DevOps Project BankApp Deployment using ArgoCD and Secrets Stored in Hashicorp Vault Multibranch Pipeline with Webhook Multicloud (AWS and Azure)

By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

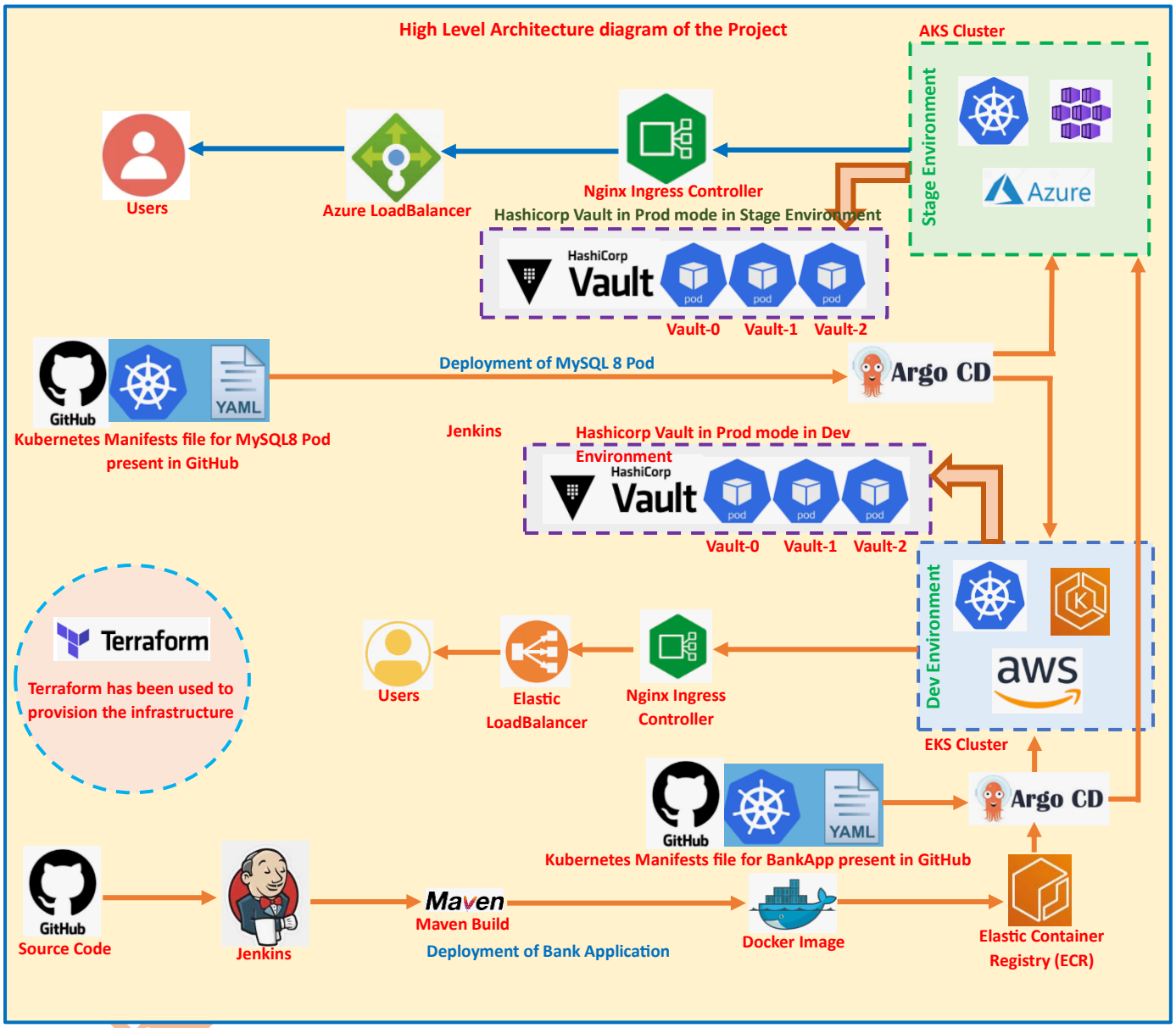
LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/

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

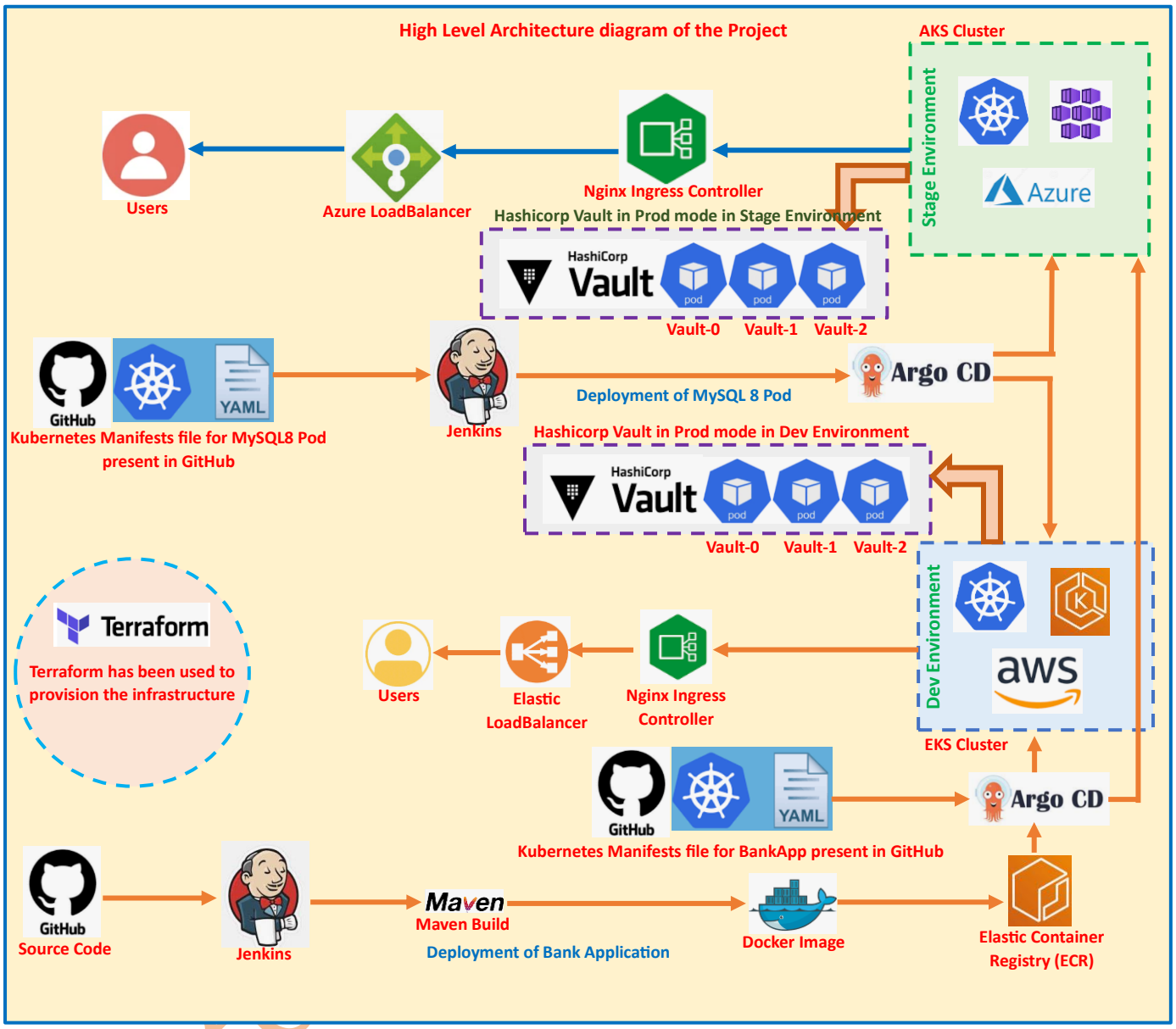
या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

DevOps Project BankApp Deployment using ArgoCD and Secrets Stored in Hashicorp Vault Multibranch Pipeline with Webhook Multicloud (AWS and Azure)

Module-1: Manual Approach

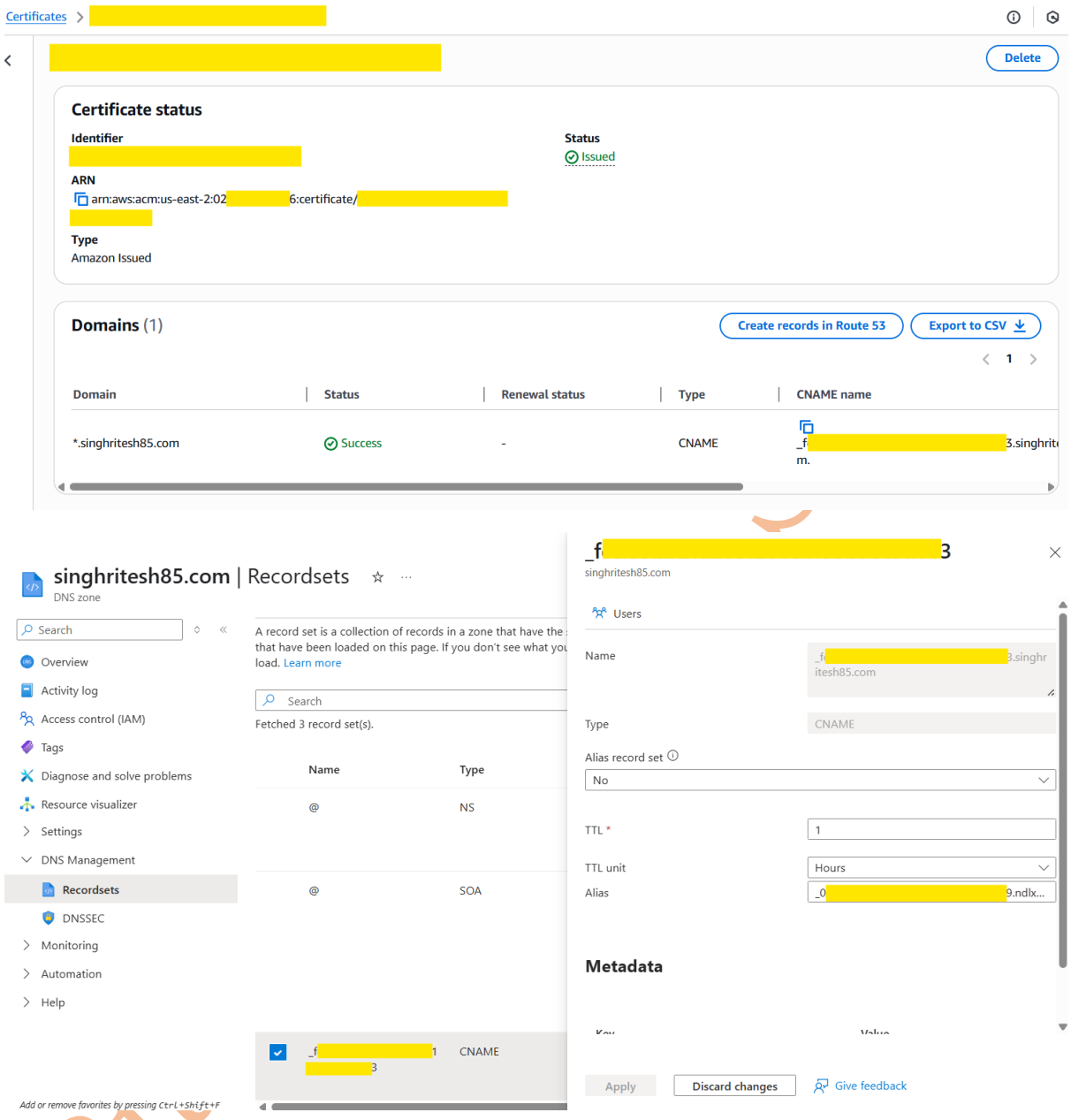


Module-2: Automation Approach



This DevOps project aims to create the Resources in AWS and Azure cloud using the Terraform Script present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-with-Secrets-stored-in-Hashicorp-Vault-Multicloud.git at the path **terraform-bankapp-multibranch-multicloud-nonprod** and setup of CICD pipeline as shown in the high-level architecture diagram drawn above. In this project I did not use the SonarQube, Nexus and Trivy Image scan if you are interested in that then you can go through the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git. In this project I used **Hashicorp Vault** in Prod mode (High Availability) to store the MySQL8 Pod credentials. I had used two ECR (Elastic Container Registries) **bankapp-1** which keeps the Docker Image for dev environment (EKS Cluster) and **bankapp-2** which keeps the Docker Image for stage environment (AKS Cluster).

To validate the SSL Certificate generated from AWS certificate manager I used DNS validation and did the entry for Azure DNS Zone record set of type CNAME as shown in the screenshot attached below.



Then I wait for around 20-25 seconds and after that SSL Certificate was issued as shown in the first screenshot attached above.

I installed terraform on Alma Linux2 Azure VM and State file was kept in the S3 Bucket and state lock had been achieved using the AWS DynamoDB. Before running the terraform script I ran the shell script present in the directory **terraform-bankapp-multibranch-multicloud-nonprod** as shown in the screenshot attached below. This shell script will install the aws cli, kubectl and helm.

```
[root@                terraform-bankapp-multibranch-multicloud-nonprod]# ./initial-setup.sh
```

Then I logged-out and login again ran the command aws configure as shown in the screenshot attached below. As it is an important step to Authenticate and Authorize the user before creating resources using terraform in the AWS Account.

```
[root@Terraform-Server terraform-bankapp-multibranch-multicloud-nonprod]# cd main/
[root@                  main]# aws configure
AWS Access Key ID [****************    ]:
AWS Secret Access Key [****************    ]:
Default region name [          ]:
Default output format [    ]:
[root@                  main]#
```

Then I installed Azure CLI and authenticated and authorize the user as shown in the screenshot attached below.

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

```
[root@                      main]# yum install azure-cli -y
```

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

Then you can run the below commands

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.eks_cluster.aws_eks_addon.vpc_cni: Still creating... [06m20s elapsed]
module.eks_cluster.aws_eks_addon.vpc_cni: Still creating... [06m30s elapsed]
module.eks_cluster.aws_eks_addon.vpc_cni: Creation complete after 6m39s [id=eks-demo-cluster-dev:vpc-cni]

Apply complete! Resources: 83 added, 0 changed, 0 destroyed.

