DevOps-Project-BankApp-AKS-EKS-GKE-WAF-CloudArmor-Performance-Testing

By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

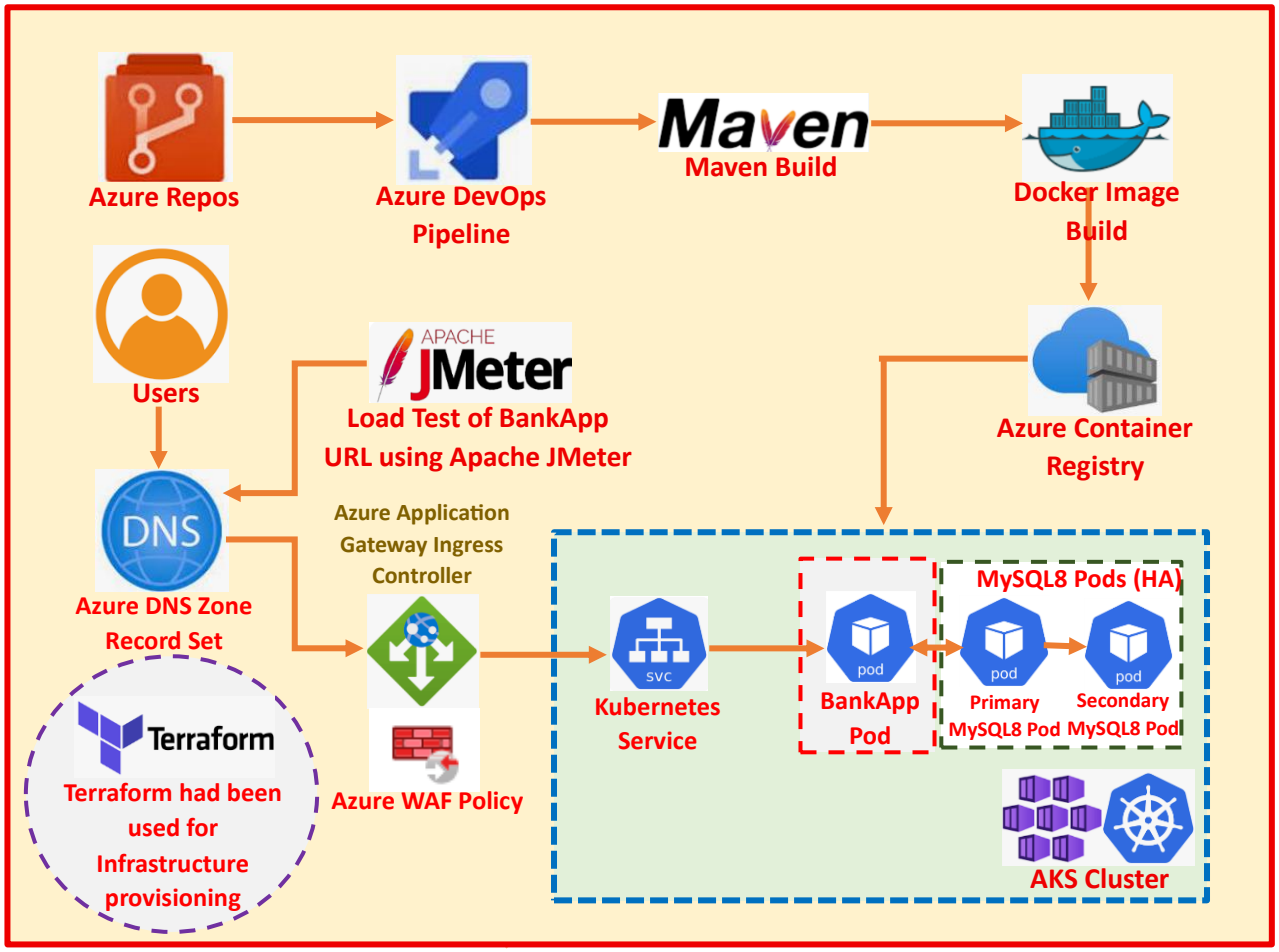
LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/

GitHub: - https://github.com/singhritesh85

या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

Module-1

DevOps-Project-BankApp-AKS-WAF-Performance-Testing



This Project deals with the creation of infrastructure using Terraform, Azure Repos was used to keep the source Code and Azure DevOps Pipeline had been used as CI/CD Tool. The build tool used in this project was Maven and Docker Image was pushed to the Azure Container Registry (ACR) which was finally deployed to the Azure Kubernetes Services (AKS). The Bank Application Pods was access using Application Gateway Ingress Controller and hence using the kubernetes service as shown in the screenshot attached above. I applied WAF Policy (to Limit the Rate) to the Bank Application URL. As this Bank Application was open to entire world so rate limit was applied through WAF Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot so that its speed becomes slow or this website hangs to refrain from such a situation WAF Policy Rate Limit was applied. Finally, I did the Performance Test (Load test) of the Bank Application URL using Apache JMeter to check how many users can access the BankApp without any hitch.

To create the infrastructure, I used Terraform and Terraform was installed on Azure VM and its state-file was kept in Azure Storage Account Container. For Azure Cloud Authentication and Authorization, I used the command **az login** before that I installed the azure-cli using the command as shown in the screenshot attached below.

```
[root@Terraform-Server ~]# yum install -y https://packages.microsoft.com/config/rhel/8/packages-microsoft-prod.rpm
```

```
[root@Terraform-Server ~]# yum install azure-cli -y
```

```
[root@Terraform-Server ~]# az login
To sign in, use a web browser to open the page [redacted] and enter the code [redacted] to authenticate.
```

Terraform script to create the infrastructure is present at the path **terraform-aks-cluster-waf-with-backup** in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. Commands as written below had been used to create infrastructure using Terraform.

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

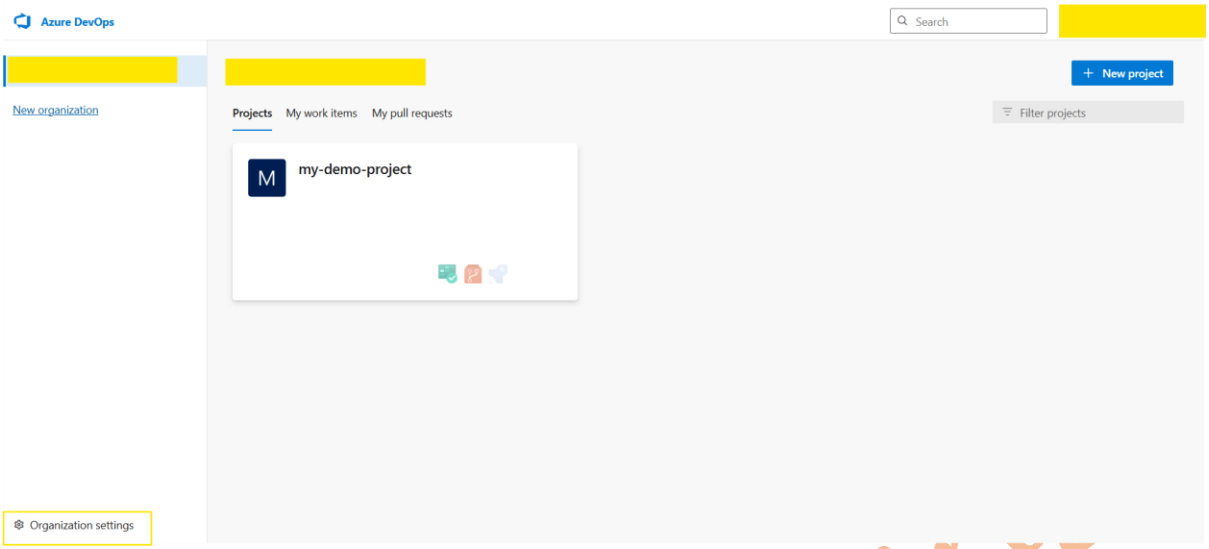
```
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [05m10s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [05m20s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [05m30s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [05m40s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [05m50s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Creation complete after 5m57s [id=/subscriptions/[redacted]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster/providers/Microsoft.KubernetesConfiguration/extensions/bankapp-extension]
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Creating...
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Creating...
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Still creating... [00m10s elapsed]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Still creating... [00m10s elapsed]
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Still creating... [00m20s elapsed]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Still creating... [00m20s elapsed]
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Creation complete after 26s [id=/subscriptions/[redacted]/resourceGroups/bankapp-rg/providers/Microsoft.Storage/storageAccounts/bankappbackupsa2025/providers/Microsoft.Authorization/roleAssignments/[redacted]]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Creation complete after 27s [id=/subscriptions/[redacted]/resourceGroups/bankapp-rg/providers/Microsoft.Storage/storageAccounts/bankappbackupsa2025/providers/Microsoft.Authorization/roleAssignments/[redacted]]
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Creating...
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Still creating... [00m10s elapsed]
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Creation complete after 17s [id=/subscriptions/[redacted]/resourceGroups/bankapp-rg/providers/Microsoft.DataProtection/backupVaults/bankapp-backup-vault/backupInstances/bankapp-backup-instance]

Apply complete! Resources: 50 added, 0 changed, 0 destroyed.

Outputs:

acr_azurevm_private_ip_and_aks_details_waf_policy_name_and_application_gateway_name = {
  "acr_login_server" = "bankappcontainer24registry.azurecr.io"
  "aks_id" = "/subscriptions/[redacted]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster"
  "aks_name" = "bankapp-cluster"
  "application_gateway_name" = "app-gtw-ingress-controller"
  "azurevm_devopsagent_privateip" = "10.[redacted]"
  "web_application_firewall_policy_name" = "bankapp-wafpolicy"
}
```

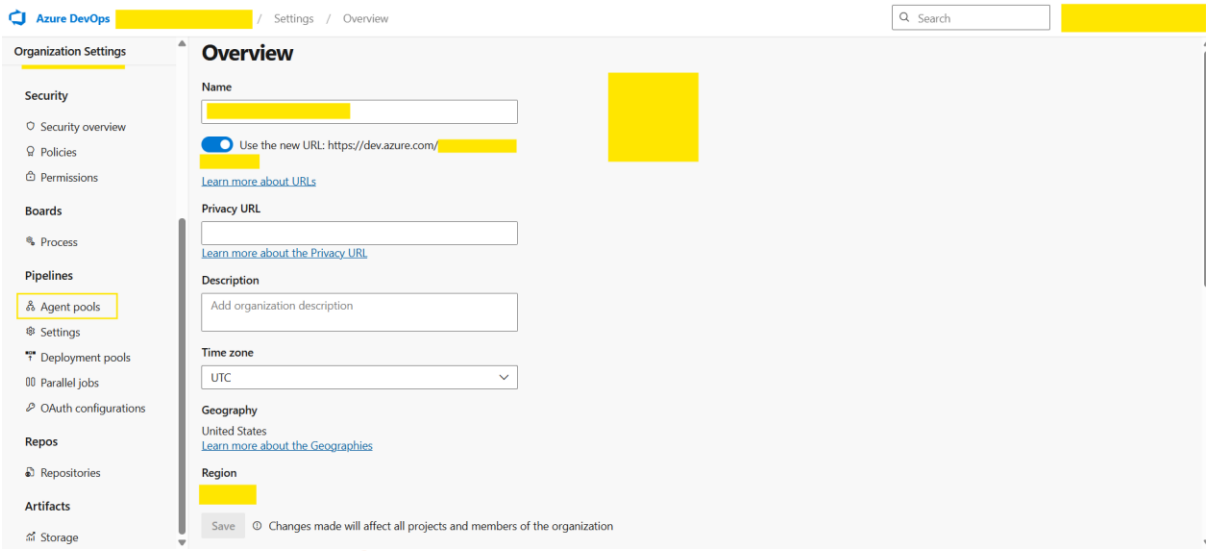
For this project I will not use SonarQube and Nexus in dev environment, you can use SonarQube and Nexus in this project. To install the same use project present in GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git.

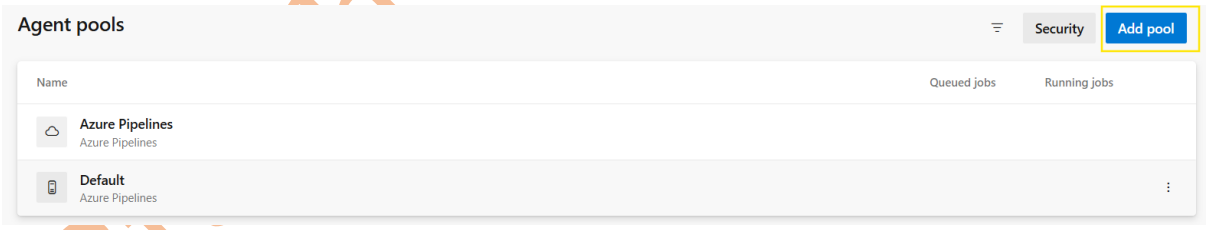
Installation of Azure DevOps Agent

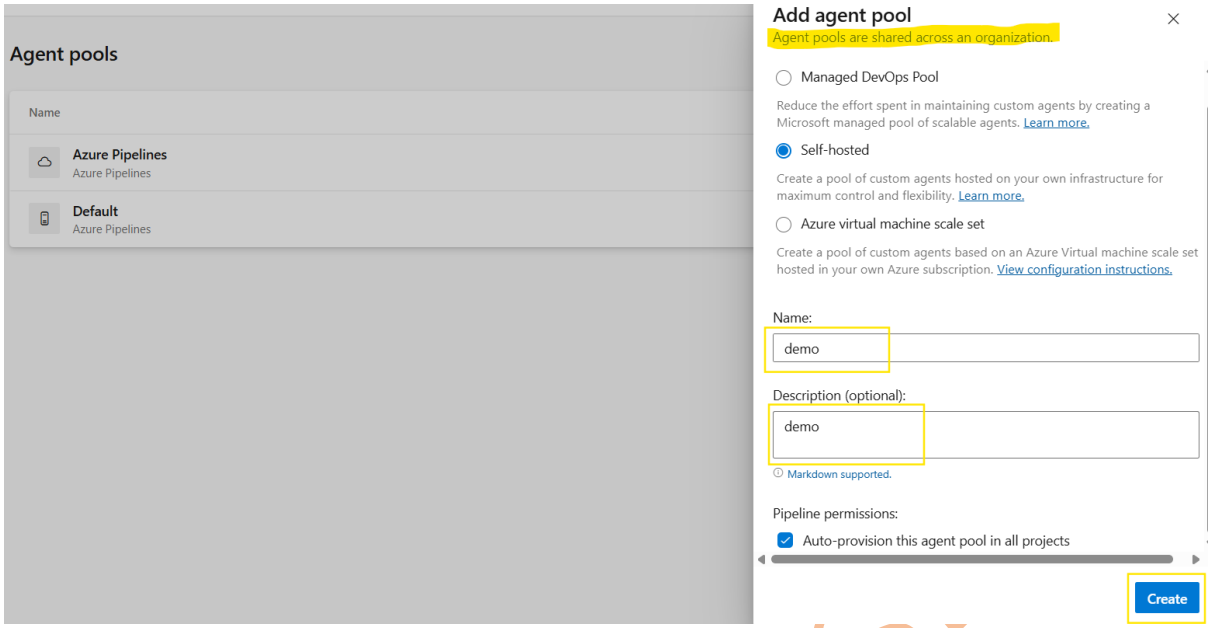
To install Azure DevOps Agent, follow the steps as written below.

To install the self-hosted agent for Azure DevOps pipeline first I opened the Azure DevOps UI and went to Organisations Settings > Agent Pools > Add Pool.

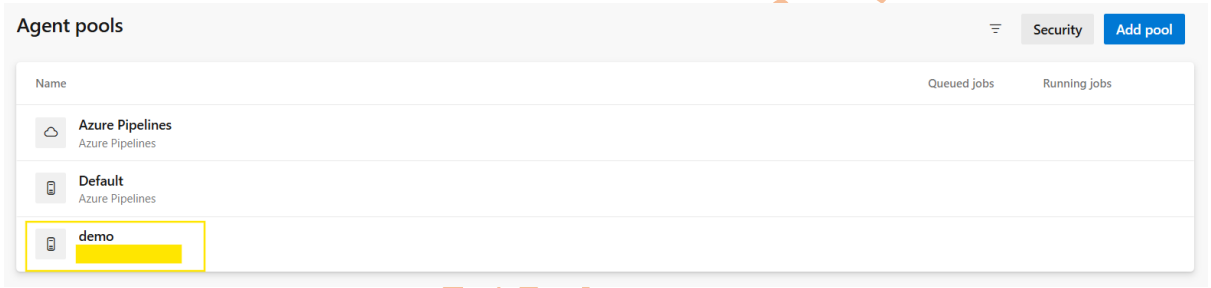

Azure DevOps
Search
New organization
New project
Projects
My work items
My pull requests
Filter projects
M
my-demo-project
Organization settings


Azure DevOps
Settings
Overview
Search
Organization Settings
Security
Security overview
Policies
Permissions
Boards
Process
Pipelines
Agent pools
Settings
Deployment pools
Parallel jobs
OAuth configurations
Repos
Repositories
Artifacts
Storage
Overview
Name
Use the new URL: https://dev.azure.com/
Learn more about URLs
Privacy URL
Learn more about the Privacy URL
Description
Add organization description
Time zone
UTC
Geography
United States
Learn more about the Geographies
Region
Save
Changes made will affect all projects and members of the organization

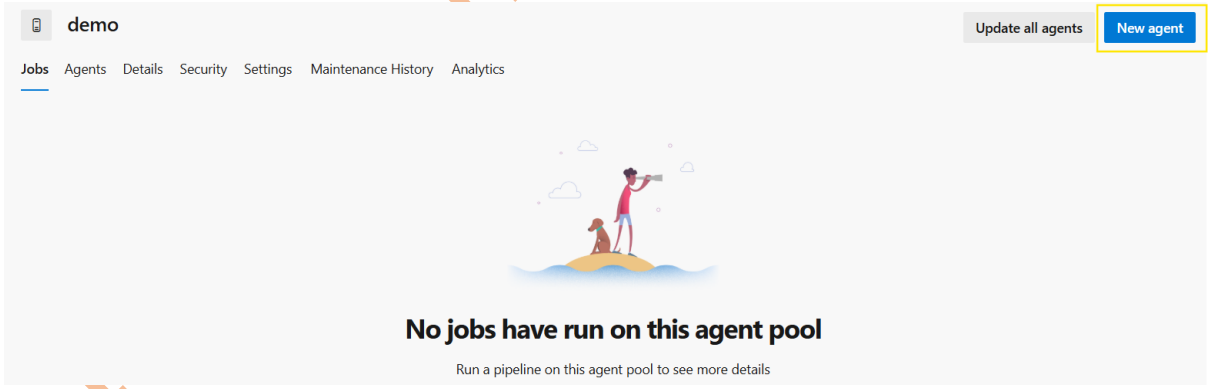

Agent pools
Security
Add pool
Name
Queued jobs
Running jobs
Azure Pipelines
Azure Pipelines
Default
Azure Pipelines



New Agent Pool had been added as shown in the screenshot attached below.



Then added the agent as shown in the screenshot attached below.



For Azure DevOps Pipeline I had used Self-hosted-Agent and followed the below procedure to install it.

```
[root@devopsagent-vm ~]# cd /opt && mkdir myagent && cd myagent
[root@devopsagent-vm myagent]# wget https://vstsagentpackage.azureedge.net/agent/[redacted]/vsts-agent-linux-x64-[redacted].tar.gz
[root@devopsagent-vm myagent]# tar -xvf vsts-agent-linux-x64-[redacted].tar.gz
[root@devopsagent-vm myagent]# rm -f vsts-agent-linux-x64-[redacted].tar.gz
[root@devopsagent-vm myagent]# ./bin/installdependencies.sh
```

```
[demo@devopsagent-vm myagent]$ sudo chown -R demo:demo /opt/myagent/
```

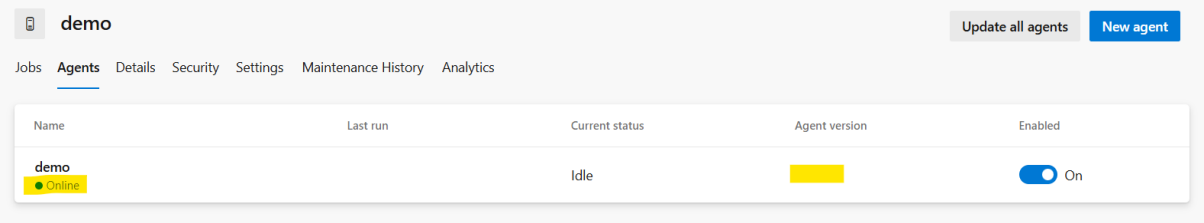
```
[demo@devopsagent-vm myagent]$ ./config.sh

  ___                  _____ _                 _
 / _ \                |  __ (_)               | (_)
/ /_\ \_____  _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \ __|
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                    | |
       agent [redacted]             |_|          [redacted]

>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sou
rces from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
  /opt/myagent/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > Y

>> Connect:

Enter server URL > [redacted]
Enter authentication type (press enter for PAT) >
Enter personal access token > [redacted]
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > demo
Enter agent name (press enter for devopsagent-vm) > demo
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2025[redacted]: Settings Saved.
```

```
[demo@devopsagent-vm myagent]$ ./env.sh
[demo@devopsagent-vm myagent]$
[demo@devopsagent-vm myagent]$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/opt/sonar-scanner/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/opt/dependency-check/bin:/usr/local
/bin
```

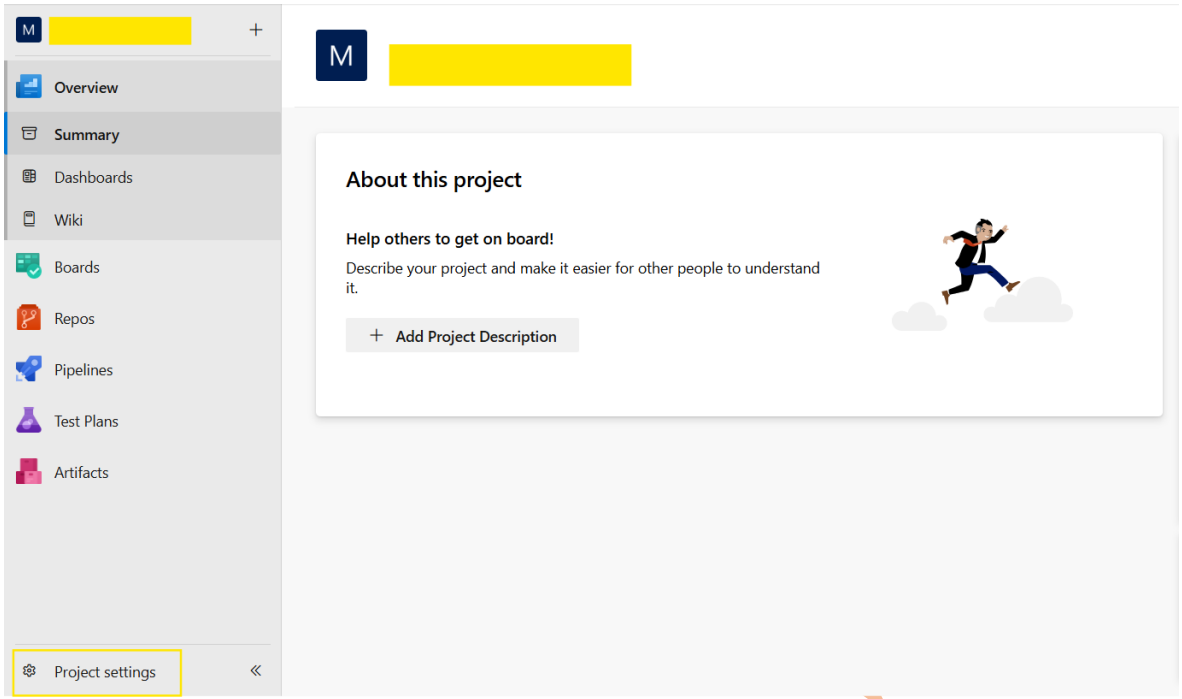
```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh install
```

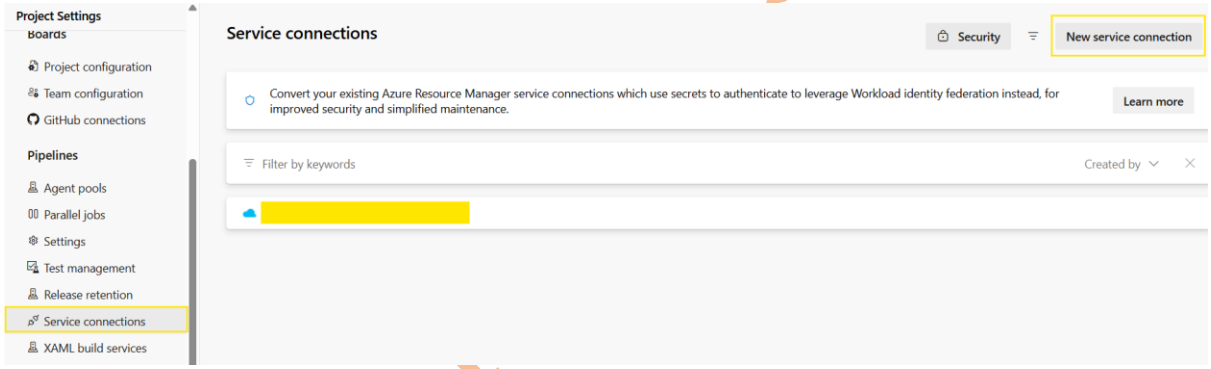
```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh start
```

Now the Agent came in online state as shown in the screenshot attached below.



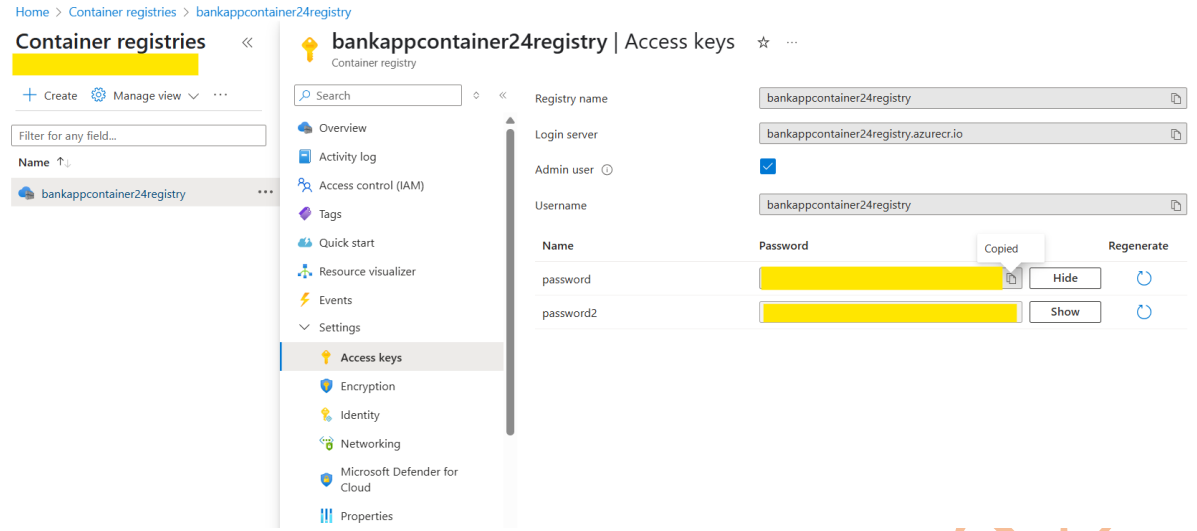
Then I went into the **Azure DevOps project > Project Settings** and created the **Service connections** for Docker Registry as shown in the screenshot attached blow.

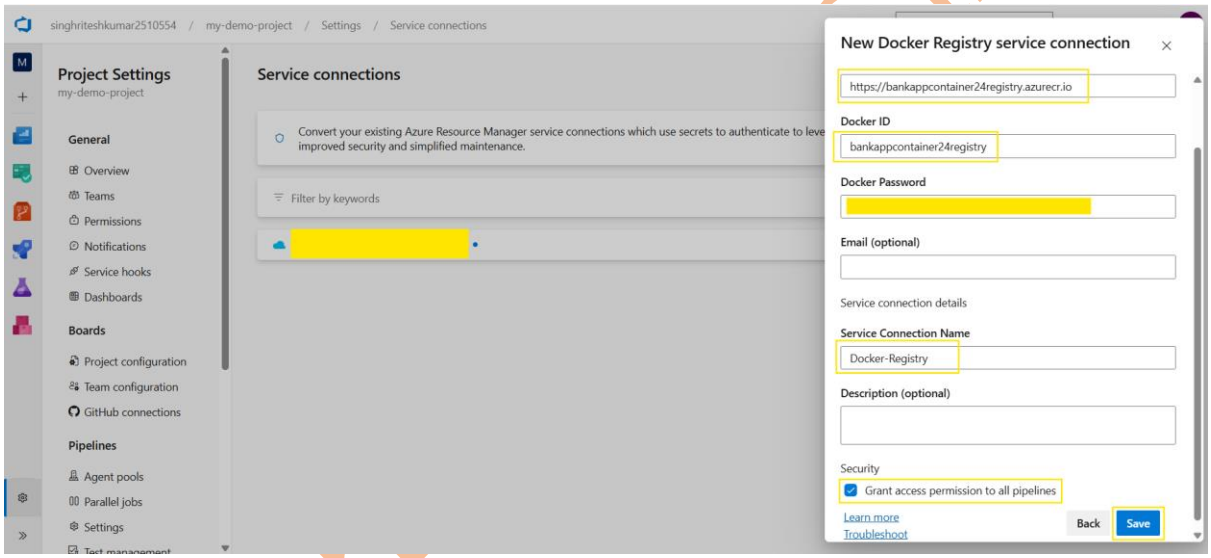




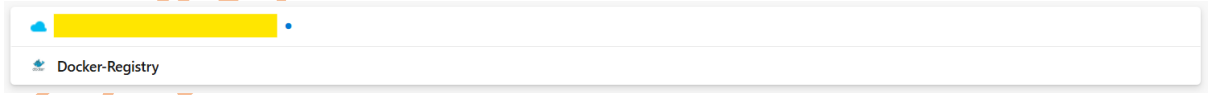
The Azure Container Registry which I had created for this project is as shown in the screenshot attached below.

Then I had created the Service Connection for Docker Registry as shown in the screenshot attached below.

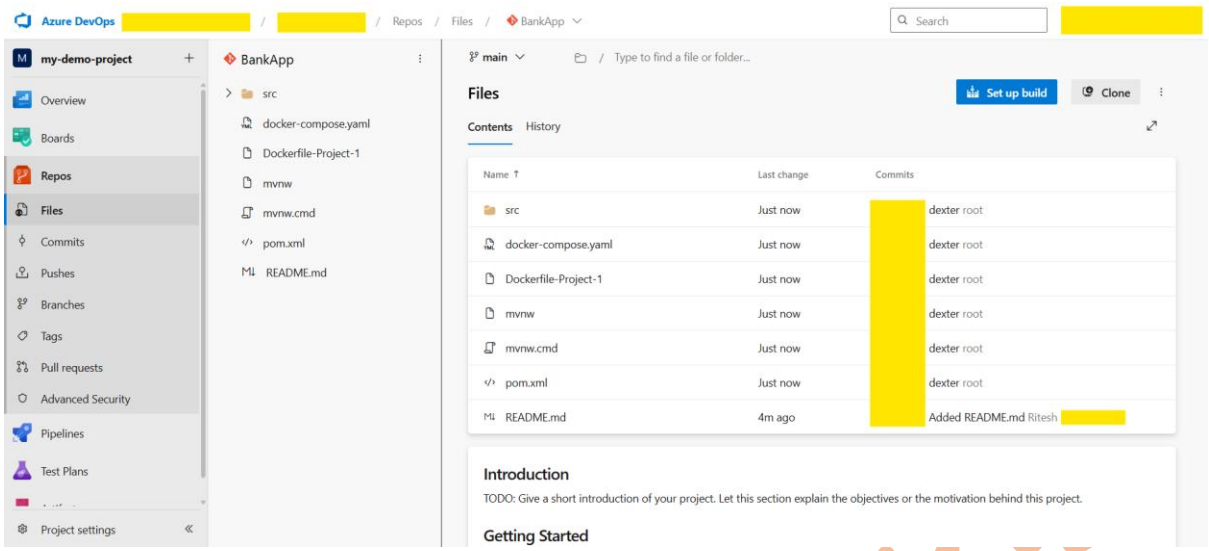




Finaly the service connections had been created as shown in the screenshot attached below.



The source Code to deploy the BankApp is present in the Azure Repos as shown in the screenshot attached below.



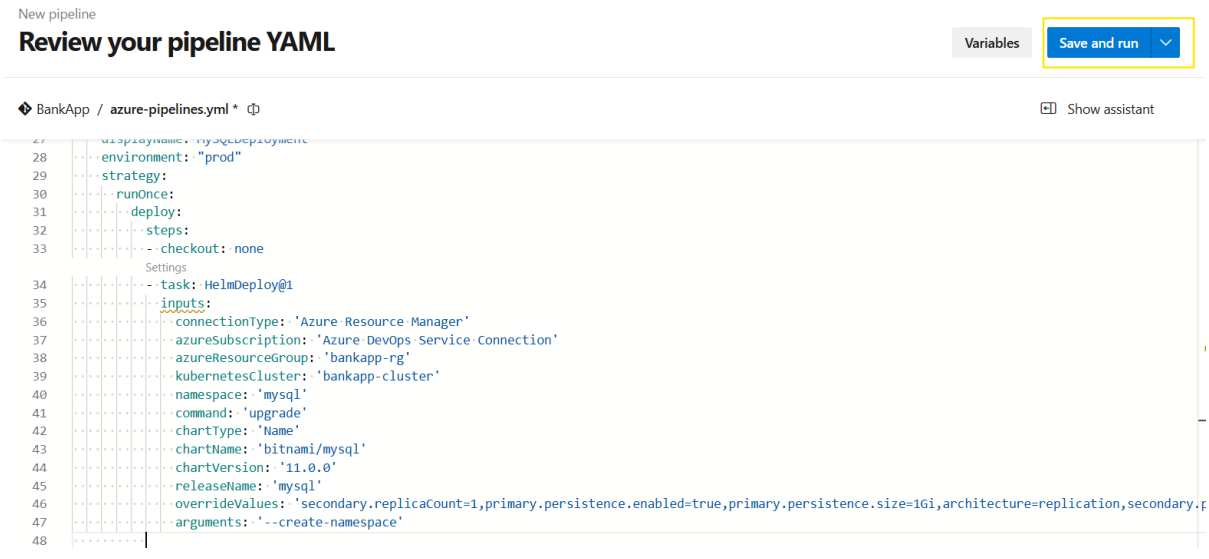
To deploy the source code in Azure DevOps I created the Azure DevOps Pipeline as shown in the screenshot attached below. Before creating Application Pods I created MySQL 8 Pods in Azure DevOps using Azure DevOps Pipeline as shown in the screenshot attached below.

Go to Azure DevOps Organisation > Project > Pipelines > Create Pipeline > Azure Repos Git (Select the Azure Repo where Source Code was present) > Starter Pipeline and Provide the below **azure-pipelines.yml**

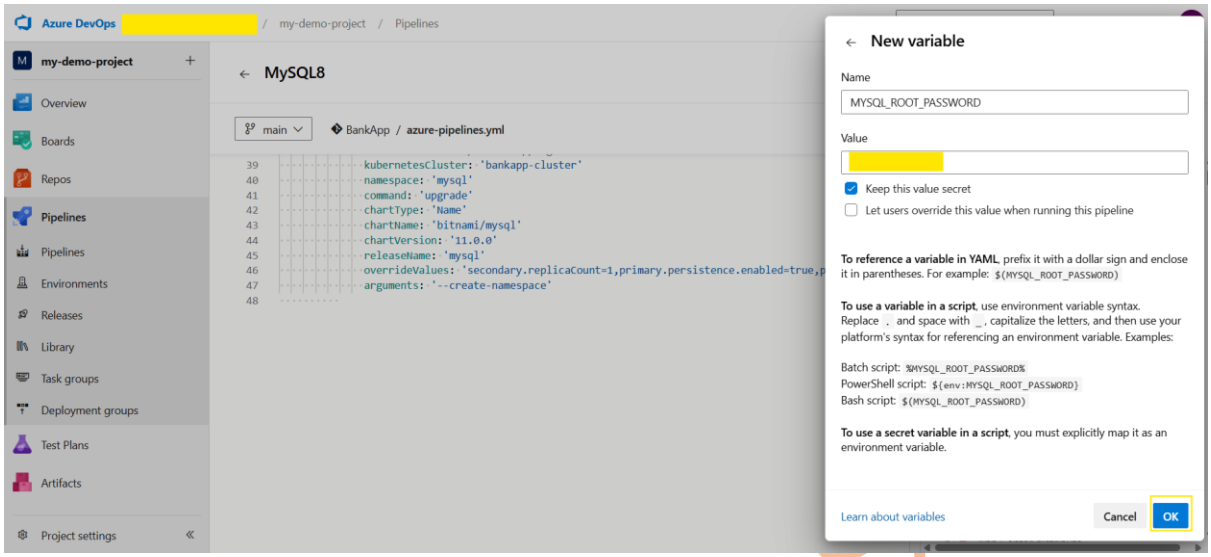
```
trigger:
- none

pr: none
pool:
  name: demo
  demands:
  - agent.name -equals demo

stages:
- stage: BitnamiHelmRepoAdd
  displayName: BitnamiHelmRepoAdd
  jobs:
  - job: BitnamiHelmRepoAdd
    displayName: BitnamiHelmRepoAdd
    steps:
    - task: CmdLine@2
      inputs:
        script: |
          helm repo add bitnami https://charts.bitnami.com/bitnami
          helm repo update
- stage: MySQLDeployment
  displayName: MySQLDeployment
  dependsOn: BitnamiHelmRepoAdd
  jobs:
  - deployment: MySQLDeployment
    displayName: MySQLDeployment
    environment: "dev"
    strategy:
      runOnce:
        deploy:
```

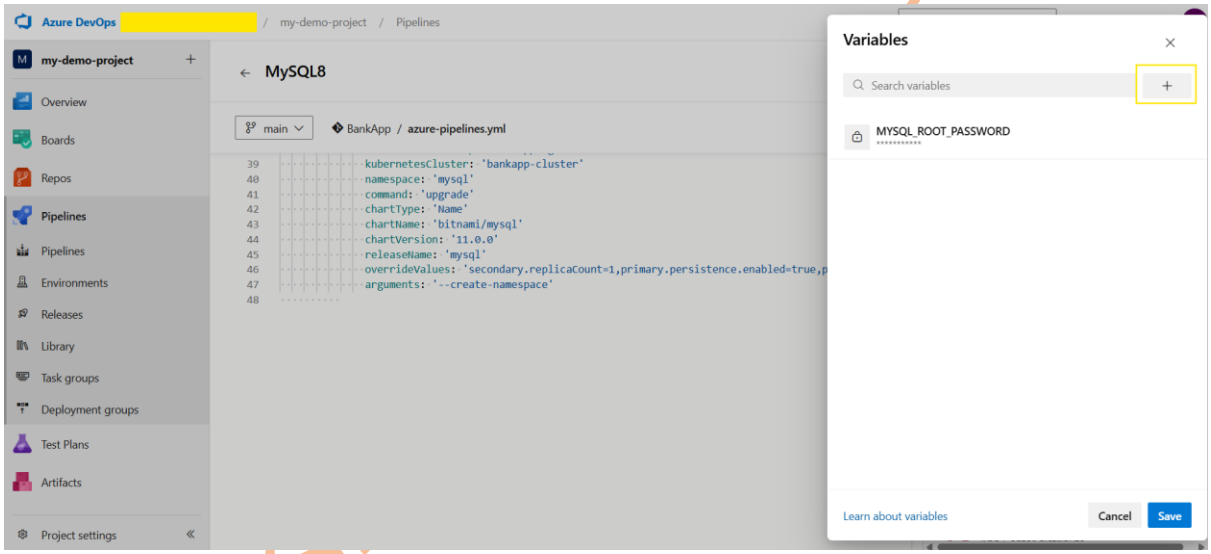
```
steps:
- checkout: none
- task: HelmDeploy@1
  inputs:
    connectionType: 'Azure Resource Manager'
    azureSubscription: 'Azure DevOps Service Connection'
    azureResourceGroup: 'bankapp-rg'
    kubernetesCluster: 'bankapp-cluster'
    namespace: 'mysql'
    command: 'upgrade'
    chartType: 'Name'
    chartName: 'bitnami/mysql'
    chartVersion: '11.0.0'
    releaseName: 'mysql'
    overrideValues:
'secondary.replicaCount=1,primary.persistence.enabled=true,primary.persistence.size=1Gi,architecture=replication,secondary.persistence.enabled=true,secondary.persistence.size=1Gi,global.storageClass=managed-csi,primary.service.type=ClusterIP,auth.rootPassword=$(MYSQL_ROOT_PASSWORD),auth.database=$(MYSQL_DATABASE)'
    arguments: '--create-namespace'
```

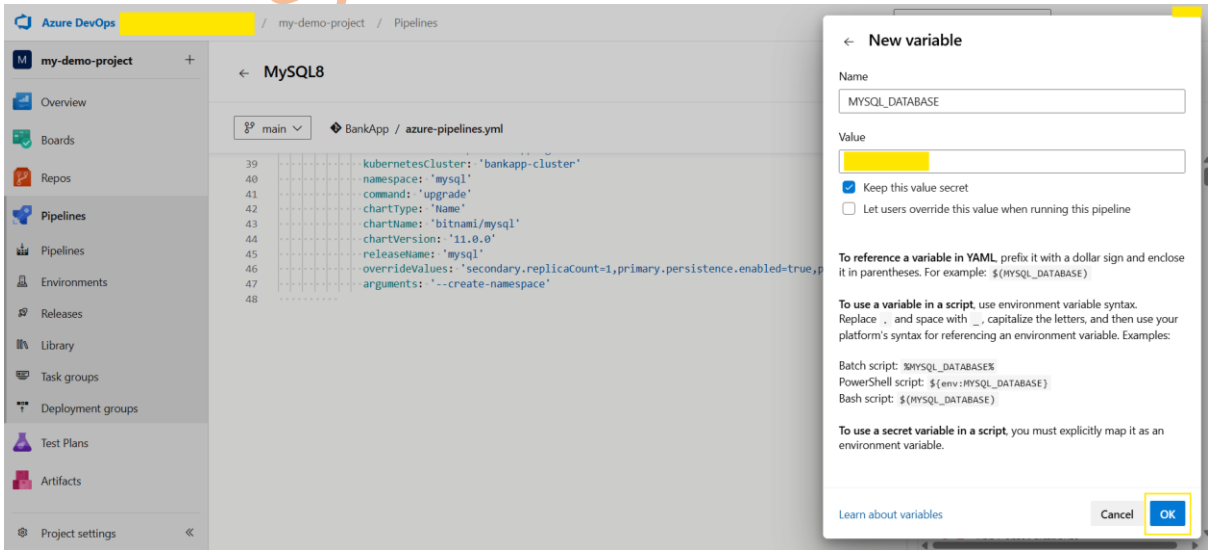
Then click on Validate and Save as shown in the screenshot attached below.



