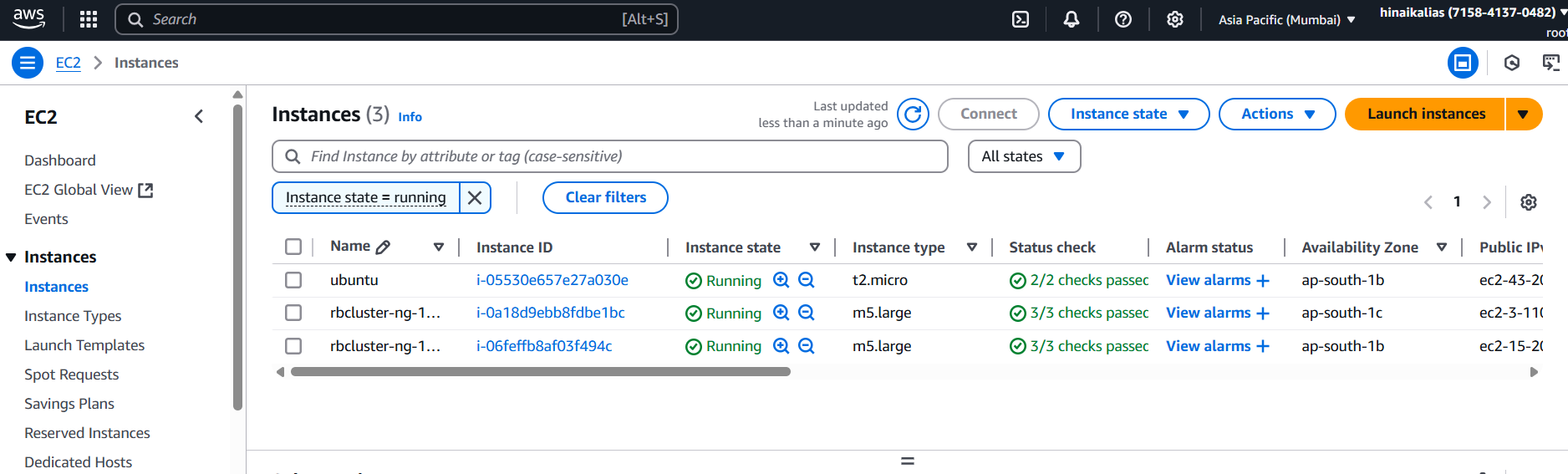


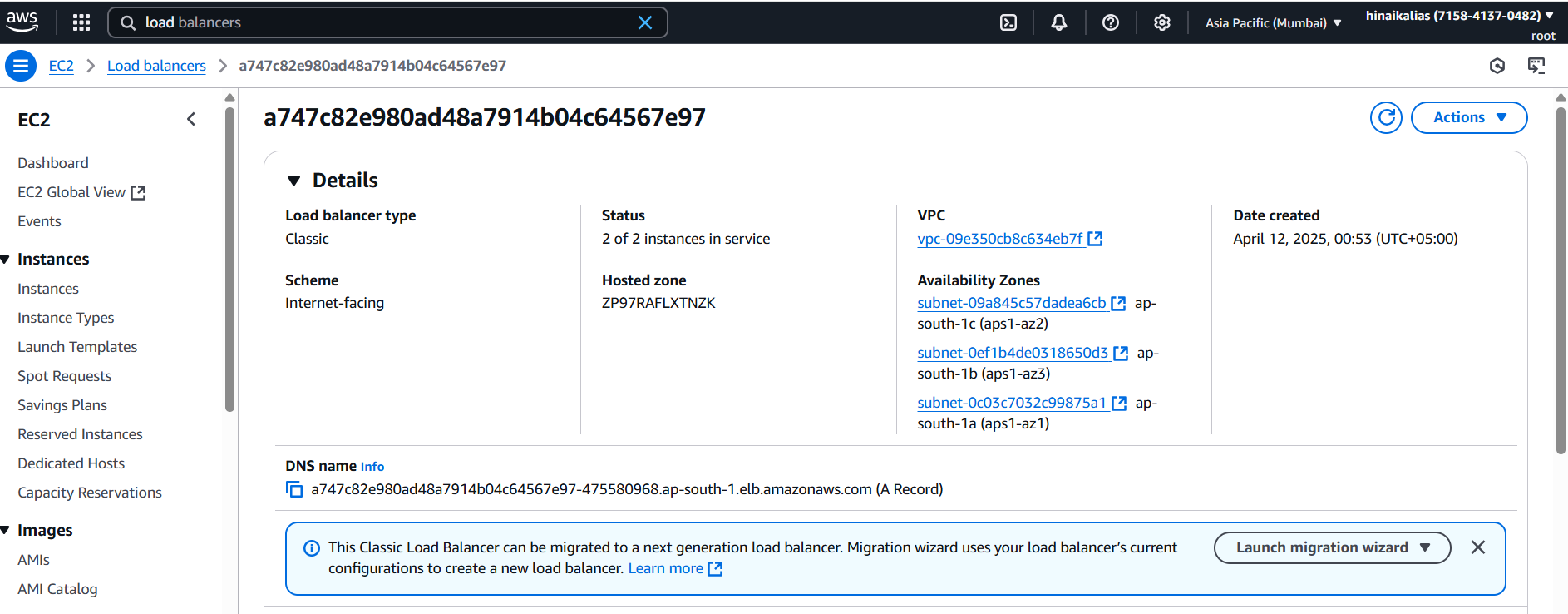
1. Cluster has been configured and full functional.