Outputs:

acr_ec2_private_ip_alb_dns = {
  "EC2_Instance_Jenkins_Master_Server_Private_IP_Address" = "10.       "
  "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10.       "
  "Jenkins_ALB_DNS_Name" = "jenkins-ms-          .us-east-2.elb.amazonaws.com"
  "registry_id" = [
    "02         6",
    "02         6",
  ]
  "repository_url" = [
    "02         6.dkr.ecr.us-east-2.amazonaws.com/bankapp-1",
    "02         6.dkr.ecr.us-east-2.amazonaws.com/bankapp-2",
  ]
}
```

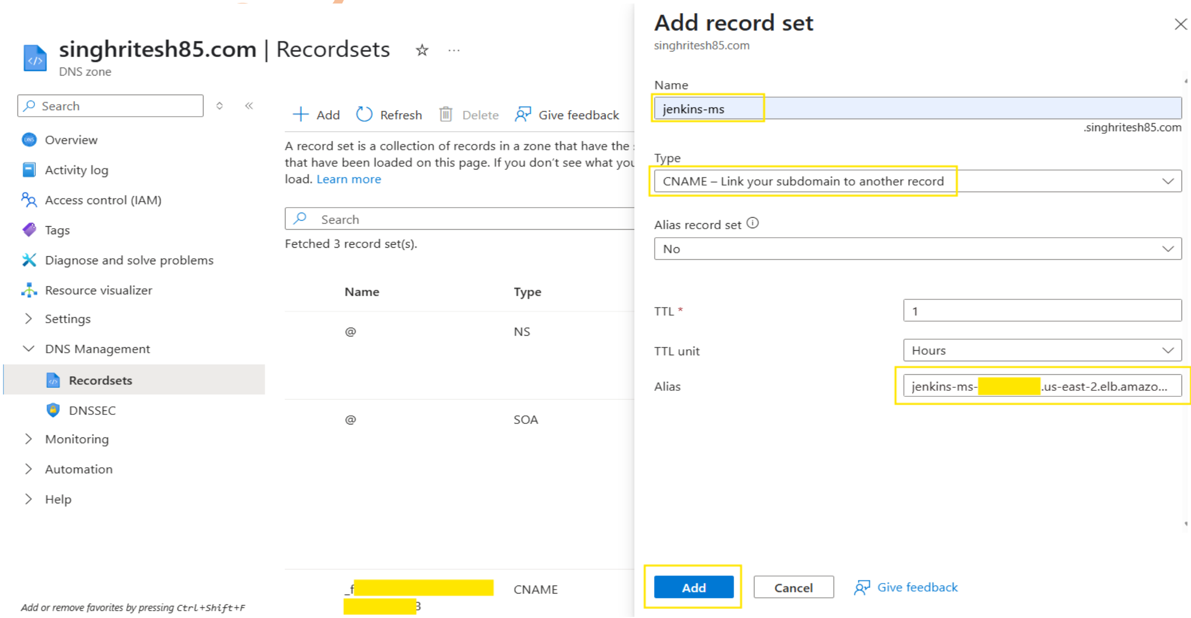
After creation of the resources a kubeconfig file was generated, I did below change in the kubeconfig file as shown in the screenshot attached below.

```
- cluster:
    certificate-authority-data: L[redacted]
    server: https://F[redacted]6.sk1.us-east-2.eks.amazonaws.com
  name: eks-demo-cluster-dev
contexts:
- context:
    cluster: aks-cluster
    user: clusterUser_aks-rg_aks-cluster
  name: aks-cluster
- context:
    cluster: eks-demo-cluster-dev
    user: arn:aws:eks:us-east-2:02[redacted]6:cluster/eks-demo-cluster-dev
  name: eks-demo-cluster-dev
current-context: aks-cluster
kind: Config
preferences: {}
users:
- name: arn:aws:eks:us-east-2:02[redacted]6:cluster/eks-demo-cluster-dev
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - --region
      - us-east-2
      - eks
      - get-token
      - --cluster-name
      - eks-demo-cluster-dev
      - --output
      - json
```

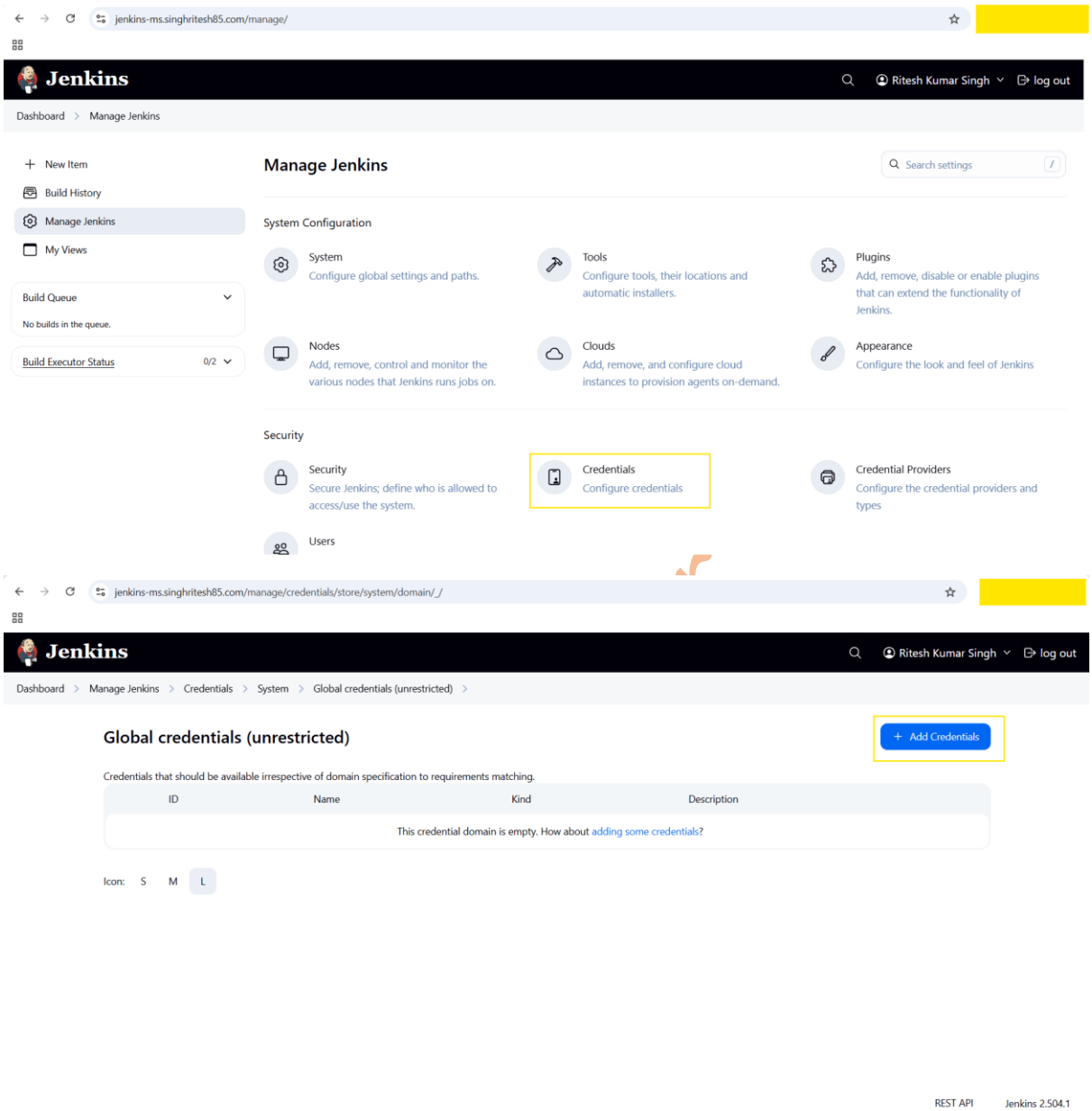
It is my suggestion before any change in the kubeconfig file take a backup of the original kubeconfig file. After creation of EKS and AKS Cluster the node in the Kubernetes Cluster can be seen as shown in the screenshot attached below.

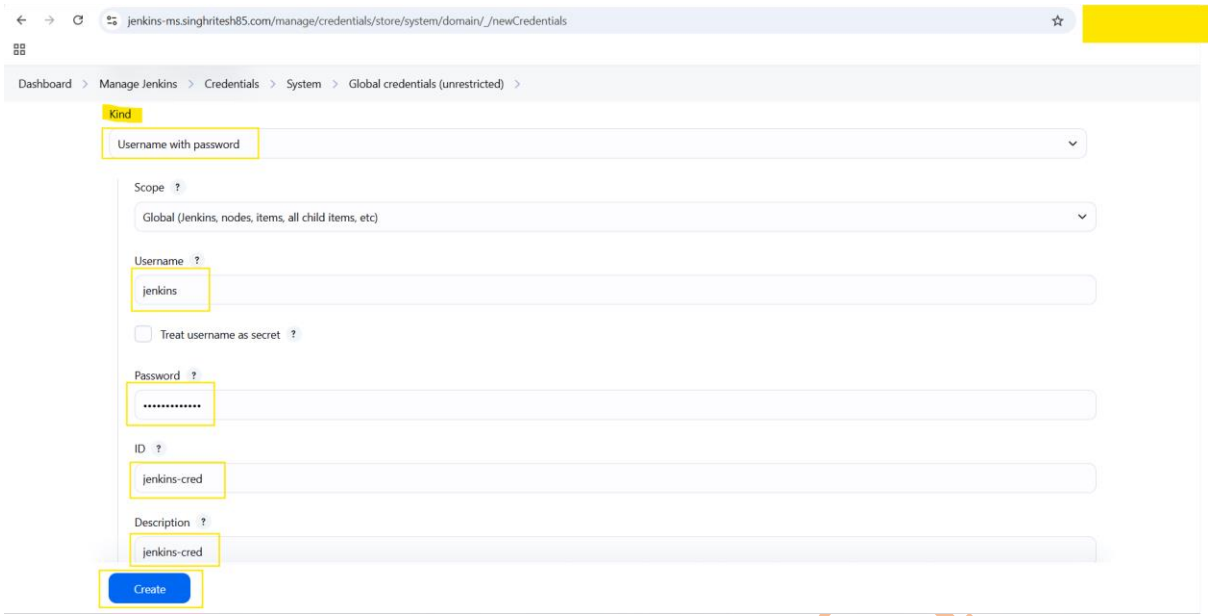
```
[root@[redacted] main]# kubectl get nodes --context=aks-cluster
NAME                                STATUS   ROLES    AGE    VERSION
aks-agentpool-[redacted]-vmss000000   Ready    <none>   121m   v1.30.0
aks-userpool-[redacted]-vmss000000    Ready    <none>   116m   v1.30.0
aks-userpool-[redacted]-vmss000001    Ready    <none>   47s    v1.30.0
aks-userpool-[redacted]-vmss000002    Ready    <none>   30s    v1.30.0
[root@[redacted] main]# kubectl get nodes --context=eks-demo-cluster-dev
NAME                                          STATUS   ROLES    AGE    VERSION
ip-10-[redacted]-167.us-east-2.compute.internal   Ready    <none>   115m   v1.30.4-eks-[redacted]
ip-10-[redacted]-65.us-east-2.compute.internal    Ready    <none>   115m   v1.30.4-eks-[redacted]
ip-10-[redacted]-11.us-east-2.compute.internal    Ready    <none>   3m44s  v1.30.4-eks-[redacted]
ip-10-[redacted]-122.us-east-2.compute.internal   Ready    <none>   3m43s  v1.30.4-eks-[redacted]
```

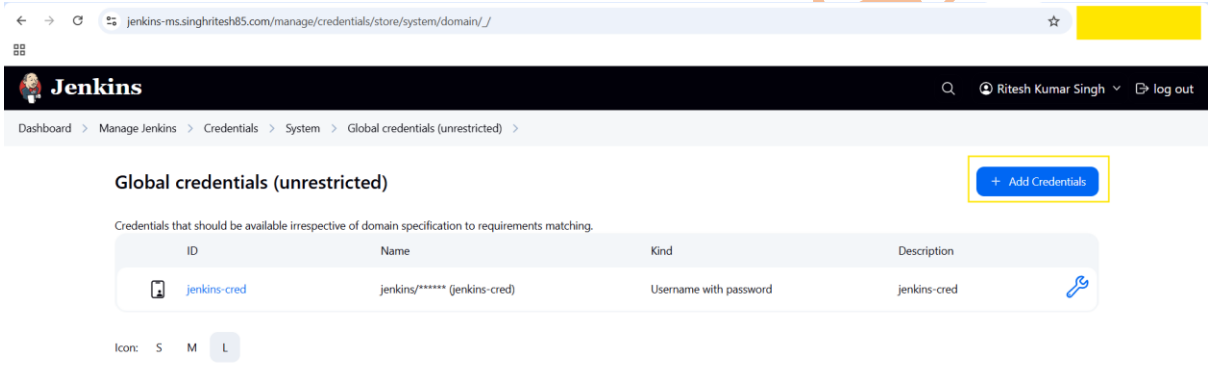
I had created the URL for Jenkins by creating the record set of Type CNAME in Azure DNS Zone using the Jenkins LoadBalancer DNS Name as shown in the screenshot attached below.

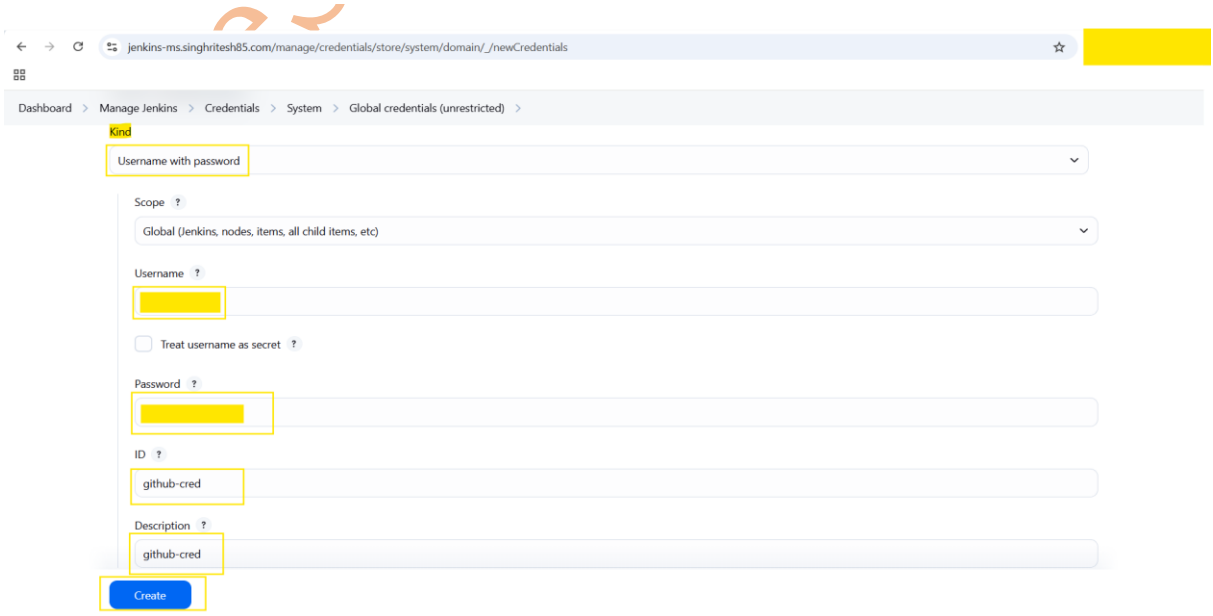


Finally, I logged-in to the Jenkins and created two credentials for Jenkins Slave and GitHub Credentials.

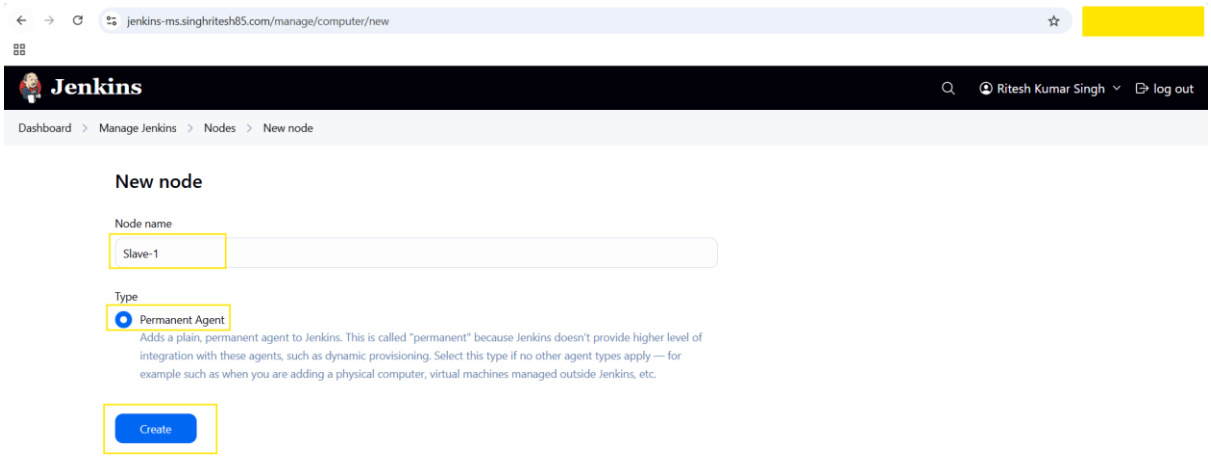


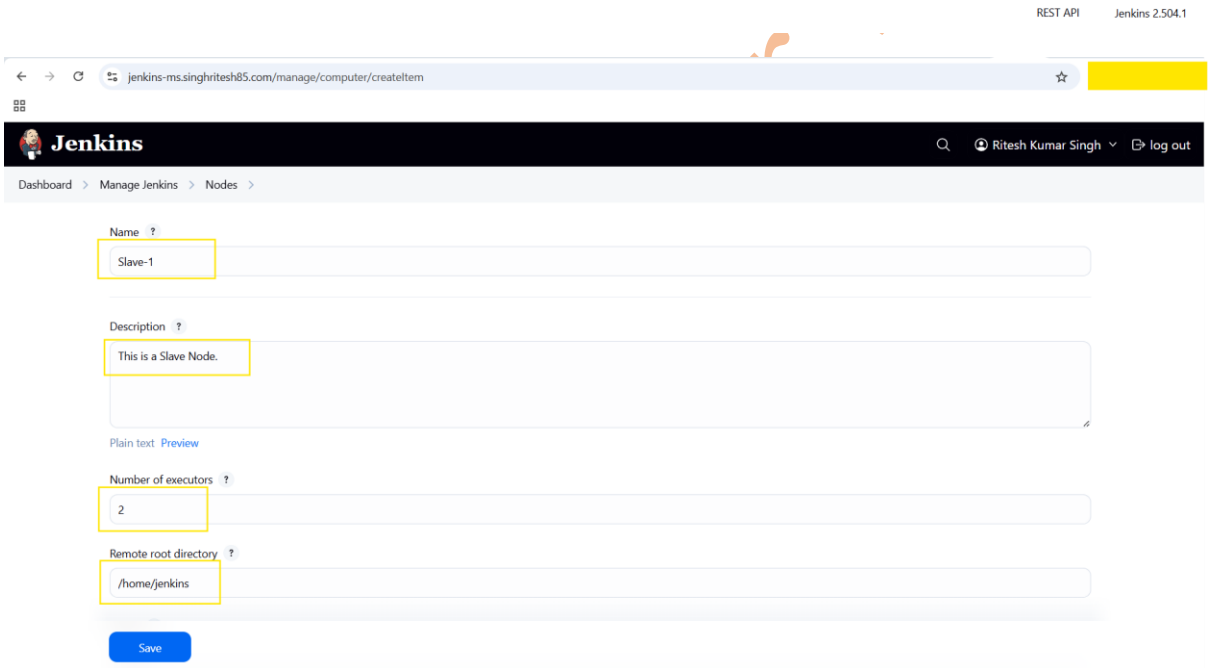

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Kind
Username with password
Scope ?
Global (Jenkins, nodes, items, all child items, etc)
Username ?
jenkins
Treat username as secret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

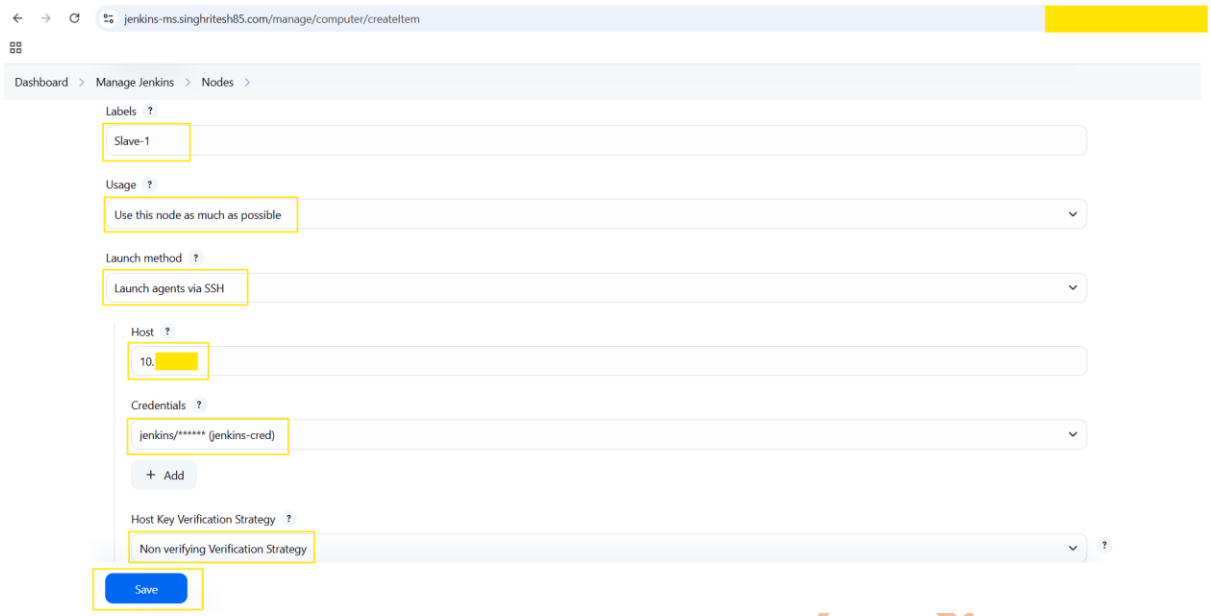

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
jenkins-cred
jenkins/****** (jenkins-cred)
Username with password
jenkins-cred
Icon: S M L
REST API
Jenkins 2.504.1

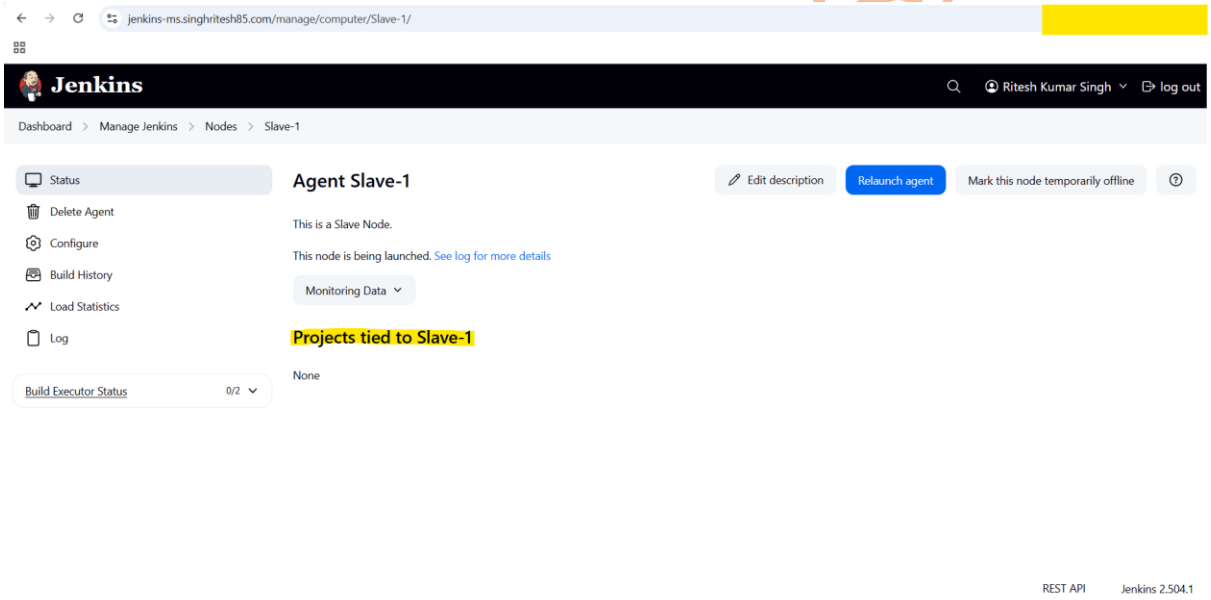

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Kind
Username with password
Scope ?
Global (Jenkins, nodes, items, all child items, etc)
Username ?
Treat username as secret ?
Password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

Then I had added Jenkins Slave Node as shown in the screenshot attached below. Go to Jenkins and then Manage Jenkins > Nodes > New Node.









Installation of Nginx Ingress Controller and Hashicorp Vault Prod in EKS and AKS Cluster

Installation of Nginx Ingress Controller and Hashicorp Vault Prod in EKS

Installation of Nginx Ingress Controller in EKS Cluster

To install Nginx Ingress Controller in EKS Cluster I used helm and below are the commands used.

kubectl create ns ingress-nginx

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02XXXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-type"="elb" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol"=http --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-ssl-ports"=https --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"