I created two variables in Azure DevOps Pipeline named as MYSQL_ROOT_PASSWORD and MYSQL_DATABASE which is as shown below.







Finally, the MySQL 8 Pods and PVC was created as shown in the screenshot attached blow.

```
[demo@devopsagent-vm ~]$ kubectl get pods -n mysql --watch
NAME                 READY   STATUS    RESTARTS   AGE
mysql-primary-0      1/1     Running   0          71s
mysql-secondary-0    1/1     Running   0          71s
```

```
[demo@devopsagent-vm ~]$ kubectl get pvc -n mysql --watch
NAME                   STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
data-mysql-primary-0   Bound    pvc-[redacted]   1Gi        RWO            managed-csi    <unset>                 2m10s
data-mysql-secondary-0 Bound    pvc-[redacted]   1Gi        RWO            managed-csi    <unset>                 2m10s
```

Screenshot for Azure DevOps Pipeline after successful execution of the Pipeline is as shown in the screenshot attached below.

Now to deploy the Bank Application Pods similarly go to Azure DevOps Organisation > Project > Pipelines > New Pipeline > Azure Repos Git (Select the Azure Repo where source code was present) > Starter Pipeline and provide the azure-pipelines-1.yml file as provided below then click on Save and Run as shown in the screenshot attached below.

```
trigger:
- main

pr: none
pool:
  name: demo
  demands:
  - agent.name -equals demo

variables:
  imagePullSecret: 'bankapp-auth'

stages:
- stage: "Build"
  displayName: Build
  jobs:
  - job: "Build"
    displayName: Build
    steps:
    - task: Maven@4
      inputs:
        mavenPomFile: 'pom.xml'
        goals: 'clean install'
        publishJUnitResults: false
        javaHomeOption: 'JDKVersion'
        mavenVersionOption: 'Default'
        mavenAuthenticateFeed: false
        effectivePomSkip: false
        sonarQubeRunAnalysis: false
- stage: DockerImageBuild
  displayName: DockerImageBuild
```

```
  dependsOn: "Build"
  jobs:
  - job: DockerImageBuild
    displayName: DockerImageBuild
    steps:
    - checkout: none
    - task: CmdLine@2
      inputs:
        script: |
          docker system prune -f --all
          docker build -t bankappcontainer24registry.azurecr.io/samplewebapp:$(Build.BuildId) -
f Dockerfile-Project-1 .
    - task: Docker@2
      inputs:
        containerRegistry: 'Docker-Registry'
        repository: 'samplewebapp'
        command: 'buildAndPush'
        Dockerfile: '**/Dockerfile-Project-1'
- stage: KubernetesDeployment
  displayName: KubernetesDeployment
  dependsOn: DockerImageBuild
  jobs:
  - deployment: KubernetesDeployment
    displayName: KubernetesDeployment
    environment: "dev"
    strategy:
      runOnce:
        deploy:
          steps:
          - checkout: none
          - task: HelmDeploy@1
            inputs:
```

connectionType: 'Azure Resource Manager'

azureSubscription: 'Azure DevOps Service Connection'

azureResourceGroup: 'bankapp-rg'

kubernetesCluster: 'bankapp-cluster'

namespace: 'bankapp'

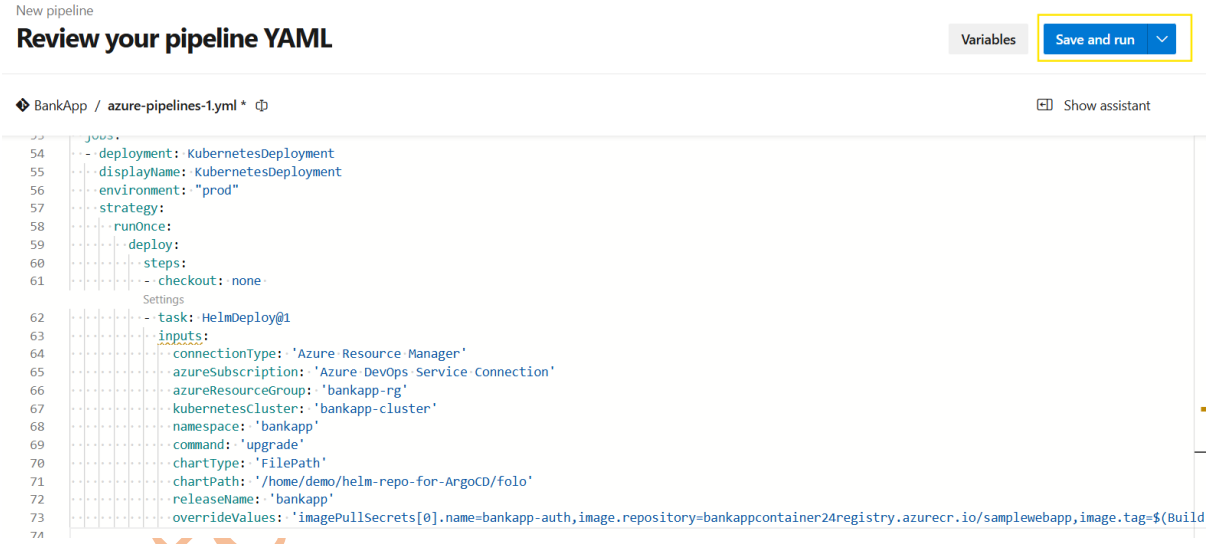
command: 'upgrade'

chartType: 'FilePath'

chartPath: '/home/demo/helm-repo-for-ArgoCD/folo'

releaseName: 'bankapp'

overrideValues: 'imagePullSecrets[0].name=bankapp-auth,image.repository=bankappcontainer24registry.azurecr.io/samplewebapp,image.tag=$(Build.BuildId),replicaCount=1,service.type=ClusterIP,service.port=80'



Before running this pipeline make sure I had created the Kubernetes Secrets to pull docker image from Azure Container Registry (ACR)

kubectl create ns bankapp && kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappcontainer24registry.azurecr.io --docker-username=bankappcontainer24registry --docker-password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -n bankapp

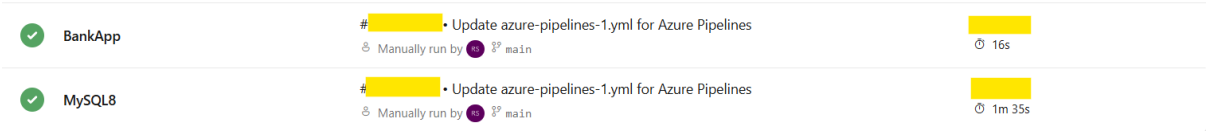
```
[demo@devopsagent-vm ~]$ kubectl create ns bankapp && kubectl create secret docker-registry boardgame-auth --docker-server=https://bankappcontainer24registry.
azurecr.io --docker-username=bankappcontainer24registry --docker-password=                                                                    -n bankapp
namespace/bankapp created
secret/boardgame-auth created
```

```
[demo@devopsagent-vm ~]$ kubectl get pods -n bankapp --watch
NAME                              READY   STATUS    RESTARTS   AGE
bankapp-folo-                     1/1     Running   0          3m47s
```

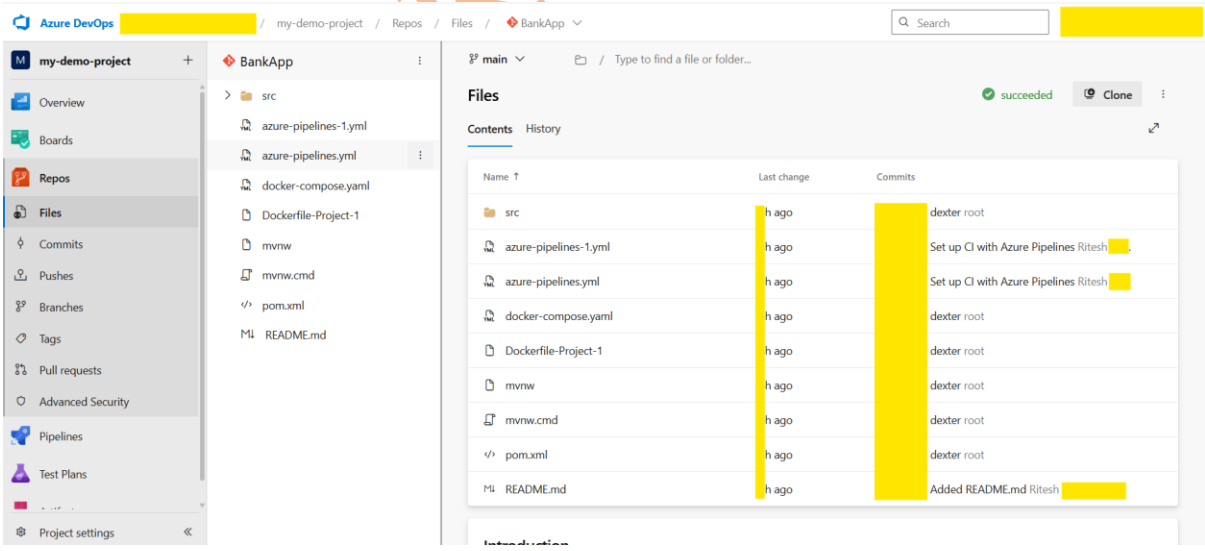
Now create the Ingress Rule as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp
secret/ingress-tls created
[demo@devopsagent-vm ~]$ cat ingress-rule.yaml
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-tls
  rules:
  - host: bankapp.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-folo
            port:
              number: 80
[demo@devopsagent-vm ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/bankapp-ingress created
[demo@devopsagent-vm ~]$ kubectl get ing -A --watch
NAMESPACE   NAME              CLASS                    HOSTS                    ADDRESS         PORTS     AGE
bankapp     bankapp-ingress   azure-application-gateway   bankapp.singhritesh85.com   4.[redacted].127   80, 443   7s
```

The screenshot for Azure DevOps Pipeline after successful execution of Azure DevOps Pipeline is as shown in the screenshot attached below.



The screenshot for Azure Repos after successful execution of the pipeline is as shown in the screenshot attached below.



cat ingress-rule.yaml

kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp

---

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: bankapp-ingress
 namespace: bankapp
 annotations:
  appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
 ingressClassName: azure-application-gateway
 tls:
 - secretName: ingress-tls
 rules:
 - host: bankapp.singhritesh85.com
  http:
   paths:
   - path: /
    pathType: Prefix
    backend:
     service:
      name: bankapp-folo
      port:
       number: 80
```

I created the Azure DNS Zone using the terraform script present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git at the path **terraform-azure-dns-zone**. The creation of Azure DNS Zone is not included in day-to-day activity but for your use I provided terraform script to create it.

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

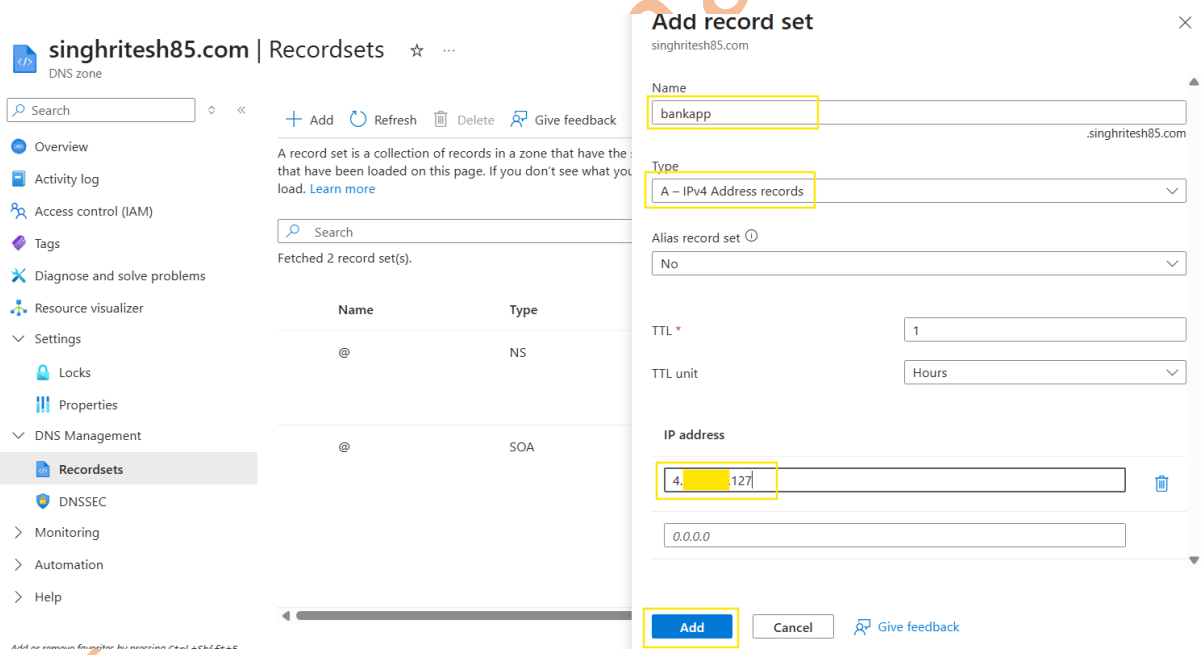
terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

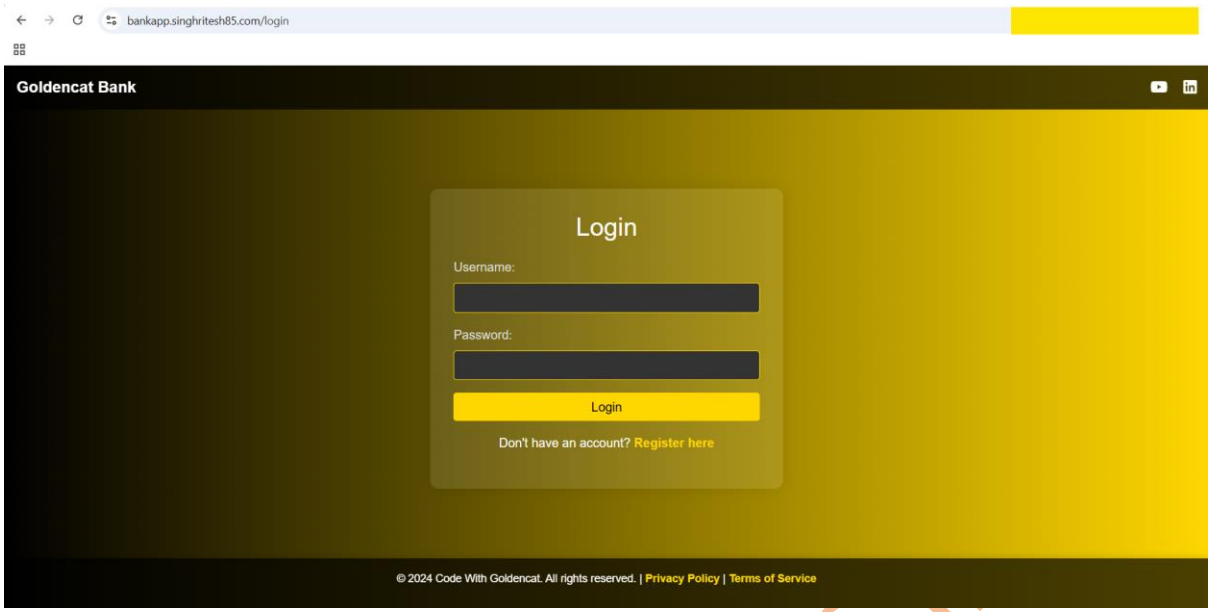
Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

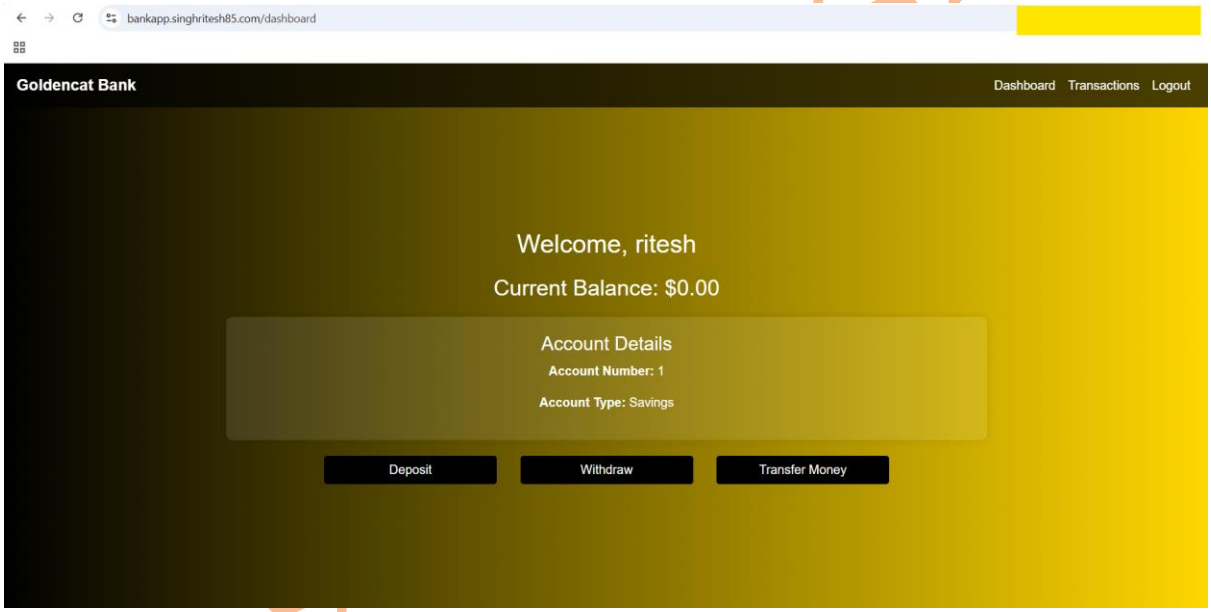
```
  + resource "azurerm_resource_group" "dns_rg" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "dns-rosource-group"
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + azure_dns_zone_name_and_nameserver = {
      + dns_zone_name = "singhritesh85.com"
      + name_servers  = (known after apply)
    }
module.dns.azurerm_resource_group.dns_rg: Creating...
module.dns.azurerm_resource_group.dns_rg: Still creating... [00m10s elapsed]
module.dns.azurerm_resource_group.dns_rg: Creation complete after 13s [id=/subscriptions/[redacted]/resourceGroups/dns-rosource-grou
p]
module.dns.azurerm_dns_zone.dns_zone: Creating...
module.dns.azurerm_dns_zone.dns_zone: Creation complete after 7s [id=/subscriptions/[redacted]/resourceGroups/dns-rosource-group/pro
viders/Microsoft.Network/dnsZones/singhritesh85.com]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

azure_dns_zone_name_and_nameserver = {
  "dns_zone_name" = "singhritesh85.com"
  "name_servers" = toset([
    "[redacted]",
    "[redacted]",
    "[redacted]",
    "[redacted]",
  ])
}
```

Then I did the entry for Public IP obtained from ingress-rule as shown in the screenshot attached above with HOST to create the record set of A-Type.



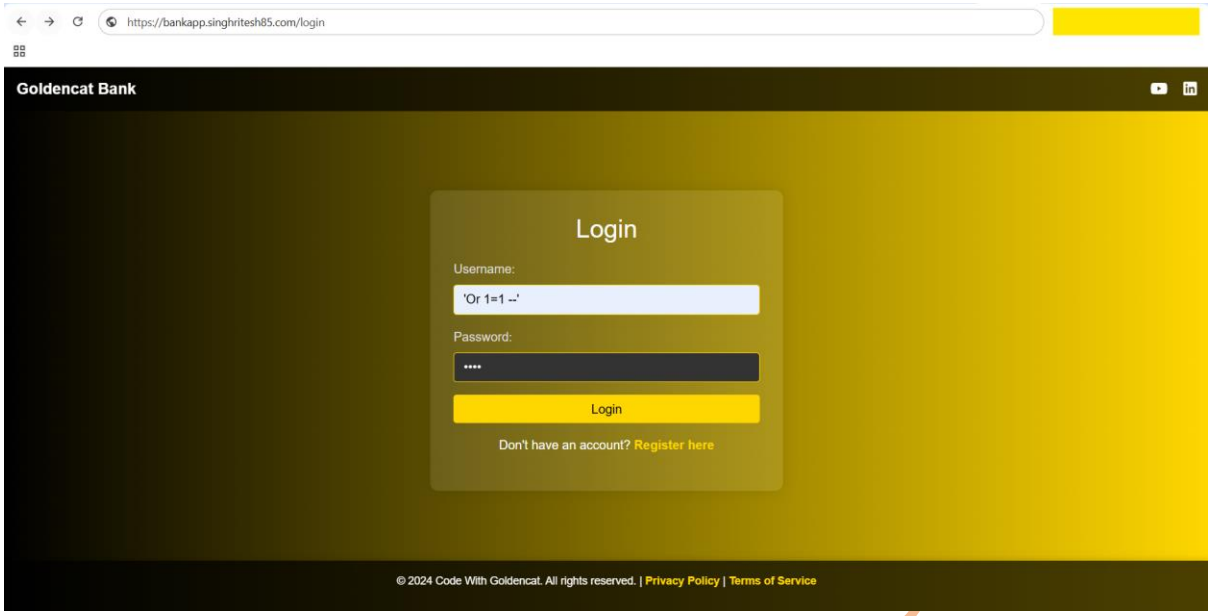
Finally, I was able to access the Bank Application as shown in the screenshot attached below.





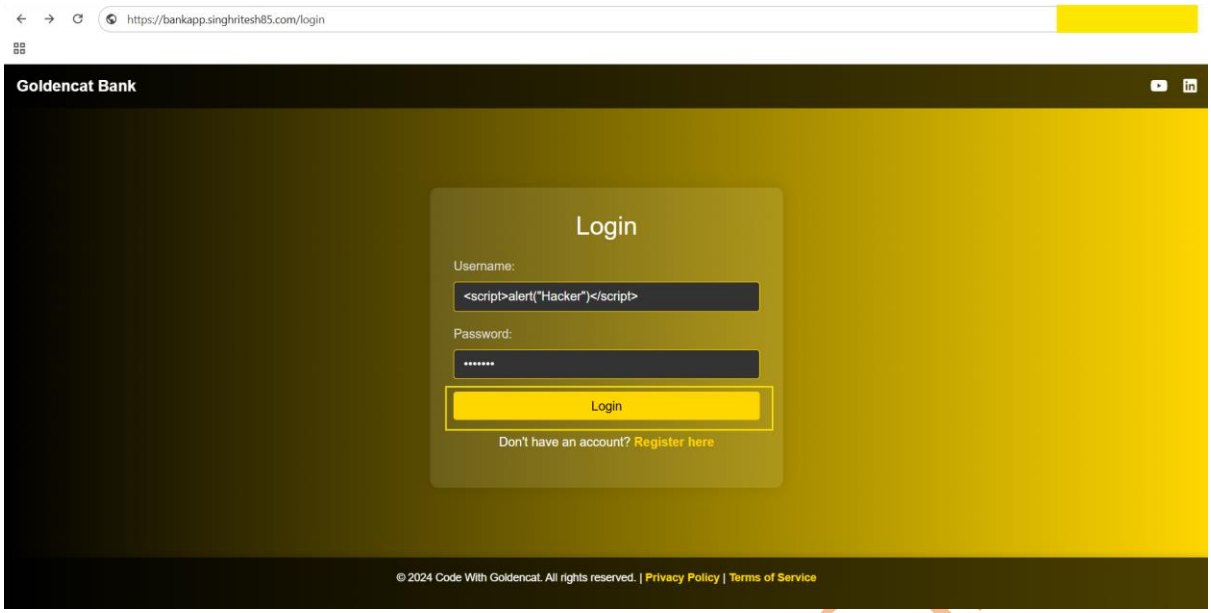
I accessed it more than 40 times in a minute and found it was block as per the WAF Policy as shown in the screenshot attached below.

This WAF Policy also protect from SQL Injection attack as shown in the screenshot attached below.



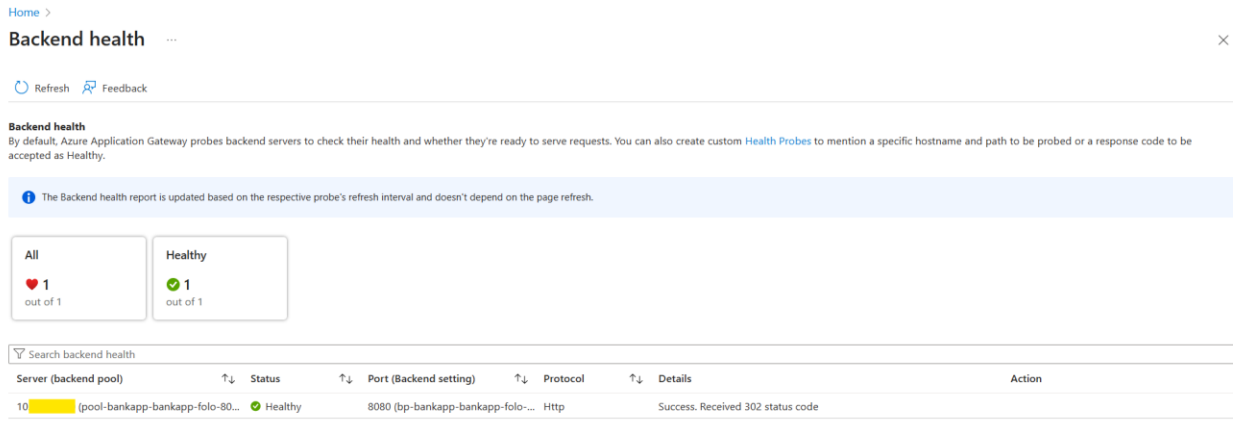
As shown in the screenshot attached above, instead of username I used the command 'OR 1=1--' (which indicates condition is always true and – is comment indicator in SQL) but found it was blocked, the Mode for WAF was used as prevention mode.

This WAF Policy also protects from XSS (Cross-Site Scripting) attack as shown in the screenshot attached below.



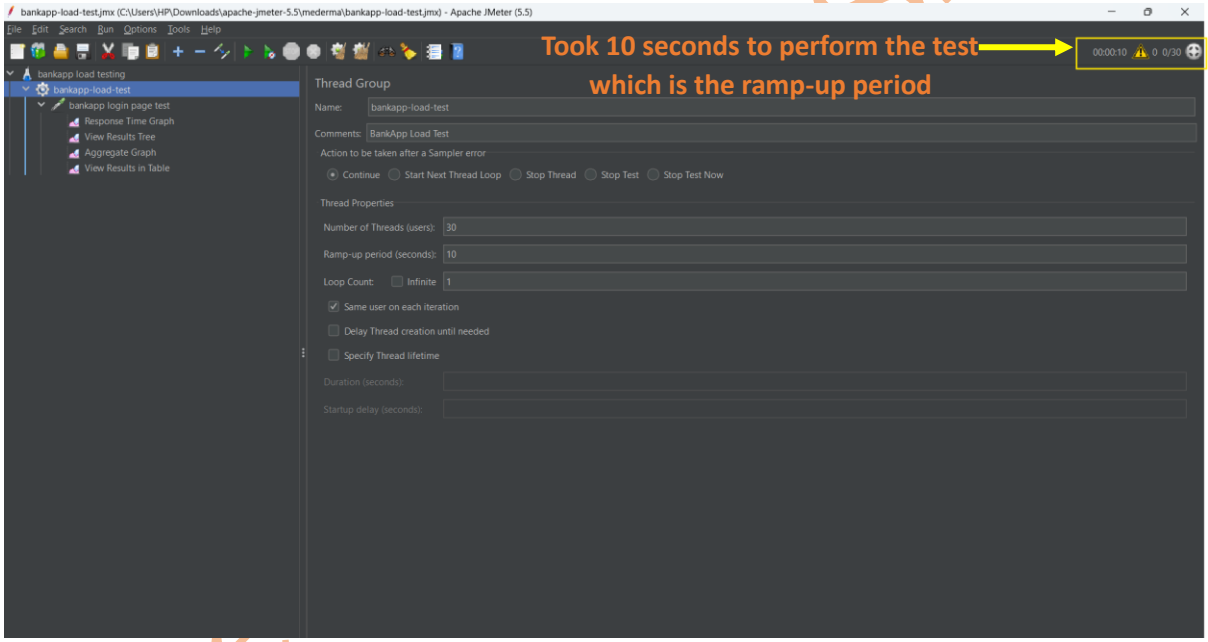
Reflected XSS happens when malicious script is part of the request as shown in the screenshot attached above. It should be reflected in the Server Response but as shown in the screenshot attached below, it was blocked with 403 Error (which means Server understood your request but refused to Grant access).

The Backend health of Application LoadBalancer is as shown in the screenshot attached below.

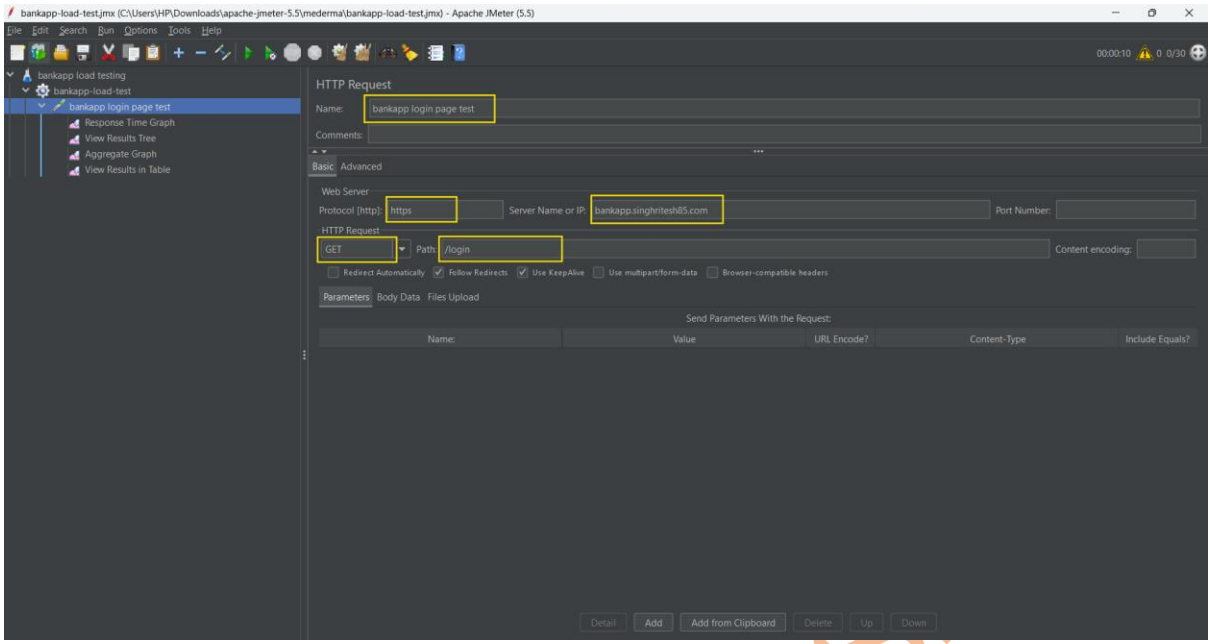


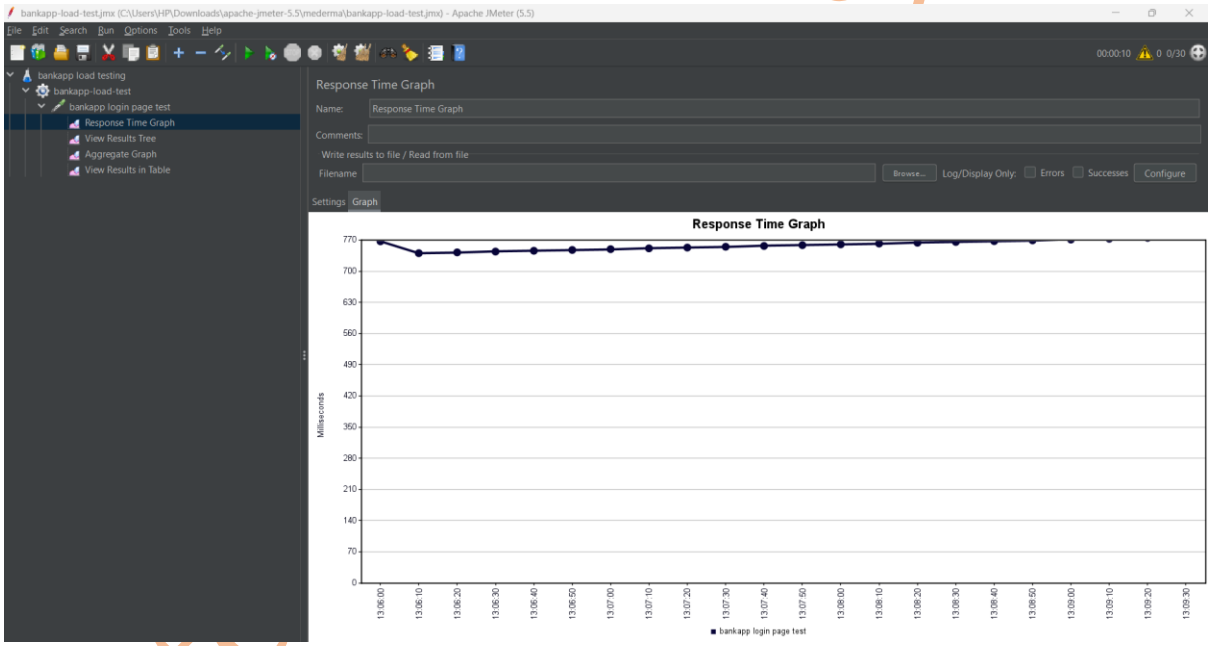
I performed the Load Test on the BankApp URL https://bankapp.singhritesh85.com/login using Apache JEMETER which results is as shown in the screenshot attached below.

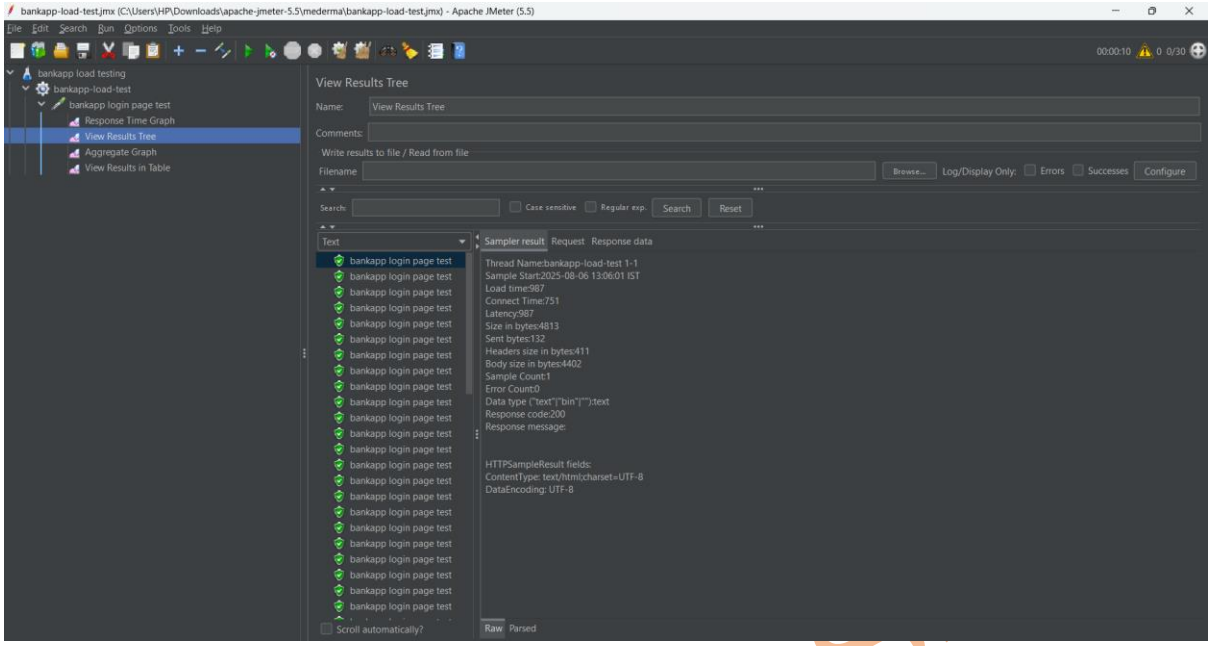
For this test I used Load Test for simultaneous users equal to 30 and below is the result.