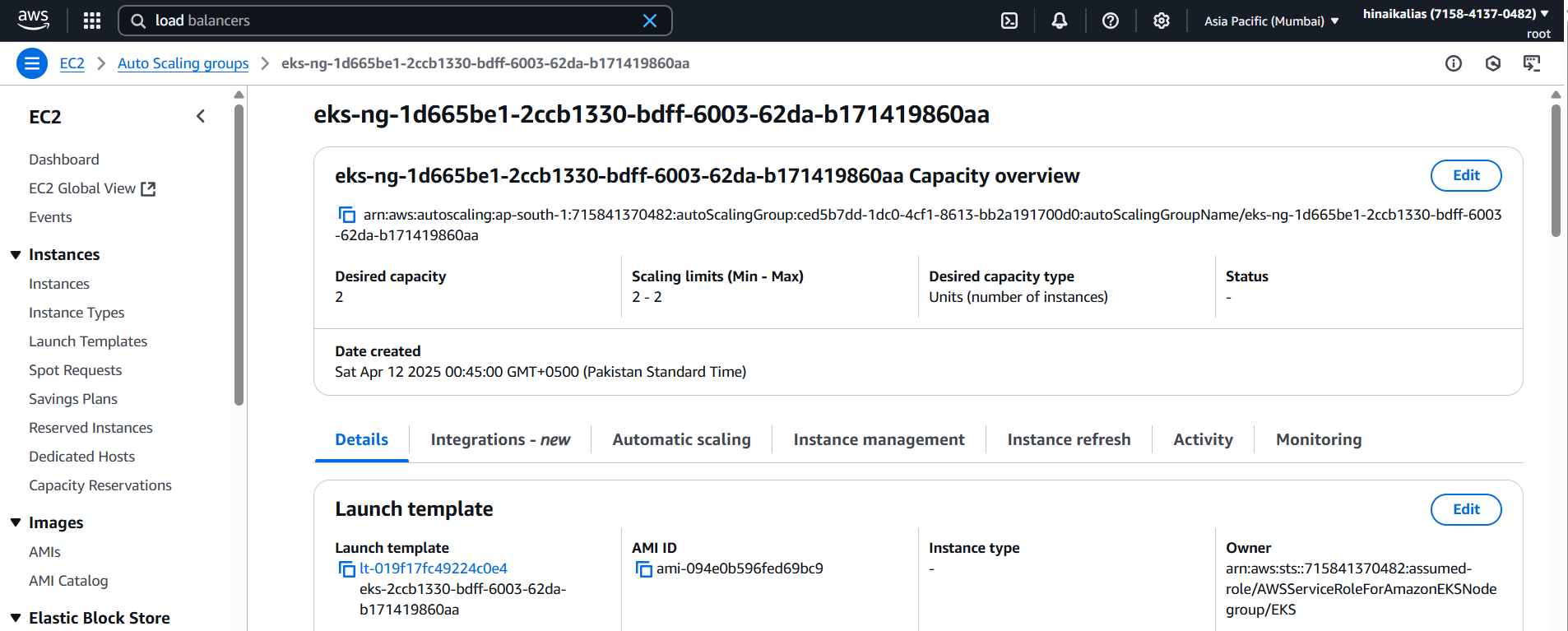


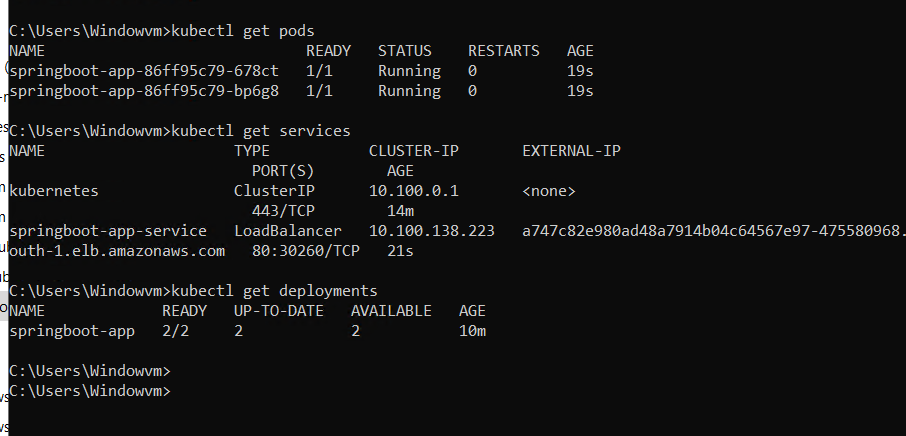


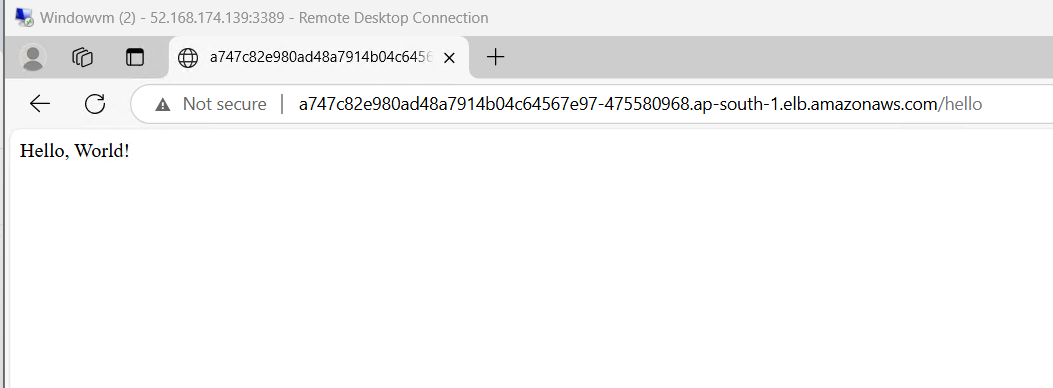








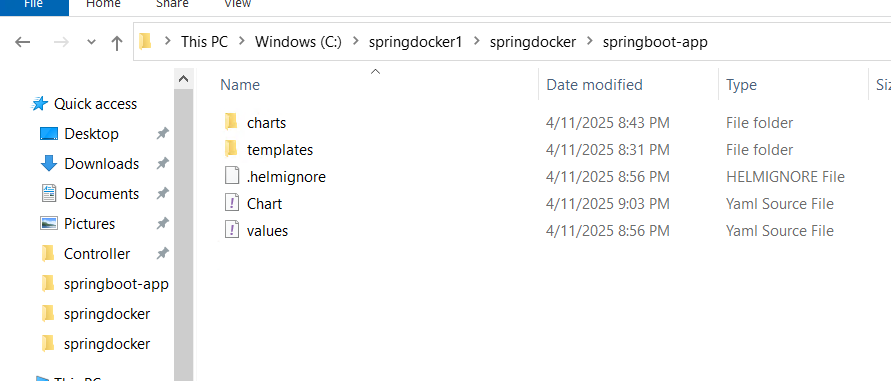


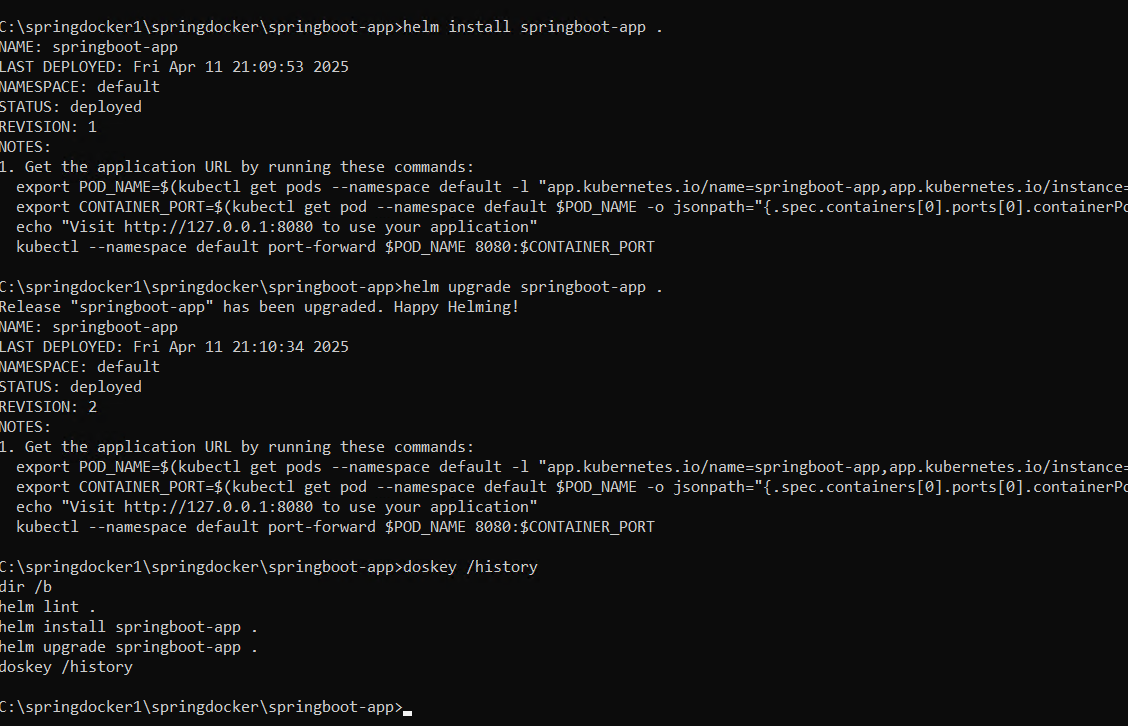


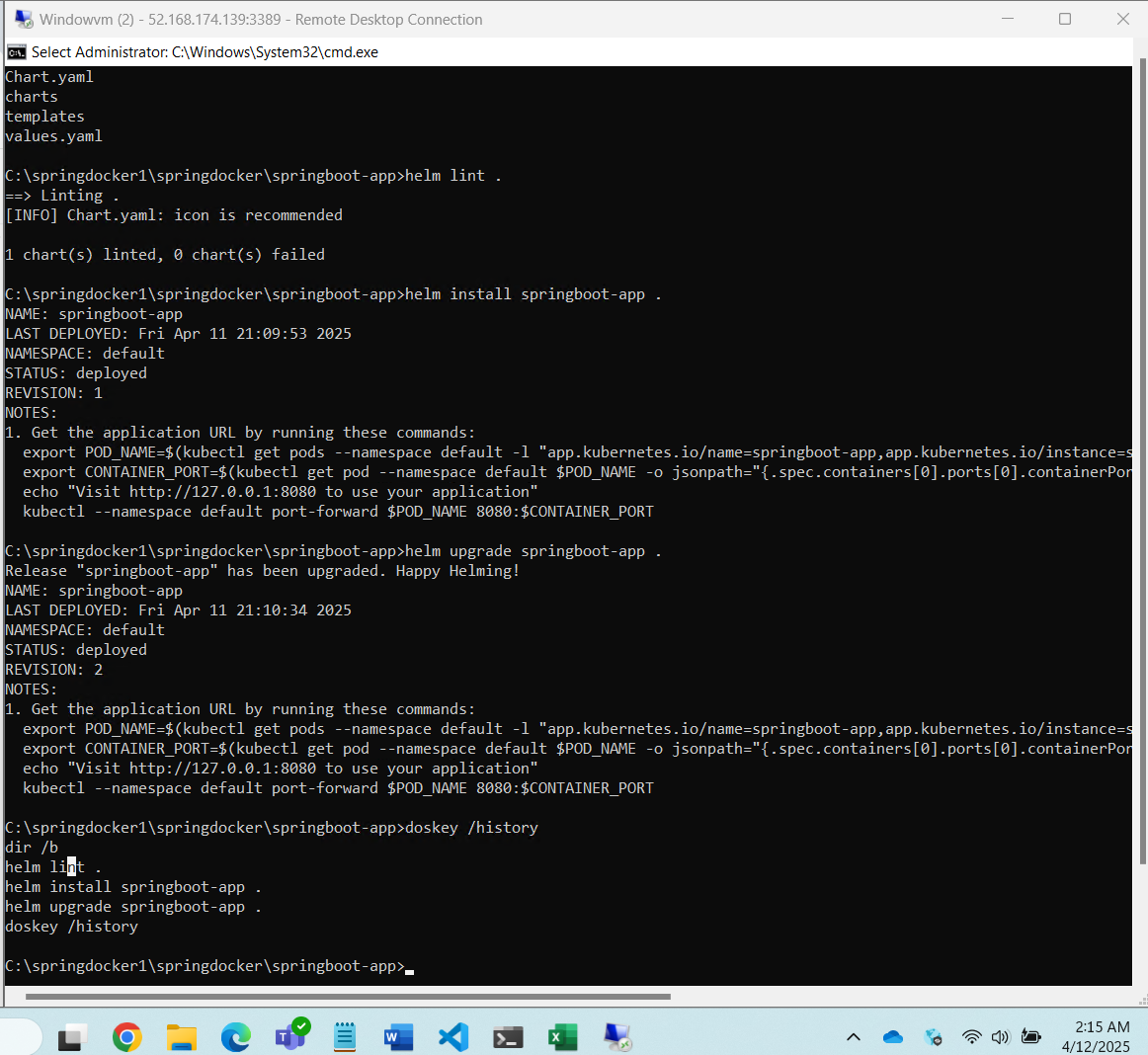
# Helm charts

Helm charts are package managers for Kubernetes applications, allowing you to define, install, and upgrade even the most complex Kubernetes apps.

1. helm upgrade springboot-app .