```
[root@                ~]# kubectl config get-contexts
CURRENT   NAME                  CLUSTER               AUTHINFO                                                NAMESPACE
*         aks-cluster           aks-cluster           clusterUser_aks-rg_aks-cluster
          eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:0          6:cluster/eks-demo-cluster-dev
[root@                ~]# kubectl config use-context eks-demo-cluster-dev
Switched to context "eks-demo-cluster-dev".
[root@                ~]# kubectl create ns ingress-nginx
namespace/ingress-nginx created
[root@                ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" 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 "ingress-nginx" chart repository
Update Complete. ⎈Happy Helming!⎈
```

```
[root@                ~]# helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\.beta\.kubern
etes\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02          6:certificate/          --set controller.service.annotations.
"service\.beta\.kubernetes\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-ba
lancer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-type"="elb" --set contr
oller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol"=http --set controller.service.annotations."service\.beta\.kuberne
tes\.io/aws-load-balancer-ssl-ports"=https --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"
```

```
[root@                ~]# kubectl get all -n ingress-nginx
NAME                                            READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-                   1/1     Running   0          2m6s

NAME                                         TYPE           CLUSTER-IP     EXTERNAL-IP                                                              PORT(S)
                  AGE
service/ingress-nginx-controller             LoadBalancer   10.     .92    a                                        0.us-east-2.elb.amazonaws.com   80:32585/
TCP,443:31666/TCP   2m6s
service/ingress-nginx-controller-admission   ClusterIP      10.     .187   <none>                                                                   443/TCP
                  2m6s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           2m6s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-             1         1         1       2m6s
```

Installation of Hashicorp Vault Prod in EKS Cluster

I had installed Hashicorp Vault Prod in EKS Cluster using the commands as shown in the screenshot attached below.

helm repo add hashicorp https://helm.releases.hashicorp.com

helm repo update

helm upgrade --install vault hashicorp/vault --namespace vault --create-namespace --set server.image.tag=1.19.0,server.ha.enabled=true,server.ha.raft.enabled=true,server.ha.replicas=3,server.dev.enabled=false,server.ui.enabled=true,server.ui.serviceType=ClusterIP,server.dataStorage.storageClass=gp2,server.dataStorage.size=2Gi

```
[root@                ~]# helm repo add hashicorp https://helm.releases.hashicorp.com
"hashicorp" 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 "ingress-nginx" chart repository
...Successfully got an update from the "hashicorp" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@                ~]# helm upgrade --install vault hashicorp/vault --namespace vault --create-namespace --set server.image.tag=1.19.0,server.ha.enabled=t
rue,server.ha.raft.enabled=true,server.ha.replicas=3,server.dev.enabled=false,server.ui.enabled=true,server.ui.serviceType=ClusterIP,server.dataStorage.storag
eClass=gp2,server.dataStorage.size=2Gi
Release "vault" does not exist. Installing it now.
NAME: vault
LAST DEPLOYED:                2025
NAMESPACE: vault
STATUS: deployed
REVISION: 1
NOTES:
Thank you for installing HashiCorp Vault!

Now that you have deployed Vault, you should look over the docs on using
Vault with Kubernetes available here:

https://developer.hashicorp.com/vault/docs


Your release is named vault. To learn more about the release, try:

  $ helm status vault
  $ helm get manifest vault
```

```
[root@                main]# kubectl get pods -n vault
NAME                                    READY   STATUS    RESTARTS   AGE
vault-0                                 0/1     Running   0          26s
vault-1                                 0/1     Running   0          26s
vault-2                                 0/1     Running   0          26s
vault-agent-injector-                   1/1     Running   0          26s
```

Now logged-in to the vault pods and executed the below commands.

==========

on pod vault-0

==========

kubectl exec -it vault-0 -n vault sh

$ vault operator init

---------- Key1

---------- Key2

---------- Key3

---------- Key4

---------- Key5

---------- Root Token

$ vault operator unseal ----------> First Time

Unseal Key:

$ vault operator unseal ----------> Second Time

Unseal Key:

$ vault operator unseal ----------> Third Time

Unseal Key:

$ vault login ----------> Should run only on vault-0

Token:

==========

Run the below commands as written below

==========

kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers

kubectl exec -ti vault-1 -n vault -- vault operator raft join http://vault-0.vault-internal:8200

kubectl exec -ti vault-2 -n vault -- vault operator raft join http://vault-0.vault-internal:8200

==========

on pod vault-1

==========

kubectl exec -it vault-1 -n vault sh

$ vault operator unseal ----------> First Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Second Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Third Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

==========

on pod vault-2

==========

kubectl exec -it vault-2 -n vault sh

$ vault operator unseal ----------> First Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Second Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Third Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

==========

Finally, Run the below command to check the leader and follower of Hashicorp Vault

==========

kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers ----------> Here you will see that vault-0 will be leader and vault-1 and vault-2 pod will be as forwarder

==========

Now check the status of three vault pods

==========

kubectl get pods -n vault

NAME READY STATUS RESTARTS AGE

vault-0 1/1 Running 0 3h46m

vault-1 1/1 Running 0 3h46m

vault-2 1/1 Running 0 3h46m

vault-agent-injector-7XXXXXXXXb-mXXXv 1/1 Running 0 3h46m

The Pods will be in the running status and READY 1/1

```
[root@                ~]# kubectl exec -it vault-0 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault operator init
Unseal Key 1: 1                                   8
Unseal Key 2: N                                   d
Unseal Key 3: F                                   x
Unseal Key 4: 2                                   G
Unseal Key 5: x                                   i

Initial Root Token: h                        4

Vault initialized with 5 key shares and a key threshold of 3. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 3 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated root key. Without at least 3 keys to
reconstruct the root key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         1/3
Unseal Nonce            8                                 f
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         2/3
Unseal Nonce            8[redacted]f
Version                 1.19.0
Build Date              2025[redacted]
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            5
Threshold               3
Version                 1.19.0
Build Date              2025-[redacted]
Storage Type            raft
Cluster Name            vault-cluster-[redacted]
Cluster ID              6[redacted]
Removed From Cluster    false
HA Enabled              true
HA Cluster              https://vault-0.vault-internal:8201
HA Mode                 active
Active Since            2025[redacted]
Raft Committed Index    37
Raft Applied Index      37
```