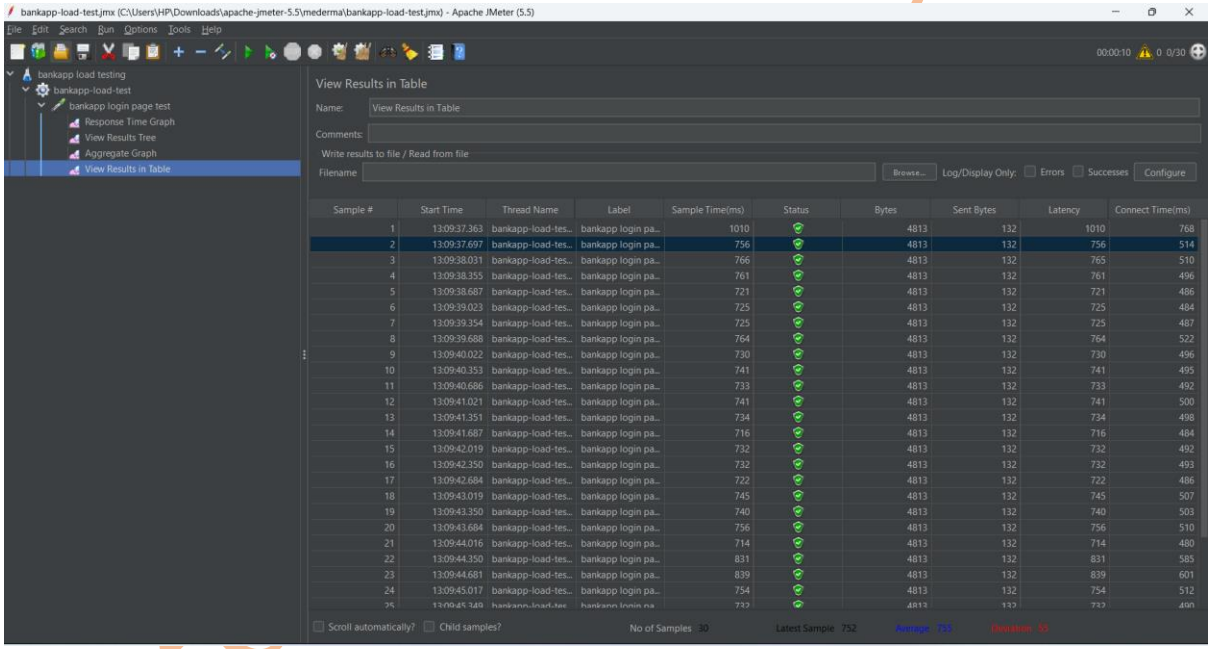


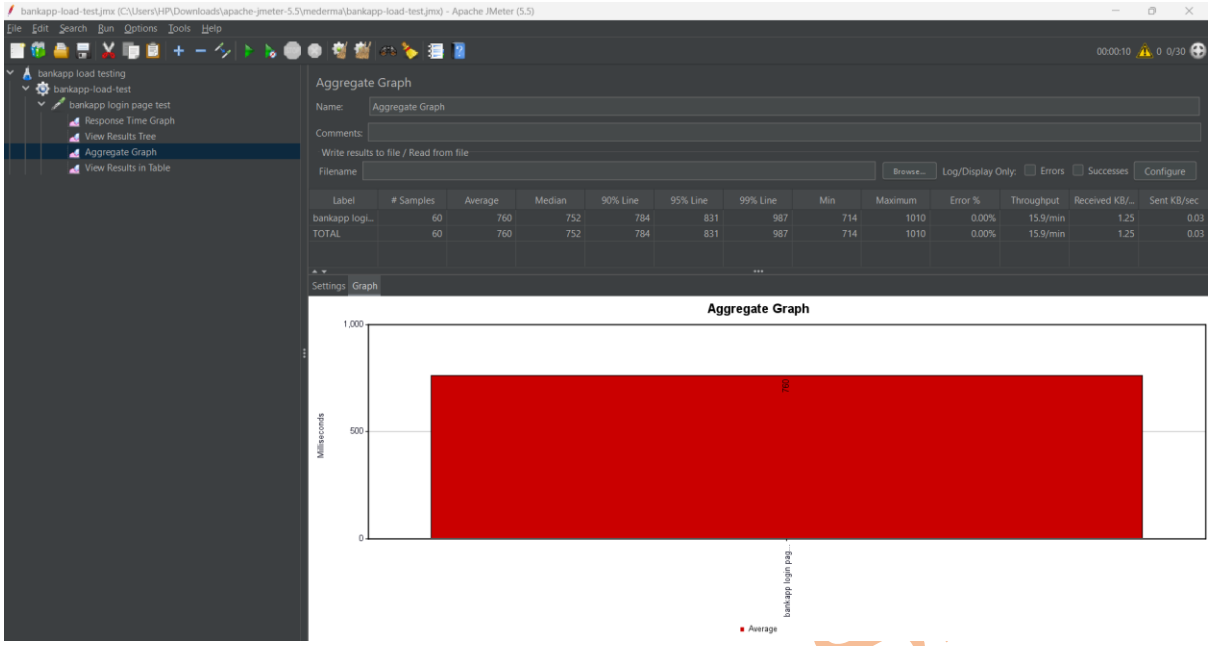
If you need, you can perform this Test in Infinite Loop but for this project I am running it for a single iteration.


bankapp-load-test.jmx (C:\Users\HP\Downloads\apache-jmeter-5.5\mederma\bankapp-load-test.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
bankapp load testing
bankapp-load-test
bankapp login page test
Response Time Graph
View Results Tree
Aggregate Graph
View Results in Table
HTTP Request
Name: bankapp login page test
Comments:
Basic Advanced
Web Server
Protocol [http]: https
Server Name or IP: bankapp.singhritesh85.com
Port Number:
HTTP Request
GET
Path: /login
Content encoding:
Redirect Automatically
Follow Redirects
Use KeepAlive
Use multipart/form-data
Browser-compatible headers
Parameters Body Data Files Upload
Send Parameters With the Request:
Name:
Value
URL Encode?
Content-Type
Include Equals?
Detail
Add
Add from Clipboard
Delete
Up
Down


bankapp-load-test.jmx (C:\Users\HP\Downloads\apache-jmeter-5.5\mederma\bankapp-load-test.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
bankapp load testing
bankapp-load-test
bankapp login page test
Response Time Graph
View Results Tree
Aggregate Graph
View Results in Table
Response Time Graph
Name: Response Time Graph
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Settings Graph
Response Time Graph
Milliseconds
bankapp login page test


bankapp-load-test.jmx (C:\Users\HP\Downloads\apache-jmeter-5.5\mederma\bankapp-load-test.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
bankapp load testing
bankapp-load-test
bankapp login page test
Response Time Graph
View Results Tree
Aggregate Graph
View Results in Table
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Search: Case sensitive Regular exp. Search Reset
Text
Sampler result Request Response data
bankapp login page test
Thread Name:bankapp-load-test 1-1
Sample Start:2025-08-06 13:06:01 IST
Load time:987
Connect Time:751
Latency:987
Size in bytes:4813
Sent bytes:132
Headers size in bytes:411
Body size in bytes:4402
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:
HTTPSampleResult fields:
ContentType: text/html;charset=UTF-8
DataEncoding: UTF-8
Scroll automatically?
Raw Parsed

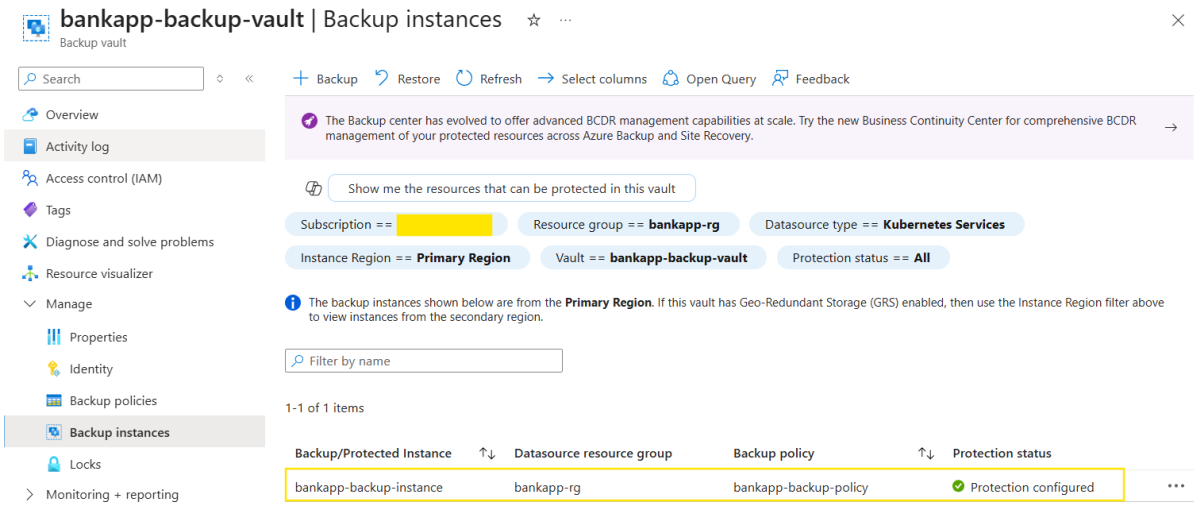

bankapp-load-test.jmx (C:\Users\HP\Downloads\apache-jmeter-5.5\mederma\bankapp-load-test.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
bankapp load testing
bankapp-load-test
bankapp login page test
Response Time Graph
View Results Tree
Aggregate Graph
View Results in Table
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Sample # Start Time Thread Name Label Sample Time(ms) Status Bytes Sent Bytes Latency Connect Time(ms)
1 13:09:37.363 bankapp-load-tes... bankapp login pa... 1010 4813 132 1010 768
2 13:09:37.697 bankapp-load-tes... bankapp login pa... 756 4813 132 756 514
3 13:09:38.031 bankapp-load-tes... bankapp login pa... 766 4813 132 765 510
4 13:09:38.355 bankapp-load-tes... bankapp login pa... 761 4813 132 761 496
5 13:09:38.687 bankapp-load-tes... bankapp login pa... 721 4813 132 721 486
6 13:09:39.023 bankapp-load-tes... bankapp login pa... 725 4813 132 725 484
7 13:09:39.354 bankapp-load-tes... bankapp login pa... 725 4813 132 725 487
8 13:09:39.688 bankapp-load-tes... bankapp login pa... 764 4813 132 764 522
9 13:09:40.022 bankapp-load-tes... bankapp login pa... 730 4813 132 730 496
10 13:09:40.353 bankapp-load-tes... bankapp login pa... 741 4813 132 741 495
11 13:09:40.686 bankapp-load-tes... bankapp login pa... 733 4813 132 733 492
12 13:09:41.021 bankapp-load-tes... bankapp login pa... 741 4813 132 741 500
13 13:09:41.351 bankapp-load-tes... bankapp login pa... 734 4813 132 734 498
14 13:09:41.687 bankapp-load-tes... bankapp login pa... 716 4813 132 716 484
15 13:09:42.019 bankapp-load-tes... bankapp login pa... 732 4813 132 732 492
16 13:09:42.350 bankapp-load-tes... bankapp login pa... 732 4813 132 732 493
17 13:09:42.684 bankapp-load-tes... bankapp login pa... 722 4813 132 722 486
18 13:09:43.019 bankapp-load-tes... bankapp login pa... 745 4813 132 745 507
19 13:09:43.350 bankapp-load-tes... bankapp login pa... 740 4813 132 740 503
20 13:09:43.684 bankapp-load-tes... bankapp login pa... 756 4813 132 756 510
21 13:09:44.016 bankapp-load-tes... bankapp login pa... 714 4813 132 714 480
22 13:09:44.350 bankapp-load-tes... bankapp login pa... 831 4813 132 831 585
23 13:09:44.681 bankapp-load-tes... bankapp login pa... 839 4813 132 839 601
24 13:09:45.017 bankapp-load-tes... bankapp login pa... 754 4813 132 754 512
Scroll automatically? Child samples?
No of Samples 30 Latest Sample 752 Average 755 Deviation 53

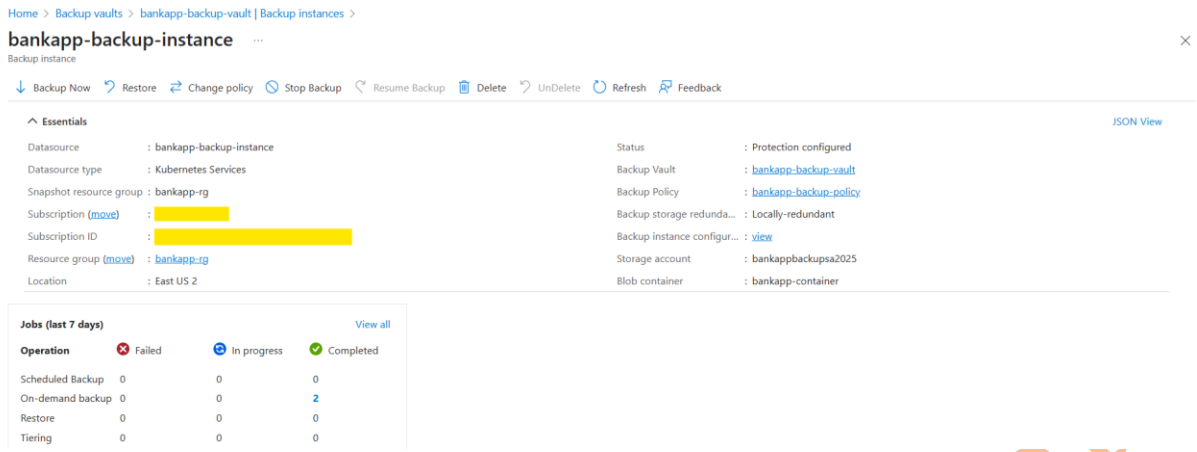


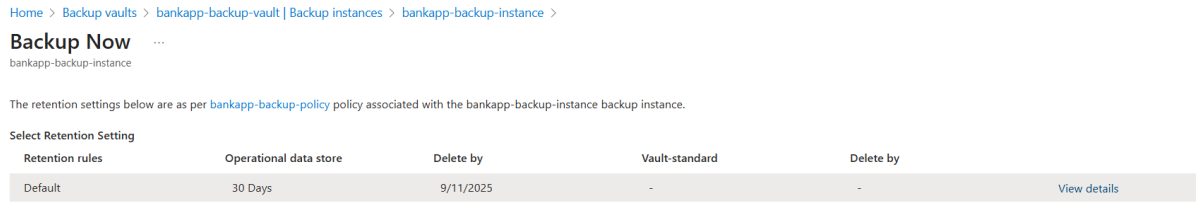
I can conclude that the overall performance of BankApp URL for the 30 users who accessed the URL https://bankapp.singhritesh85.com/login in a ramp-up time of 10 seconds was satisfactory.

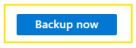
Upgrade the AKS cluster from version 1.32 to 1.33

Before Upgrading AKS Cluster from 1.32 to 1.33 I took Backup of the AKS Cluster as shown in the screenshot attached below.

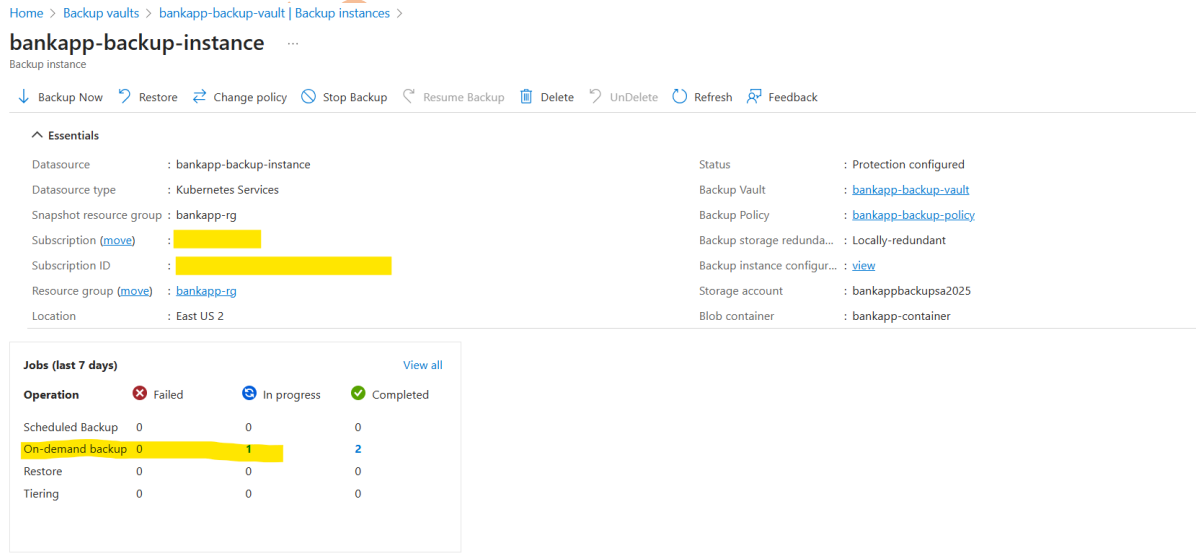




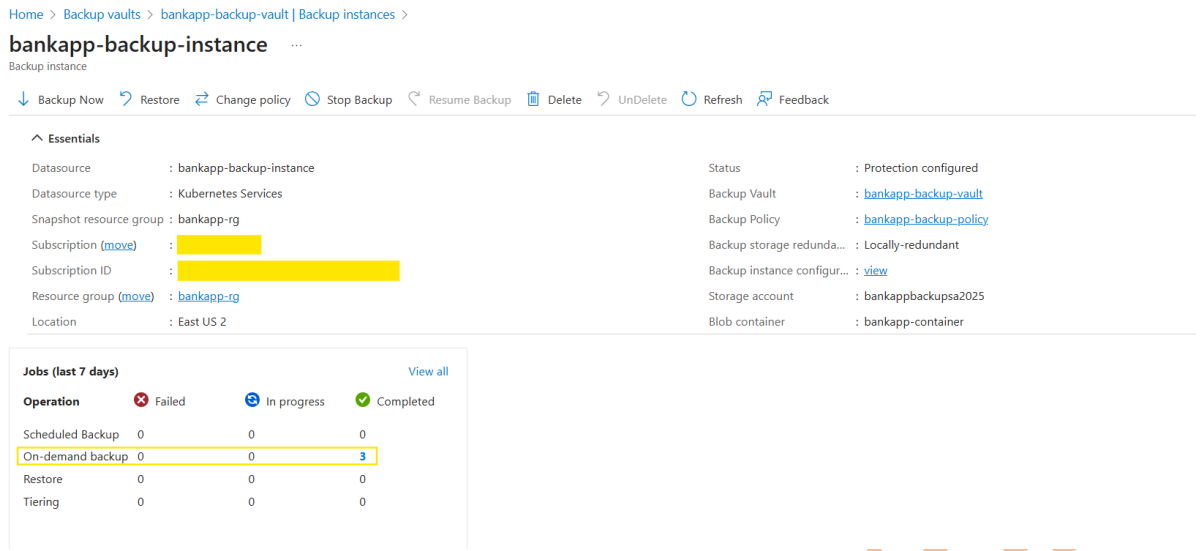




As shown in the screenshot attached below the backup of AKS Cluster had been started.



After waiting for sometime the Backup was completed as shown in the screenshot attached below.



As per the terraform.tfvars file present in the terraform script, I changed the terraform script main.tf file to upgrade AKS Cluster from 1.32 to 1.33.

```
kubernetes_version = ["1.26.6", "1.26.10", "1.27.3", "1.27.7", "1.28.0", "1.28.3", "1.28.5", "1.29.0", "1.29.2", "1.30.0", "1.30.12", "1.31.8", "1.32.4", "1.33.0"]
```

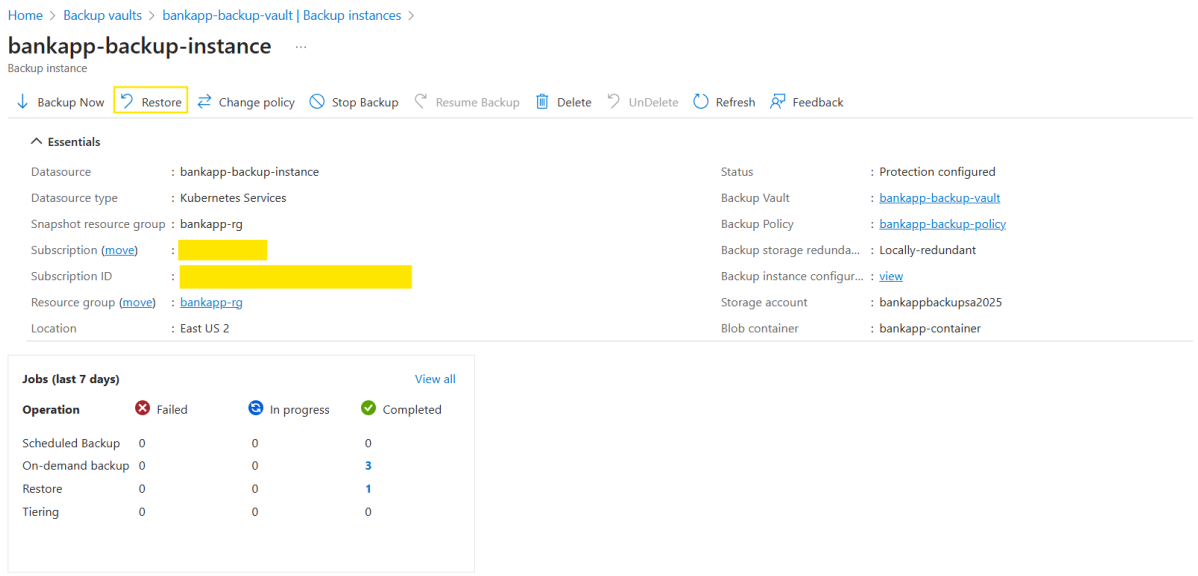
```
module "aks" {

  source = "../module"
  prefix = var.prefix
  location = var.location[1]
  kubernetes_version = var.kubernetes_version[13]
```

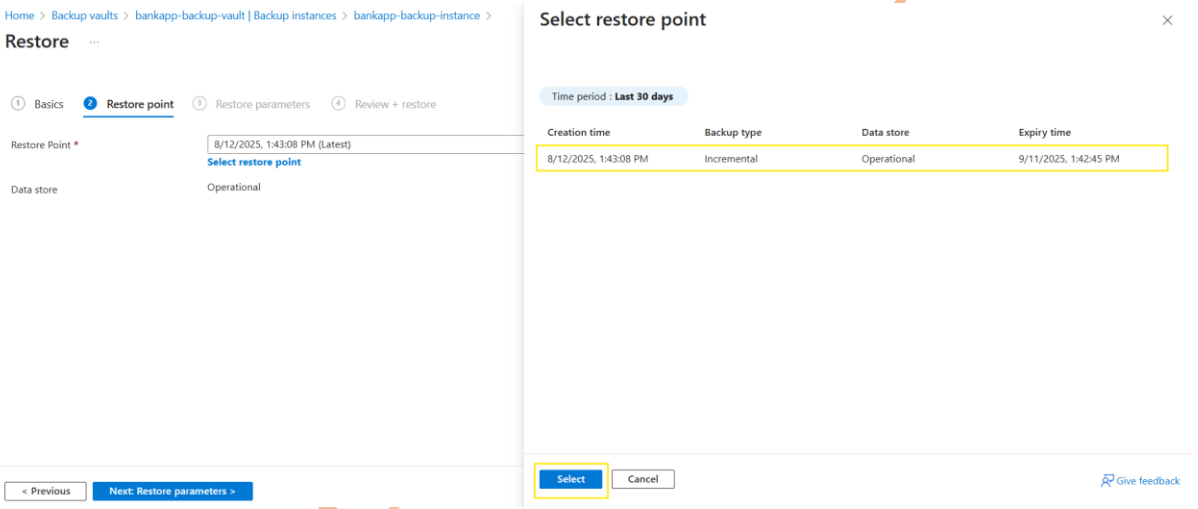
Then upgraded the Kubernetes Cluster by running the command **terraform plan** followed by **terraform apply**. For upgrade AKS Cluster you can import the AKS Cluster separately and then upgrade the same but here I am using the same terraform script as for this project it will not affect other resources.

```
[root@                    ~]# kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
aks-agentpool-42233949-vmss000000   Ready    <none>   m     v1.33.0
aks-agentpool-42233949-vmss000001   Ready    <none>   m     v1.33.0
aks-agentpool-42233949-vmss000002   Ready    <none>   m     v1.33.0
aks-userpool-30227447-vmss000001    Ready    <none>   m     v1.33.0
```

As shown in the screenshot attached above the AKS Cluster had been upgraded from 1.32 to 1.33. I had mentioned above that I took the on-demand backup before the upgrade. For this project after upgrade BankApp was working properly. But here for demonstration of Backup restore I assumed the BankApp pods was deleted which I restored with the Backup taken as shown in the screenshot attached below.

```
[root@                    ~]# kubectl get all -n bankapp
No resources found in bankapp namespace.
```


Home > Backup vaults > bankapp-backup-vault | Backup instances >
bankapp-backup-instance
Backup instance
Backup Now
Restore
Change policy
Stop Backup
Resume Backup
Delete
UnDelete
Refresh
Feedback
Essentials
Datasource : bankapp-backup-instance
Datasource type : Kubernetes Services
Snapshot resource group : bankapp-rg
Subscription (move) :
Subscription ID :
Resource group (move) : bankapp-rg
Location : East US 2
Status : Protection configured
Backup Vault : bankapp-backup-vault
Backup Policy : bankapp-backup-policy
Backup storage redunda... : Locally-redundant
Backup instance configur... : view
Storage account : bankappbackupsa2025
Blob container : bankapp-container
Jobs (last 7 days)
View all
Operation
Failed
In progress
Completed
Scheduled Backup 0 0 0
On-demand backup 0 0 3
Restore 0 0 1
Tiering 0 0 0


Home > Backup vaults > bankapp-backup-vault | Backup instances > bankapp-backup-instance >
Restore
Basics
Restore point
Restore parameters
Review + restore
Restore Point *
8/12/2025, 1:43:08 PM (Latest)
Select restore point
Data store
Operational
< Previous
Next: Restore parameters >
Select restore point
Time period : Last 30 days
Creation time
Backup type
Data store
Expiry time
8/12/2025, 1:43:08 PM
Incremental
Operational
9/11/2025, 1:42:45 PM
Select
Cancel
Give feedback

Home > Backup vaults > bankapp-backup-vault | Backup instances > bankapp-backup-instance >

Restore ...

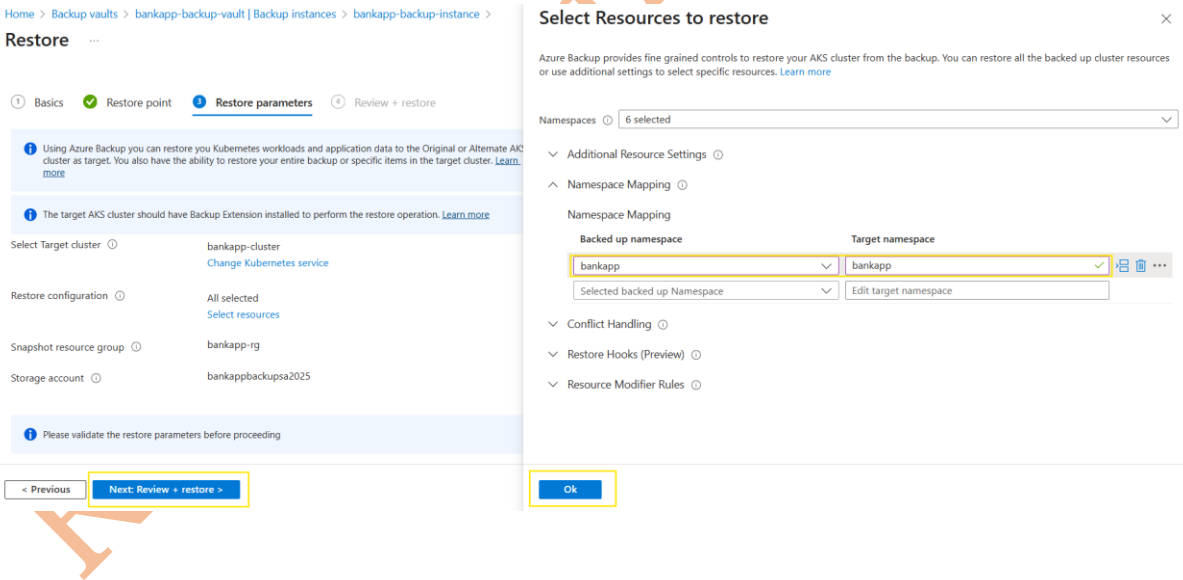
1 Basics 2 Restore point 3 Restore parameters 4 Review + restore

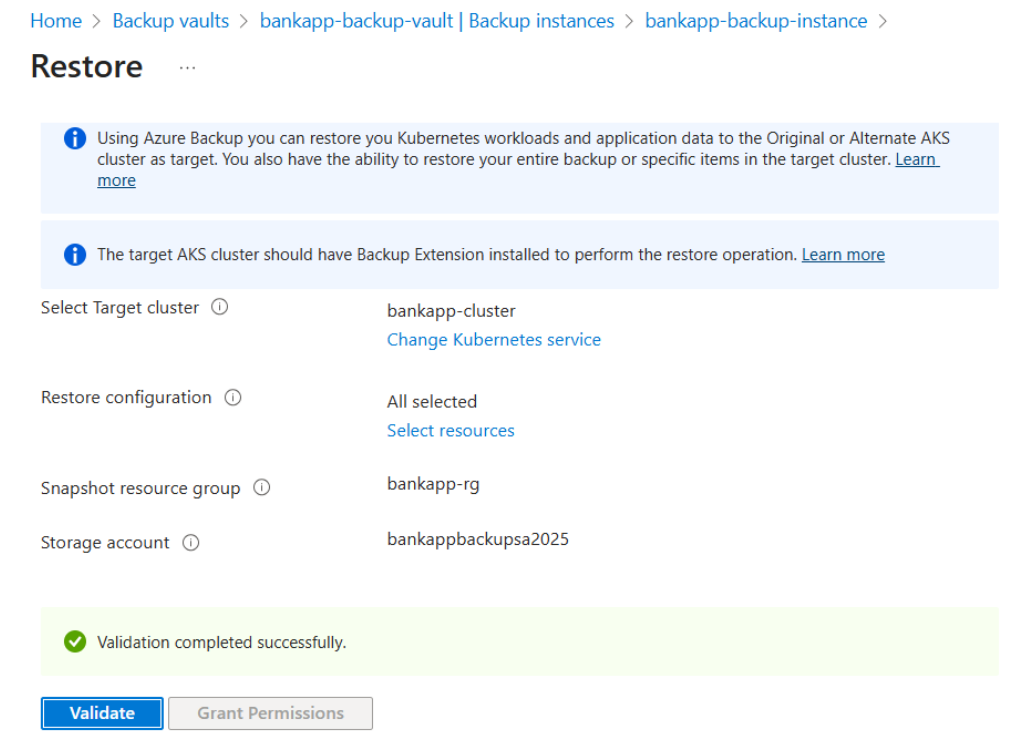
Restore Point * 8/12/2025, 1:43:08 PM

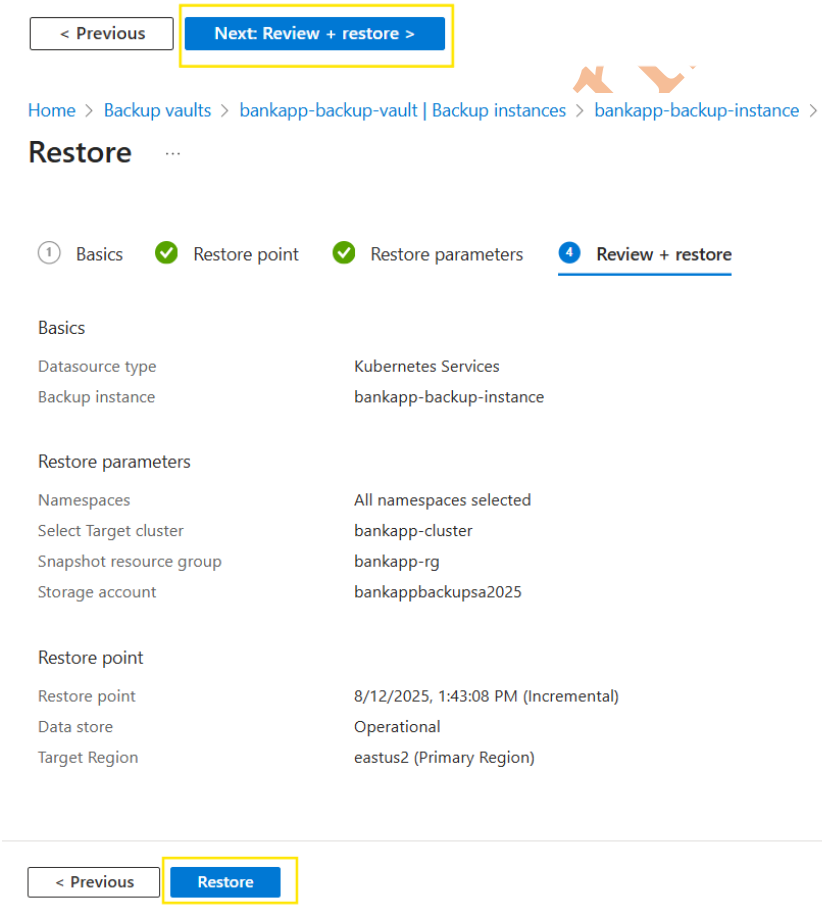
Select restore point

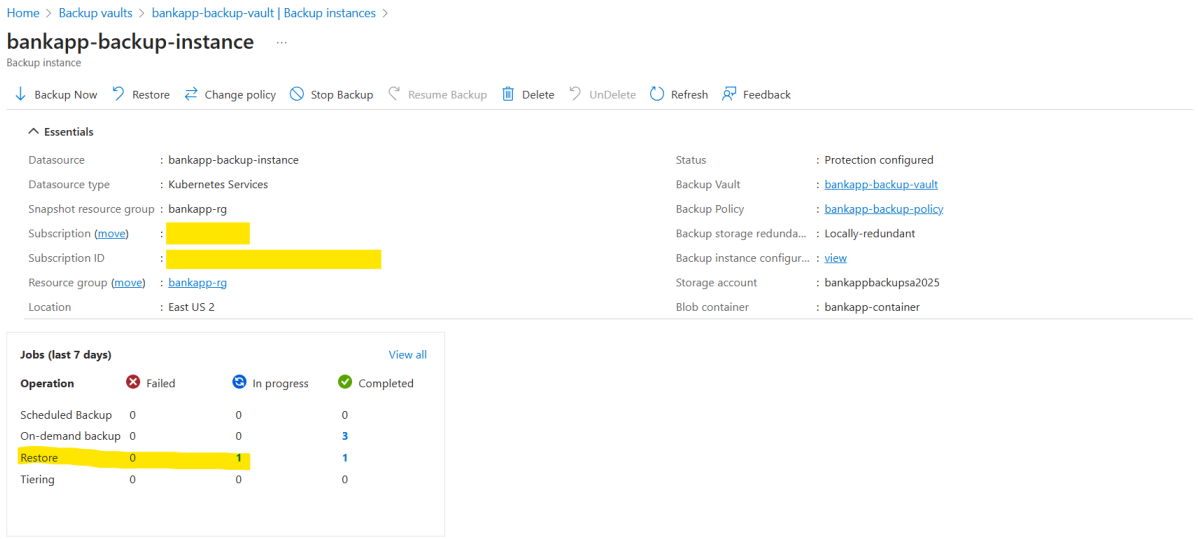
Data store Operational

< Previous Next: Restore parameters >

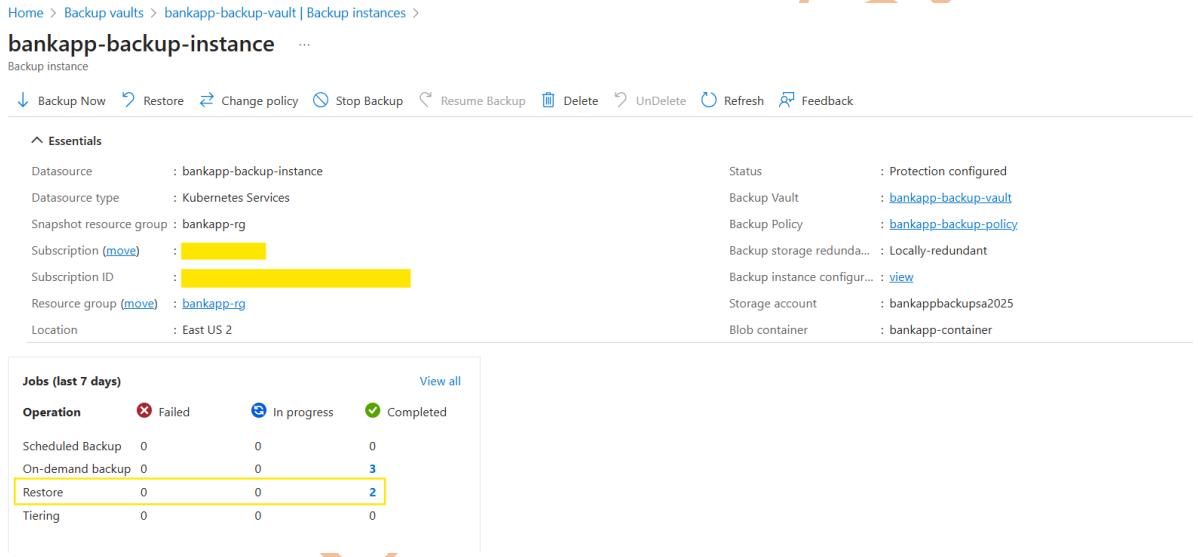



Home > Backup vaults > bankapp-backup-vault | Backup instances > bankapp-backup-instance >
Restore
Using Azure Backup you can restore you Kubernetes workloads and application data to the Original or Alternate AKS cluster as target. You also have the ability to restore your entire backup or specific items in the target cluster. Learn more
The target AKS cluster should have Backup Extension installed to perform the restore operation. Learn more
Select Target cluster
bankapp-cluster
Change Kubernetes service
Restore configuration
All selected
Select resources
Snapshot resource group
bankapp-rg
Storage account
bankappbackupsa2025
Validation completed successfully.
Validate
Grant Permissions


< Previous
Next: Review + restore >
Home > Backup vaults > bankapp-backup-vault | Backup instances > bankapp-backup-instance >
Restore
1 Basics
Restore point
Restore parameters
4 Review + restore
Basics
Datasource type
Kubernetes Services
Backup instance
bankapp-backup-instance
Restore parameters
Namespaces
All namespaces selected
Select Target cluster
bankapp-cluster
Snapshot resource group
bankapp-rg
Storage account
bankappbackupsa2025
Restore point
Restore point
8/12/2025, 1:43:08 PM (Incremental)
Data store
Operational
Target Region
eastus2 (Primary Region)
< Previous
Restore

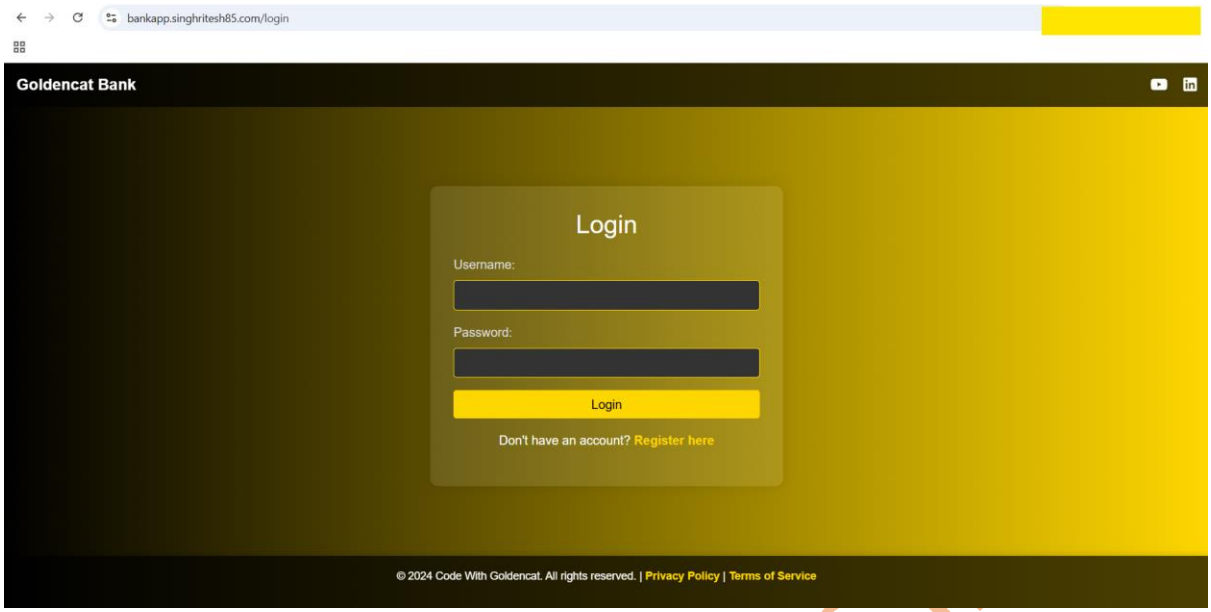


As shown in the screenshot attached below the backup was restored successfully.




```
[root@                    ~]# kubectl get all -n bankapp
NAME                                READY   STATUS    RESTARTS   AGE
pod/bankapp-folo-                   1/1     Running   0          4m54s

NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/bankapp-folo    ClusterIP   10.0.176.25    <none>        80/TCP    4m51s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp-folo    1/1     1            1           4m19s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-folo-              1         1         1       4m52s
```

Finally, I was able to access the Application as shown in the screenshot attached below.



During Rancher session https://www.linkedin.com/posts/ritesh-kumar-singh-41113128b_rancher-for-you-by-explicate-devops-activity-7305817947615174656-OGvw?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEZR-A0B3BvuC7NszexuvafVPo-bTaDUJI8 I explained you how to take the backup and restore using Kasten-k10 which you can also use for migrating your application from one cloud to another cloud. Here I used Azure Service to Backup and restore.