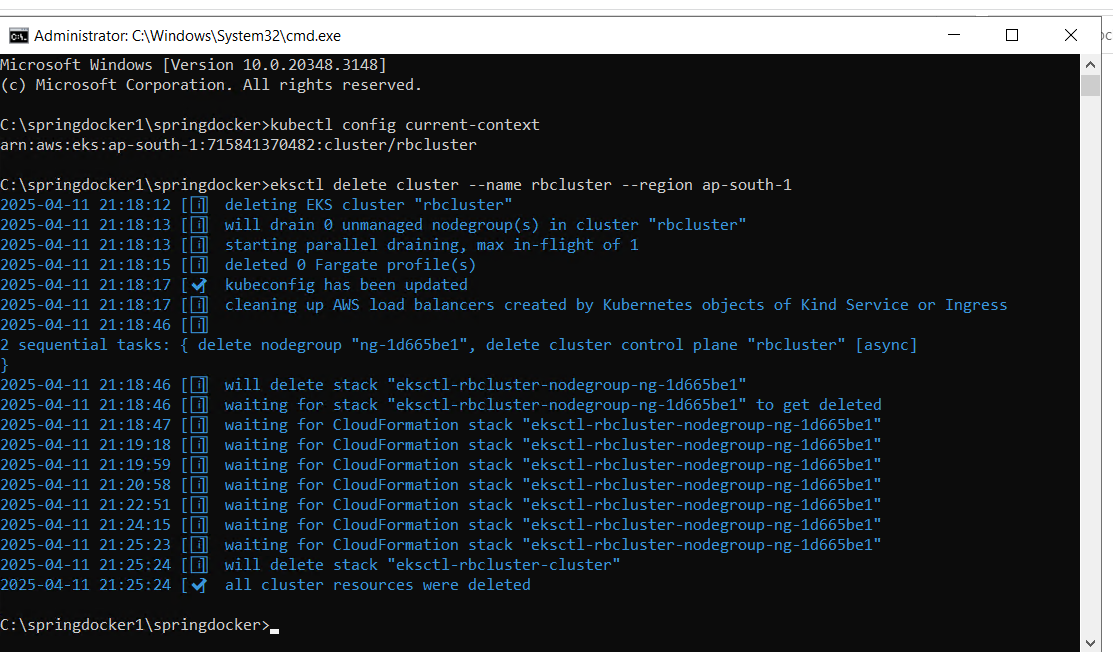






# Destroy the cluster

1. eksctl delete cluster --name rbcluster --region ap-south-1



# Terraform Kubernetes

To create an Amazon EKS (Elastic Kubernetes Service) cluster, several AWS resources are required to ensure proper networking, security, and compute capabilities.

* **VPC** – to isolate your cluster and its networking.
* **Private Subnets** – for deploying worker nodes (one per Availability Zone for high availability).
* **Public Subnet** – to allow internet access (e.g., for NAT gateway or load balancer).
* **Internet Gateway** – attached to the VPC to enable outbound internet access.
* **NAT Gateway** – to allow private subnet instances (nodes) to reach the internet.
* **Route Table** – configured to route traffic appropriately between subnets and gateways.
* **IAM Role for EKS Control Plane** – to allow EKS to manage AWS resources.
* **IAM Role for Worker Nodes** – for node instances to communicate with AWS APIs.
* **EKS Cluster** – the managed Kubernetes control plane.
* **1 Managed Node Group (EC2)** or **Fargate Profile** – to run containerized workloads.
* **Security Groups** – to control inbound/outbound traffic to cluster and nodes.
* **kubectl + aws-iam-authenticator** – on your local/dev machine to interact with the cluster.