```
/ $ vault login
Token (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                h[redacted]4
token_accessor       [redacted]
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

```
[root@[redacted] main]# kubectl exec -it vault-1 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ [redacted]

/ $ vault operator unseal
Unseal Key (will be hidden):
Key                    Value
---                    -----
Seal Type              shamir
Initialized            true
Sealed                 true
Total Shares           5
Threshold              3
Unseal Progress        1/3
Unseal Nonce           [redacted]
Version                1.19.0
Build Date             2025[redacted]
Storage Type           raft
Removed From Cluster   false
HA Enabled             true
/ $ [redacted]
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                    Value
---                    -----
Seal Type              shamir
Initialized            true
Sealed                 true
Total Shares           5
Threshold              3
Unseal Progress        2/3
Unseal Nonce           [redacted]
Version                1.19.0
Build Date             2025[redacted]
Storage Type           raft
Removed From Cluster   false
HA Enabled             true
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         0/3
Unseal Nonce            n/a
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $
```

```
[root@ main]# kubectl exec -it vault-2 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         1/3
Unseal Nonce
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         2/3
Unseal Nonce
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

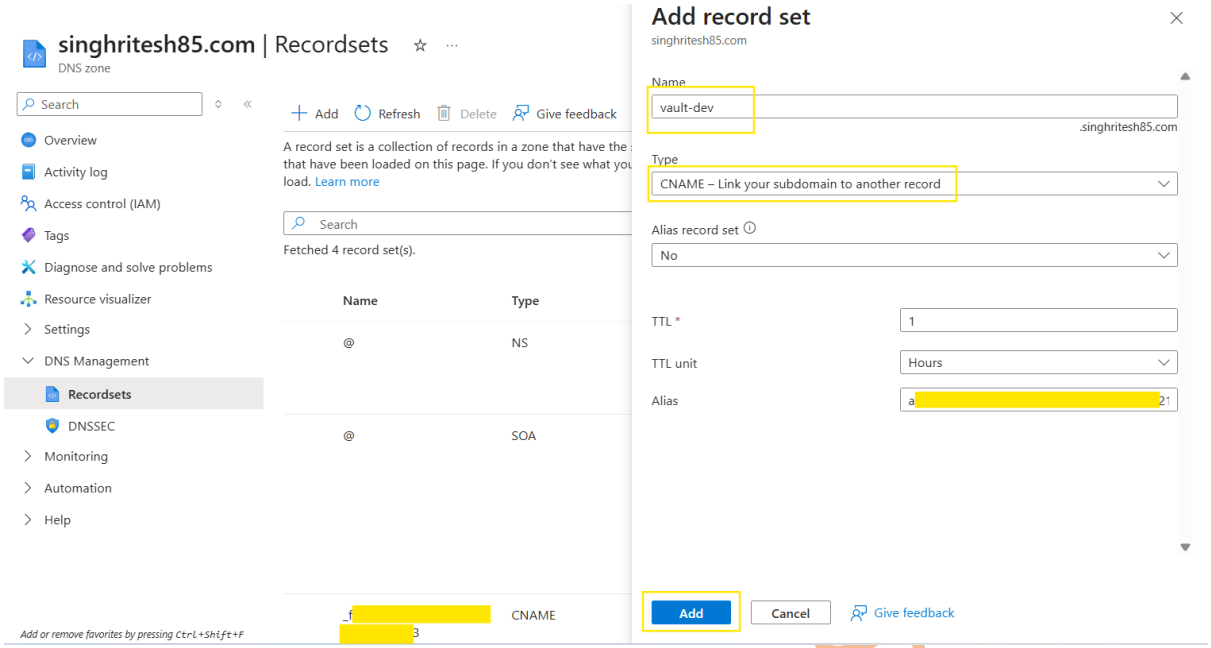
```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         0/3
Unseal Nonce            n/a
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $
```

```
[root@ main]# kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers
Node                                    Address                         State       Voter
----                                    -------                         -----       -----
                                        vault-0.vault-internal:8201     leader      true
                                        vault-1.vault-internal:8201     follower    true
                                        vault-2.vault-internal:8201     follower    true
```

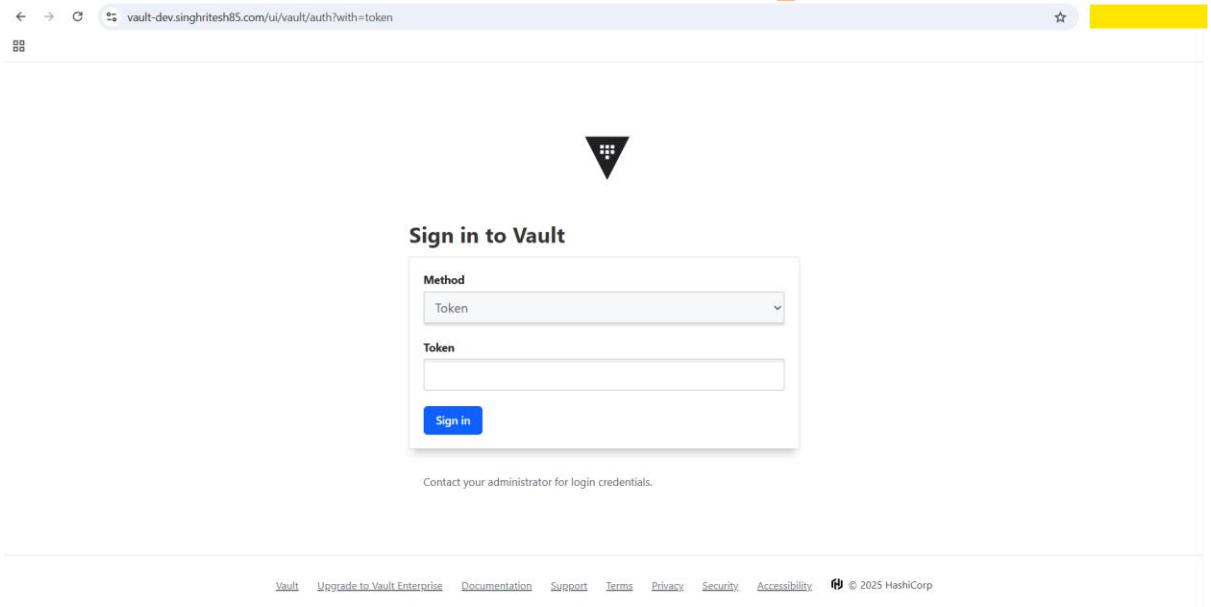
Then I created the Ingress Rule and finally a Record Set of Type CNAME in Azure DNS Zone to access the vault in prod mode in dev cluster.

```
[root@ ~]# kubectl apply -f vault-dev-ingress-rule.yaml --context=eks-demo-cluster-dev
ingress.networking.k8s.io/vault-ingress created
[root@ ~]# kubectl get ing -A --watch --context=eks-demo-cluster-dev
NAMESPACE   NAME            CLASS   HOSTS                       ADDRESS                                                          PORTS   AGE
vault       vault-ingress   nginx   vault-dev.singhritesh85.com   a0.us-east-2.elb.amazonaws.com   80      15s
^C[root@ ~]#
[root@ ~]#
[root@ ~]# cat vault-dev-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: vault-dev.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: vault-active
            port:
              number: 8200
```

```
cat vault-dev-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: vault-dev.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: vault-active
            port:
              number: 8200
```



Finally, I was able to access the Hashicorp vault in prod mode in dev cluster (EKS Cluster) as shown in the screenshot attached below.



Then I installed Nginx Ingress Controller and Hashicorp Vault in Prod mode in Stage Cluster (AKS Cluster).

Installation of Nginx Ingress Controller and Hashicorp Vault in Prod mode in AKS Cluster

Installation of Nginx Ingress Controller in AKS Cluster

To install Nginx Ingress Controller in AKS Cluster I used the command as written below.

kubectl create ns ingress-nginx

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx

helm upgrade ingress-nginx ingress-nginx/ingress-nginx --namespace ingress-nginx --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-probe-request-path"=/healthz --set controller.service.externalTrafficPolicy=Local

```
[root@ ~]# kubectl config get-contexts
CURRENT   NAME                  CLUSTER               AUTHINFO                                                  NAMESPACE
          aks-cluster           aks-cluster           clusterUser_aks-rg_aks-cluster
*         eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:0        6:cluster/eks-demo-cluster-dev
[root@ ~]# kubectl config use-context aks-cluster
Switched to context "aks-cluster".
[root@ ~]# kubectl create ns ingress-nginx
```

```
[root@ ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" already exists with the same configuration, skipping
[root@ ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "hashicorp" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@ ~]# helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx
```

```
[root@ ~]# helm upgrade ingress-nginx ingress-nginx/ingress-nginx --namespace ingress-nginx --set controller.service.annotations."service\.bet
a\.kubernetes\.io/azure-load-balancer-health-probe-request-path"=/healthz --set controller.service.externalTrafficPolicy=Local
```

```
[root@ ~]# kubectl get all -n ingress-nginx
NAME                                        READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-               1/1     Running   0          5m43s

NAME                                         TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
service/ingress-nginx-controller             LoadBalancer   10.    .227    4.      .216    80:32191/TCP,443:31275/TCP   5m43s
service/ingress-nginx-controller-admission   ClusterIP      10.    .213    <none>          443/TCP                      5m43s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           5m43s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-             1         1         1       5m43s
```

Installation of Hashicorp Vault in Prod mode in AKS Cluster (Stage Environment)

To install Hashicorp Vault in AKS Cluster I used the commands as written below.

helm repo add hashicorp https://helm.releases.hashicorp.com

helm repo update

helm install vault hashicorp/vault --namespace vault --create-namespace --set server.image.tag=1.19.0,server.ha.enabled=true,server.ha.raft.enabled=true,server.ha.replicas=3,server.dev.enabled=false,server.ui.enabled=true,server.ui.serviceType=ClusterIP

kubectl get pods -n vault

```
[root@                ~]# helm repo add hashicorp https://helm.releases.hashicorp.com
"hashicorp" already exists with the same configuration, skipping
[root@                ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "hashicorp" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@                ~]# helm install vault hashicorp/vault --namespace vault --create-namespace --set server.image.tag=1.19.0,server.ha.enabled=true,server
.ha.raft.enabled=true,server.ha.replicas=3,server.dev.enabled=false,server.ui.enabled=true,server.ui.serviceType=ClusterIP
NAME: vault
LAST DEPLOYED:                  2025
NAMESPACE: vault
STATUS: deployed
REVISION: 1
NOTES:
Thank you for installing HashiCorp Vault!

Now that you have deployed Vault, you should look over the docs on using
Vault with Kubernetes available here:

https://developer.hashicorp.com/vault/docs


Your release is named vault. To learn more about the release, try:

  $ helm status vault
  $ helm get manifest vault
```

```
[root@                ~]# kubectl get pods -n vault --watch
NAME                                    READY   STATUS    RESTARTS   AGE
vault-0                                 0/1     Running   0          110s
vault-1                                 0/1     Running   0          110s
vault-2                                 0/1     Running   0          110s
vault-agent-injector-                   1/1     Running   0          110s
```

Now logged-in to the vault pods in AKS Cluster and executed the below commands.

==========

on pod vault-0

==========

kubectl exec -it vault-0 -n vault sh

$ vault operator init

---------- Key1

---------- Key2

---------- Key3

---------- Key4

---------- Key5

---------- Root Token

$ vault operator unseal ----------> First Time

Unseal Key:

$ vault operator unseal ----------> Second Time

Unseal Key:

$ vault operator unseal ----------> Third Time

Unseal Key:

$ vault login ----------> Should run only on vault-0

Token:

==========

Then run the below commands as written below

==========

kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers

kubectl exec -ti vault-1 -n vault -- vault operator raft join http://vault-0.vault-internal:8200

kubectl exec -ti vault-2 -n vault -- vault operator raft join http://vault-0.vault-internal:8200

==========

on pod vault-1

==========

kubectl exec -it vault-1 -n vault sh

$ vault operator unseal ----------> First Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Second Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Third Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

==========

on pod vault-2

==========

kubectl exec -it vault-2 -n vault sh

$ vault operator unseal ----------> First Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Second Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

$ vault operator unseal ----------> Third Time

Unseal Key: ----------> Unseal Key should be one which you obtained from vault-0

==========

Finally, Run the below command to check the leader and follower in vault

==========

kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers ----------> Here you will see that vault-0 will be leader and vault-1 and vault-2 pod will be as forwarder

==========

Now check the status of three vault pods

==========

kubectl get pods -n vault

```
NAME                                    READY   STATUS    RESTARTS   AGE
vault-0                                 1/1     Running   0          3h46m
vault-1                                 1/1     Running   0          3h46m
vault-2                                 1/1     Running   0          3h46m
vault-agent-injector-7594d5f8bb-m486v   1/1     Running   0          3h46m
```

The Pods will be in the running status and READY 1/1

```
[root@                    ~]# kubectl exec -it vault-0 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault operator init
Unseal Key 1: X                                   F
Unseal Key 2: O                                   J
Unseal Key 3: 8                                   Z
Unseal Key 4: 8                                   +
Unseal Key 5: o                                   C

Initial Root Token: h                         h

Vault initialized with 5 key shares and a key threshold of 3. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 3 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated root key. Without at least 3 keys to
reconstruct the root key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         1/3
Unseal Nonce
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         2/3
Unseal Nonce            [redacted]
Version                 1.19.0
Build Date              2025[redacted]
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            5
Threshold               3
Version                 1.19.0
Build Date              2025[redacted]
Storage Type            raft
Cluster Name            vault-cluster-[redacted]
Cluster ID              [redacted]
Removed From Cluster    false
HA Enabled              true
HA Cluster              https://vault-0.vault-internal:8201
HA Mode                 active
Active Since            2025[redacted]
Raft Committed Index    37
Raft Applied Index      37
```

```
/ $ vault login
Token (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                h[redacted]h
token_accessor       z[redacted]g
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

```
[root@[redacted] ~]# kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers
Node                                    Address                        State     Voter
----                                    -------                        -----     -----
[redacted]    vault-0.vault-internal:8201    leader    true
[root@[redacted] ~]# kubectl exec -ti vault-1 -n vault -- vault operator raft join http://vault-0.vault-internal:8200
Key       Value
---       -----
Joined    true
[root@Terraform-Server ~]# kubectl exec -ti vault-2 -n vault -- vault operator raft join http://vault-0.vault-internal:8200
Key       Value
---       -----
Joined    true
```

```
[root@[redacted] ~]# kubectl exec -it vault-1 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         1/3
Unseal Nonce            [redacted]
Version                 1.19.0
Build Date              2025[redacted]
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         2/3
Unseal Nonce            [redacted]
Version                 1.19.0
Build Date              2025[redacted]
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         0/3
Unseal Nonce            n/a
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

```
[root@ ~]# kubectl exec -it vault-2 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         1/3
Unseal Nonce
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         2/3
Unseal Nonce
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
```

```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               3
Unseal Progress         0/3
Unseal Nonce            n/a
Version                 1.19.0
Build Date              2025
Storage Type            raft
Removed From Cluster    false
HA Enabled              true
/ $
[root@                  ~]#
```

```
[root@                  ~]# kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers
Node                    Address                        State       Voter
----                    -------                        -----       -----
                        vault-0.vault-internal:8201    leader      true
                        vault-1.vault-internal:8201    follower    true
                        vault-2.vault-internal:8201    follower    true
```

To create the Ingress Rule, I created the Kubernetes Secrets first as shown in the screenshot attached below.

kubectl create secret tls vault-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n vault