Source Code: **https://github.com/singhritesh85/Bank-App-AKS-WAF.git**

GitHub Repo: **https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git**

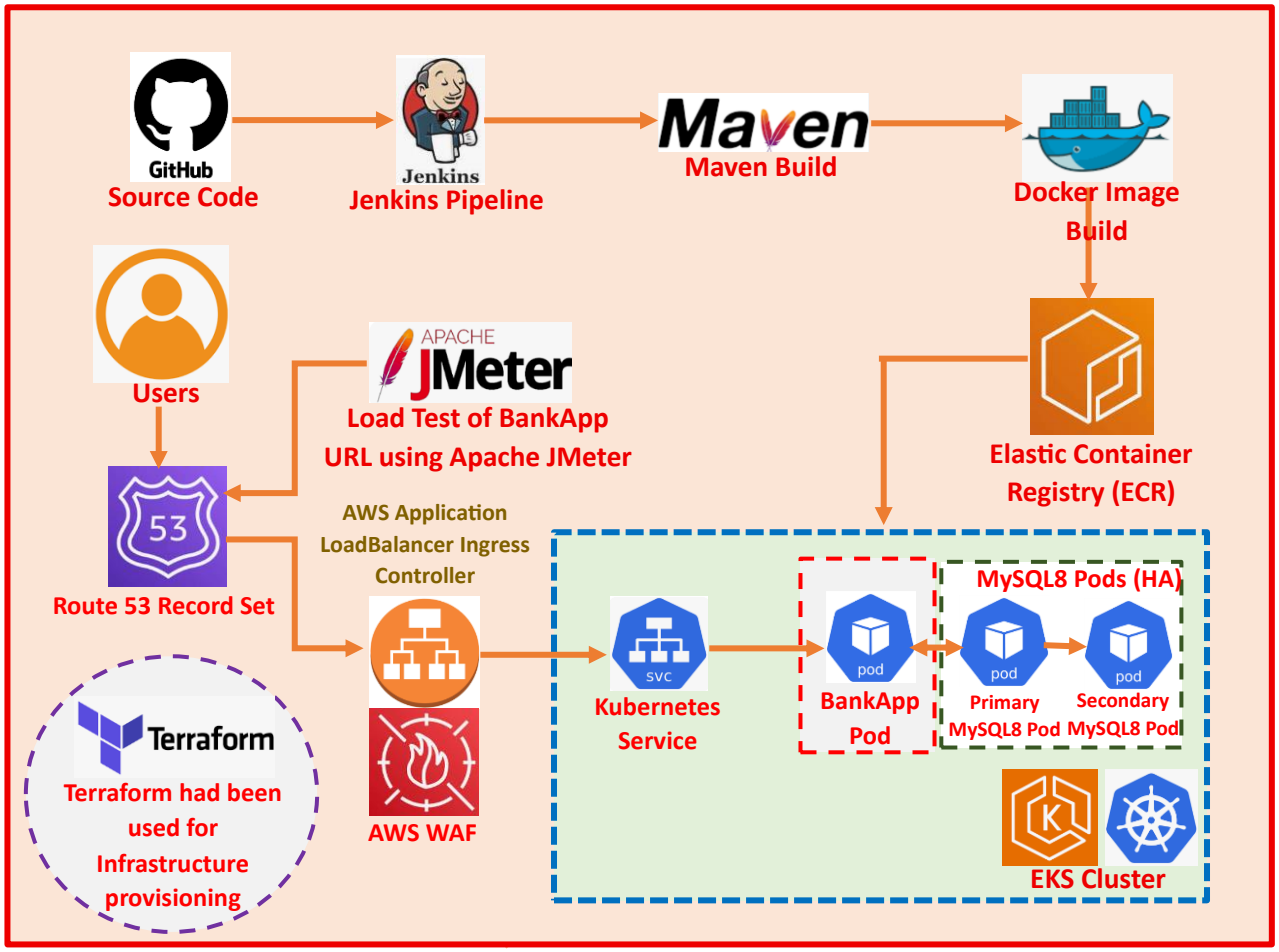
Helm Chart: **https://github.com/singhritesh85/helm-repo-for-ArgoCD.git**

Terraform Script: **https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git**

References: **https://github.com/Goldencat98/Bank-App.git**

Module-2

DevOps-Project-BankApp-EKS-WAF-Performance-Testing



This Project deals with the creation of infrastructure using Terraform, GitHub was used to keep the source Code and Jenkins had been used as CI/CD Tool. The build tool used in this project was Maven and Docker Image was pushed to the Elastic Container Registry (ECR) which was finally deployed to the Elastic Kubernetes Services (EKS). The Bank Application Pods was accessed using Application LoadBalancer Ingress Controller and hence using the kubernetes service as shown in the screenshot attached above. I applied WAF Policy (to Limit the Rate) to the Bank Application URL and to prevent from SQL Injection. As this Bank Application was open to entire world so rate limit and SQL Injection was applied through WAF Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot (by the Hackers) so that its speed becomes slow or this website hangs to refrain from such a situation WAF Policy Rate Limit was applied. Finally, I did the Performance Test (Load test) of the Bank Application URL using Apache JMeter to check how many users can access the BankApp without any hitch.

To create the infrastructure, I used Terraform and Terraform was installed on EC2 Instance and its state-file was kept in S3 Bucket. For AWS Cloud, Authentication and Authorization, I used the RBAC (Role-Based-Access-Control).

Terraform script to create the infrastructure is present at the path **terraform-route53-hosted-zone-with-acm-certificate** (this terraform script is handy for you to create the Route53 hosted-zone and AWS Certificate Manager SSL Certificate) in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. Commands as written below had been used to create infrastructure using Terraform.

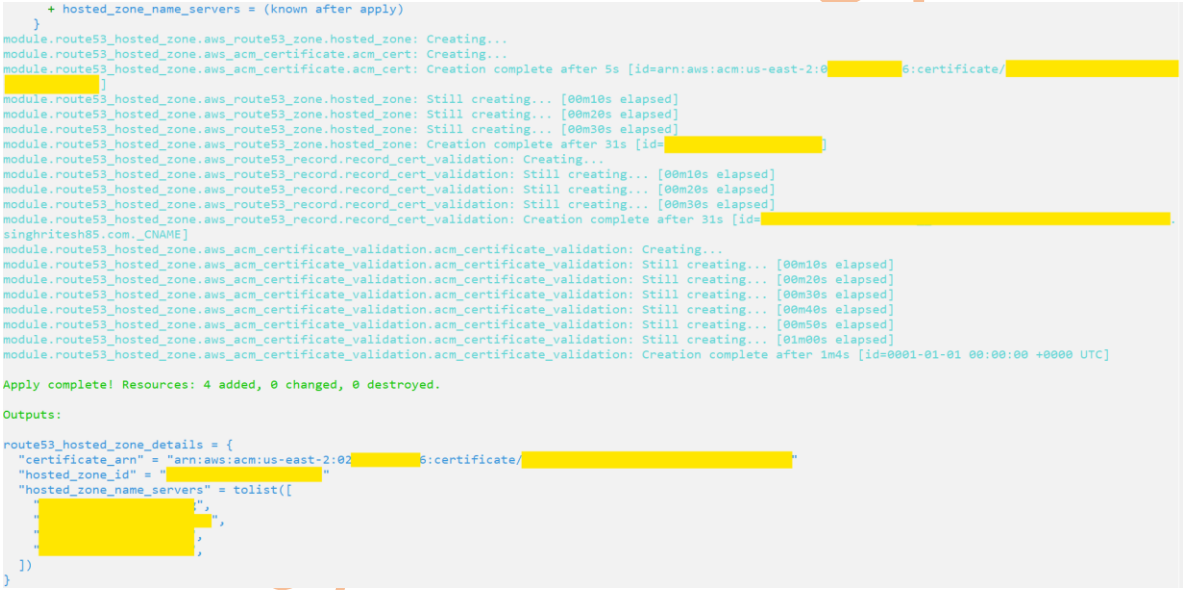
terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

At this point after executing the command **terraform apply -auto-approve** make sure you do the entry for Nameservers of the Route53 created Hosted Zone in your domain name service provider's Nameserver. After doing such an entry the terraform script will give the output as written below.



Now the Route53 Hosted Zone and AWS Certificate Manager SSL Certificate became ready. Then I created the other AWS Resources using the terraform script present at the path **terraform-eks-cluster-waf-with-velero-backup** in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git.

```
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Creation complete after 45s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m50s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m50s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m50s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Creation complete after 55s [id=eks-demo-cluster-dev:snapshot-controller]
module.eks_cluster.aws_eks_addon.metrics_server: Creation complete after 55s [id=eks-demo-cluster-dev:metrics-server]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Creation complete after 55s [id=eks-demo-cluster-dev:aws-ebs-csi-driver]

Apply complete! Resources: 100 added, 0 changed, 0 destroyed.

Outputs:

ecr_ec2_private_ip_eks_details = {
  "alb_dns_name" = "jenkins-ms-[redacted].us-east-2.elb.amazonaws.com"
  "alb_ingress_controller_role_arn" = "arn:aws:iam::02[redacted]6:role/aws-alb-ingress-controller-role"
  "aws_wafv2_web_acl_arn" = "arn:aws:wafv2:us-east-2:02[redacted]6:regional/webacl/aws-wafv2-rate-limit/[redacted]"
  "eks_cluster_endpoint" = "https://[redacted].gr7.us-east-2.eks.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "jenkins_master_instance_id" = "i-[redacted]"
  "jenkins_master_private_ip" = "10.[redacted]"
  "jenkins_slave_instance_id" = "i-[redacted]"
  "jenkins_slave_private_ip" = "10.[redacted]"
  "karpenter_node_spun_iam_role_arn" = "arn:aws:iam::0[redacted]6:role/karpenter-eks-noderole"
  "registry_id" = "02[redacted]6"
  "repository_url" = "02[redacted]6.dkr.ecr.us-east-2.amazonaws.com/bankapp"
  "velero_backup_iam_role_arn" = "arn:aws:iam::02[redacted]6:role/eks-velero-backup-role"
  "velero_backup_s3_bucket_name" = "dexter-velero-backup"
}
```

To install karpenter I edited the configmap **aws-auth** in the namespace **kube-system** as shown in the screenshot present below.

```
[root@[redacted] ~]# kubectl edit cm aws-auth -n kube-system
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::02[redacted]6:role/eks-nodegroup-role-dev
      groups:
      - system:bootstrappers
      - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
    - rolearn: arn:aws:iam::02[redacted]6:role/karpenter-eks-noderole
      groups:
      - system:bootstrappers
      - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2025[redacted]"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "[redacted]"
  uid: [redacted]
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT (paste) --
```

I installed the karpenter controller using helm as shown in the screenshot attached below.

helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.2" --namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks\.amazonaws\.com/role-arn=arn:aws:iam::02XXXXXXXXX6:role/karpenter-controller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi

```
[root@ ~]# helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.2" --namespace "karpenter" --create-namespace
 --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks\.amazonaws\.com/role-arn=arn:aws:iam::027330342406:role/karpenter-con
troller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controlle
r.resources.limits.memory=1Gi
```

Then ran the kubernetes Manifests file **karpenter.yaml** present at the path **terraform-eks-cluster-waf-with-velero-backup** in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. For your reference **karpenter.yaml** I had also provided below.

```
[root@                    ]# kubectl apply -f karpenter.yaml
nodepool.karpenter.sh/general-purpose created
ec2nodeclass.karpenter.k8s.aws/default created
```

```
# This example NodePool will provision general purpose instances
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: general-purpose
  annotations:
    kubernetes.io/description: "General purpose NodePool for generic workloads"
spec:
  template:
    spec:
      requirements:
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
        - key: kubernetes.io/os
          operator: In
          values: ["linux"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]   #["spot"] ### You can select on-demand or spot instance depending on your requirement.
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["t"]   # Interested to launch t series instance          #["c", "m", "r"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["1"]   # Instances launched will be greated than 1    ###["2"]   # Instances launched will be greated than 2
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
```

```
    name: default
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: default
  annotations:
    kubernetes.io/description: "General purpose EC2NodeClass for running Amazon Linux 2023 nodes"
spec:
  role: "karpenter-eks-noderole" # replace with your karpenter noderole
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "eks-demo-cluster-dev" # replace with your cluster name
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "eks-demo-cluster-dev" # replace with your cluster name
  amiSelectorTerms:
    - alias: al2023@latest # Amazon Linux 2023
```

```
[root@                    ~]# kubectl get pods -n karpenter
NAME                         READY   STATUS    RESTARTS   AGE
karpenter-                   1/1     Running   0          7m20s
karpenter-                   1/1     Running   0          7m20s
```

To install velero using helm I used the commands as written below.

helm repo add vmware-tanzu https://vmware-tanzu.github.io/helm-charts/

helm repo update

helm install velero vmware-tanzu/velero --namespace velero --create-namespace --set configuration.backupStorageLocation[0].provider=aws --set configuration.backupStorageLocation[0].bucket=dexter-velero-backup --set configuration.volumeSnapshotLocation[0].config.region=us-east-2 --set configuration.volumeSnapshotLocation[0].provider=aws --set serviceAccount.create=true --set serviceAccount.name=velero-server --set serviceAccount.server.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::02XXXXXXXXX6:role/eks-velero-backup-role" --set initContainers[0].name=velero-plugin-for-aws --set initContainers[0].image="velero/velero-plugin-for-aws:v1.12.2" --set initContainers[0].volumeMounts[0].mountPath="/target" --set initContainers[0].volumeMounts[0].name="plugins" --version=10.1.0 --set credentials.useSecret=false

```
[root@            ~]# helm repo add vmware-tanzu https://vmware-tanzu.github.io/helm-charts/
"vmware-tanzu" has been added to your repositories
[root@            ~]#     helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "vmware-tanzu" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@            ~]#     helm install velero vmware-tanzu/velero --namespace velero --create-namespace --set configuration.backupStorageLocation[0].provide
r=aws --set configuration.backupStorageLocation[0].bucket=dexter-velero-backup --set configuration.volumeSnapshotLocation[0].config.region=us-east-2 --set con
figuration.volumeSnapshotLocation[0].provider=aws --set serviceAccount.create=true --set serviceAccount.name=velero-server --set serviceAccount.server.annotat
ions."eks\.amazonaws\.com/role-arn"="arn:aws:iam::02        6:role/eks-velero-backup-role" --set initContainers[0].name=velero-plugin-for-aws --set initConta
iners[0].image="velero/velero-plugin-for-aws:v1.12.2" --set initContainers[0].volumeMounts[0].mountPath="/target" --set initContainers[0].volumeMounts[0].name
="plugins" --version=10.1.0 --set credentials.useSecret=false
```

```
[root@                ~]#   kubectl get pods -n velero
NAME                        READY   STATUS    RESTARTS   AGE
velero-                     1/1     Running   0          66s
```

To install AWS Application LoadBalancer Ingress Controller using helm I used the command as written below.

helm repo add eks https://aws.github.io/eks-charts

helm repo update

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=eks-demo-cluster-dev --set serviceAccount.create=true --set region=us-east-2 --set vpcId=vpc-0XXXXXXXXXXXXXXX2 --set serviceAccount.name=aws-load-balancer-controller --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::02XXXXXXXXX6:role/aws-alb-ingress-controller-role" --set enableWafv2=true

```
[root@            ~]# helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
[root@            ~]#     helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@            ~]# helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=eks-demo-cluster-dev --set
 serviceAccount.create=true --set region=us-east-2 --set vpcId=vpc-0              2 --set serviceAccount.name=aws-load-balancer-controller --set serviceAccou
nt.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::02        6:role/aws-alb-ingress-controller-role" --set enableWafv2=true
```

```
[root@                ~]# kubectl get pods -n kube-system|grep -i "aws-load-balancer-controller"
aws-load-balancer-controller-          1/1     Running   0          96s
aws-load-balancer-controller-          1/1     Running   0          96s
```

Then I installed ArgoCD and ArgoCD CLI. To install ArgoCD CLI run the commands as written below.

curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

rm argocd-linux-amd64

Following steps I used to install the ArgoCD.

kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

To the default password of Admin user run the command as written below.

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d

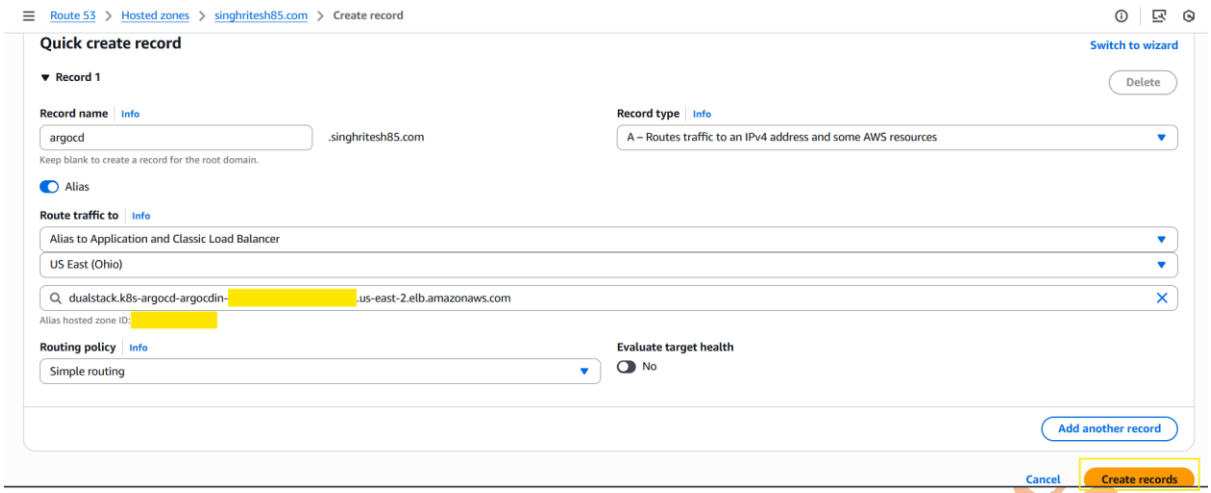
```
[root@                    ~]# kubectl create namespace argocd
namespace/argocd created
```

```
[root@              ~]# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

[root@              ~]# kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
Y              9[root@              ~]#
```

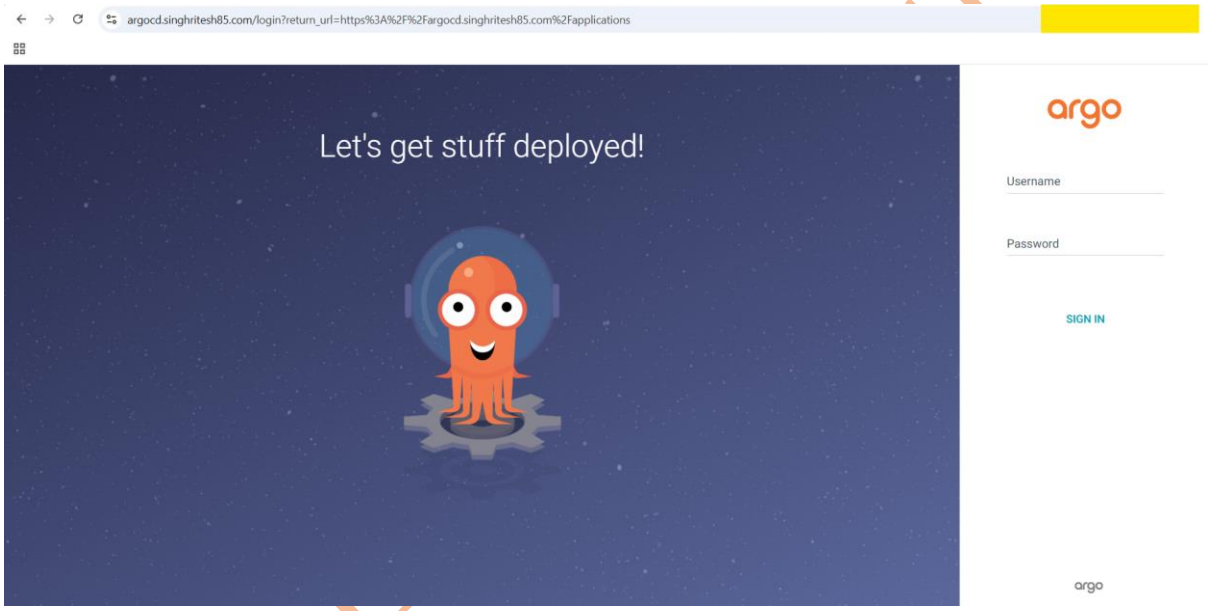
To access ArgoCD I used the Ingress Rule as written below. It will create an AWS Application LoadBalancer which entry I will do in the Route53 to create the Record Set of A-Type as shown in the screenshot attached below. Finally using that URL it is possible to access the ArgoCD as shown below.

```
[root@              ~]# kubectl apply -f argocd-ingress-rule.yaml
ingress.networking.k8s.io/argocd-ingress created
[root@              ~]# cat argocd-ingress-rule.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-ingress
  namespace: argocd
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:0         6:certificate/
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
spec:
  ingressClassName: alb
  rules:
    - host: argocd.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: argocd-server
                port:
                  number: 443
[root@              ~]# kubectl get ing -n argocd --watch
NAME             CLASS   HOSTS                   ADDRESS                                                                  PORTS   AGE
argocd-ingress   alb     argocd.singhritesh85.com   k8s-argocd-argocdin-         -         .us-east-2.elb.amazonaws.com   80      19s
```

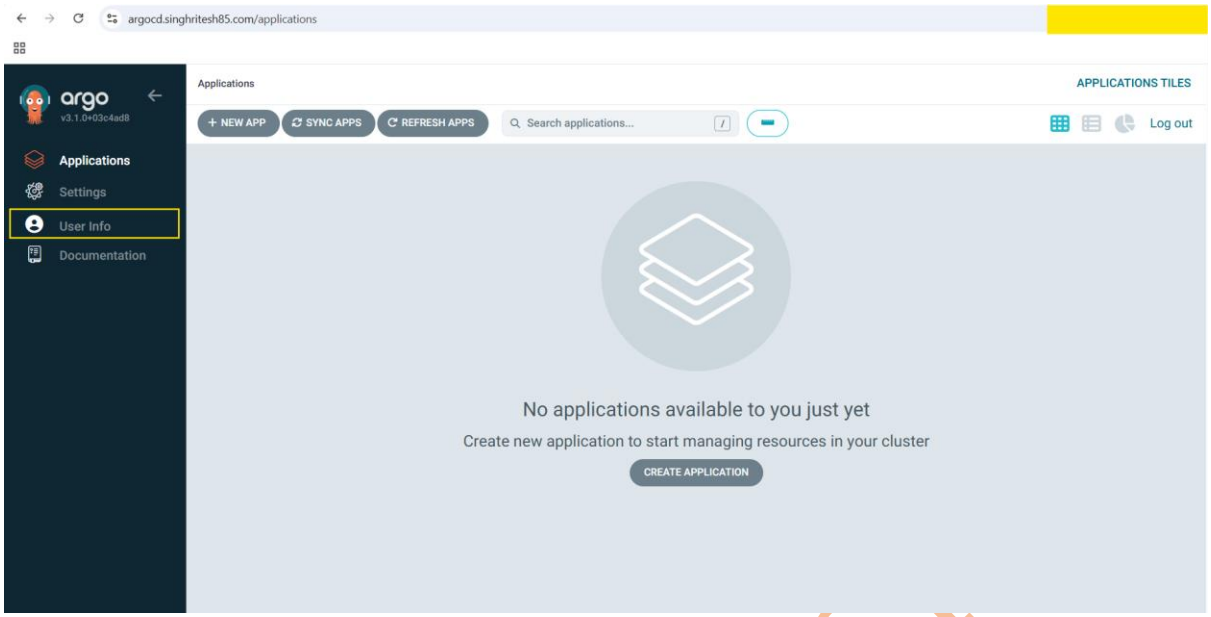
```
cat argocd-ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-ingress
  namespace: argocd
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-
2:02XXXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
spec:
  ingressClassName: alb
  rules:
    - host: argocd.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: argocd-server
                port:
                  number: 443
```

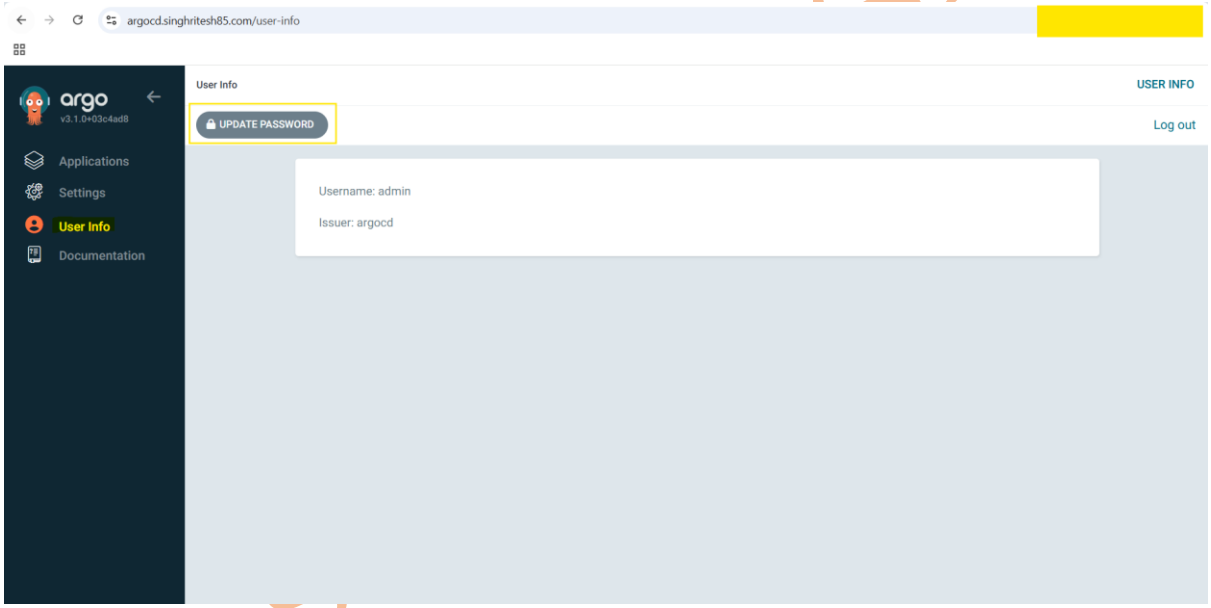


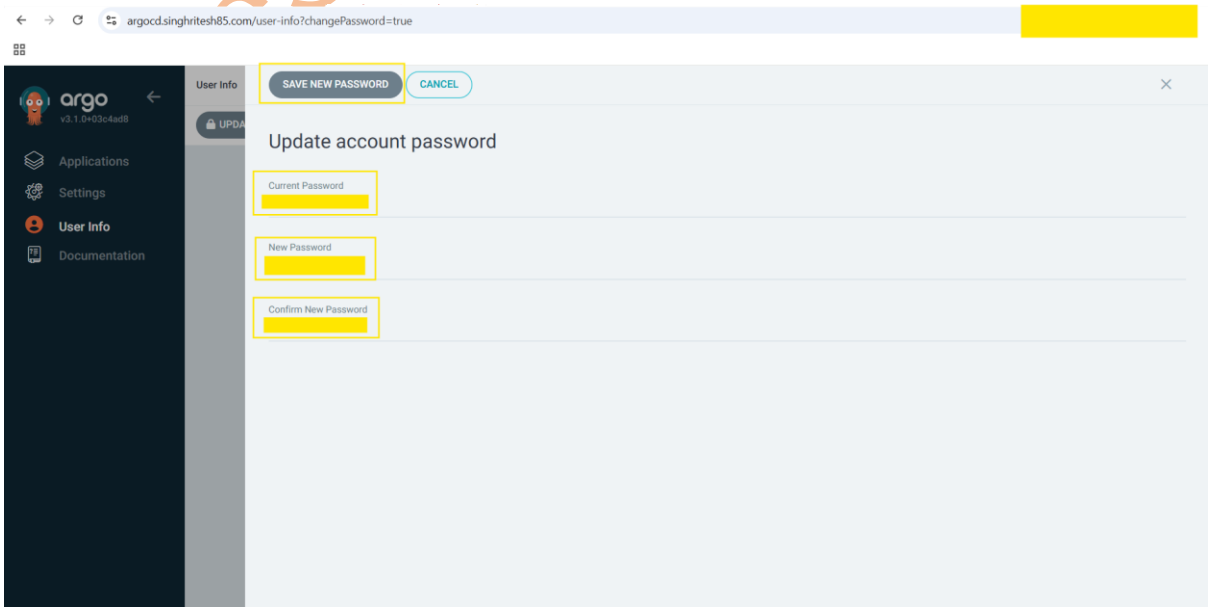
Finally, I was able to access the ArgoCD as shown in the screenshot attached below.

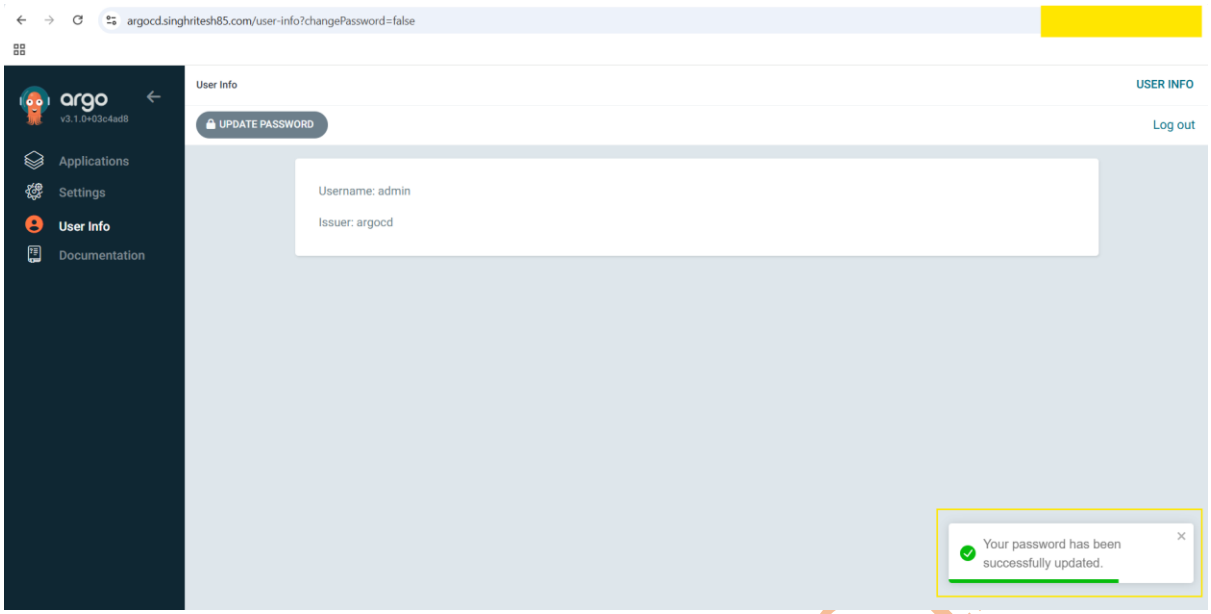


To Login for the first time, I used admin user and its default password, after first Login I changed the default password of admin user as shown in the screenshot attached below.

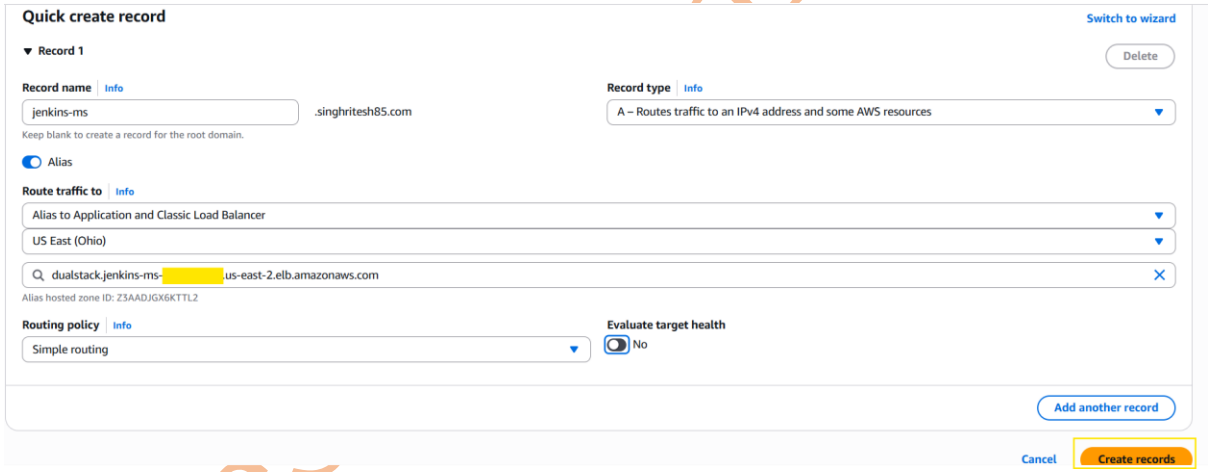

argocd.singhritesh85.com/applications
argo
v3.1.0+03c4ad8
Applications
Settings
User Info
Documentation
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
No applications available to you just yet
Create new application to start managing resources in your cluster
CREATE APPLICATION


argocd.singhritesh85.com/user-info
argo
v3.1.0+03c4ad8
Applications
Settings
User Info
Documentation
User Info
USER INFO
UPDATE PASSWORD
Log out
Username: admin
Issuer: argocd


argocd.singhritesh85.com/user-info?changePassword=true
argo
v3.1.0+03c4ad8
Applications
Settings
User Info
Documentation
User Info
SAVE NEW PASSWORD
CANCEL
Update account password
Current Password
New Password
Confirm New Password

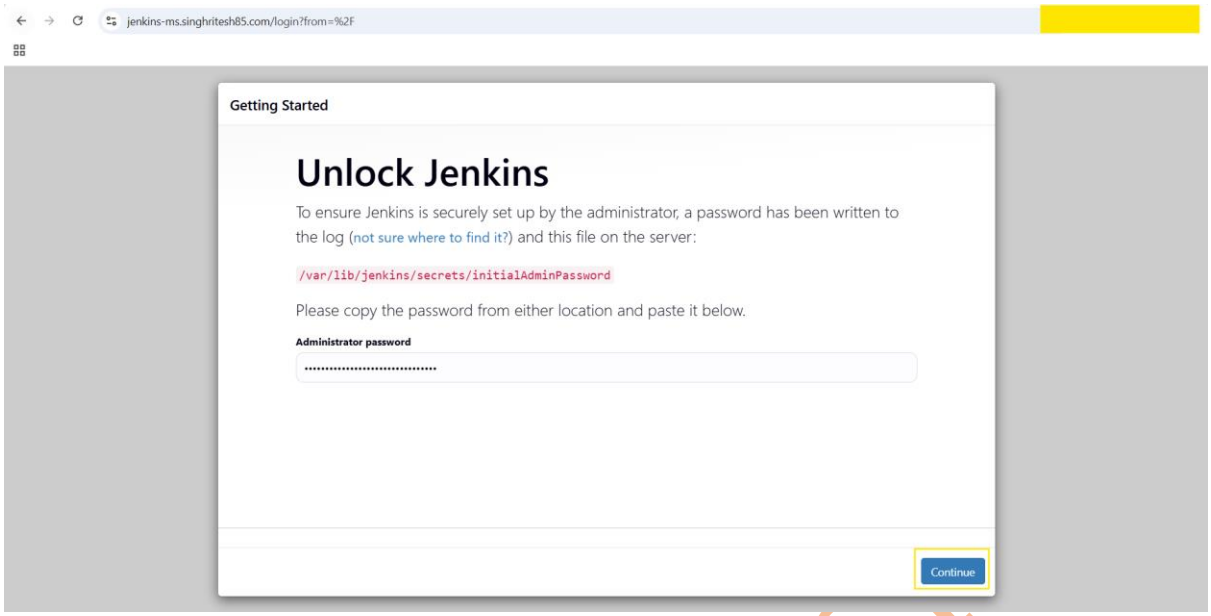


To create the URL for accessing Jenkins I did the entry for Jenkins LoadBalancer DNS Name in Route53 to create the record set of A-Type as shown in the screenshot attached below.