```
[root@                  ~]# kubectl create secret tls vault-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n vault
secret/vault-tls-secret created
```

Then I created the Ingress Rule and finally a Record Set of A Type in Azure DNS Zone to access the vault in prod mode in dev cluster.

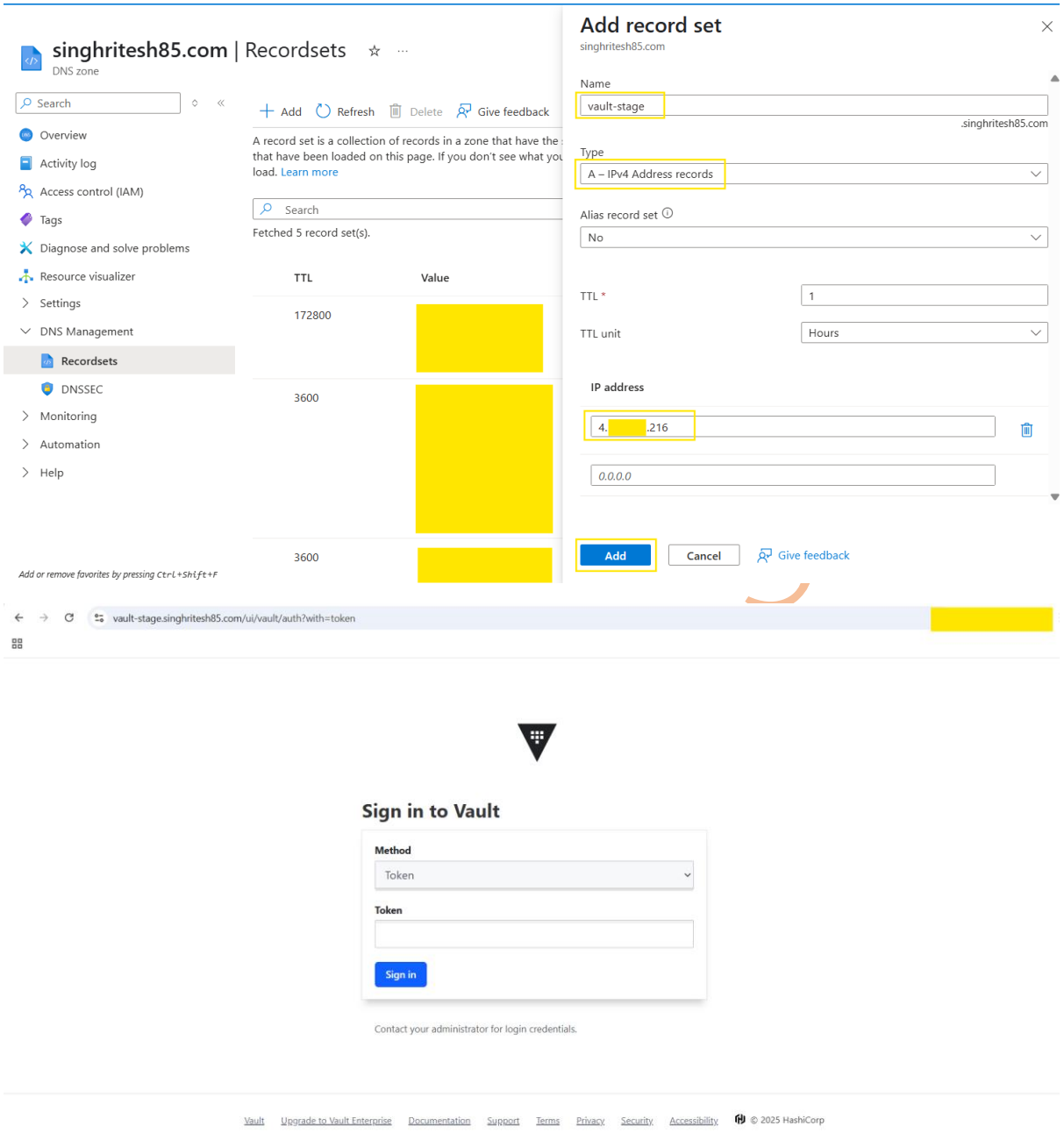
```
[root@                    ~]# kubectl apply -f vault-stage-ingress-rule.yaml --context=aks-cluster
ingress.networking.k8s.io/vault-ingress created
[root@                    ~]# kubectl get ing -A --watch --context=aks-cluster
NAMESPACE   NAME            CLASS   HOSTS                          ADDRESS   PORTS     AGE
vault       vault-ingress   nginx   vault-stage.singhritesh85.com              80, 443   4s
vault       vault-ingress   nginx   vault-stage.singhritesh85.com   4.        .216   80, 443   10s
^C[root@                  ~]#
[root@                    ~]# cat vault-stage-ingress-rule.yaml
# kubectl create secret tls vault-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n vault
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - vault-stage.singhritesh85.com
    secretName: vault-tls-secret
  rules:
  - host: vault-stage.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: vault-active
            port:
              number: 8200
```

cat vault-stage-ingress-rule.yaml

kubectl create secret tls vault-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n vault

---

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - vault-stage.singhritesh85.com
    secretName: vault-tls-secret
  rules:
  - host: vault-stage.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: vault-active
            port:
              number: 8200
```

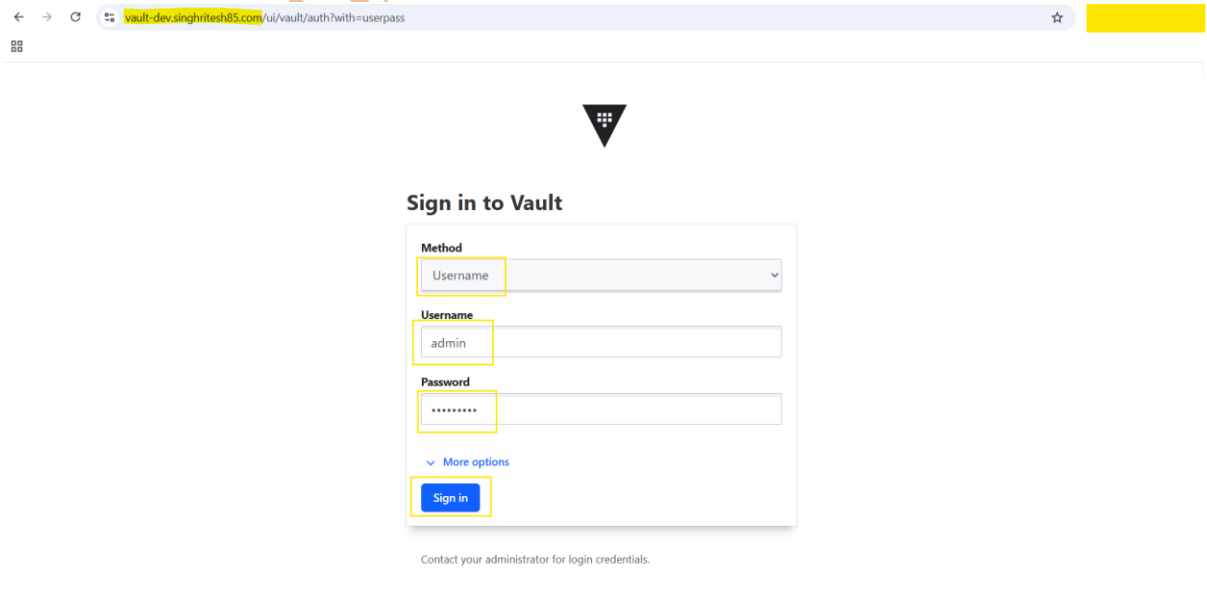


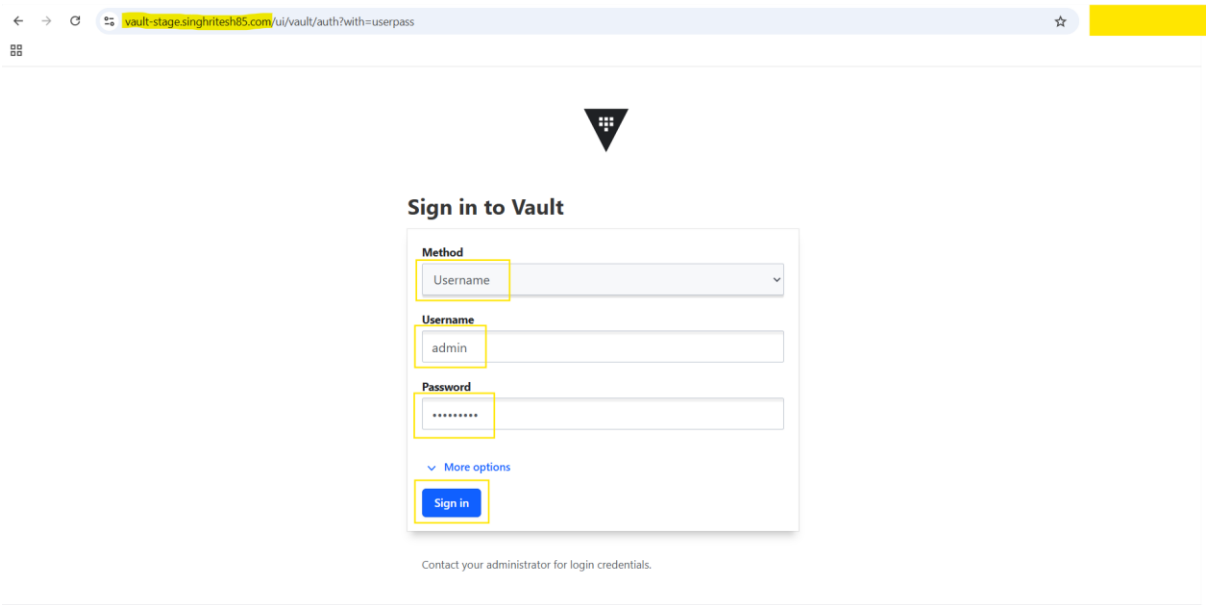
For the first time login method into the vault is Token. The Token which I got from the vault-0 pod in dev cluster and stage cluster was used to login into the Hashicorp vault prod mode in dev cluster and stage cluster respectively. Then I enabled the vault auth userpass and created a username with password with Administrator privileges which I will use to login in to the Hashicorp Vault prod mode in dev cluster (EKS Cluster) and stage cluster (AKS Cluster) as shown in the screenshot attached below.

```
$ vault auth enable userpass


$ vault policy write admin -<<EOF
path "*" {
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
EOF


$ vault write auth/userpass/users/admin password=Admin@123 policies=admin
```

```
[root@                ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $  vault auth enable userpass
Success! Enabled userpass auth method at: userpass/
/ $ vault policy write admin -<<EOF
> path "*" {
>   capabilities = ["create", "read", "update", "delete", "list", "sudo"]
> }
> EOF
Success! Uploaded policy: admin
/ $ vault write auth/userpass/users/admin password=Admin@123 policies=admin
Success! Data written to: auth/userpass/users/admin
/ $
[root@                ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault auth enable userpass
Success! Enabled userpass auth method at: userpass/
/ $ vault policy write admin -<<EOF
> path "*" {
>   capabilities = ["create", "read", "update", "delete", "list", "sudo"]
> }
> EOF
Success! Uploaded policy: admin
/ $ vault write auth/userpass/users/admin password=Admin@123 policies=admin
Success! Data written to: auth/userpass/users/admin
```

Then I logged-in with the created username and password as shown in the screenshot attached below.



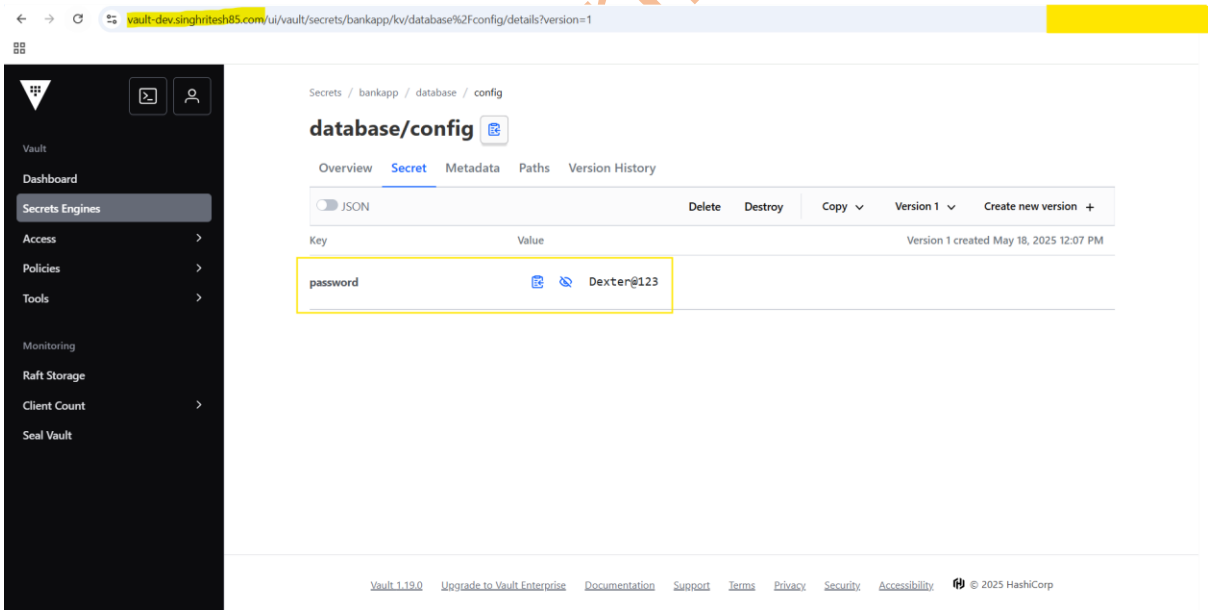


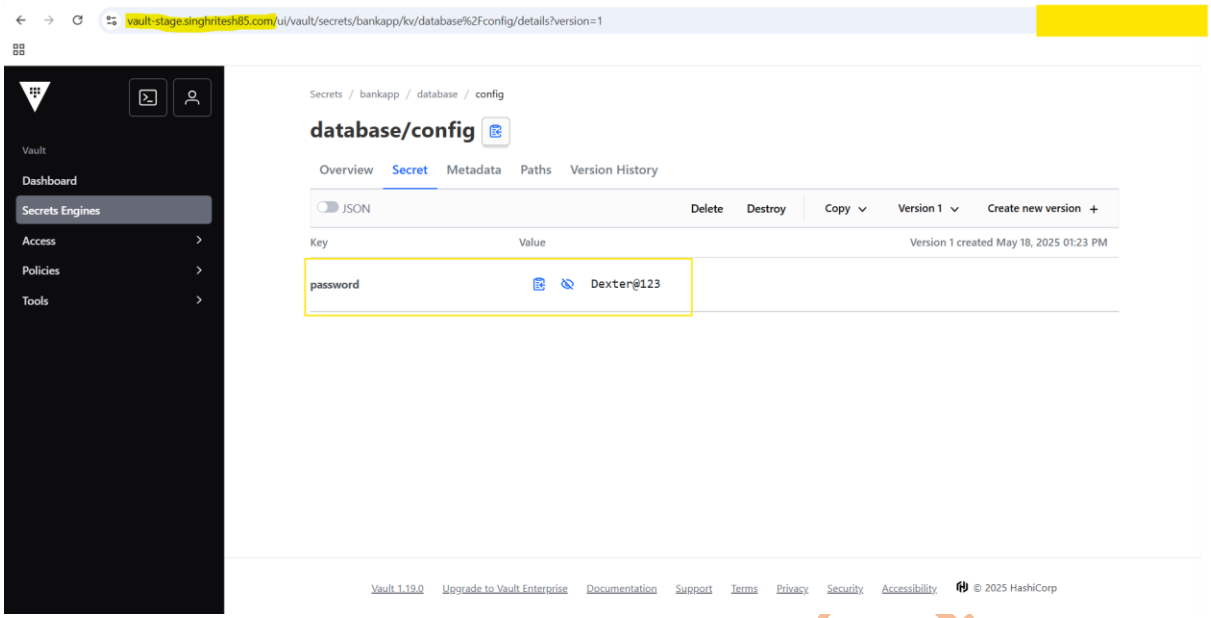
Then created the Hashicorp vault secrets in dev cluster (EKS Cluster) and stage cluster (AKS Cluster) as shown in the screenshot attached below.

```
[root@                ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault secrets enable -path=bankapp kv-v2
Success! Enabled the kv-v2 secrets engine at: bankapp/
/ $
/ $ vault kv put bankapp/database/config password=Dexter@123
======== Secret Path ========
bankapp/data/database/config

======= Metadata =======
Key                Value
---                -----
created_time       2025-05-18T06:37:57.871570522Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1
/ $
/ $ vault kv get bankapp/database/config
======== Secret Path ========
bankapp/data/database/config

======= Metadata =======
Key                Value
---                -----
created_time       2025-05-18T06:37:57.871570522Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
password    Dexter@123
```

```
[root@                    ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
cubbyhole/    cubbyhole    cubbyhole_947c03b7    per-token private secret storage
identity/     identity     identity_47365a9f     identity store
sys/          system       system_ea37fcb0       system endpoints used for control, policy and debugging
/ $ vault secrets enable -path=bankapp kv-v2
Success! Enabled the kv-v2 secrets engine at: bankapp/
/ $ vault kv put bankapp/database/config password=Dexter@123
======== Secret Path ========
bankapp/data/database/config

======= Metadata =======
Key                Value
---                -----
created_time       2025
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1
/ $ vault kv get bankapp/database/config
======== Secret Path ========
bankapp/data/database/config

======= Metadata =======
Key                Value
---                -----
created_time       2025
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
password    Dexter@123
```

You can verify the same secrets in your dev cluster (EKS Cluster) and stage cluster (AKS Cluster) Hashicorp Vault UI as shown in the screenshot attached below.





Create and apply the policy in vault to read secrets from Hashicorp Vault in EKS and AKS Cluster

Next, I had created a policy which allowed reading secrets. This policy was attached to a role, which could be used to grant access to specific Kubernetes service accounts.

```
vault policy write bankapp-policy -<<EOF
path "bankapp/data/database/config" {
  capabilities = ["read"]
}
EOF
```

```
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
```

```
/ $ vault policy write bankapp-policy -<<EOF
> path "bankapp/data/database/config" {
>   capabilities = ["read"]
> }
> EOF
Success! Uploaded policy: bankapp-policy
```

```
/ $ vault policy list
admin
bankapp-policy
default
root
/ $ vault policy read bankapp-policy
path "bankapp/data/database/config" {
  capabilities = ["read"]
}
/ $
```

vault policy list

vault policy read <policy-name>

```
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
```

```
/ $ vault policy write bankapp-policy -<<EOF
> path "bankapp/data/database/config" {
>   capabilities = ["read"]
> }
> EOF
```

```
/ $ vault policy list
admin
bankapp-policy
default
root
/ $ vault policy read bankapp-policy
path "bankapp/data/database/config" {
  capabilities = ["read"]
}
/ $
```

Enable the Kubernetes authentication method in Vault in EKS and AKS Cluster as shown in the screenshot attached below.

Vault auth enable kubernetes

```
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault auth enable kubernetes
Success! Enabled kubernetes auth method at: kubernetes/
/ $
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault auth enable kubernetes
Success! Enabled kubernetes auth method at: kubernetes/
/ $
```

Configure Vault to communicate with the Kubernetes API server on EKS and AKS Cluster as shown in the screenshot attached below.

```
vault write auth/kubernetes/config \
  token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
  kubernetes_host=https://${KUBERNETES_PORT_443_TCP_ADDR}:443 \
  kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

```
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault write auth/kubernetes/config \
>    token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
>    kubernetes_host=https://${KUBERNETES_PORT_443_TCP_ADDR}:443 \
>    kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
Success! Data written to: auth/kubernetes/config
/ $
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault write auth/kubernetes/config \
>    token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
>    kubernetes_host=https://${KUBERNETES_PORT_443_TCP_ADDR}:443 \
>    kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
Success! Data written to: auth/kubernetes/config
/ $
```

Then I created a role named as bankapp that binds the policy bankapp-policy to a Kubernetes service account(bankapp-sa) in a specific namespace mysql. This allows the service account to access secrets stored in Hashicorp Vault

```
vault write auth/kubernetes/role/bankapp bound_service_account_names=bankapp-sa
bound_service_account_namespaces=mysql policies=bankapp-policy ttl=1h
```

```
[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.

/ $ vault write auth/kubernetes/role/bankapp bound_service_account_names=bankapp-sa bound_service_account_namespaces=mysql policies=bankapp-policy ttl=1h
Success! Data written to: auth/kubernetes/role/bankapp

[root@ ~]# kubectl exec -it vault-0 -n vault sh --context=aks-cluster
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.

/ $ vault write auth/kubernetes/role/bankapp bound_service_account_names=bankapp-sa bound_service_account_namespaces=mysql policies=bankapp-policy ttl=1h
Success! Data written to: auth/kubernetes/role/bankapp
/ $
```

The vault secrets will be accessed from mysql database which will exist in mysql namespace so here I had created a namespace mysql then service account named as vault-sa in namespace mysql on EKS and AKS Cluster.

```
kubectl create ns mysql

kubectl create sa bankapp-sa-n mysql
```

```
[root@ ~]# kubectl config get-contexts
CURRENT   NAME                  CLUSTER               AUTHINFO                                                NAMESPACE
          aks-cluster           aks-cluster           clusterUser_aks-rg_aks-cluster
*         eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:02        06:cluster/eks-demo-cluster-dev
[root@ ~]# kubectl create ns mysql
namespace/mysql created
[root@ ~]# kubectl create sa bankapp-sa -n mysql
serviceaccount/bankapp-sa created
[root@ ~]# kubectl config use-context aks-cluster
Switched to context "aks-cluster".
[root@ ~]# kubectl create ns mysql
namespace/mysql created
[root@ ~]# kubectl create sa bankapp-sa -n mysql
serviceaccount/bankapp-sa created
```

Now I install ArgoCD in EKS Cluster (dev Cluster) and then created the ingress rule for ArgoCD to access it by adding the DNS Name of the LoadBalancer in Azure DNS Zone by creating a Record Set of CNAME Type then added AKS Cluster into ArgoCD as shown in the screenshot attached below.

Installation of ArgoCD

kubectl create namespace argocd

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

You can get the default password of admin user in ArgoCD using the command as written below

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

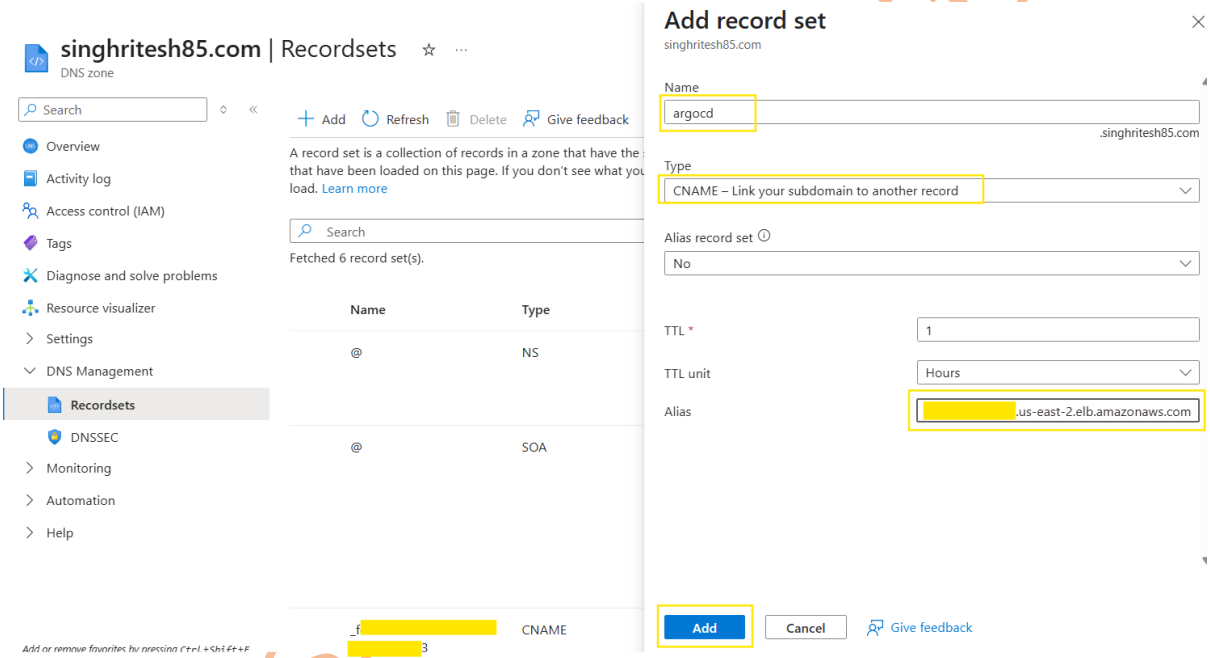
```
[root@                    ~]# kubectl config get-contexts
CURRENT   NAME                  CLUSTER               AUTHINFO                                                NAMESPACE
*         aks-cluster           aks-cluster           clusterUser_aks-rg_aks-cluster
          eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:02          6:cluster/eks-demo-cluster-dev
[root@                    ~]# kubectl config use-context eks-demo-cluster-dev
Switched to context "eks-demo-cluster-dev".
[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
k              D[root@                    ~]#
```

cat argocd-ingress-rule.yaml

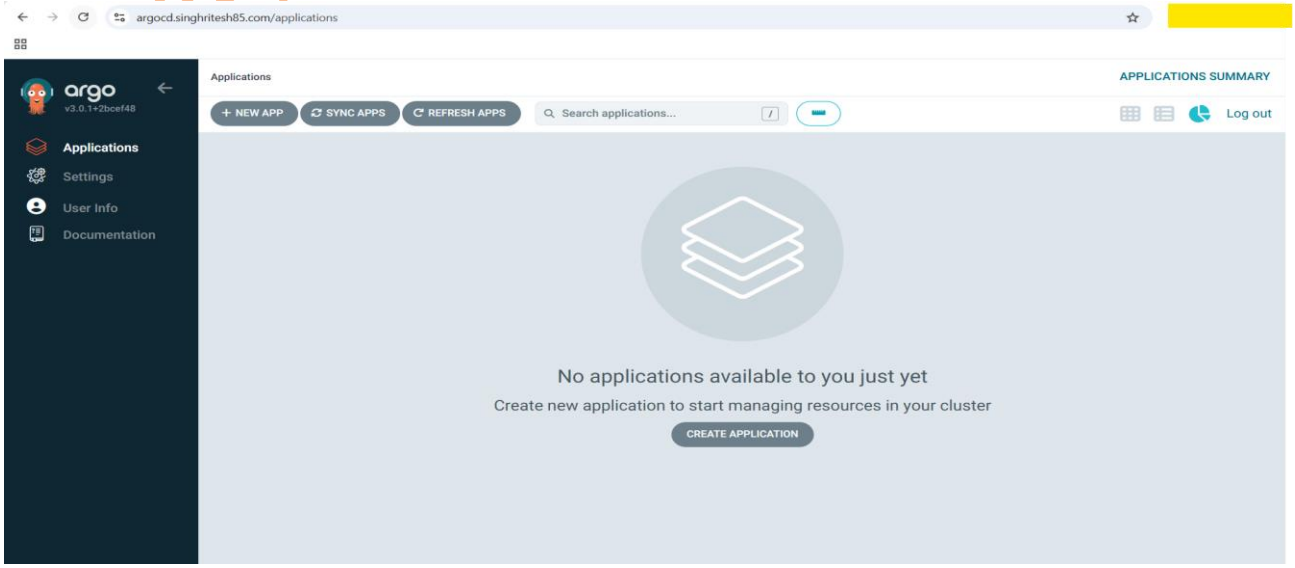
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ### You can use this option for this particular case for ArgoCD but not for all
#    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  ingressClassName: nginx
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server   ### Provide your service Name
            port:
              number: 80     #### Provide your service port for this particular example you can also choose 443
```

```
[root@                    ~]# kubectl apply -f argocd-ingress-rule.yaml
ingress.networking.k8s.io/minimal-ingress created
[root@                    ~]# cat argocd-ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ###  You can use this option for this particular case for ArgoCD but not for all
#    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  ingressClassName: nginx
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server   ### Provide your service Name
            port:
              number: 80     #### Provide your service port for this particular example you can also choose 443
```

```
[root@                    ~]# kubectl get ing -A --watch
NAMESPACE   NAME              CLASS   HOSTS                    ADDRESS                                                          PORTS   AGE
argocd      minimal-ingress   nginx   argocd.singhritesh85.com   a                                  6.us-east-2.elb.amazonaws.com   80      85s
vault       vault-ingress     nginx   vault-dev.singhritesh85.com   a                                  6.us-east-2.elb.amazonaws.com   80      140m
```



Finally, I was able to login into the ArgoCD and successfully changed its password from default admin.