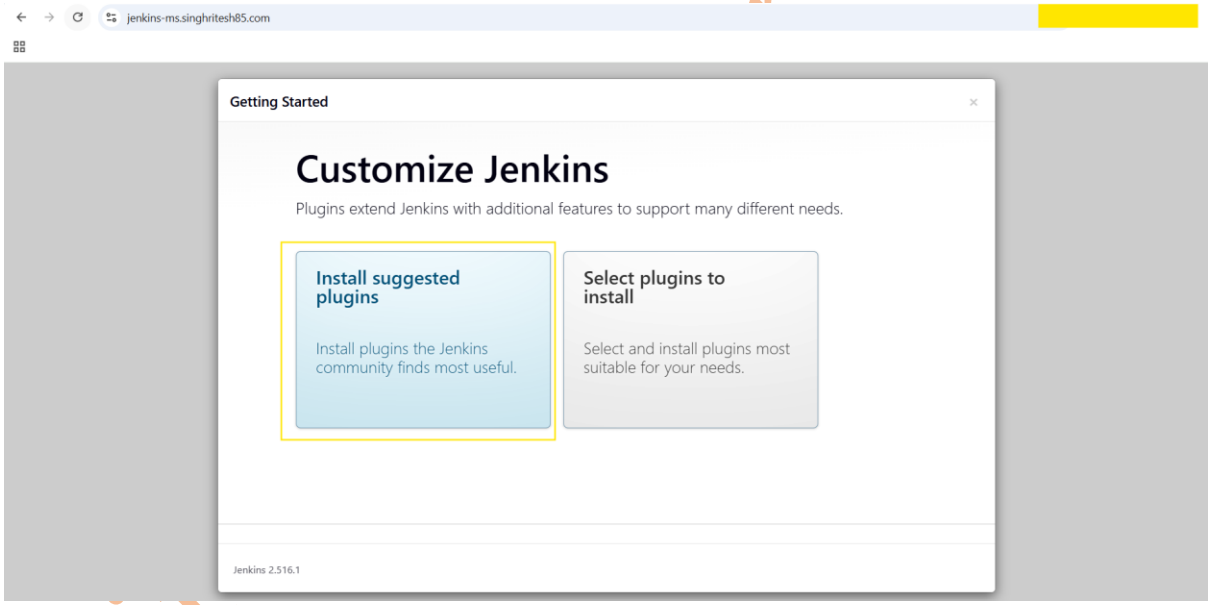


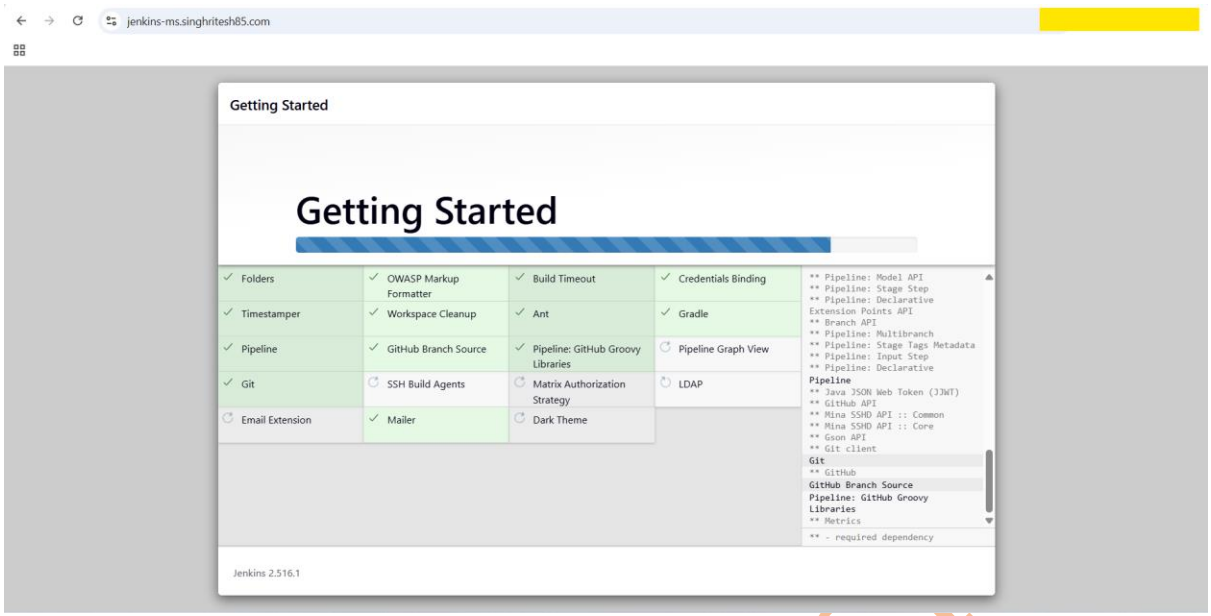
For first access to Jenkins, it required initial admin password which was present on Jenkins-Server at the path **/var/lib/jenkins/secrets/initialAdminPassword**.

```
[root@jenkins-master ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
9                                   2
```

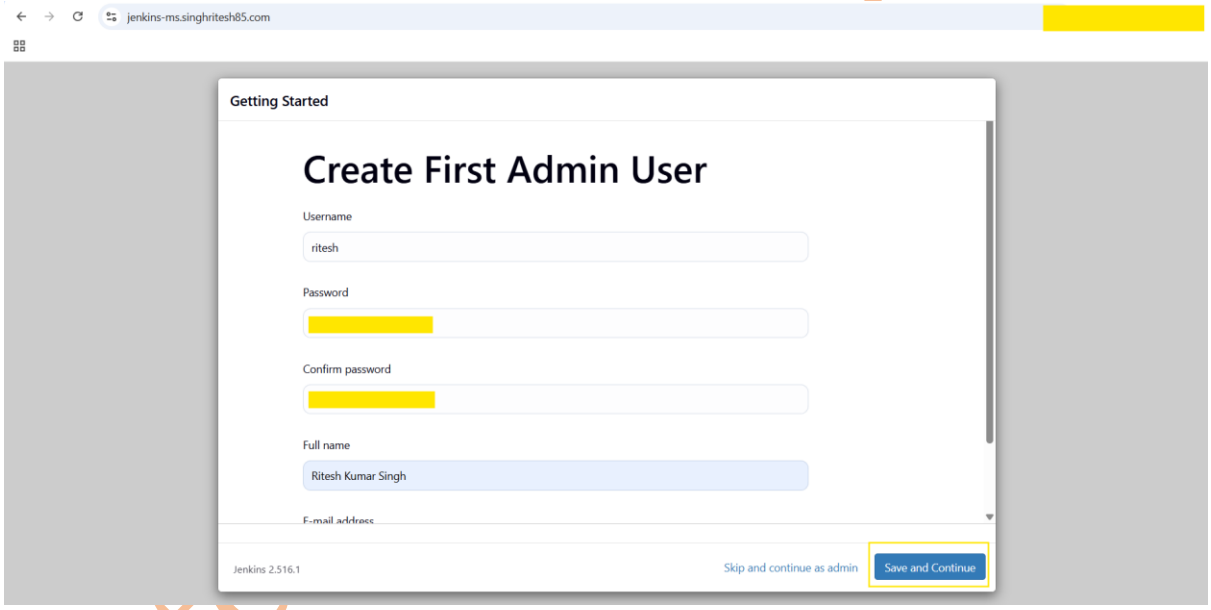


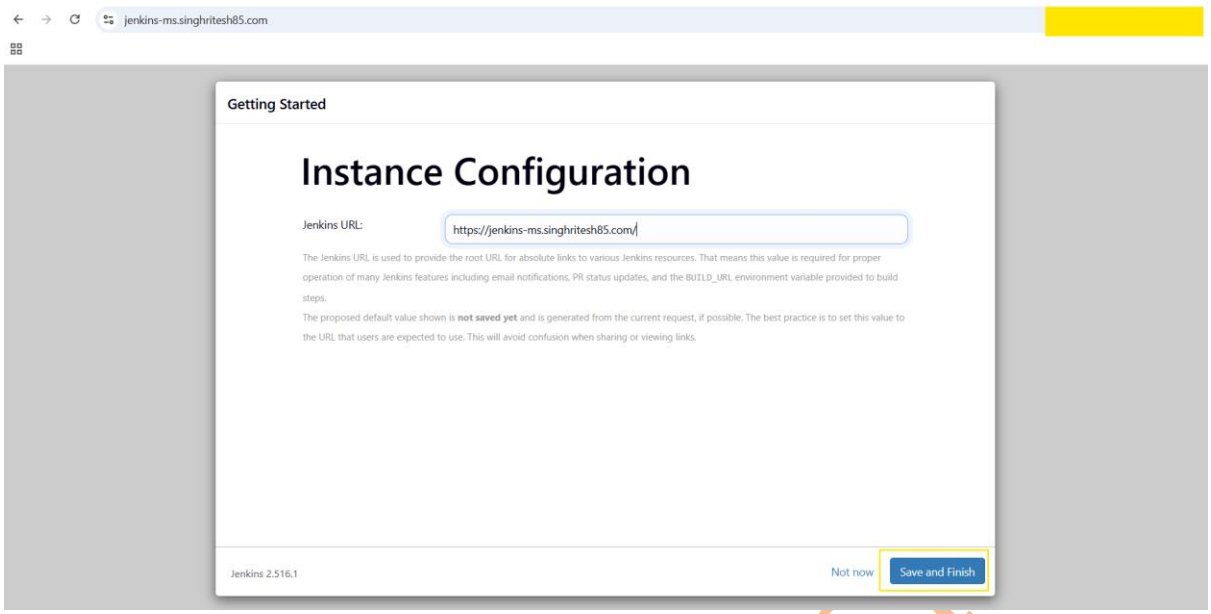
Then it will ask to install the plugins, I went with the option of suggested plugins and install then as shown in the screenshot attached below.

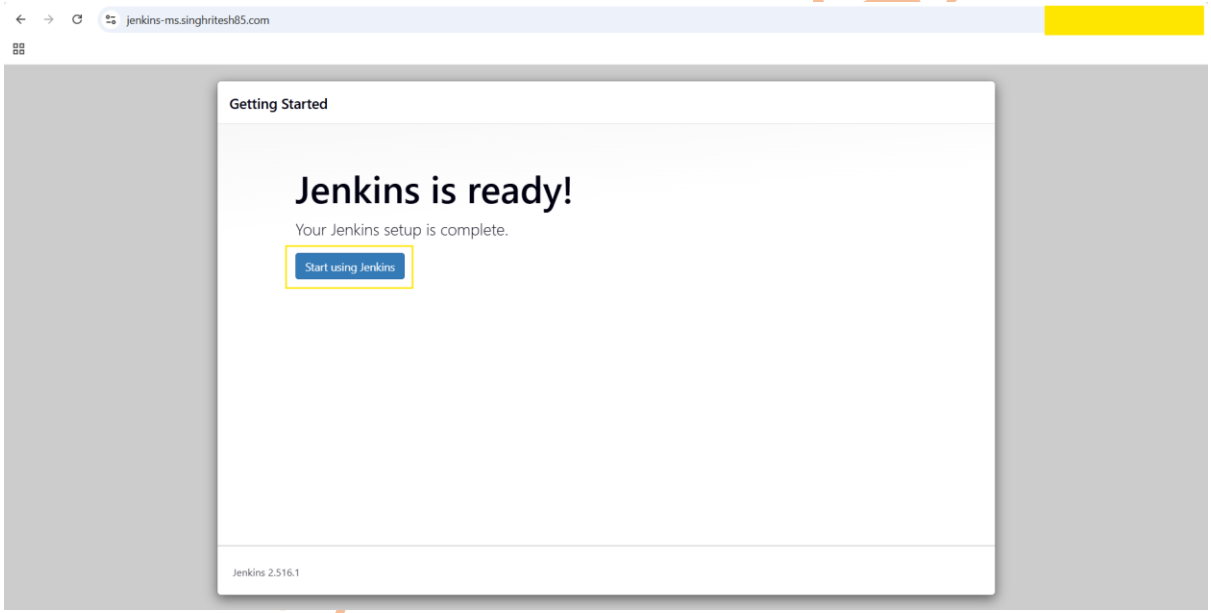




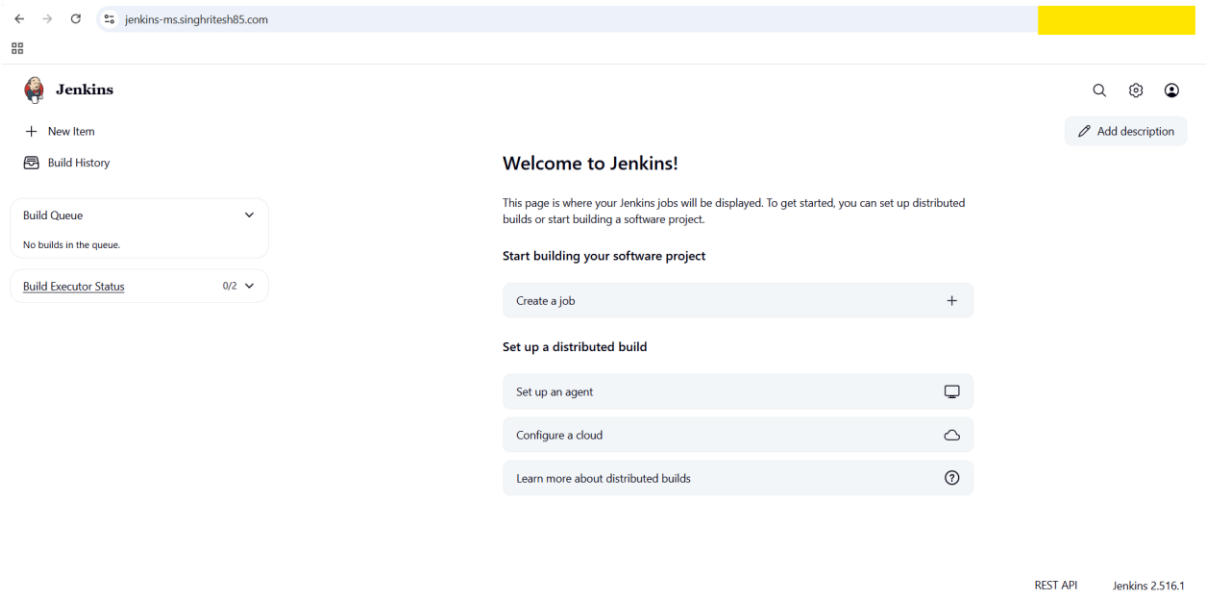
Provided username and password of Admin user as shown in the screenshot attached below.



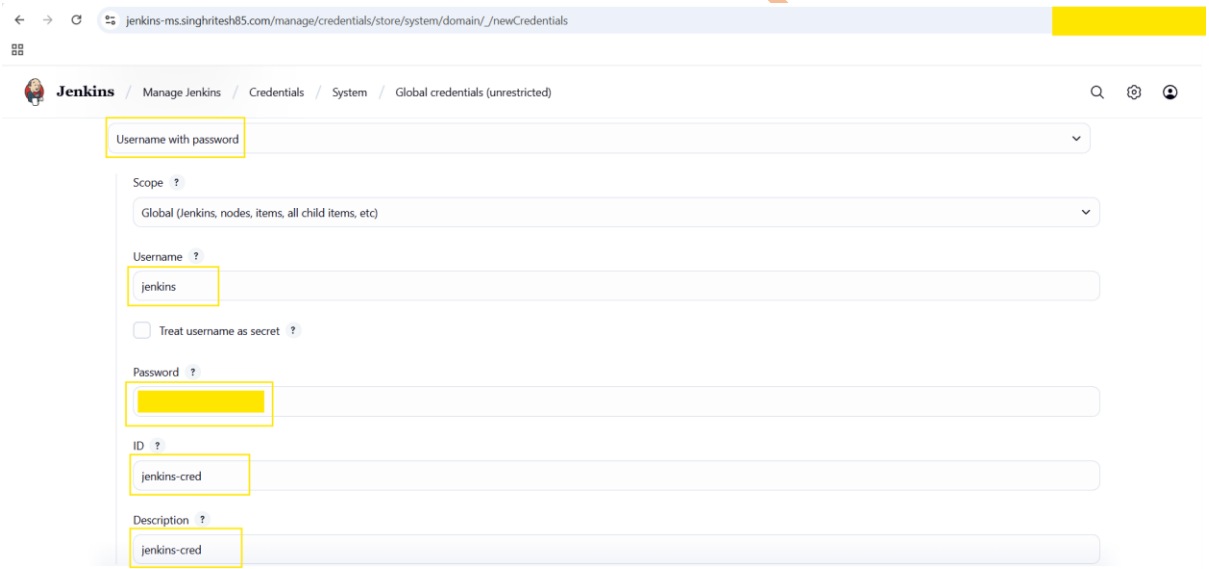


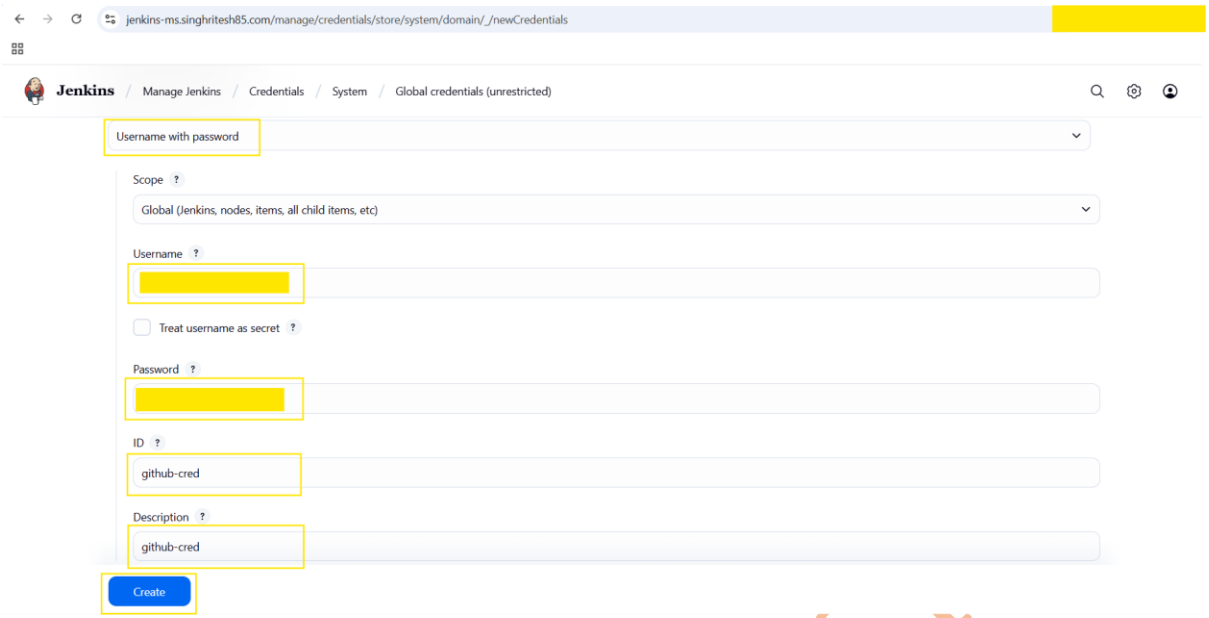


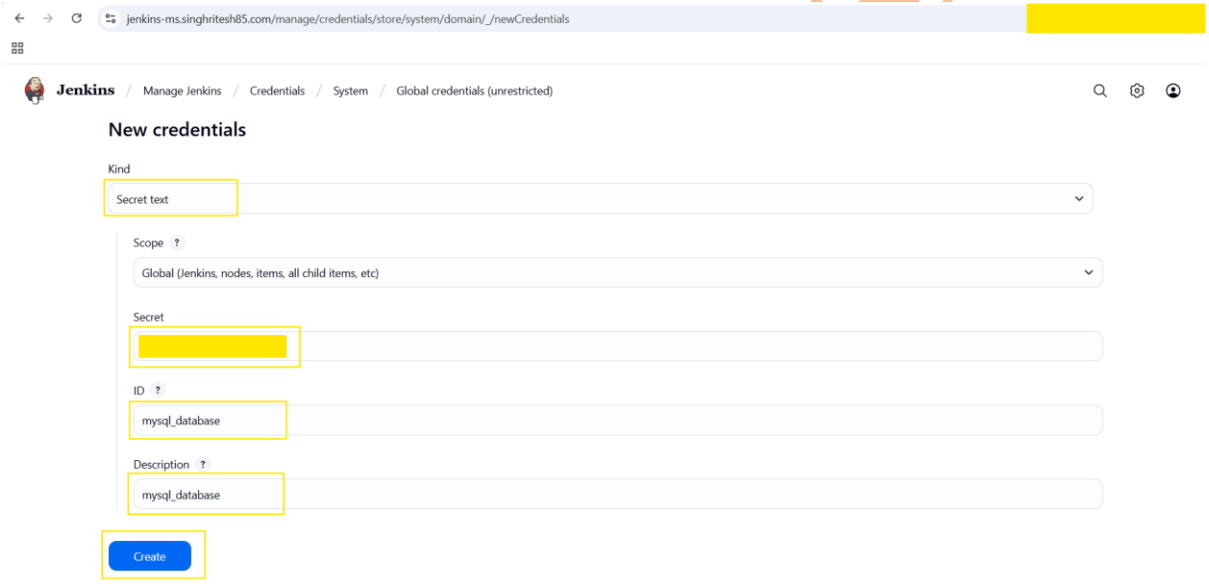
Finally, I was able to access Jenkins as shown in the screenshot attached below.

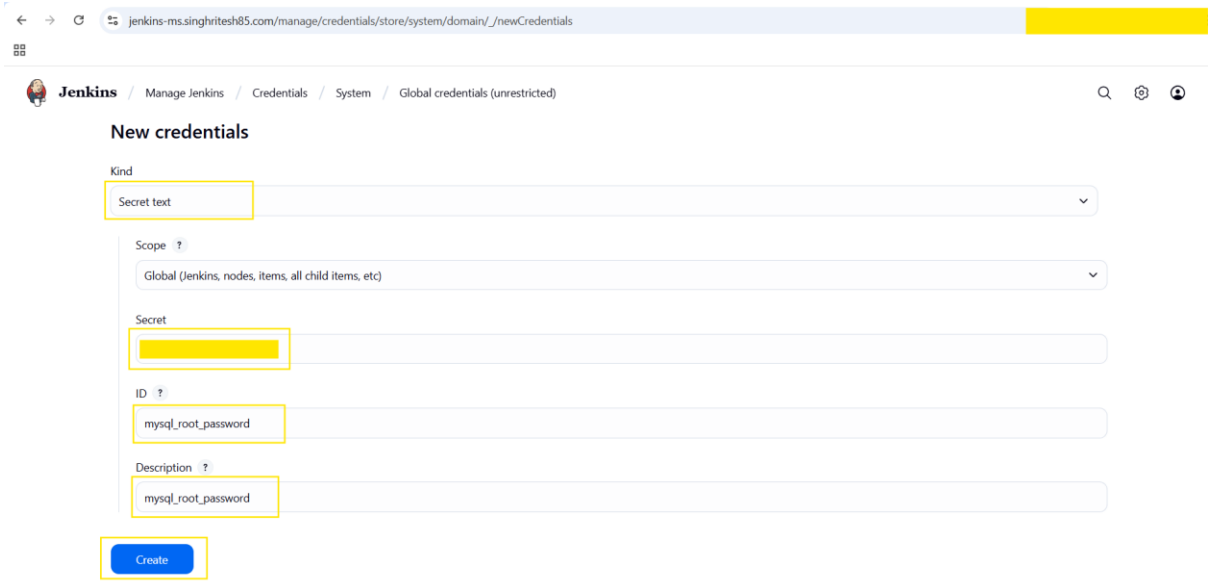


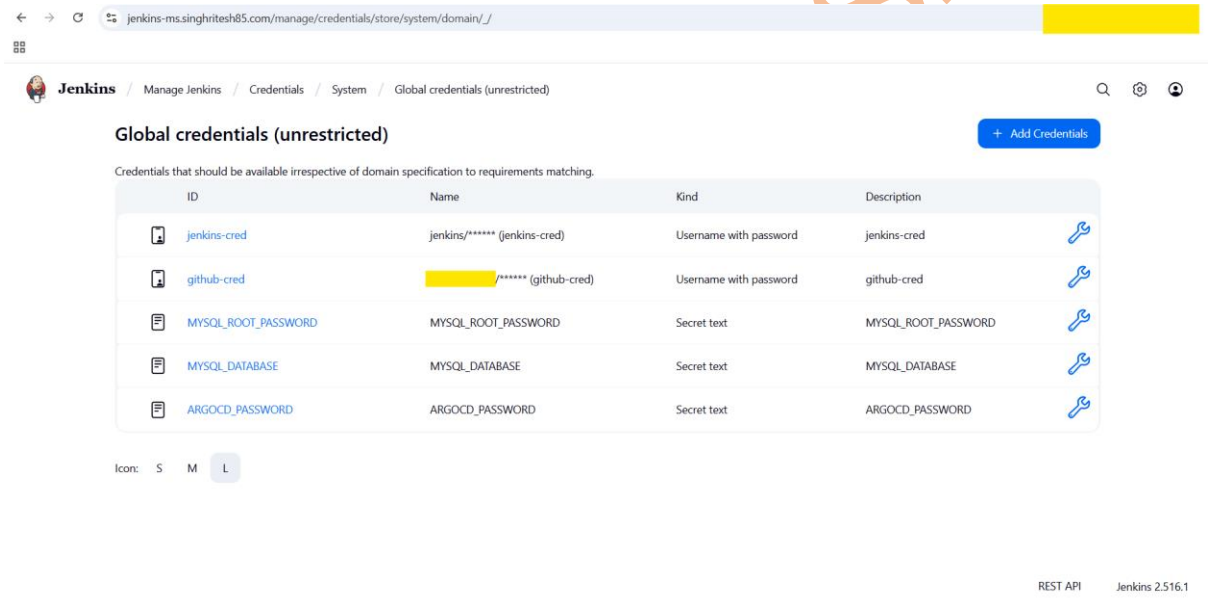
After Login into Jenkins, I created the Jenkins Credentials jenkins-cred, github-cred of **kind Username with Password** and **ARGOCD_PASSWORD**, **MYSQL_DATABASE** and **MYSQL_ROOT_PASSWORD** of kind secret text.



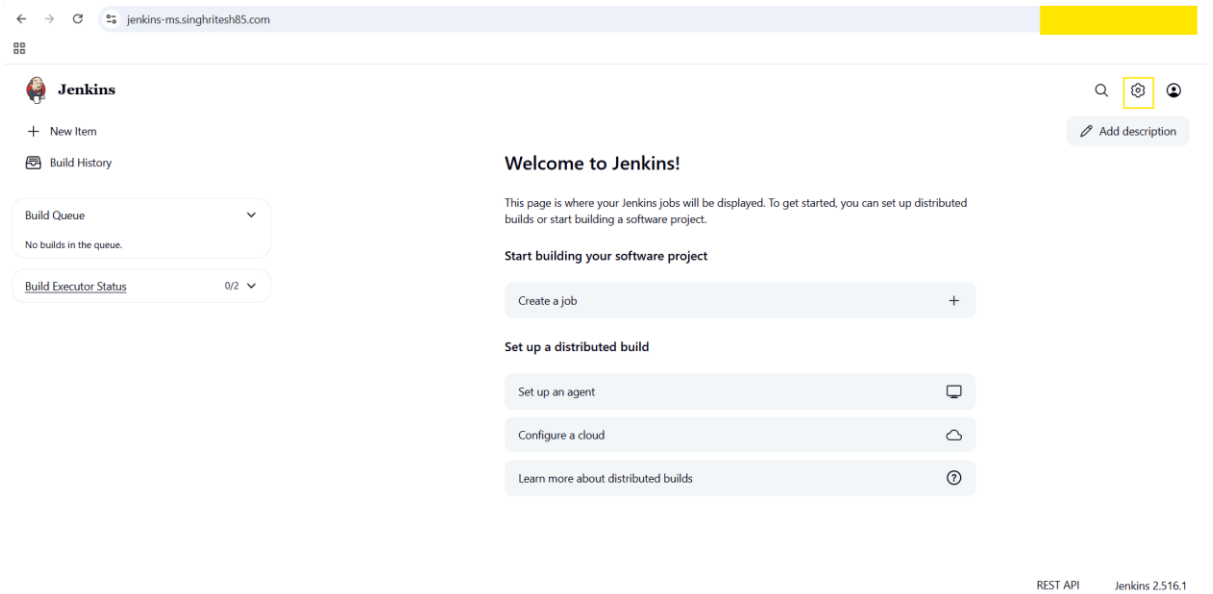

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials
Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted)
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
Treat username as secret
Password
ID
github-cred
Description
github-cred
Create

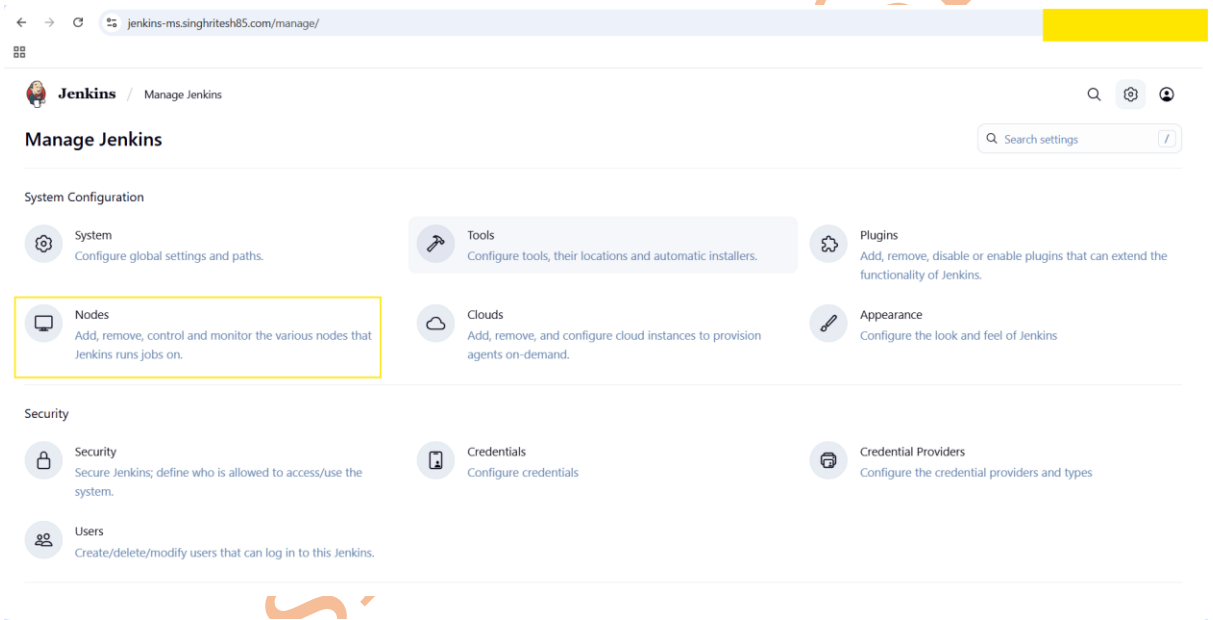

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials
Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted)
New credentials
Kind
Secret text
Scope
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID
mysql_database
Description
mysql_database
Create

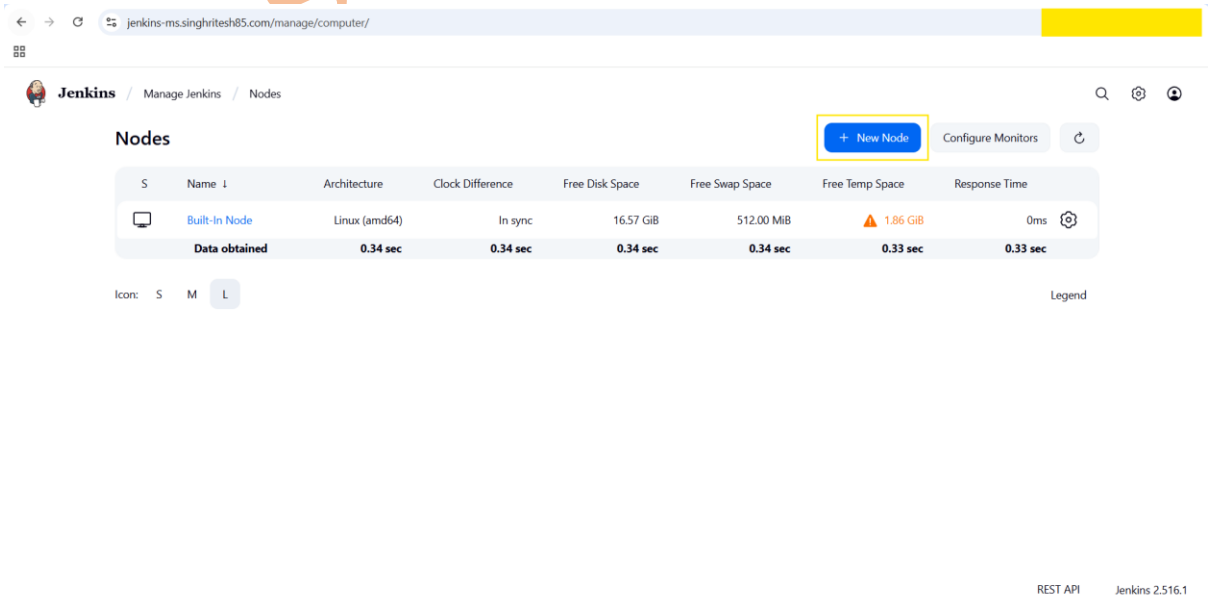


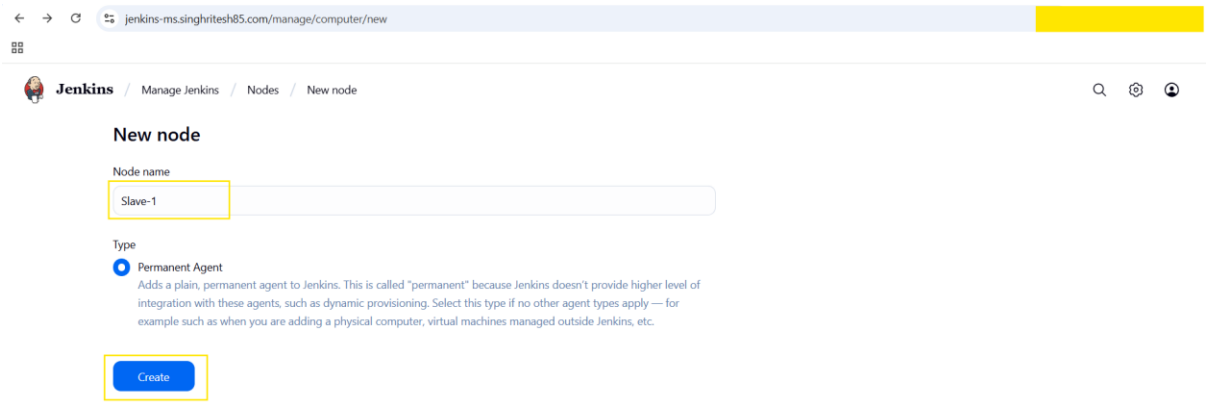


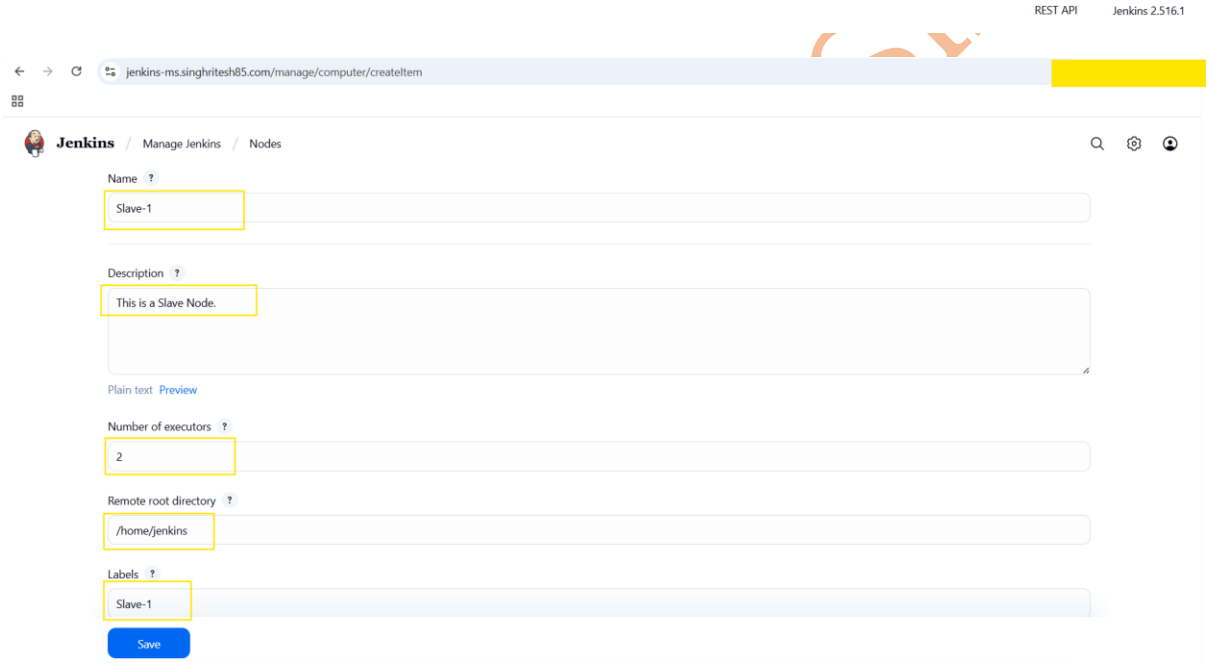
Then I added the Jenkins-Slave to the Jenkins-Master as shown in the screenshot attached below.

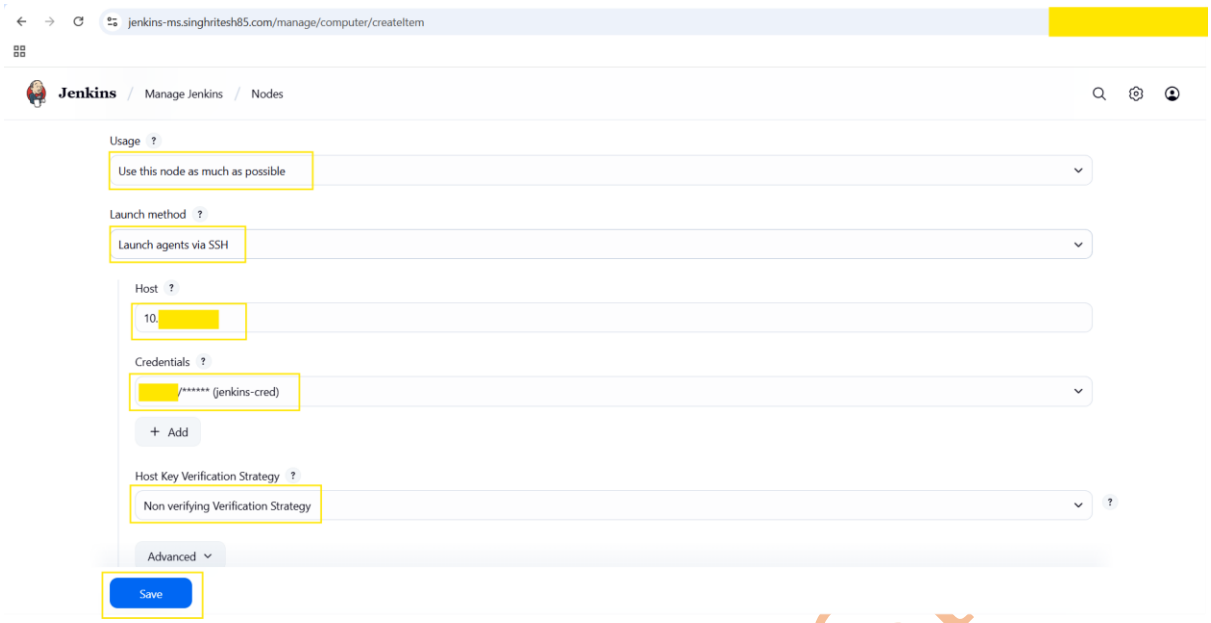

jenkins-ms.singhritesh85.com
Jenkins
New Item
Build History
Build Queue
No builds in the queue.
Build Executor Status 0/2
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds
REST API
Jenkins 2.516.1


jenkins-ms.singhritesh85.com/manage/
Jenkins / Manage Jenkins
Manage Jenkins
Search settings
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Appearance
Configure the look and feel of Jenkins
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users
Create/delete/modify users that can log in to this Jenkins.


jenkins-ms.singhritesh85.com/manage/computer/
Jenkins / Manage Jenkins / Nodes
Nodes
New Node
Configure Monitors
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
Built-In Node
Linux (amd64)
In sync
16.57 GiB
512.00 MiB
1.86 GiB
0ms
Data obtained
0.34 sec
0.34 sec
0.34 sec
0.34 sec
0.33 sec
0.33 sec
Icon: S M L
Legend
REST API
Jenkins 2.516.1

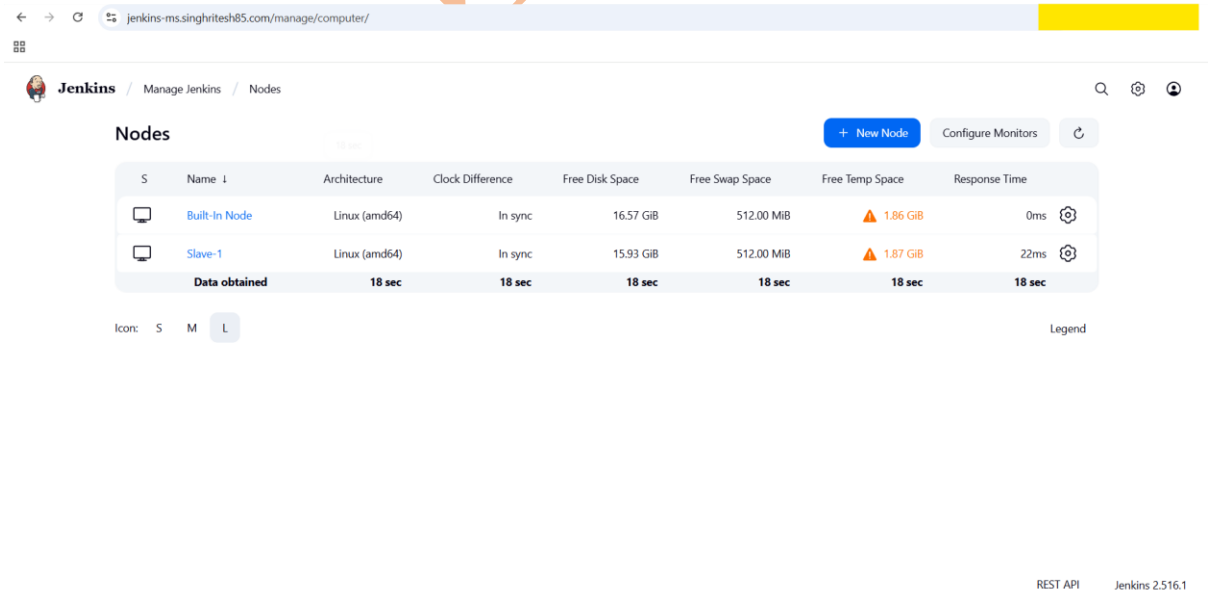

jenkins-ms.singhritesh85.com/manage/computer/new
Jenkins / Manage Jenkins / Nodes / New node
New node
Node name
Slave-1
Type
Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
Create


REST API
Jenkins 2.516.1
jenkins-ms.singhritesh85.com/manage/computer/createItem
Jenkins / Manage Jenkins / Nodes
Name
Slave-1
Description
This is a Slave Node.
Plain text Preview
Number of executors
2
Remote root directory
/home/jenkins
Labels
Slave-1
Save



Below commands I executed on Jenkins-Master and Jenkins-Slave to add swap space of 512 MiB.

```
[root@jenkins-slave ~]# fallocate -l 512M /swapfile
[root@jenkins-slave ~]# chmod 600 /swapfile
[root@jenkins-slave ~]# mkswap /swapfile
Setting up swapspace version 1, size = 512 MiB (536866816 bytes)
no label, UUID=
[root@jenkins-slave ~]# vim /etc/fstab
[root@jenkins-slave ~]# swapon -a
[root@jenkins-slave ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           3.7Gi       454Mi       850Mi       0.0Ki       2.5Gi       3.0Gi
Swap:          511Mi          0B       511Mi
[root@jenkins-slave ~]# cat /etc/fstab
#
UUID=                                     /           xfs    defaults,noatime  1   1
UUID=           /boot/efi       vfat    defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2

/swapfile swap swap defaults 0 0
```



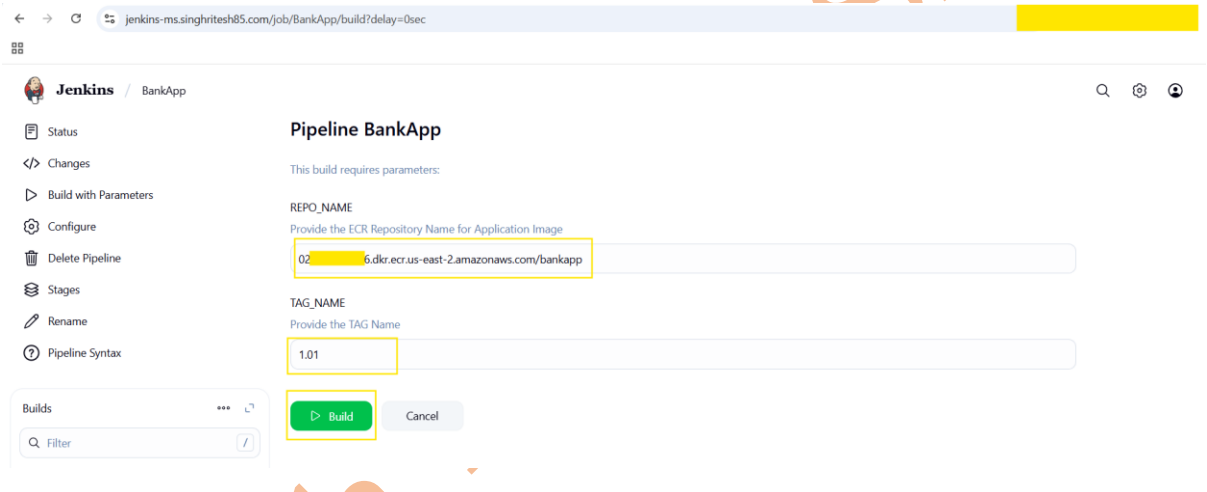
For this project I have not configured SonarQube and Nexus if you require to use it during the pipeline in your project you can refer the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git.