Then I had installed ArgoCD CLI on Jenkins Slave Node using the commands as shown in the screenshot attached 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

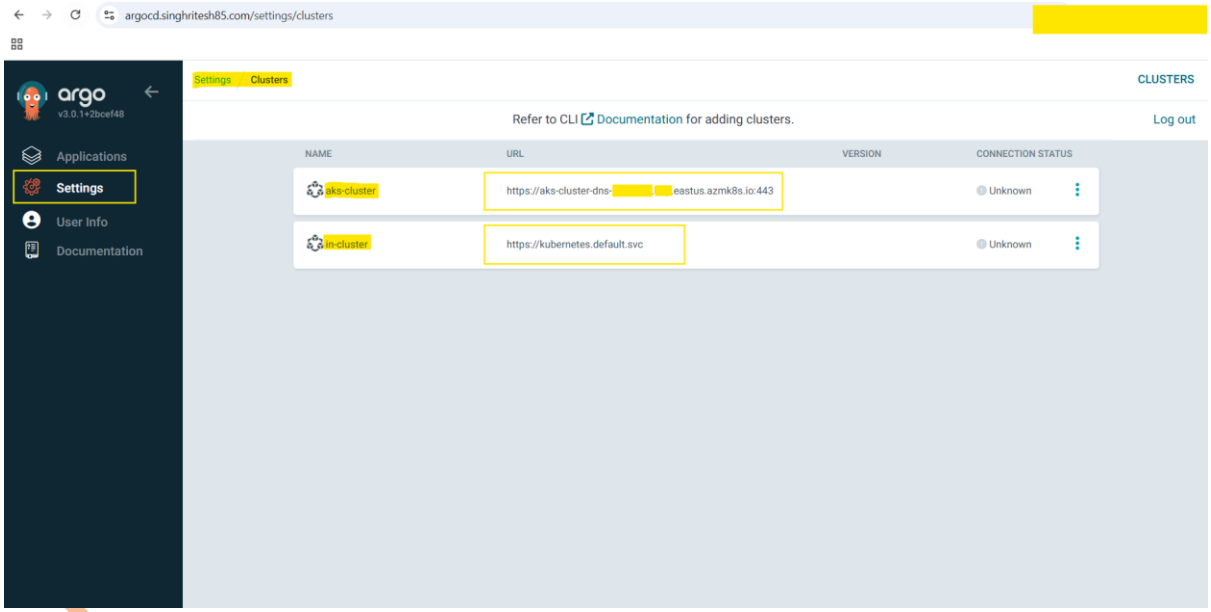
```
[jenkins@ ~]$ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
[jenkins@ ~]$ sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
[jenkins@ ~]$ rm argocd-linux-amd64
```

Then I had added the AKS Cluster using the command **argocd cluster add aks-cluster** as shown in the screenshot attached below.

argocd login argocd.singhritesh85.com --username admin --password Admin@123 --skip-test-tls --grpc-web

argocd cluster add aks-cluster

```
[jenkins@ ~]$ argocd login argocd.singhritesh85.com --username admin --password Admin@123 --skip-test-tls --grpc-web
'admin:login' logged in successfully
Context 'argocd.singhritesh85.com' updated
[jenkins@ ~]$ argocd cluster add aks-cluster
WARNING: This will create a service account `argocd-manager` on the cluster referenced by context `aks-cluster` with full cluster level privileges. Do you wan
t to continue [y/N]? y
{"level":"info","msg":"ServiceAccount \"argocd-manager\" created in namespace \"kube-system\"","time":"2025-"}
{"level":"info","msg":"ClusterRole \"argocd-manager-role\" created","time":"2025-"}
{"level":"info","msg":"ClusterRoleBinding \"argocd-manager-role-binding\" created","time":"2025-"}
{"level":"info","msg":"Created bearer token secret for ServiceAccount \"argocd-manager\"","time":"2025-"}
Cluster 'https://aks-cluster-dns-.eastus.azmk8s.io:443' added
```

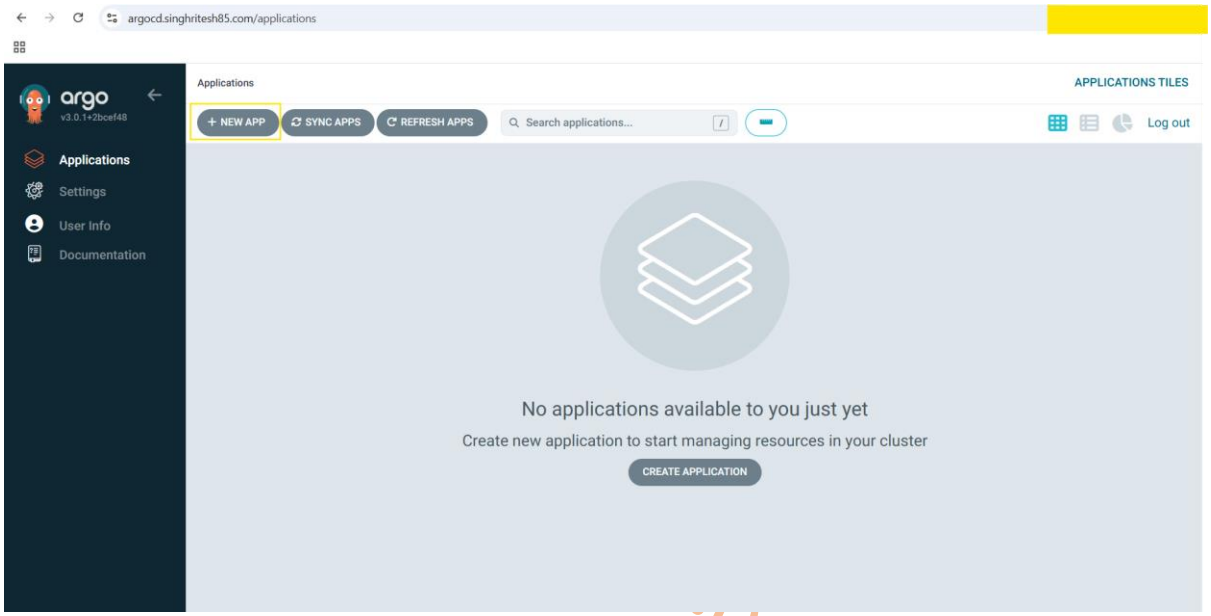


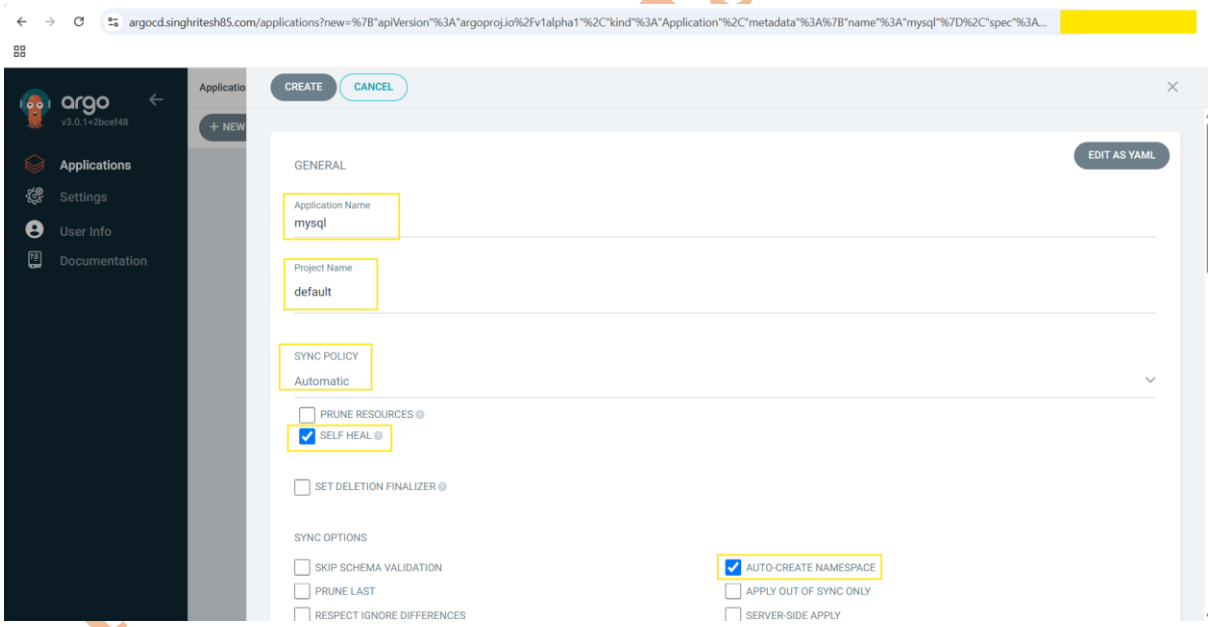
On Jenkins Slave Node I had provided Authentication and Authorization using Access Key and Secret Key as shown in the screenshot attached below.