For this project to refrain from higher cloud cost I used **t3.medium** instance type for Jenkins-Master and Jenkins-Slave.

Then I created the Jenkins Job to create the Pods as shown in the screenshot attached below. The source code is present in the GitHub Repo https://github.com/singhritesh85/Bank-App-multibranch.git. To create Jenkins Job for MySQL Pods deployment, Jenkinsfile is present at the path **mysql/Jenkinsfile** and to create the Jenkins Job for BankApp Pods deployment, Jenkinsfile is present at the path **Jenkinsfile** in the root directory. First run the Jenkins Job for MySQL Pod deployment and then run the Jenkins Job for BankApp Pod deployment.

```
[jenkins@jenkins-slave ~]$ kubectl get pods -n mysql --watch
NAME                READY   STATUS    RESTARTS   AGE
mysql-primary-0     1/1     Running   0          29s
mysql-secondary-0   1/1     Running   0          32s
```

To run the Jenkins Job for BankApp I provided string parameters as shown in the screenshot attached below.




```
[jenkins@jenkins-slave ~]$ kubectl get pods -n bankapp --watch
NAME                              READY   STATUS    RESTARTS   AGE
bankapp-folo-                     1/1     Running   0          3m1s
```

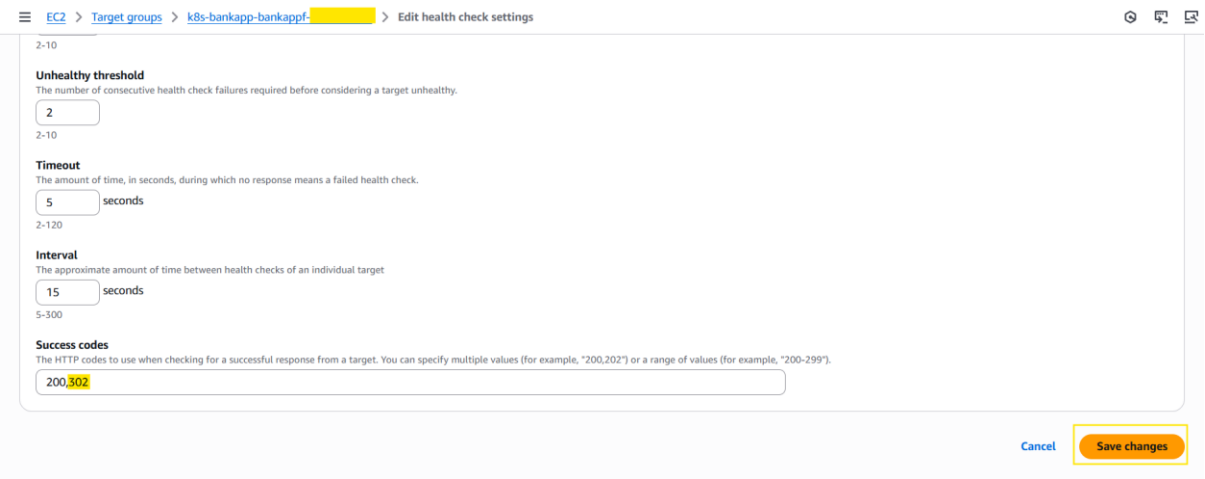
```
[jenkins@jenkins-slave ~]$ cat ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:026:certificate/
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb-ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/wafv2-acl-arn: arn:aws:wafv2:us-east-2:026:regional/webacl/aws-wafv2-rate-limit/
spec:
  ingressClassName: alb
  rules:
    - host: bankapp.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: bankapp-folo
                port:
                  number: 80
```

```
cat ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:02XXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb-ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/wafv2-acl-arn: arn:aws:wafv2:us-east-2:02XXXXXXXX6:regional/webacl/aws-wafv2-rate-limit/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
spec:
  ingressClassName: alb
  rules:
    - host: bankapp.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: bankapp-folo
                port:
                  number: 80
```

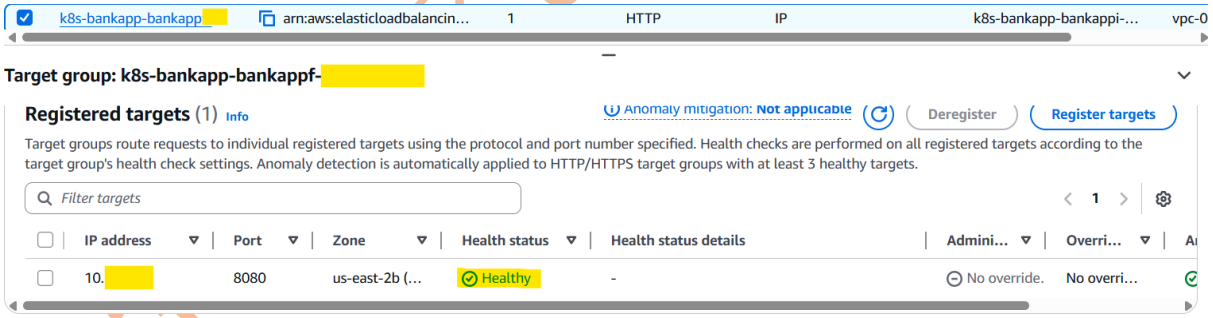
```
[jenkins@jenkins-slave ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/bankapp-ingress created
```

```
[jenkins@jenkins-slave ~]$ kubectl get ing -n bankapp --watch
NAME              CLASS   HOSTS                    ADDRESS                                                                  PORTS   AGE
bankapp-ingress   alb     bankapp.singhritesh85.com   k8s-bankapp-bankappi-                    .us-east-2.elb.amazonaws.com   80      3m4s
```

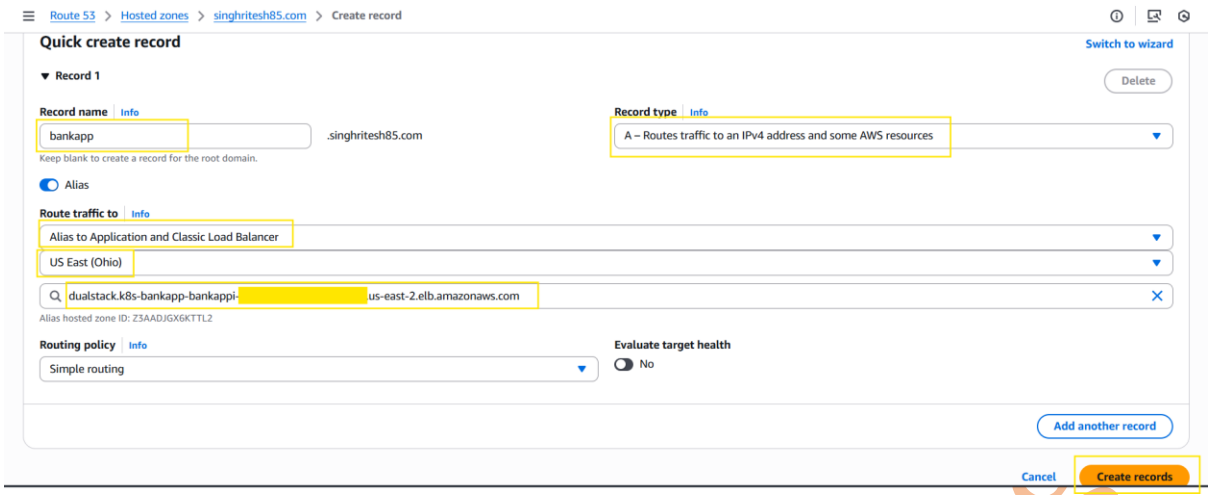
After creation of each Ingress Rule results an Application LoadBalancer, similarly after creating the ingress rule to access the bankapp will result a new Application LoadBalancer which and when I checked its Target Group health check I fount it was Unhealth with the reason of **Health checks failed with these codes: [302]**. This is not a major problem and it can be resolved by adding a code 302 in health check setting success code as shown in the screenshot attached below.

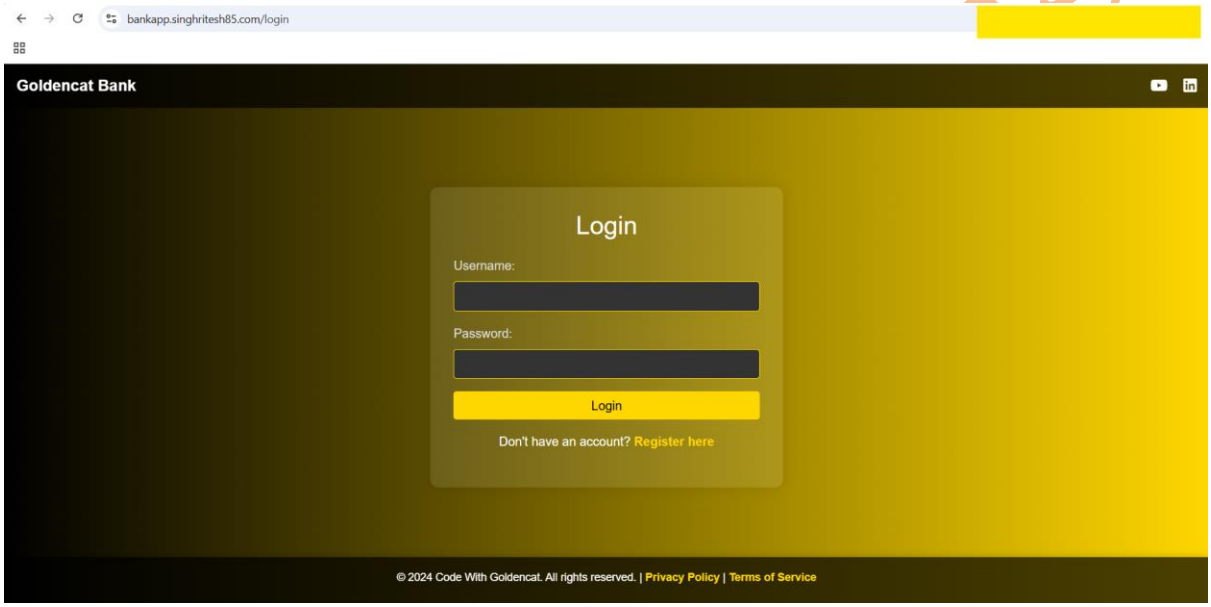


First understand what does this Unhealthy state means, it shows Moved Temporarily redirect which means requested resource has been temporarily moved to a different URL. When we access the Application on https://bankapp.singhritesh85.com it redirects to https://bankapp.singhritesh85.com/login and it was performing the heath check on path / and not on /login. After I white listed the success code of AWS Application LoadBalancer health check with 302 this problem got resolved.

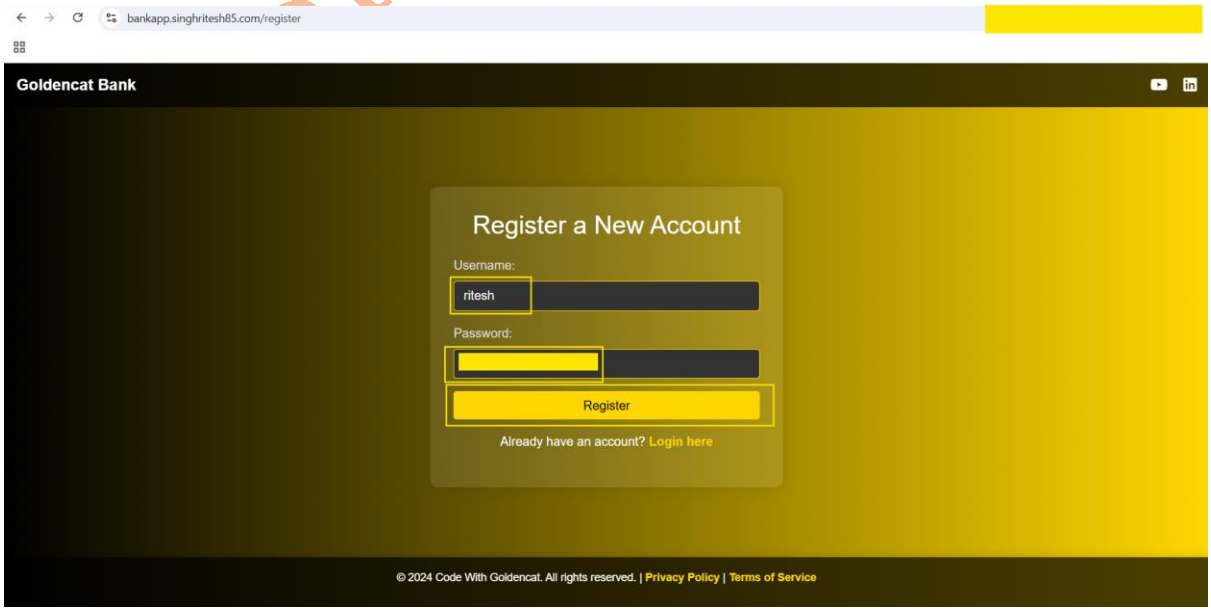


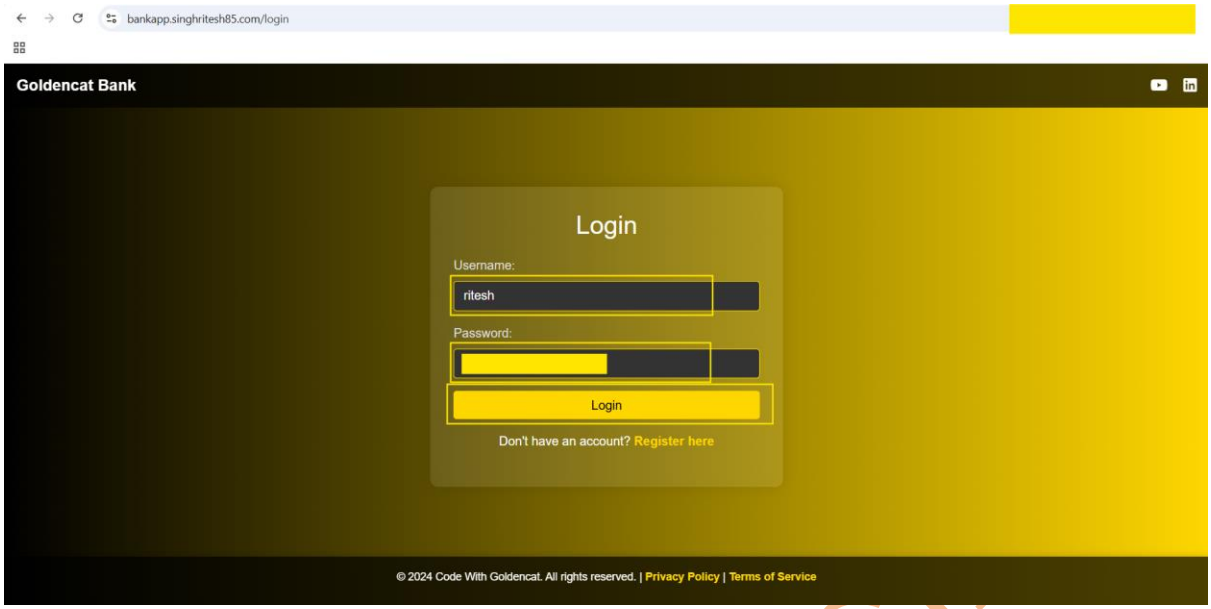
Finally, the Target Group is in healthy state as shown in the screenshot attached above. With the created Application LoadBalancer DNS Name and HOST of the ingress rule I created a Route53 record set of A-Type. Using the created URL I was able to access the Bank Application as shown in the screenshot attached below.

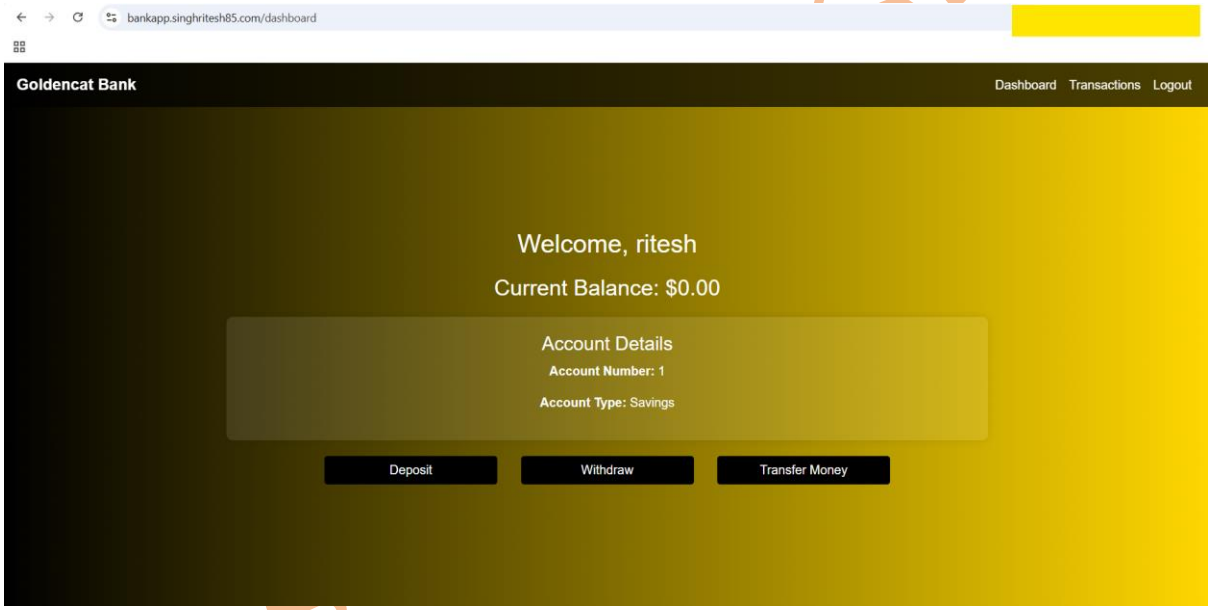




Then I registered a user and logged-in with the same user and found the entry of MySQL8.




bankapp.singhritesh85.com/login
Goldencat Bank
Login
Username:
ritesh
Password:
Login
Don't have an account? Register here
© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service


bankapp.singhritesh85.com/dashboard
Goldencat Bank
Dashboard Transactions Logout
Welcome, ritesh
Current Balance: $0.00
Account Details
Account Number: 1
Account Type: Savings
Deposit
Withdraw
Transfer Money

```
[jenkins@jenkins-slave ~]$ kubectl exec -it mysql-primary-0 -- /bin/bash -n mysql
Error from server (NotFound): pods "mysql-primary-0" not found
[jenkins@jenkins-slave ~]$ kubectl get pods -n mysql
NAME                READY   STATUS    RESTARTS      AGE
mysql-primary-0     1/1     Running   0             49m
mysql-secondary-0   1/1     Running   1 (48m ago)   49m
[jenkins@jenkins-slave ~]$ kubectl exec -it mysql-primary-0 -- /bin/bash -n mysql
Error from server (NotFound): pods "mysql-primary-0" not found
[jenkins@jenkins-slave ~]$ kubectl exec -it mysql-primary-0 -n mysql -- /bin/bash
Defaulted container "mysql" out of: mysql, preserve-logs-symlinks (init)
I have no name!@mysql-primary-0:/$ mysql -h localhost -u root --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 635
Server version: 8.4.0 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| bankappdb          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
mysql> use bankappdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_bankappdb |
+--------------------+
| account            |
| transaction        |
+--------------------+
2 rows in set (0.00 sec)

mysql> select * from account;
+----+---------+--------------------------------------------------------------+----------+
| id | balance | password                                                     | username |
+----+---------+--------------------------------------------------------------+----------+
|  1 |    0.00 | [redacted]                                                   | ritesh   |
+----+---------+--------------------------------------------------------------+----------+
1 row in set (0.00 sec)
```

Using velero it is possible to create the backup of EKS Cluster, for the current project I am creating backup of full EKS cluster using velero.

```
[root@                ~]# velero schedule create backup --schedule="30 */12 * * *" --ttl=720h
Schedule "backup" created successfully.
[root@                ~]# velero schedule get
NAME     STATUS    CREATED                         SCHEDULE        BACKUP TTL   LAST BACKUP   SELECTOR   PAUSED
backup   Enabled   2025-08-20 08:24:27 +0000 UTC   30 */12 * * *   720h0m0s     n/a           <none>     false
```

It will create the backup at 12:30 AM in night and 12:30 PM in day time UTC.

You can get the velero backup storage location as shown in the screenshot attached below.

```
[root@                ~]# velero backup-location get
NAME      PROVIDER   BUCKET/PREFIX          PHASE       LAST VALIDATED                  ACCESS MODE   DEFAULT
default   aws        dexter-velero-backup   Available   2025-08-20 08:26:32 +0000 UTC   ReadWrite     true
```

For this project I created the manual backup by executing the command **velero backup create backup-`date +"%Y%m%d%H%M%S"`** manually as shown in the screenshot attached below.

```
[root@                ~]# velero backup create backup-`date +"%Y%m%d%H%M%S"`
Backup request "backup-20250820082659" submitted successfully.
Run `velero backup describe backup-20250820082659` or `velero backup logs backup-20250820082659` for more details.
```

Velero backup successfully created as shown in the screenshot attached below.

```
[root@                ~]# velero backup get
NAME                   STATUS      ERRORS   WARNINGS   CREATED                         EXPIRES   STORAGE LOCATION   SELECTOR
backup-20250820082659   Completed   0        0          2025-08-20 08:27:00 +0000 UTC   29d       default            <none>
```

After this manual backup I upgraded the EKS version from 1.32 to 1.33 for this you can use terraform import for the eks cluster and change the version or in the existing terraform script you do changes as shown in the screenshot attached below then run the terraform script. As I am aware here after doing changes in the current terraform script it will not affect the other existing AWS resources So I did change in the current terraform script.

```
name     = "${var.eks_cluster}-${var.env}"
role_arn = aws_iam_role.eksdemorole.arn
version = var.kubernetes_version[10]
```

```
ami_type        = var.ami_type[0]
capacity_type   = var.capacity_type[0]
release_version = var.release_version[10]
```

Just after creation of EKS Cluster the nodes present in the cluster is as shown below.

```
[root@                ~]# kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-10-        us-east-2.compute.internal      Ready    <none>   7m55s   v1.32.7-eks-3abbec1
ip-10-        .us-east-2.compute.internal     Ready    <none>   7m55s   v1.32.7-eks-3abbec1
```

Just before Kubernetes version upgrade the Nodes present in EKS Cluster and eks version is as shown in the screenshot attached below. The below screenshot shows there were three nodes created by karpenter.

```
[root@                ~]# kubectl get nodes
NAME                                        STATUS   ROLES    AGE     VERSION
ip-10-          us-east-2.compute.internal   Ready    <none>   63m     v1.32.7-eks-3abbec1
ip-10-          us-east-2.compute.internal   Ready    <none>   77m     v1.32.7-eks-3abbec1
ip-10-          .us-east-2.compute.internal  Ready    <none>   77m     v1.32.7-eks-3abbec1
ip-10-          .us-east-2.compute.internal  Ready    <none>   9m21s   v1.32.7-eks-3abbec1
ip-10-          .us-east-2.compute.internal  Ready    <none>   9m24s   v1.32.7-eks-3abbec1
```

After the upgrade EKS version is as shown below.

```
[root@                ~]# kubectl get nodes
NAME                                        STATUS   ROLES    AGE     VERSION
ip-10-          us-east-2.compute.internal   Ready    <none>   20m     v1.33.0-eks-802817d
ip-10-          us-east-2.compute.internal   Ready    <none>   18m     v1.33.0-eks-802817d
ip-10-          us-east-2.compute.internal   Ready    <none>   5m5s    v1.33.3-eks-3abbec1
ip-10-          .us-east-2.compute.internal  Ready    <none>   3m22s   v1.33.3-eks-3abbec1
ip-10-          .us-east-2.compute.internal  Ready    <none>   3m7s    v1.33.3-eks-3abbec1
```

After upgrade make sure to check all the pods present on the EKS Cluster in Running state using the command **kubectl get pods -A**. No Pod should be in pending state after the upgrade. In the screenshot attached above you can see two nodes are with the version 1.33.0 (as initially I created two nodes in the node group the targeted minor version is 1.33.0 these two nodes are those nodes) and three nodes are created with karpenter and Karpenter will look for the most up-to-date EKS-optimized AMI available for that minor version and hence these nodes are provisioned with the minor version 1.33.3.

After this upgrade of EKS version from 1.32 to 1.33 I was able to access the BankApp successfully. But consider the scenario when both the mysql and bankapp namespace is not present on the upgraded EKS version. In that case I retrieved the namespace mysql and bankapp as shown in the screenshot attached below.

velero restore create namespace-mysql --from-backup backup-20250819093307 --include-namespaces mysql

velero restore create namespace-bankapp --from-backup backup-20250819093307 --include-namespaces bankapp

```
[jenkins@jenkins-slave ~]$ kubectl get pods -n mysql
No resources found in mysql namespace.
[jenkins@jenkins-slave ~]$ kubectl get pods -n bankapp
No resources found in bankapp namespace.
```

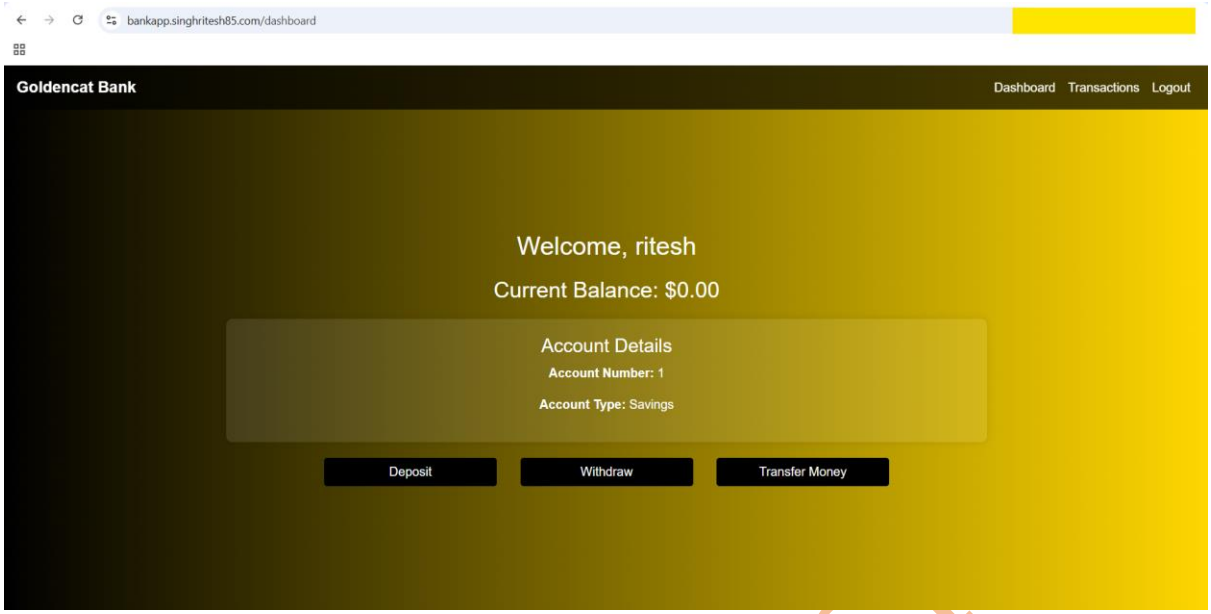
```
[root@                ~]# velero restore create namespace-mysql --from-backup backup-20250820082659 --include-namespaces mysql
Restore request "namespace-mysql" submitted successfully.
Run `velero restore describe namespace-mysql` or `velero restore logs namespace-mysql` for more details.
```

When I checked MySQL Pods are up and in Running state then I restored bankapp pods as shown in the screenshot attached below.

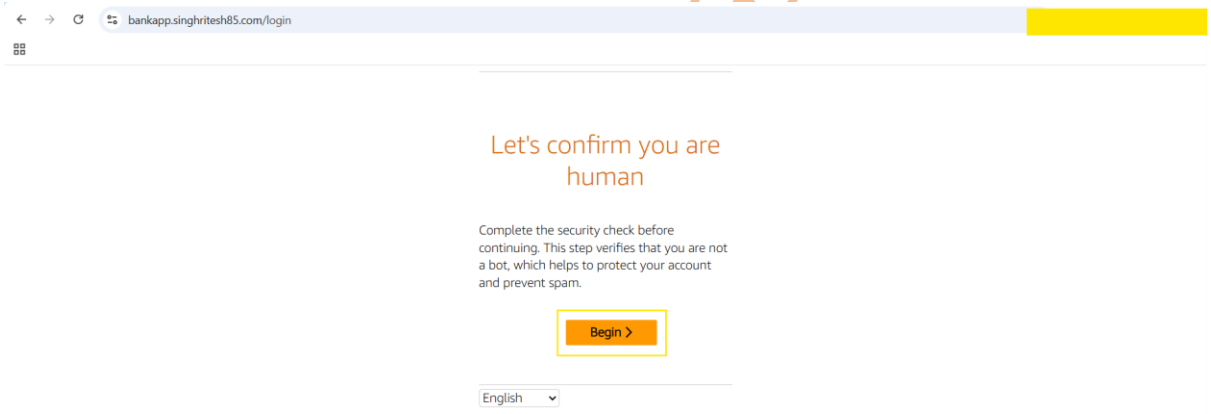
```
[root@                ~]# velero restore get
NAME              BACKUP                   STATUS      STARTED                         COMPLETED                       ERRORS   WARNINGS   CREATED
SELECTOR
namespace-mysql   backup-20250820082659    Completed   2025-08-20 08:31:42 +0000 UTC   2025-08-20 08:31:48 +0000 UTC   0        1          2025-08-20 08:31:42
```

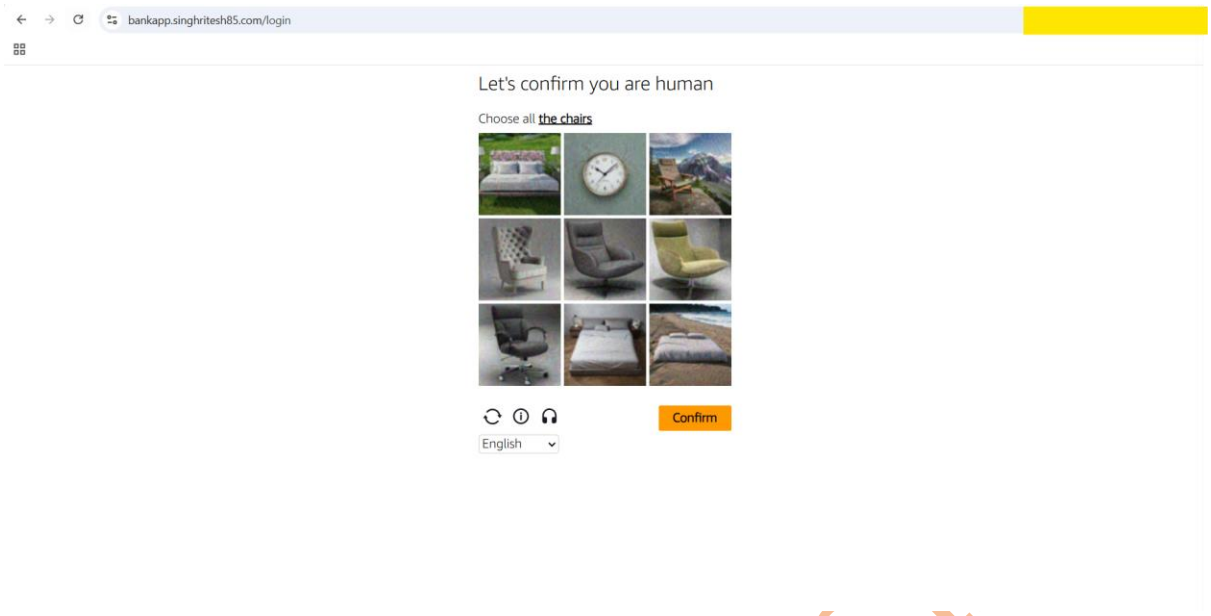
```
[root@                ~]# velero restore create namespace-bankapp --from-backup backup-20250820082659 --include-namespaces bankapp
Restore request "namespace-bankapp" submitted successfully.
Run `velero restore describe namespace-bankapp` or `velero restore logs namespace-bankapp` for more details.
```

As this time a new AWS ALB gets created for BankApp after the restore then do its entry in Route53 to create the A-Type record set as discussed above. Finally, I was able to access the BankApp and the user was also existed in the MySQL database as earlier.

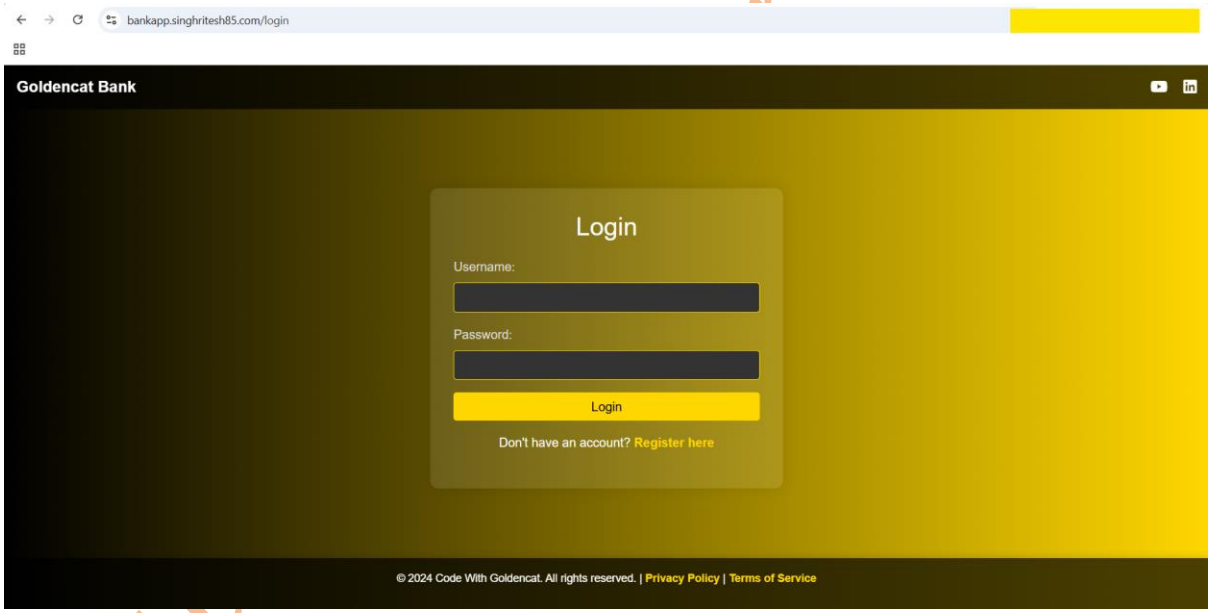


To test the Captcha and Rate Limit, I access the website https://bankapp.singhritesh85.com/login more than 30 and 40 times respectively in a minute and found the below results.





After Validating Captcha, you will be able to access the Bank Application as shown in the screenshot attached below.



When the request more than 40, rate-limit blocks the Bank Application and will show the message **error: Too Many Requests** as shown in the screenshot attached below.

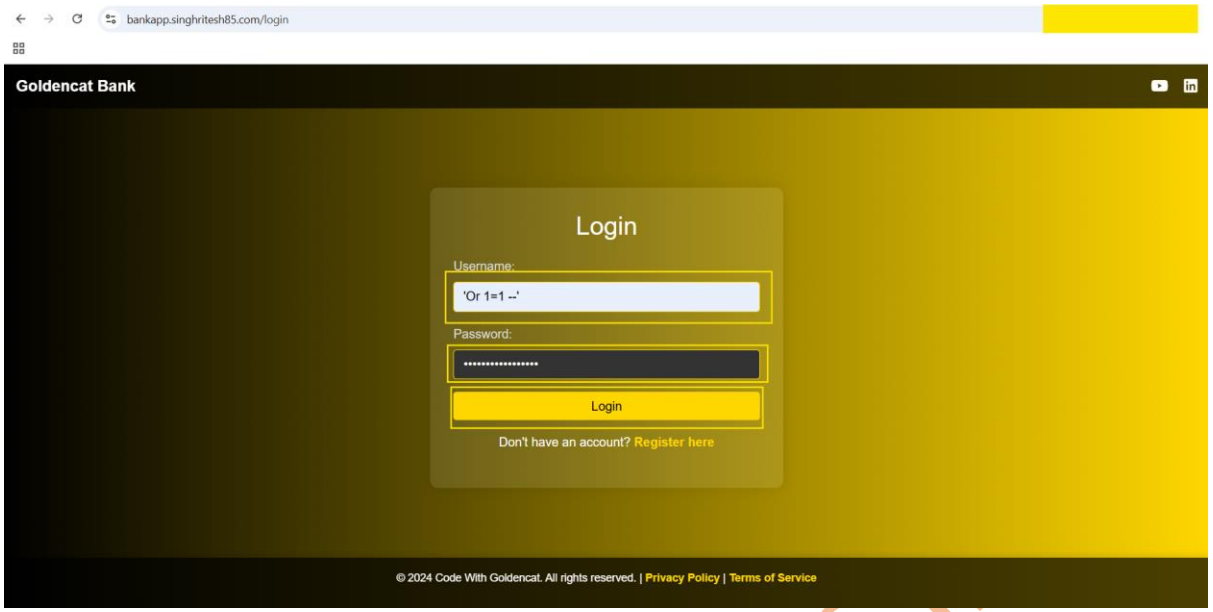
You can also test the same with the shell-script as well which is provided below.

```
[jenkins@jenkins-slave ~]$ cat test.sh
#!/bin/bash

while true
do
curl -k https://bankapp.singhritesh85.com/login
done
```

```
[jenkins@jenkins-slave ~]$ sh test.sh
```

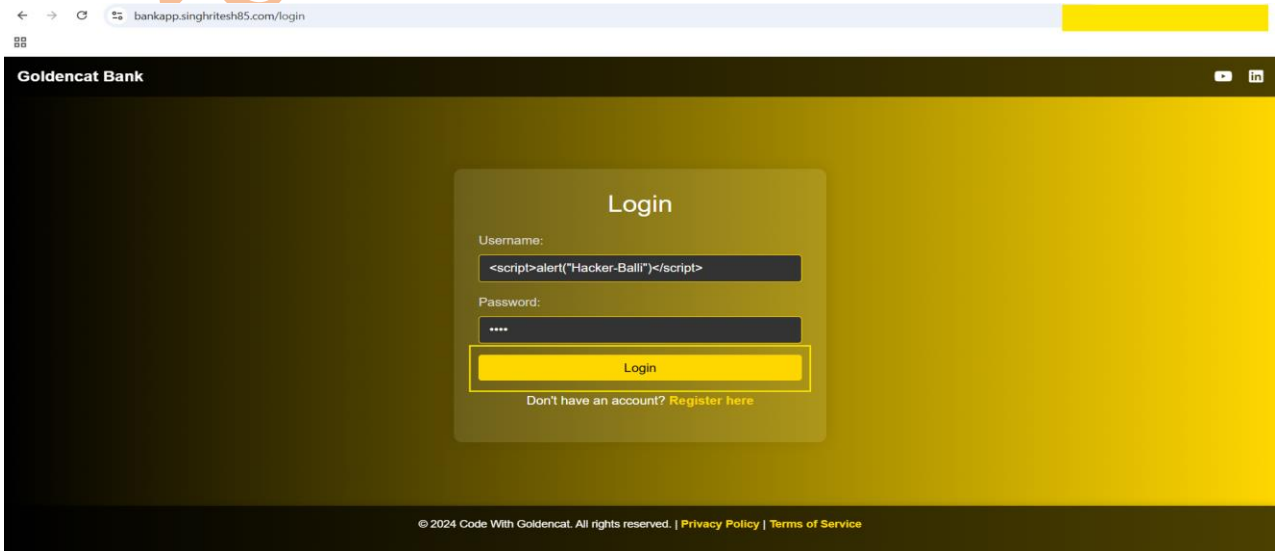
```
    <h2 class="text-center">Login</h2>
    <form method="post" action="/login">
        <div class="form-group">
            <label>Username:</label>
            <input type="text" class="form-control" name="username" required />
        </div>
        <div class="form-group">
            <label>Password:</label>
            <input type="password" class="form-control" name="password" required />
        </div>
        <button type="submit" class="btn btn-block">Login</button>
    </form>
    <p class="text-center mt-3">Don't have an account? <a href="/register" class="custom-link">Register here</a></p>


</div>

<!-- Footer -->
<footer class="footer">
    <p>&copy; 2024 Code With Goldencat. All rights reserved. | <a href="#" class="custom-link">Privacy Policy</a> | <a href="#" class="custom-link">Terms of S
ervice</a></p>
</footer>

</body>
</html>
{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
}{
    "error":"Too Many Requests."
```

To test SQL Injection, I provider the user as **'Or 1=1 --'** and any of the password and found 403 error as shown in the screenshot attached below.



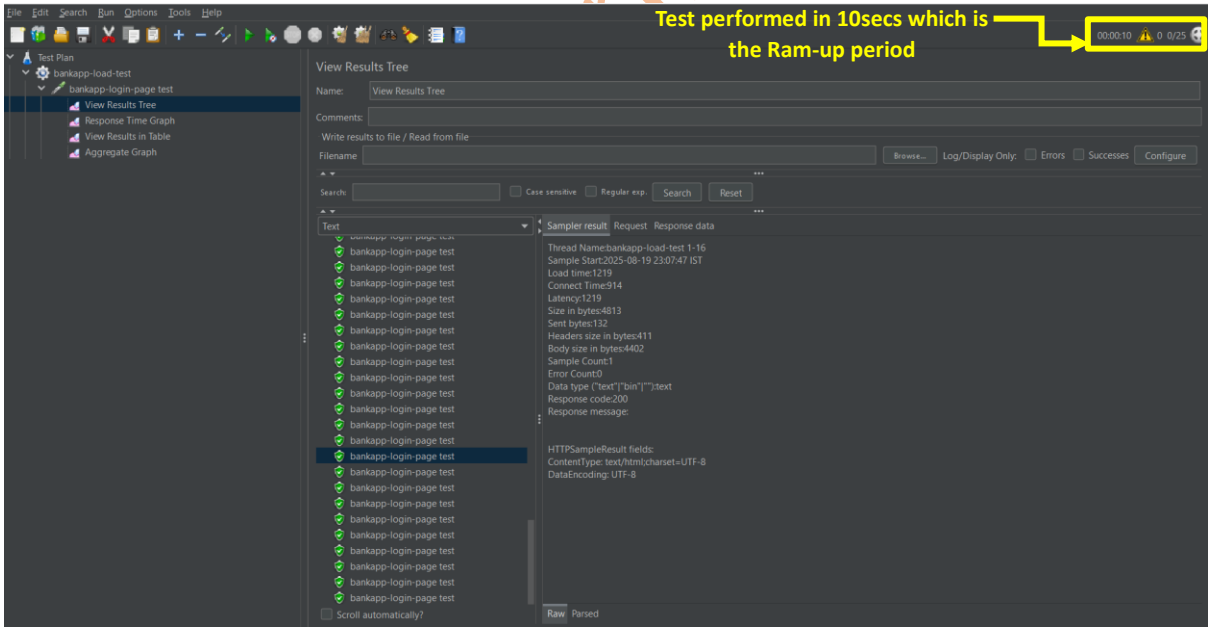
403 Forbidden

This WAF web ACL created using terraform also protects from XSS (Cross-Site Scripting) attack as shown in the screenshot attached below.

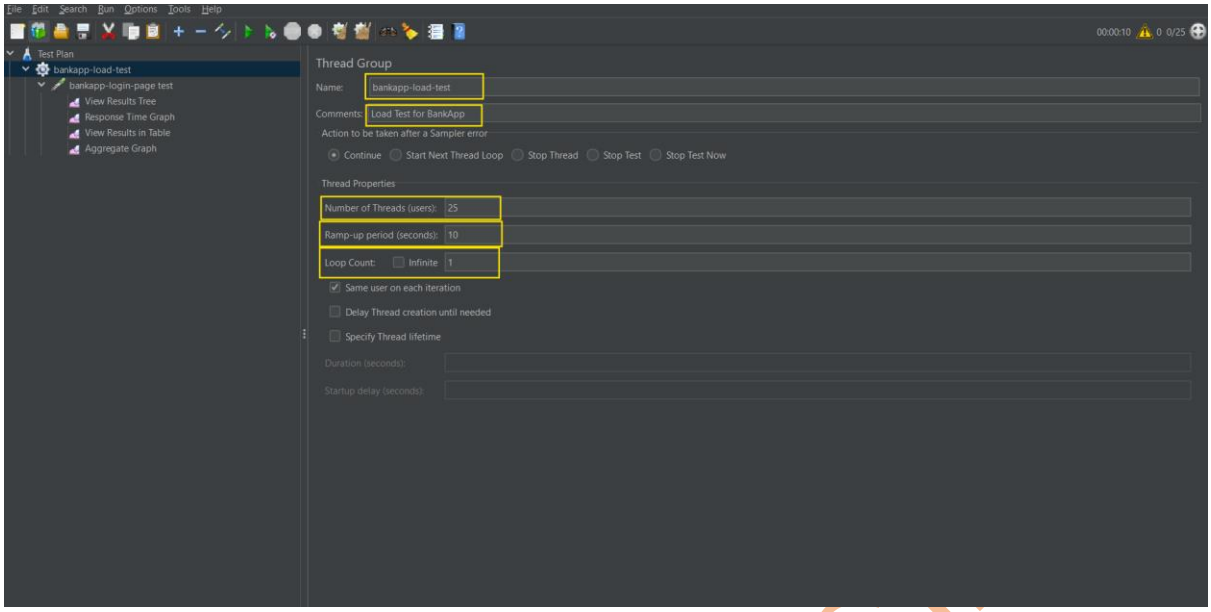


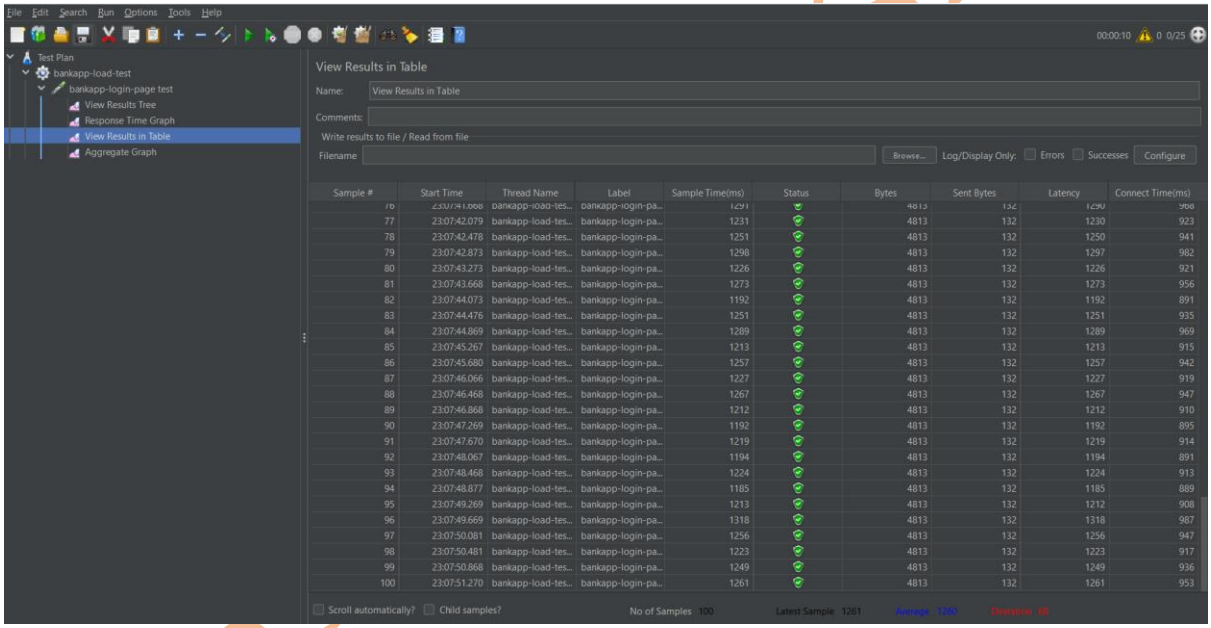
Reflected XSS happens when malicious script is part of the request as shown in the screenshot attached above. It should be reflected in the Server Response but as shown in the screenshot attached below, it was blocked with 403 Error (which means Server understood your request but refused to Grant access).

Finally, I did the Load Test using Apache JMETER as shown in the screenshot attached below.



If you need, you can perform this Test in Infinite Loop but for this project I am running it for a single iteration.





The above Load Test I performed for 25 simultaneous users in Ramp-up time of 10 seconds (in a time frame of 10 seconds). I can conclude that the overall performance of BankApp URL for the 25 users who accessed the URL https://bankapp.singhritesh85.com/login in a ramp-up time of 10 seconds was satisfactory.

To ensure high-availability of BankApp during voluntary disruptions (Node maintenance or upgrade and cluster scaling operations) you can implement pod disruption budget (PDB), but make sure you have adequate number of pods available with the Application otherwise it will through an error **PodEvictionFailure** during NodeGroup upgrade or you need to do this activity during non-production hours and delete the PDB while performing the upgrade and the create PDB after the upgrade activity. I had already discussed PDB, you can refer the GitHub Repo https://github.com/singhritesh85/update-eks-cluster.git.

However, for your reference the pod disruption budget yaml manifests file as shown in the screenshot attached below.

```
cat pdb-stateless.yaml
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: bankapp-pdb
  namespace: bankapp
spec:
  selector:
    matchLabels:
      app.kubernetes.io/instance: bankapp
  minAvailable: 33%

cat pdb-statefull.yaml
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: mysql-pdb
  namespace: mysql
spec:
  selector:
    matchLabels:
      app.kubernetes.io/instance: mysql
  minAvailable: 33%
```

For this project I had opted only one pod for stateful and stateless applications So I had not implemented the PDB here because when you will take the **minAvailabe** in PDB as **33%** for one pod it will be rounded to one and hence it will not allow to upgrade that node on which this pod was scheduled. I had provided here the PDB manifests file just for your reference, if you need to implement it then scale-up the pods for stateful and stateless applications (you should have adequate number of pods available with the application) then implement it otherwise you may face error **PodEvictionFailure** during EKS version upgrade (during NodeGroup upgrade).

Source Code: https://github.com/singhritesh85/Bank-App-multibranch.git (Branch: - main)

GitHub Repo: https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git

Helm Chart: https://github.com/singhritesh85/helm-repo-for-ArgoCD.git

https://github.com/singhritesh85/helm-repo-for-bitnami.git

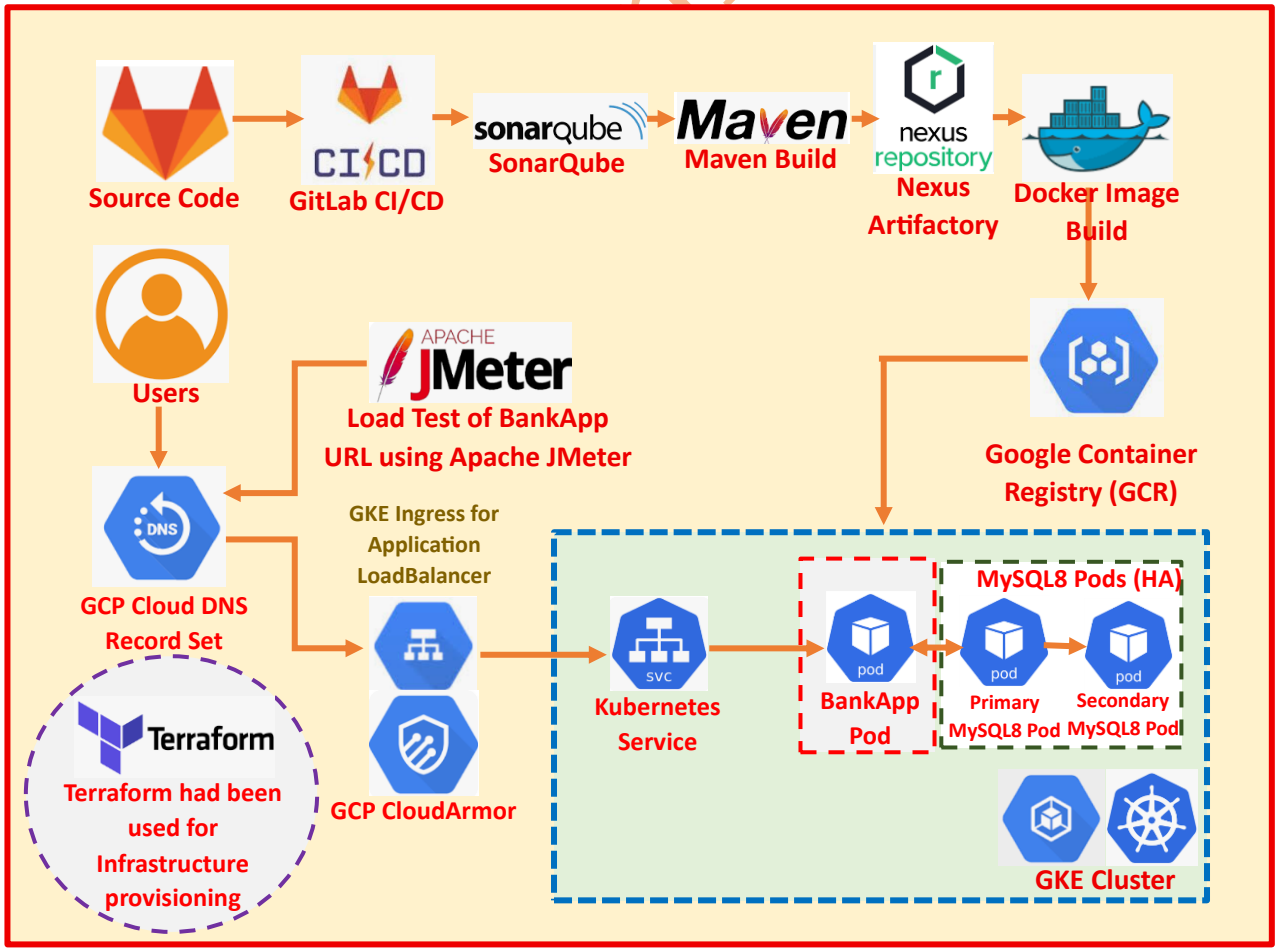
Terraform Script: https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git

References: https://github.com/Goldencat98/Bank-App.git

https://github.com/singhritesh85/update-eks-cluster.git

Module-3

DevOps-Project-BankApp-GKE-CloudArmor-Performance-Testing



This Project deals with the creation of infrastructure using Terraform, GitLab was used to keep the source Code and GitLab CI/CD had been used as CI/CD Tool. SonarQube was used for Code Analysis, the build tool used in this project was Maven, Sonatype Nexus Artifactory was used to keep the Artifacts and Docker Image was pushed to the Google Container Registry (GCR, recently GCR is replaced by Google Artifact Registry so in this project Google Artifact Registry used instead of GCR) which was finally deployed to the Google Kubernetes Engine (GKE) Cluster. The Bank Application Pods was accessed using GKE Ingress Application LoadBalancer and hence using the kubernetes service as shown in the screenshot attached above. I applied GCP Cloud Armor Policy (to Limit the Rate) to the Bank Application URL and to prevent from SQL Injection and prevent from cross-site scripting (XSS). As this Bank Application was open to entire world so rate limit and SQL Injection was applied through GCP Cloud Armor Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot (by the Hackers) so that its speed becomes slow or this website hangs to refrain from such a situation GCP Cloud Armor Policy Rate Limit was applied. Finally, I did the Performance Test (Load test) of the Bank Application URL using Apache JMeter to check how many users can access the BankApp without any hitch.

To create the infrastructure, I used Terraform and Terraform was installed on GCP VM Instance and its state-file was kept in GCP Bucket. Terraform script to create the infrastructure is present at the path **terraform-gcp-cloud-dns-aws-acm-certificate** (this terraform script is handy for you to create the GCP Cloud DNS zone and AWS Certificate Manager SSL Certificate) in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. Commands as written below had been used to create infrastructure using Terraform.

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

At this point after executing the command **terraform apply -auto-approve** make sure you do the entry for Nameservers of the GCP Cloud DNS Zone created zone in your domain name service provider's Nameserver. After doing such an entry the terraform script will give the output as written below. For Authentication and Authorization of AWS Account, I installed **aws cli** on GCP VM instance and created an IAM user in AWS account and used its access key and secret key as shown in the screenshot attached below.

```
[root@terraform-server ~]# cat /etc/os-release
NAME="Rocky Linux"
VERSION="8.10 (Green Obsidian)"
ID="rocky"
ID_LIKE="rhel centos fedora"
VERSION_ID="8.10"
PLATFORM_ID="platform:el8"
PRETTY_NAME="Rocky Linux 8.10 (Green Obsidian)"
ANSI_COLOR="0;32"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:rocky:rocky:8:GA"
HOME_URL="https://rockylinux.org/"
BUG_REPORT_URL="https://bugs.rockylinux.org/"
SUPPORT_END="2029-05-31"
ROCKY_SUPPORT_PRODUCT="Rocky-Linux-8"
ROCKY_SUPPORT_PRODUCT_VERSION="8.10"
REDHAT_SUPPORT_PRODUCT="Rocky Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="8.10"
```

To install aws cli I installed first python3.9 as shown in the screenshot attached below.

```
[root@terraform-server ~]# yum install -y python3.9
```

```
[root@terraform-server ~]# ln -s /usr/bin/python3.9 /usr/bin/python
```

```
[root@terraform-server ~]# python --version
Python 3.9.20
```

Then installed aws-cli using the commands as shown below.

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

```
[root@terraform-server ~]# vim ~/.bashrc
[root@terraform-server ~]# logout
[ritesh@terraform-server ~]$ sudo -i
[root@terraform-server ~]# cat ~/.bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

PATH="$PATH:/usr/local/bin"
[root@terraform-server ~]# echo $PATH
/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/root/bin
[root@terraform-server ~]# aws --version
aws-cli/2.27.50 Python/3.13.4 Linux/4.18.0-553.58.1.el8_10.x86_64 exe/x86_64.rocky.8
```

```
[root@terraform-server main]# aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-2
Default output format [None]: text
```

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

```
module.gcp_cloud_dns_zone_aws_certificate.google_dns_managed_zone.dexter_public_zone: Creating...
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate.acm_cert: Creating...
module.gcp_cloud_dns_zone_aws_certificate.google_dns_managed_zone.dexter_public_zone: Creation complete after 0s [id=projects/                /managedZones
/bankapp-public-zone]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate.acm_cert: Creation complete after 6s [id=arn:aws:acm:us-east-2:02        6:certificate/
]
module.gcp_cloud_dns_zone_aws_certificate.google_dns_record_set.record_cert_validation: Creating...
module.gcp_cloud_dns_zone_aws_certificate.google_dns_record_set.record_cert_validation: Creation complete after 2s [id=projects/                /managedZon
es/bankapp-public-zone/rrsets/                .singhritesh85.com./CNAME]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Creating...
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m10s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m20s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m30s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m40s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m50s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m00s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m10s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m20s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m30s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m40s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m50s elapsed]
module.gcp_cloud_dns_zone_aws_certificate.aws_acm_certificate_validation.acm_certificate_validation: Creation complete after 1m55s [id=
UTC]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_dns_zone_and_aws_cerificate_arn_details = {
  "certificate_arn" = "arn:aws:acm:us-east-2:02        6:certificate/                "
  "cloud_dns_zone_id" = "projects/                /managedZones/bankapp-public-zone"
  "gcp_cloud_dns_zone_name_servers" = tolist([
    "                ",
    "                ",
    "                ",
    "                ",
  ])
}
```

Now the GCP Cloud DNS Zone and AWS Certificate Manager SSL Certificate became ready. Then I created the GitLab Server using the terraform script **terraform-gitlab-server** present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. I had created the snapshots of EBS Volume attached to the GitLab Server using the Lifecycle Manager Policy and created the snapshots at 11:00 AM and 11:00 PM UTC (twice a day). The below steps and screenshot show how I had executed the terraform script.

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources

```
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitaly: (pid 31883) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-exporter: (pid 31891) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-kas: (pid 31115) 280s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-workhorse: (pid 31859) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: logrotate: (pid 30721) 310s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: nginx: (pid 31867) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: node-exporter: (pid 31876) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgres-exporter: (pid 31924) 14s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgresql: (pid 30903) 287s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: prometheus: (pid 31901) 17s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: puma: (pid 31254) 148s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis: (pid 30767) 304s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis-exporter: (pid 31893) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: sidekiq: (pid 31281) 143s
module.gitlab.null_resource.gitlab_server (remote-exec): run: alertmanager: (pid 31916) 17s; run: log: (pid 31662) 97s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitaly: (pid 31883) 21s; run: log: (pid 30836) 298s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-exporter: (pid 31891) 21s; run: log: (pid 31548) 117s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-kas: (pid 31115) 283s; run: log: (pid 31133) 280s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-workhorse: (pid 31859) 22s; run: log: (pid 31418) 137s
module.gitlab.null_resource.gitlab_server (remote-exec): run: logrotate: (pid 30721) 313s; run: log: (pid 30729) 312s
module.gitlab.null_resource.gitlab_server (remote-exec): run: nginx: (pid 31867) 22s; run: log: (pid 31452) 130s
module.gitlab.null_resource.gitlab_server (remote-exec): run: node-exporter: (pid 31876) 21s; run: log: (pid 31511) 123s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgres-exporter: (pid 31924) 17s; run: log: (pid 31705) 87s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgresql: (pid 30903) 290s; run: log: (pid 30927) 287s
module.gitlab.null_resource.gitlab_server (remote-exec): run: prometheus: (pid 31901) 20s; run: log: (pid 31619) 105s
module.gitlab.null_resource.gitlab_server (remote-exec): run: puma: (pid 31254) 151s; run: log: (pid 31262) 150s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis: (pid 30767) 307s; run: log: (pid 30781) 304s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis-exporter: (pid 31893) 21s; run: log: (pid 31580) 111s
module.gitlab.null_resource.gitlab_server (remote-exec): run: sidekiq: (pid 31281) 145s; run: log: (pid 31349) 144s
module.gitlab.null_resource.gitlab_server: Creation complete after 13m16s [                    ]

Apply complete! Resources: 39 added, 0 changed, 0 destroyed.

Outputs:

ec2_private_ip_alb_dns = {
  "EC2_Instance_GitLab_Server_Private_IP_Address" = "10.      "
  "GitLab_ALB_DNS_Name" = "gitlab-          .us-east-2.elb.amazonaws.com"
}
```

Then I created the other GCP Resources using the terraform script present at the path **terraform-gke-cluster-cloudarmor-with-backup** in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. As a part of disaster recovery, I created the Lifecycle Policy to create the EBS backed AMI periodically 11:00 AM and 11:00 PM UTC (twice daily) for SonarQube and Nexus Server, snapshot schedule for gitlab-runner and GKE backup plan to take the backup daily twice a day (11:00 AM UTC and 11:00 PM UTC).

terraform init -------> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate --------> verify that terraform configuration file is correct or not

terraform plan ---------> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

The final GKE Cluster is as shown in the screenshot attached below.

```
[root@                    ~]# gcloud --version
Google Cloud SDK 526.0.0
alpha 2025.06.06
beta 2025.06.06
bq 2.1.18
bundled-python3-unix 3.12.9
core 2025.06.06
gcloud-crc32c 1.0.0
gke-gcloud-auth-plugin 0.5.10
gsutil 5.34
kubectl 1.32.4
```

```
[root@                    ~]# gcloud auth login

You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

You can run:

  $ gcloud config set account `ACCOUNT`

to switch accounts if necessary.

Your credentials may be visible to others with access to this
virtual machine. Are you sure you want to authenticate with
your personal account?

Do you want to continue (Y/n)?  Y

Go to the following link in your browser, and complete the sign-in prompts:

                                                                                

Once finished, enter the verification code provided in your browser:

You are now logged in as [      @gmail.com].
Your current project is [                ].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```

Status	Name	Location	Tier	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
✓	bankapp-gke-cluster	us-central1	Standard	2	4	8 GB		goog-terra... : true

```
[root@                    ~]# kubectl get nodes
NAME                                                  STATUS   ROLES    AGE     VERSION
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   5m6s    v1.32.6-gke.1125000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   4m53s   v1.32.6-gke.1125000
```

For its CI/CD deployment you can refer the project present in GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-CICD-using-GitLab-GCP-and-AWS.git.

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n mysql --watch
NAME                READY   STATUS    RESTARTS   AGE
mysql-primary-0     1/1     Running   0          58s
mysql-secondary-0   1/1     Running   0          58s
```

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n bankapp
NAME                                READY   STATUS    RESTARTS   AGE
bankapp-folo-                       1/1     Running   0          20m
```

To create Ingress Rule with Cloud Armor Policy, use the procedure as written below.

cat backendconfig-bankapp.yaml

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
     namespace: bankapp
     name: bankapp-backendconfig
spec:
    securityPolicy:
    name: bankapp-gke-cloud-armor-policy
```

kubectl get backendconfig -n bankapp

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl edit svc -n banapp-folo -n bankapp
```

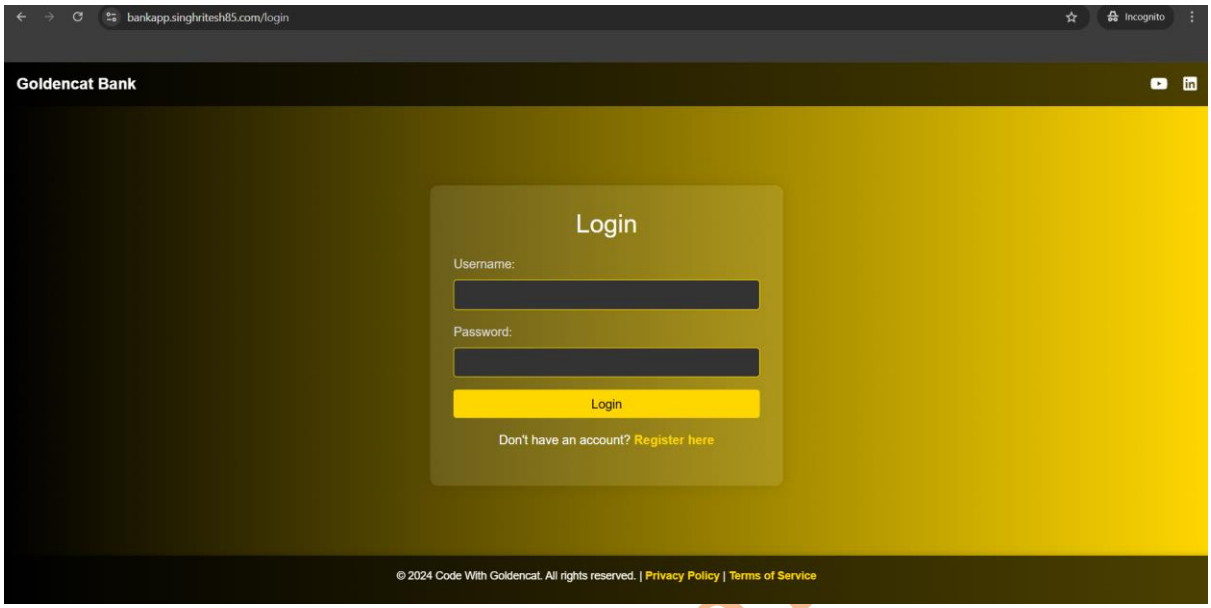
```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    cloud.google.com/neg: '{"ingress":true}'
    cloud.google.com/neg-status: '{"network_endpoint_groups":{"80":"[redacted]-bankapp-bankapp-folo-[redacted]"},"zones":["us-central1-a","us-central1-c"]
}'
    meta.helm.sh/release-name: bankapp
    meta.helm.sh/release-namespace: bankapp
    cloud.google.com/backend-config: '{"default": "bankapp-gke-cloud-armor-policy"}'
  creationTimestamp: "2025-[redacted]"
  labels:
    app.kubernetes.io/instance: bankapp
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: folo
    app.kubernetes.io/version: 1.16.0
    helm.sh/chart: folo-0.1.0
  name: bankapp-folo
  namespace: bankapp
  resourceVersion: "[redacted]"
  uid: [redacted]
spec:
  clusterIP: [redacted]
  clusterIPs:
  - [redacted]
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
```

cat ingress-rule.yaml

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dexter-ingress
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: "gce" # Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
#    cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'
spec:
  ingressClassName: "gce"
  tls:
  - hosts:
    - bankapp.singhritesh85.com
    secretName: ingress-tls
  rules:
  - host: "bankapp.singhritesh85.com"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-folo
            port:
              number: 80
```

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get ing -A --watch
NAMESPACE   NAME             CLASS   HOSTS                      ADDRESS          PORTS     AGE
bankapp     dexter-ingress   gce     bankapp.singhritesh85.com                   80, 443   4m52s
```

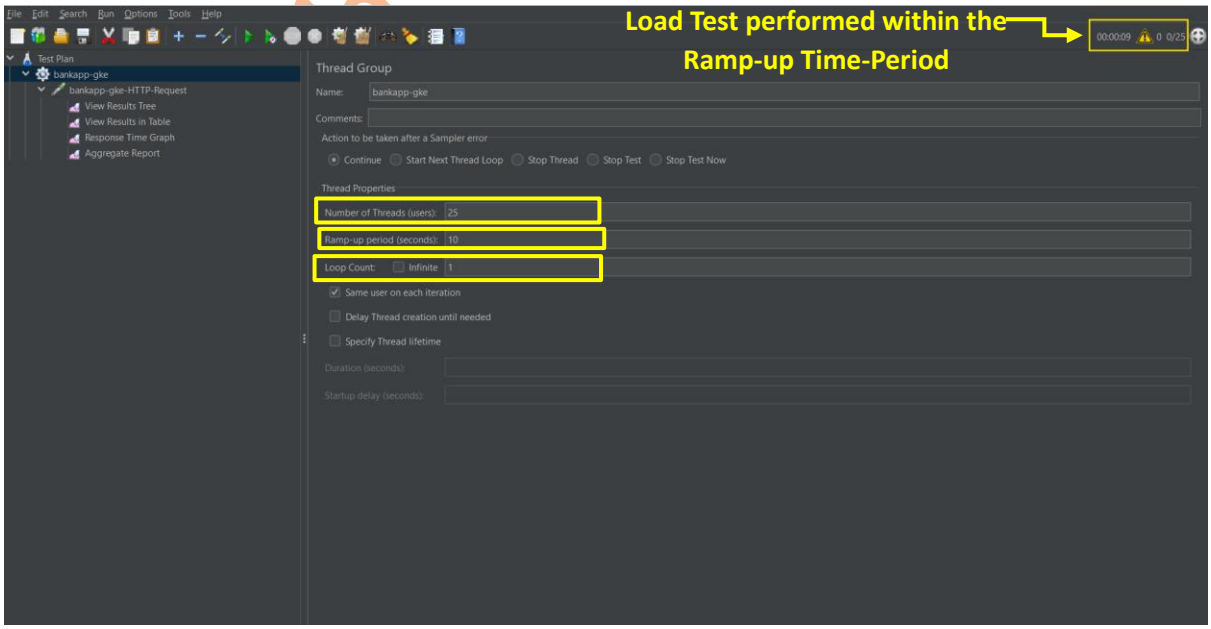
Do the entry for obtained External IP Address with HOSTS in GCP DNS Zone to create the Record Set of A-Type as I explained in the project https://github.com/singhritesh85/DevOps-Project-BankApp-CICD-using-GitLab-GCP-and-AWS.git. Finally, you will be able to access the BankApp as shown in the screenshot attached below.

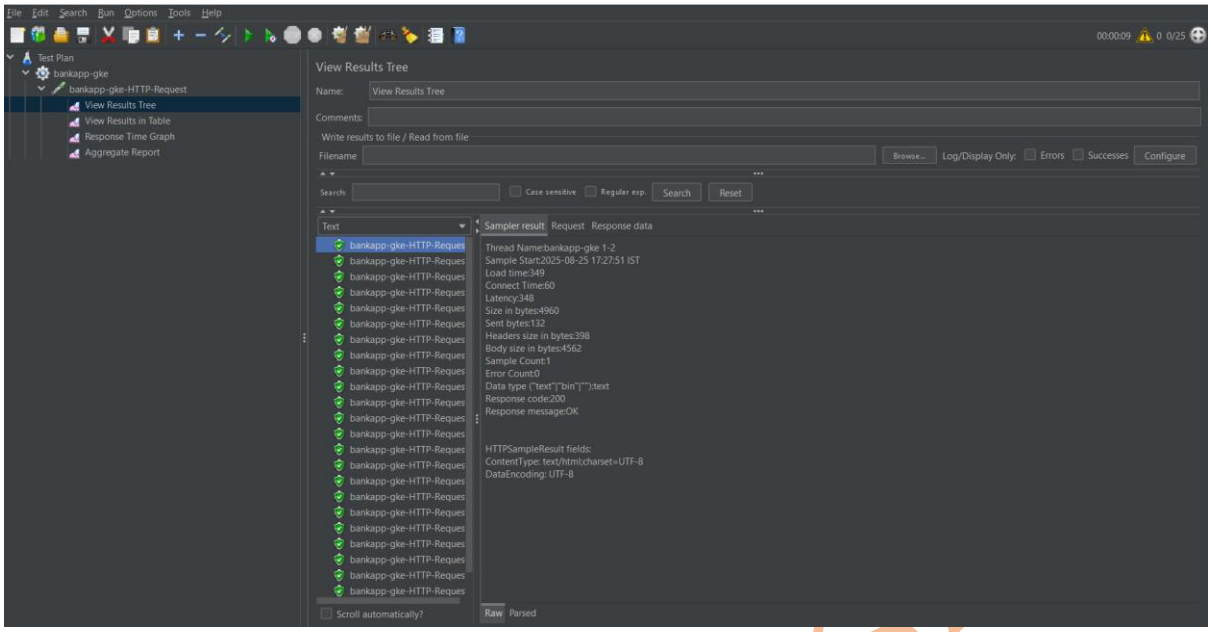


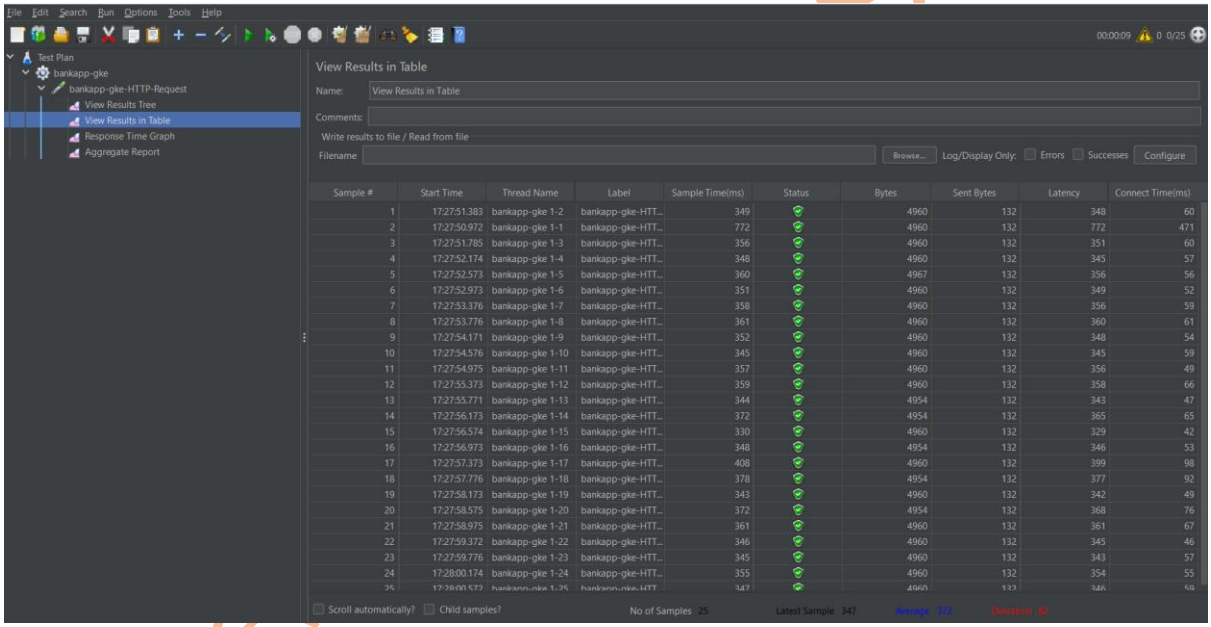
You can test after accessing the BankApp URL more than 40 times in a minute, it will be blocked with Too Many Requests error. The SQL Injection protection had been applied and as explained in Azure and AWS section Cross-Site scripting protection had been applied. It was already applied using the terraform script.

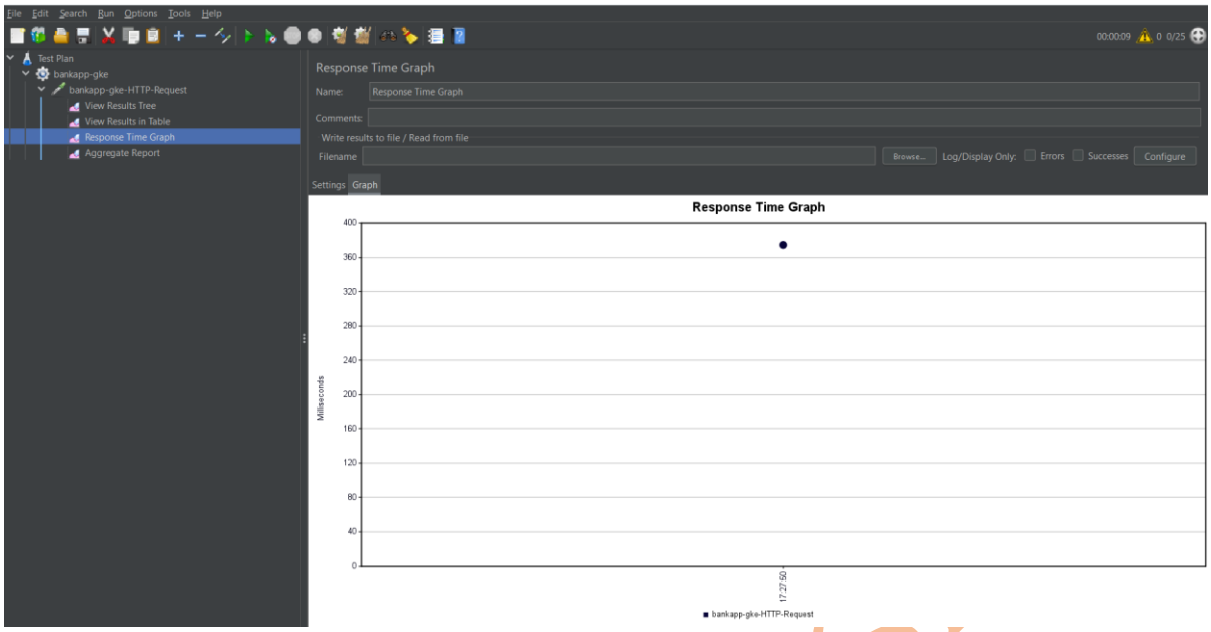
Then I did the Load Test using Apache JMETER as shown in the screenshot attached below.

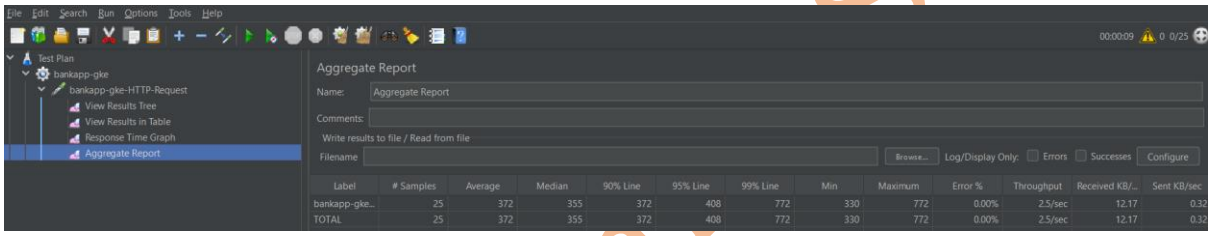
If you need, you can perform this Test in Infinite Loop but for this project I am running it for a single iteration.




File Edit Search Run Options Tools Help
Test Plan
bankapp-gke
bankapp-gke-HTTP-Request
View Results Tree
View Results in Table
Response Time Graph
Aggregate Report
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Search: Case sensitive Regular exp. Search Reset
Text
Sampler result Request Response data
Thread Name:bankapp-gke 1-2
Sample Start:2025-08-25 17:27:51 IST
Load time:349
Connect Time:60
Latency:348
Size in bytes:4960
Sent bytes:132
Headers size in bytes:398
Body size in bytes:4562
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:OK
HTTPSampleResult fields:
ContentType: text/html;charset=UTF-8
DataEncoding: UTF-8
Scroll automatically?
Raw Parsed


File Edit Search Run Options Tools Help
Test Plan
bankapp-gke
bankapp-gke-HTTP-Request
View Results Tree
View Results in Table
Response Time Graph
Aggregate Report
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Sample # Start Time Thread Name Label Sample Time(ms) Status Bytes Sent Bytes Latency Connect Time(ms)
1 17:27:51.383 bankapp-gke 1-2 bankapp-gke-HTT... 349 4960 132 348 60
2 17:27:50.972 bankapp-gke 1-1 bankapp-gke-HTT... 772 4960 132 772 471
3 17:27:51.785 bankapp-gke 1-3 bankapp-gke-HTT... 356 4960 132 351 60
4 17:27:52.174 bankapp-gke 1-4 bankapp-gke-HTT... 348 4960 132 345 57
5 17:27:52.573 bankapp-gke 1-5 bankapp-gke-HTT... 360 4967 132 356 56
6 17:27:52.973 bankapp-gke 1-6 bankapp-gke-HTT... 351 4960 132 349 52
7 17:27:53.376 bankapp-gke 1-7 bankapp-gke-HTT... 358 4960 132 356 59
8 17:27:53.776 bankapp-gke 1-8 bankapp-gke-HTT... 361 4960 132 360 61
9 17:27:54.171 bankapp-gke 1-9 bankapp-gke-HTT... 352 4960 132 348 54
10 17:27:54.576 bankapp-gke 1-10 bankapp-gke-HTT... 345 4960 132 345 59
11 17:27:54.975 bankapp-gke 1-11 bankapp-gke-HTT... 357 4960 132 356 49
12 17:27:55.373 bankapp-gke 1-12 bankapp-gke-HTT... 359 4960 132 358 66
13 17:27:55.771 bankapp-gke 1-13 bankapp-gke-HTT... 344 4954 132 343 47
14 17:27:56.173 bankapp-gke 1-14 bankapp-gke-HTT... 372 4954 132 365 65
15 17:27:56.574 bankapp-gke 1-15 bankapp-gke-HTT... 330 4960 132 329 42
16 17:27:56.973 bankapp-gke 1-16 bankapp-gke-HTT... 348 4954 132 346 53
17 17:27:57.373 bankapp-gke 1-17 bankapp-gke-HTT... 408 4960 132 399 98
18 17:27:57.776 bankapp-gke 1-18 bankapp-gke-HTT... 378 4954 132 377 92
19 17:27:58.173 bankapp-gke 1-19 bankapp-gke-HTT... 343 4960 132 342 49
20 17:27:58.575 bankapp-gke 1-20 bankapp-gke-HTT... 372 4954 132 368 76
21 17:27:58.975 bankapp-gke 1-21 bankapp-gke-HTT... 361 4960 132 361 67
22 17:27:59.372 bankapp-gke 1-22 bankapp-gke-HTT... 346 4960 132 345 46
23 17:27:59.776 bankapp-gke 1-23 bankapp-gke-HTT... 345 4960 132 343 57
24 17:28:00.174 bankapp-gke 1-24 bankapp-gke-HTT... 355 4960 132 354 55
Scroll automatically? Child samples? No of Samples 25 Latest Sample 347





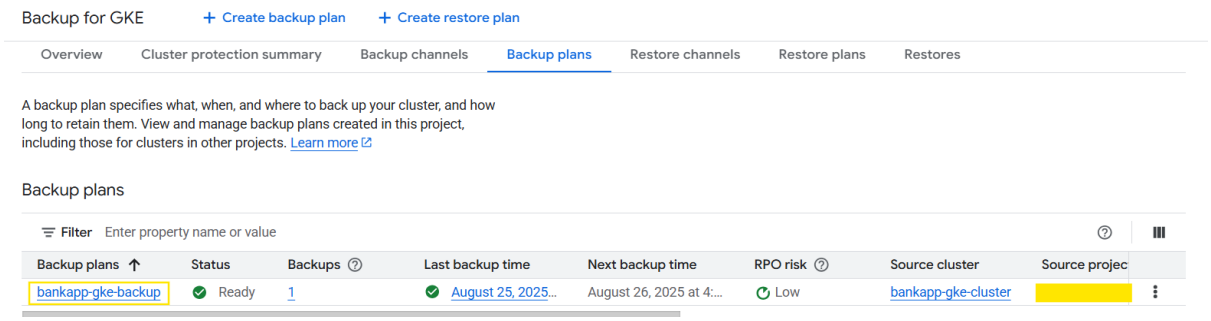
The above Load Test I performed for 25 simultaneous users in Ramp-up time of 10 seconds (within a time frame of 10 seconds). I can conclude that the overall performance of BankApp URL for the 25 users who accessed the URL https://bankapp.singhritesh85.com/login within a ramp-up time of 10 seconds was satisfactory.

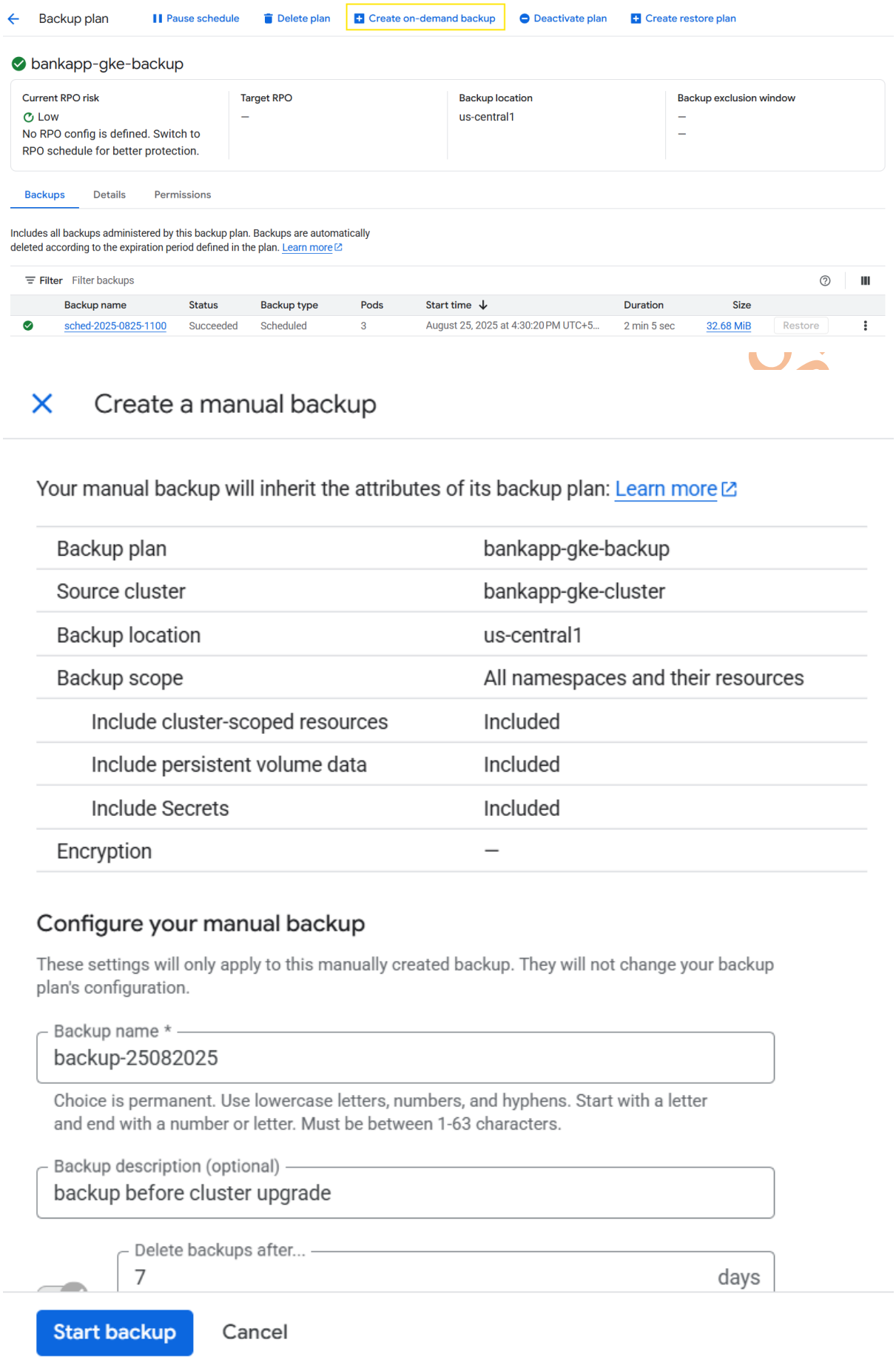
Finally, I will upgrade the GKE version from 1.32 to 1.33. The below screenshot shows the number of nodes and kubernetes version present before the upgrade.

```
[root@                    ~]# kubectl get nodes
NAME                                                  STATUS   ROLES    AGE   VERSION
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   72m   v1.32.6-gke.1125000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   11m   v1.32.6-gke.1125000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   72m   v1.32.6-gke.1125000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   11m   v1.32.6-gke.1125000
```

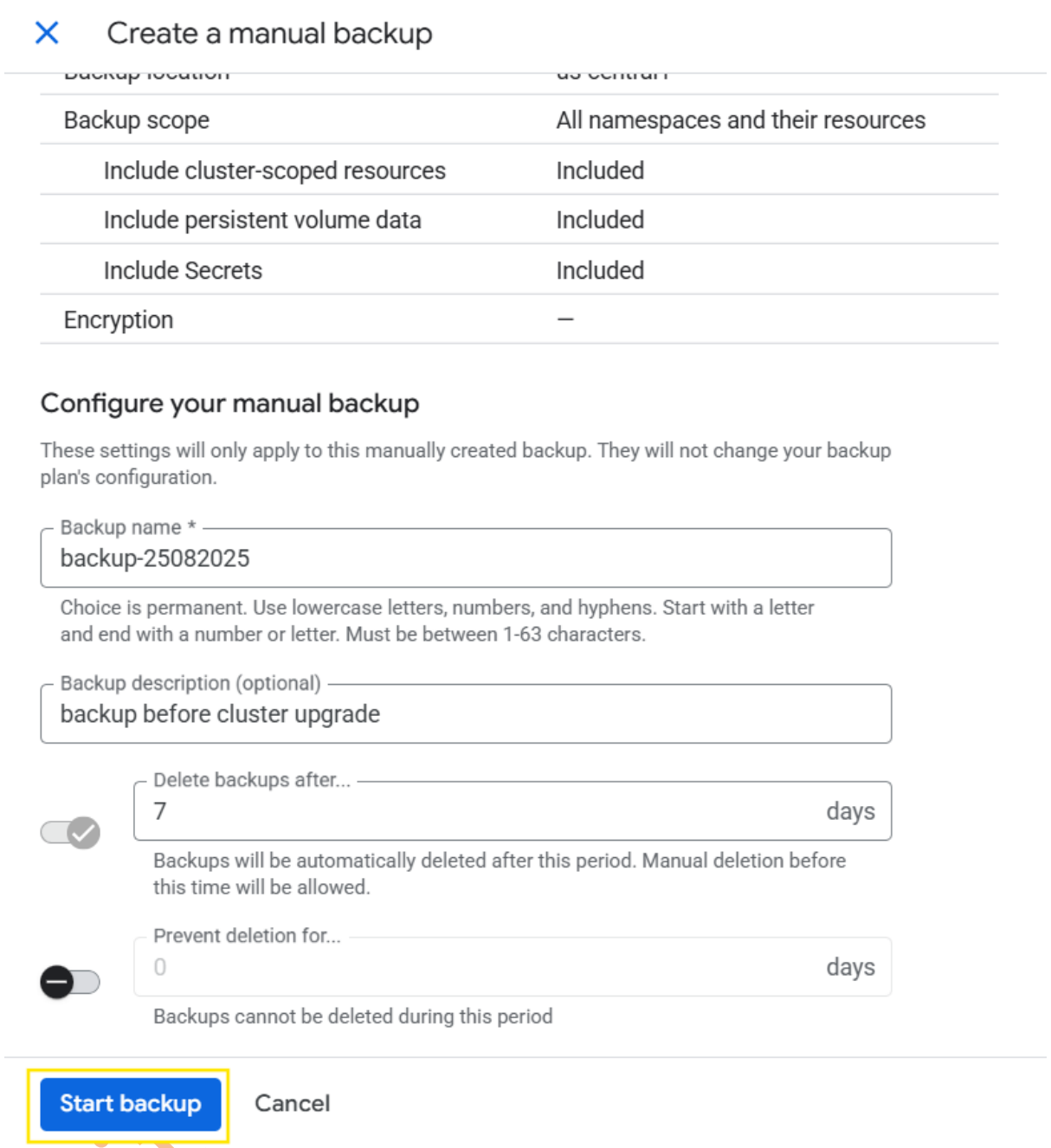
The GKE Cluster was created with two nodes and later two more nodes were created using the GKE Cluster Autoscaler.

To GKE Cluster version upgrade first step is the take the full manual backup of entire cluster as shown in the screenshot attached below.

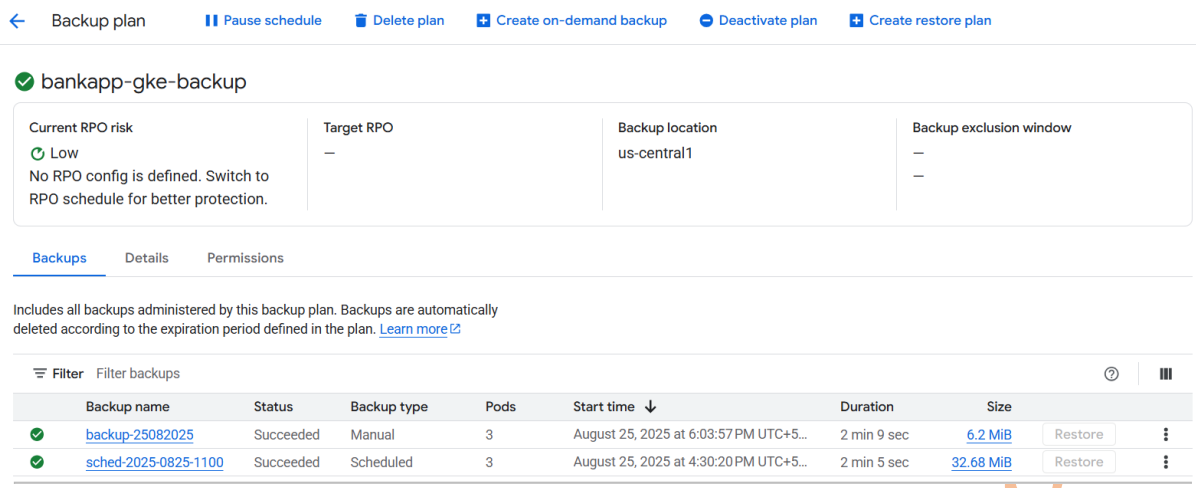




Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/ || GitHub: - https://github.com/singhritesh85



The backup had been completed as shown in the screenshot attached below.



To upgrade GKE Cluster I changed the parameters in terraform as shown in the screenshot attached below.

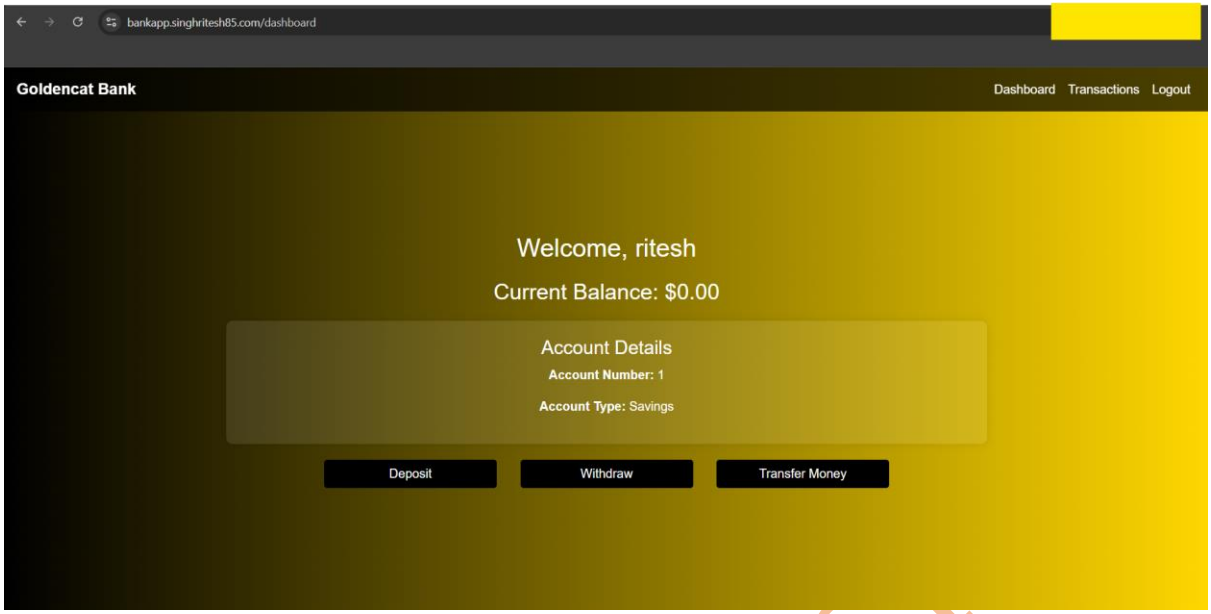
```
min_master_version = ["1.33.2", "1.32.6", "1.30.12"] ###["v1.33.1-gke.1386000", "v1.32.4-gke.1415000", "v1.30.12-gke.1246000"]
node_version = ["1.33.2", "1.32.6", "1.30.12"]       ###["v1.33.1-gke.1386000", "v1.32.4-gke.1415000", "v1.30.12-gke.1246000"]
```

```
module "gke" {

  source = "../module"
  project_name = var.project_name
  gcp_region = var.gcp_region[1]
  prefix = var.prefix
  ip_range_subnet = var.ip_range_subnet
  master_ip_range = var.master_ip_range
  min_master_version = var.min_master_version[0]
  node_version = var.node_version[0]
  pods_ip_range = var.pods_ip_range
  services_ip_range = var.services_ip_range
  ip_public_range_subnet = var.ip_public_range_subnet
  machine_type = var.machine_type
```

It is also possible to import the existing cluster and then upgrade using terraform but as I am aware it will not affect the existing resources so I upgraded the GKE version using the existing terraform script present at the path **terraform-gke-cluster-cloudarmor-with-backup** (same terraform script using which I created the GKE Cluster) in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git. Run the terraform commands **terraform plan** followed by **terraform apply -auto-approve**. Finally, the GKE Cluster had been upgraded from 1.32 to 1.33 as shown in the screenshot attached below.

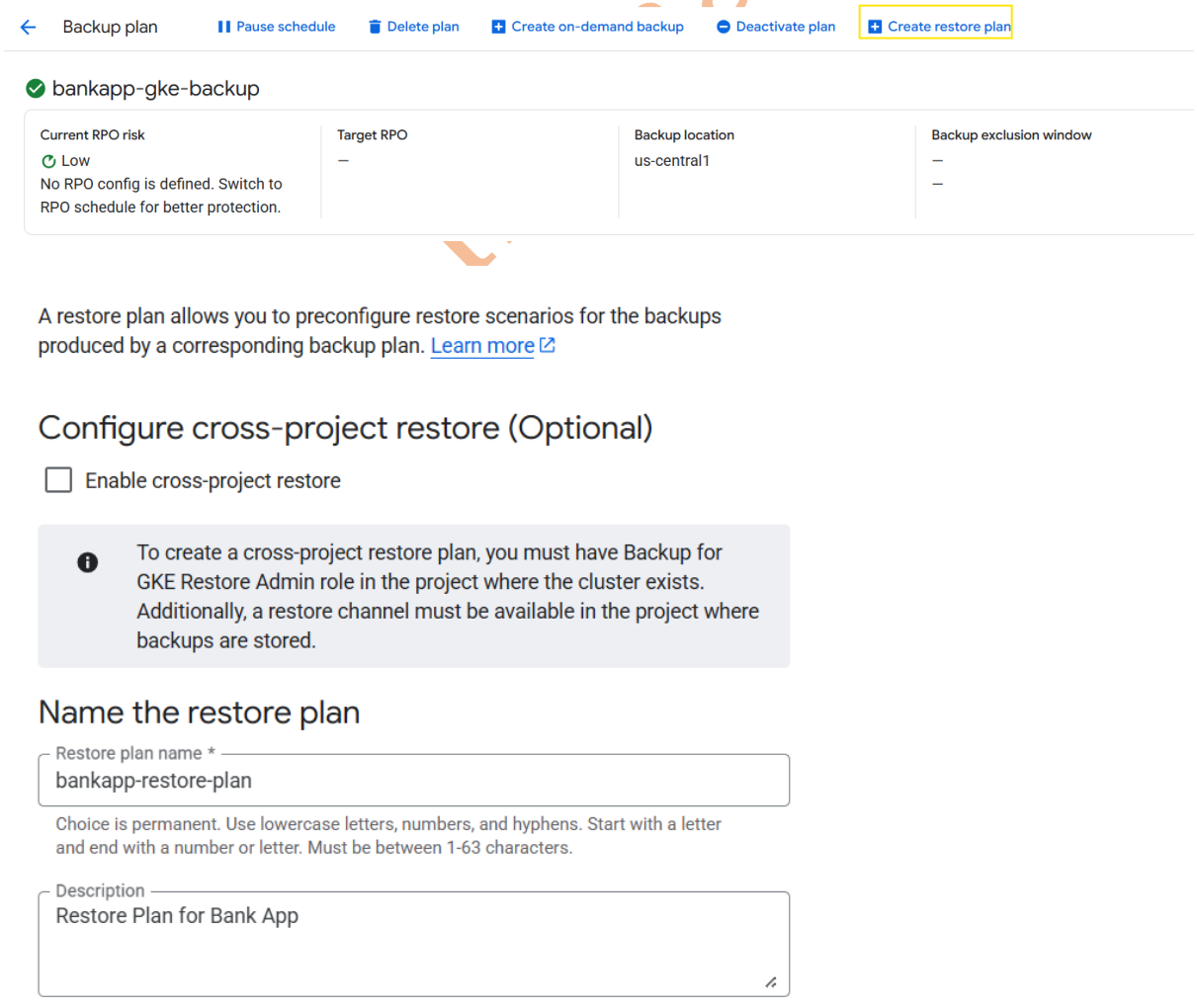
```
[root@                    ~]# kubectl get nodes
NAME                                                  STATUS   ROLES    AGE   VERSION
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   54m   v1.33.2-gke.1240000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   50m   v1.33.2-gke.1240000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   74m   v1.33.2-gke.1240000
gke-bankapp-gke-clus-bankapp-linux-no-                Ready    <none>   82m   v1.33.2-gke.1240000
```

After the upgrade I found MySQL pods and bankapp pods were in Running state and I was able to access the BankApp as shown in the screenshot attached below.



I consider a case when MySQL and bankapp pods did not come up in Running state then how to restore it from the taken backup. For this demonstration I deleted the namespace mysql and bankapp and the restored it using the created restore plan.

Create restore plan as shown in the screenshot attached below.



Choose a corresponding backup plan

The restore plan can only be used to restore backups produced by the selected backup plan.

Backup plan *
bankapp-gke-backup

Choice is permanent

Choose a target cluster

Backups can only be restored into this cluster.

Cluster *
bankapp-gke-cluster (us-central1)

Choice is permanent

Backups will be restored into the source cluster. You can choose how to handle any conflicts in the next steps.

Next

Choose namespaced resources

Choose which namespaced resources in the backup you want to restore.

- All namespaced resources
  Restore all namespaced resources found in the backup.
- Selected namespaced resources
  Restore resources from the selected namespaces in the backup.

Namespace 1 *
mysql

+ Add namespace

- Selected protected applications
  Restore resources from the selected protected applications in the backup.
- No namespaced resources
  Do not restore any namespaced resources found in the backup.

Choose namespaced resources

Choose which namespaced resources in the backup you want to restore.

- ○ All namespaced resources
  Restore all namespaced resources found in the backup.
- ◉ Selected namespaced resources
  Restore resources from the selected namespaces in the backup.

Namespace 1 *
mysql

Namespace 2 *
bankapp

+ Add namespace

- ○ Selected protected applications
  Restore resources from the selected protected applications in the backup.
- ○ No namespaced resources
  Do not restore any namespaced resources found in the backup.

✓ Individual resource | Namespace and ProtectedApplication

Choose what to do if a resource already exists in the target cluster.

- ◉ Merge skip (non-destructive)
  If a specific resource already exists, skip restoring the resource from the backup.
- ○ Merge replace volume(destructive)
  If a specific resource already exists, skip restoring that resource but replace the underlying volume using the volume data restore policy.
- ○ Merge replace(destructive)
  If a specific resource already exists, replace that resource with the one from backup and the associated volume data by following the volume data restore policy

Define volume data restore policy

Choose how the system should restore volume data to satisfy any persistent volume claims (PVCs) contained within the selected namespaces. Learn more

- (●) **Provision new volumes and restore volume data from backup**
  Create a new persistent disk and restore volume data from the volume backup.
- ( ) **Don't restore volume data**
  If the target cluster contains an unbound persistent volume that can satisfy the PVC requirements, they will be bound together. Otherwise, a new PV will be dynamically provisioned.
- ( ) **Reuse existing volumes containing your data**
  Any PVCs will be bound to persistent volumes that reference existing persistent disks containing the data you need.

Define policy override for specific volume types

Next

Choose cluster-scoped resources

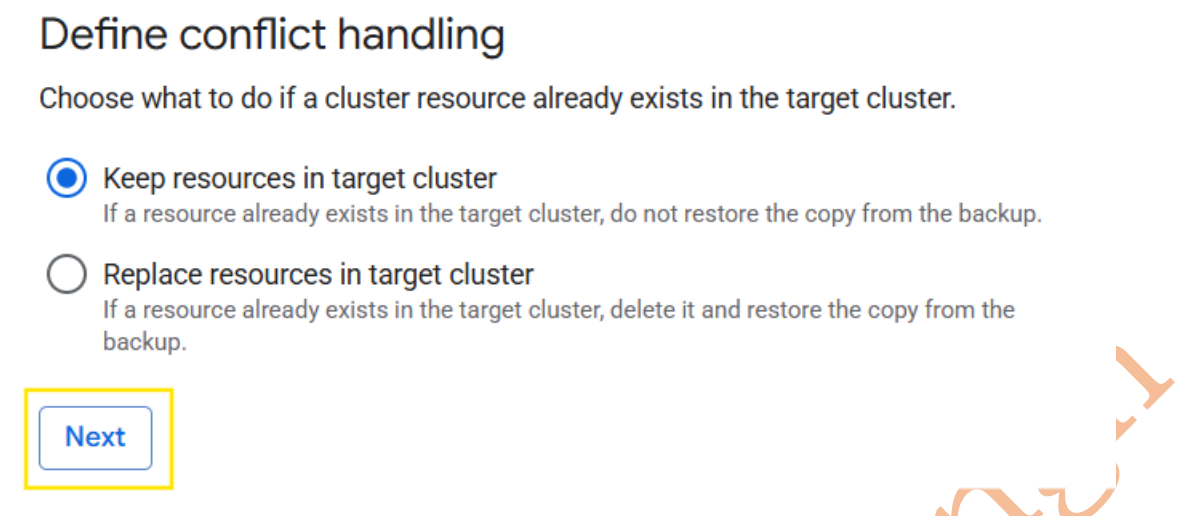
Choose the cluster-scoped resources to restore by specifying API groups and object kinds (GroupKinds), then decide how conflicts should be resolved.

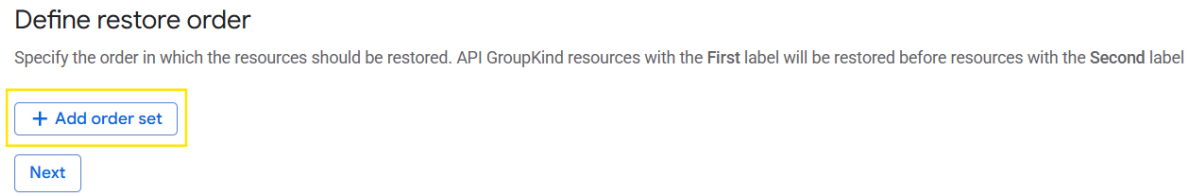
- ( ) **All cluster-scoped resources**
  Restore all cluster-scoped resources found in the backup.
- (●) **Selected cluster-scoped resources**
  Restore selected cluster-scoped resources found in the backup.

API group 1 *	Object kind 1 *
apiextensions.k8s.io	CustomResourceDefinition
API group 2 *	Object kind 2 *
storage.k8s.io	StorageClass

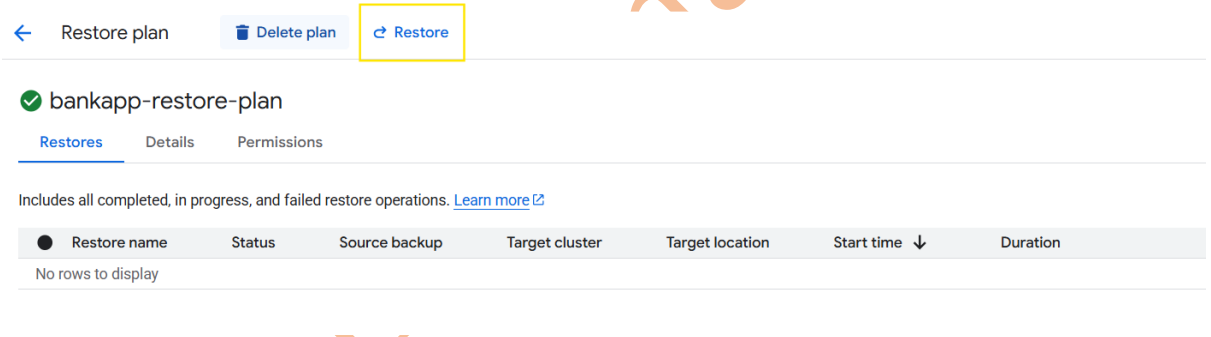
+ Add GroupKind

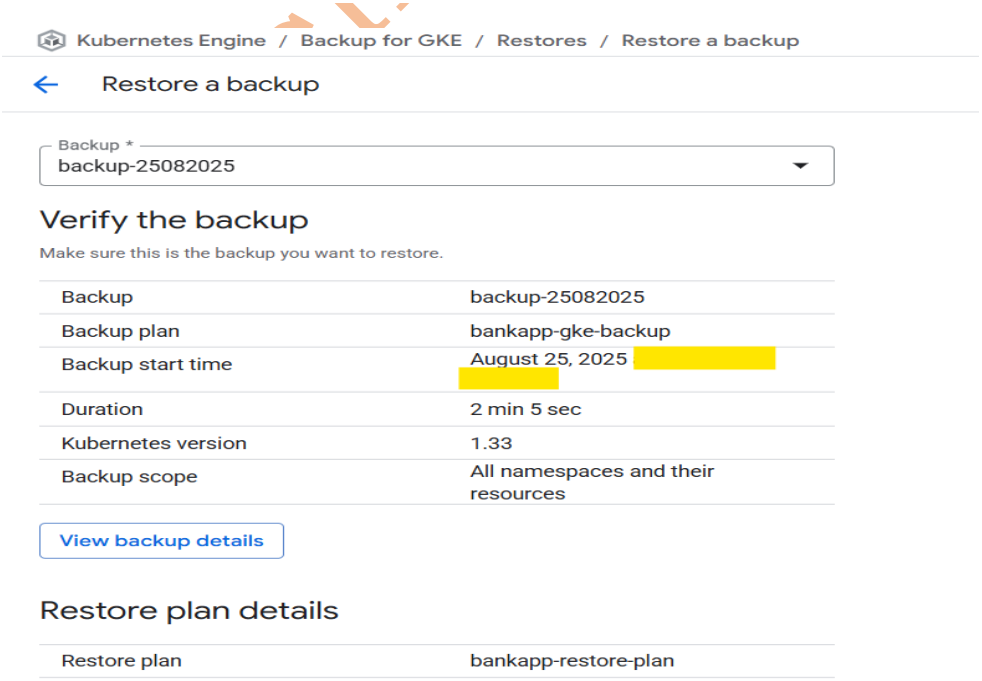
- ( ) **No cluster-scoped resources**
  Do not restore any cluster-scoped resources found in the backup.

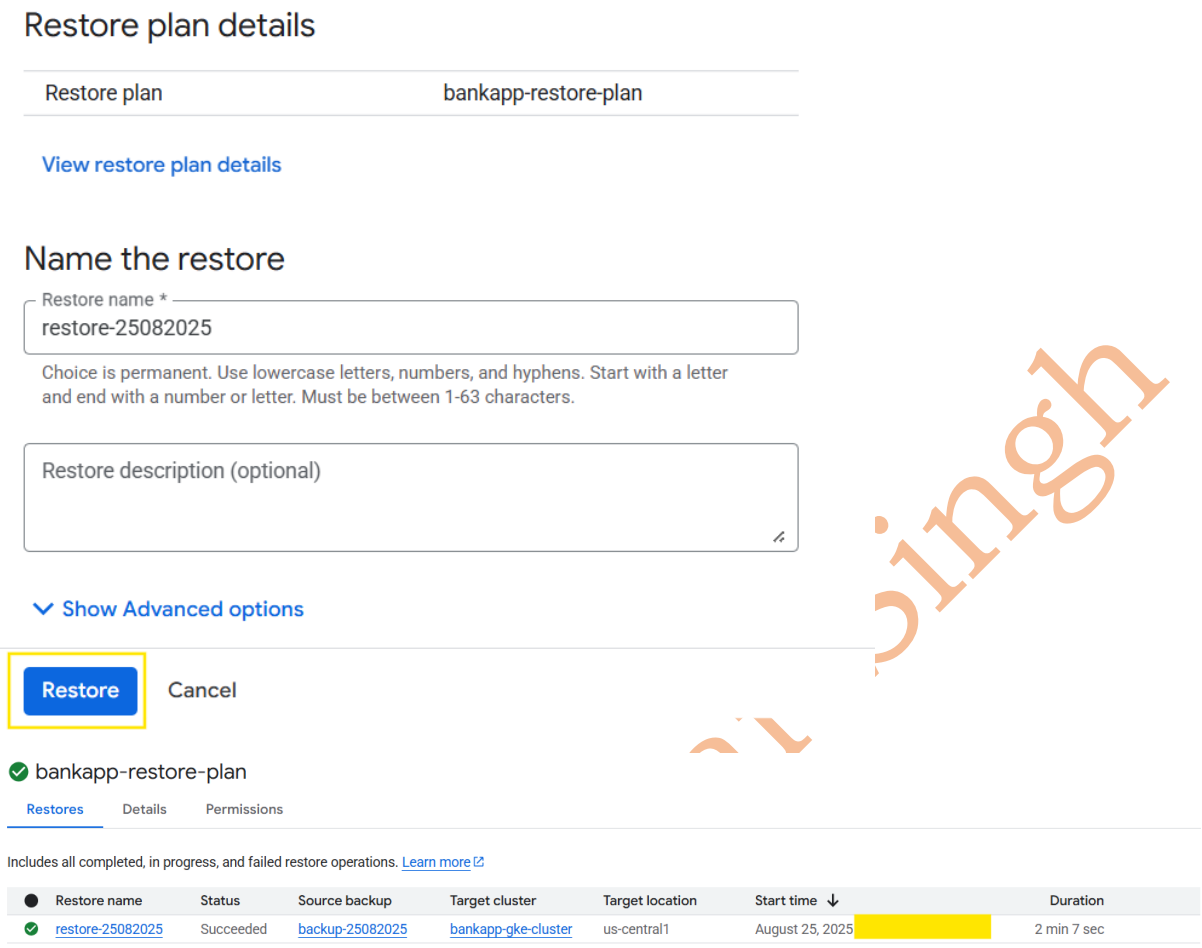




Restore Plan is created as shown in the screenshot attached below.

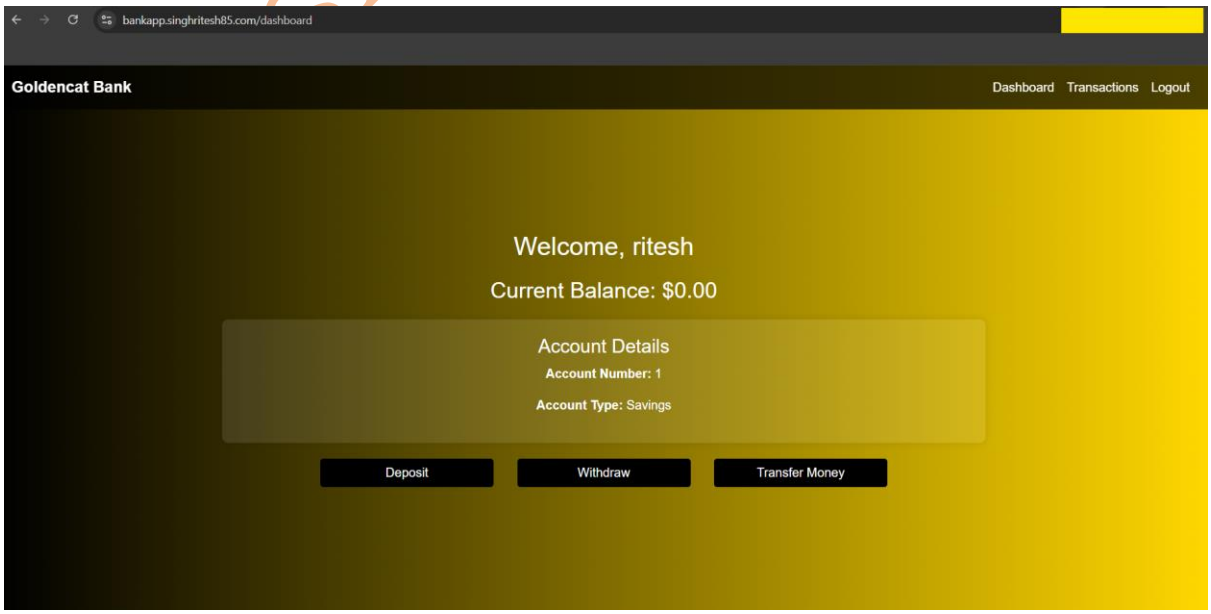






After Restore GKE Ingress Application LoadBalancer will be created again and the External IP Address will be added in the Record Set as I did earlier. Finally, I was able to access the BankApp as shown in the screenshot attached below.

After the restore I was able to login into BankApp using the same user.



Source Code: https://github.com/singhritesh85/Bank-App-GKE.git

GitHub Repo: https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git

Helm Chart: https://github.com/singhritesh85/helm-repo-for-GKE.git

https://github.com/singhritesh85/helm-repo-for-bitnami-GKE.git

Terraform Script: https://github.com/singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest.git

References: https://github.com/Goldencat98/Bank-App.git