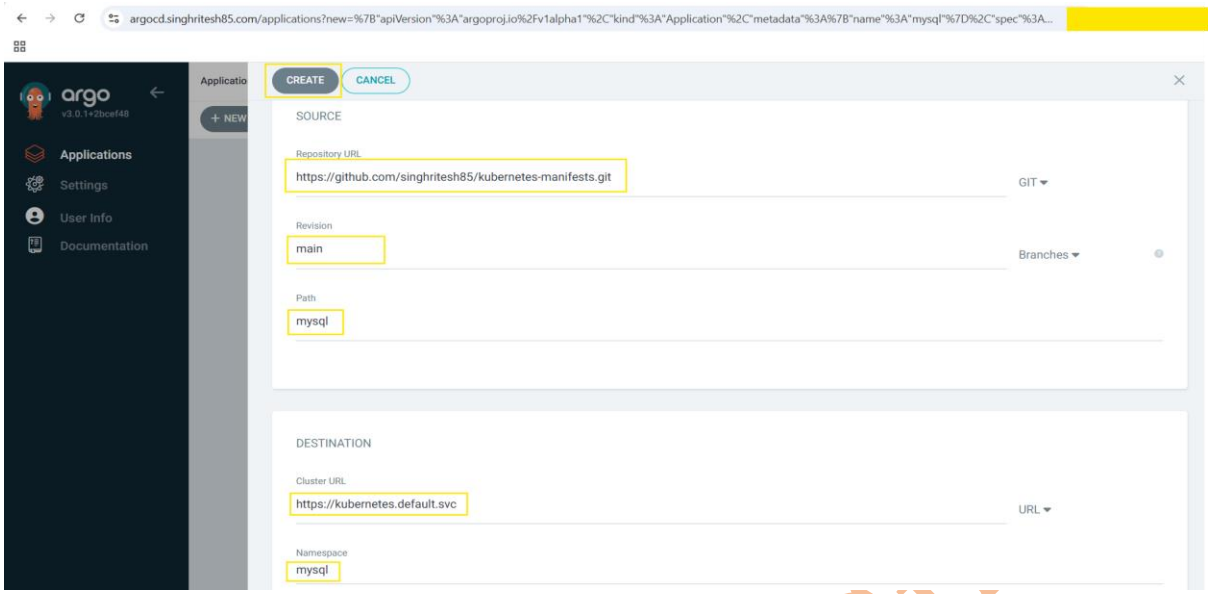
```
[jenkins@ ~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Module-1: Manual Approach

This project is basically a multibranch pipeline project in which only dev and stage environment exists. I had created mysql pod in dev and stage environment using the ArgoCD as shown in the screenshot attached below.

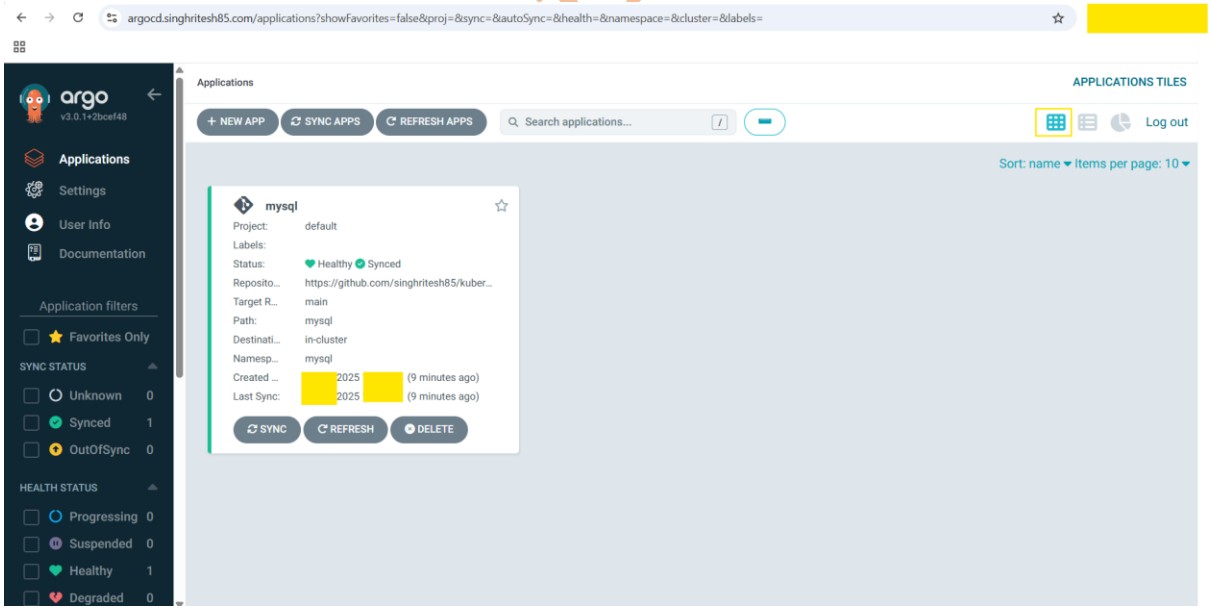


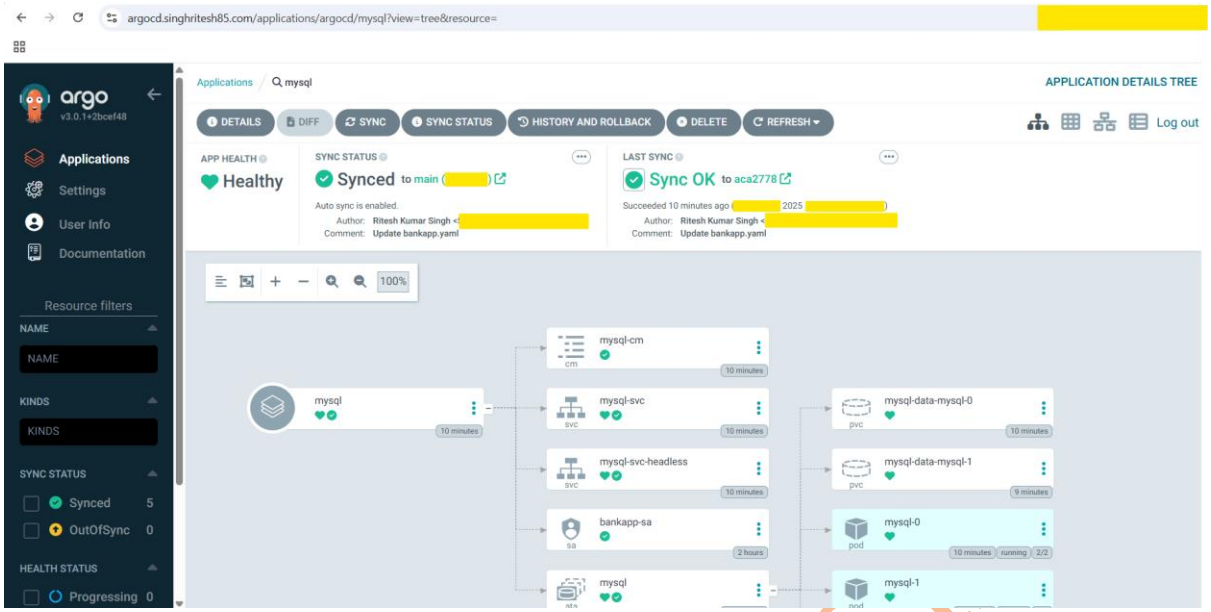




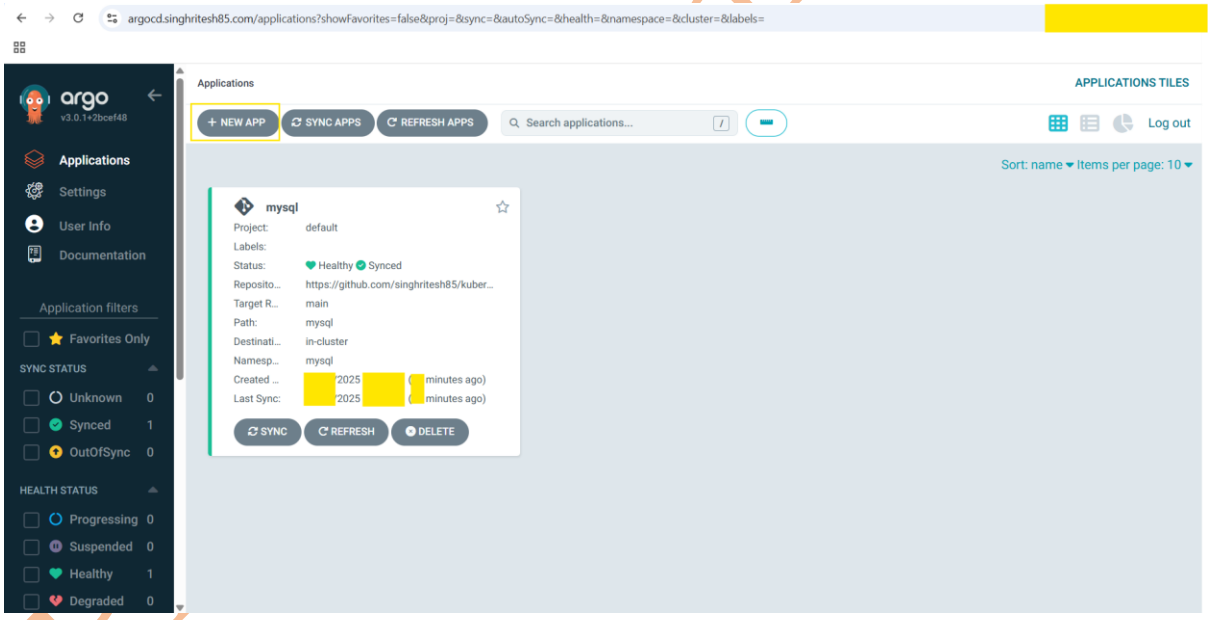
After creation of the Application mysql as shown in screenshot attached below the mysql pods had been created on dev cluster (EKS Cluster) as shown in the screenshot attached below.

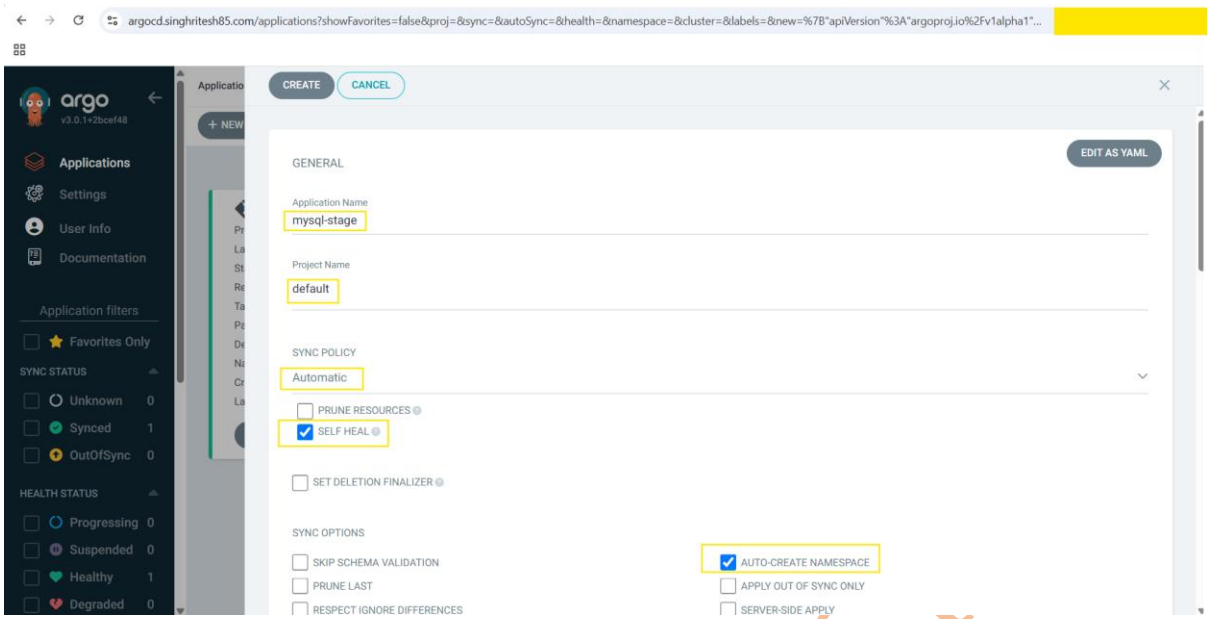
```
[jenkins@ ~]$ kubectl get pods -n mysql
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   2/2     Running   0
mysql-1   2/2     Running   0
[jenkins@ ~]$ kubectl get svc -n mysql
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
mysql-svc            ClusterIP   10.  .41    <none>        3306/TCP
mysql-svc-headless   ClusterIP   None         <none>        3306/TCP
[jenkins@ ~]$ kubectl get pvc -n mysql
NAME                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-data-mysql-0   Bound    pvc-                                       1Gi        RWO            gp2            <unset>
mysql-data-mysql-1   Bound    pvc-                                       1Gi        RWO            gp2            <unset>
```

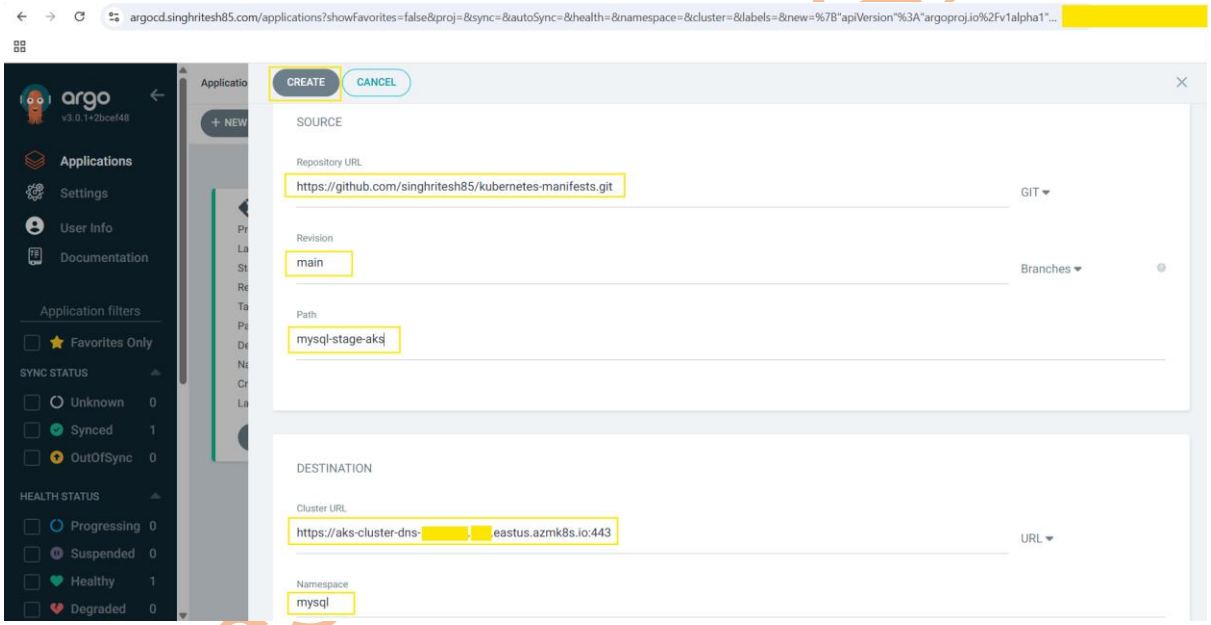




Then I had created **New App** in ArgoCD to deploy pods on AKS Cluster (Stage Environment) as shown in the screenshot attached below.

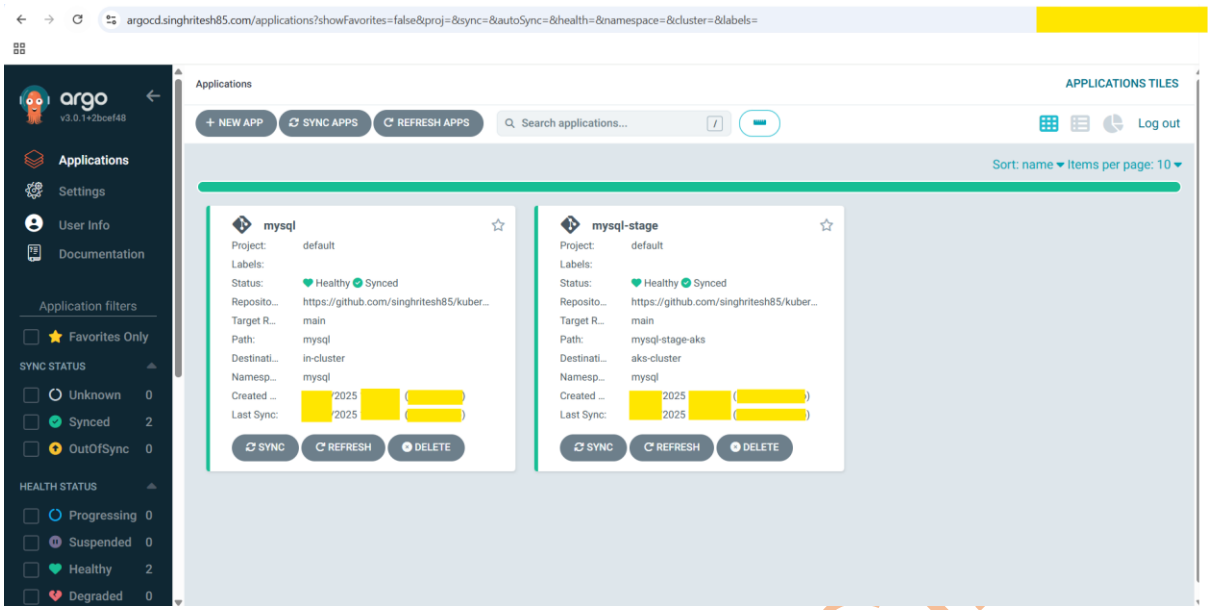






After creation of the Application **mysql-stage** as shown in screenshot attached below the mysql pods had been created on stage cluster (AKS Cluster) as shown in the screenshot attached below.

```
[jenkins@[redacted] ~]$ kubectl get pods -n mysql --watch --context=aks-cluster
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   2/2     Running   0          118s
mysql-1   2/2     Running   0          63s
^C[jenkins@[redacted] ~]$
[jenkins@[redacted] ~]$
[jenkins@[redacted] ~]$ kubectl get pods -n mysql --watch --context=aks-cluster
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   2/2     Running   0          2m3s
mysql-1   2/2     Running   0          68s
^C[jenkins@[redacted] ~]$ kubectl get svc -n mysql --watch --context=aks-cluster
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
mysql-svc           ClusterIP   10.[redacted].47   <none>        3306/TCP   2m13s
mysql-svc-headless  ClusterIP   None          <none>        3306/TCP   2m13s
^C[jenkins@[redacted] ~]$
[jenkins@[redacted] ~]$
[jenkins@[redacted] ~]$ kubectl get pvc -n mysql --watch --context=aks-cluster
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-data-mysql-0  Bound    pvc-[redacted]                             1Gi        RWO            managed-csi    <unset>                 2m21s
mysql-data-mysql-1  Bound    pvc-[redacted]                             1Gi        RWO            managed-csi    <unset>                 86s
```



To build the Bank Application code presented in the GitHub Repo https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git and created Docker Image then pushed the Docker Image to ECR (Elastic Container Registry). This Docker Image will be used to deploy pods in EKS and AKS Cluster.

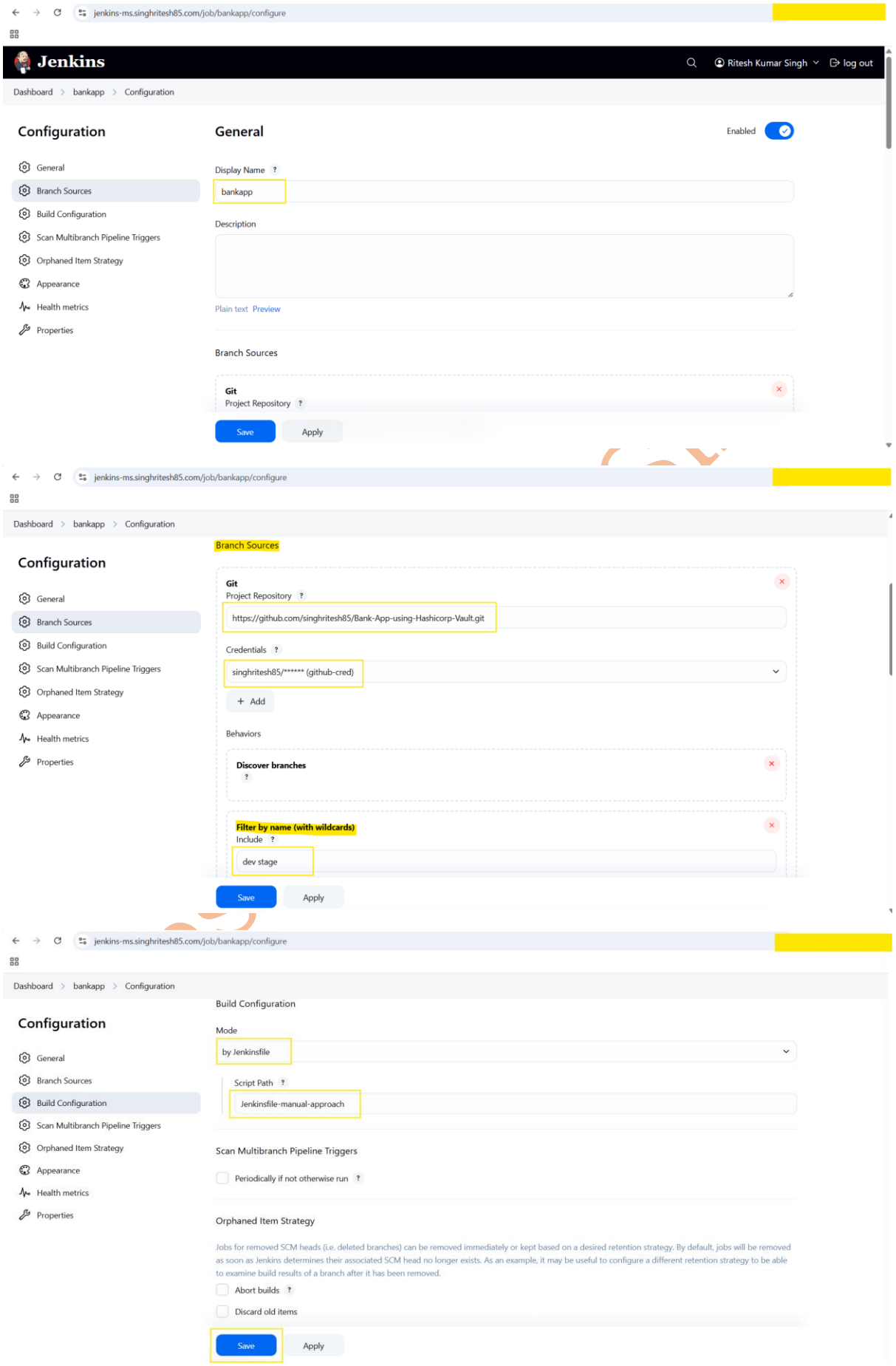
To access Docker Image from ECR (Elastic Container Registry) I created kubernetes secret in AKS Cluster as shown in the screenshot attached below.

kubectl create secret docker-registry regcred --docker-server=02XXXXXXXXX6.dkr.ecr.us-east-2.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password) --namespace=bankapp --context=aks-cluster

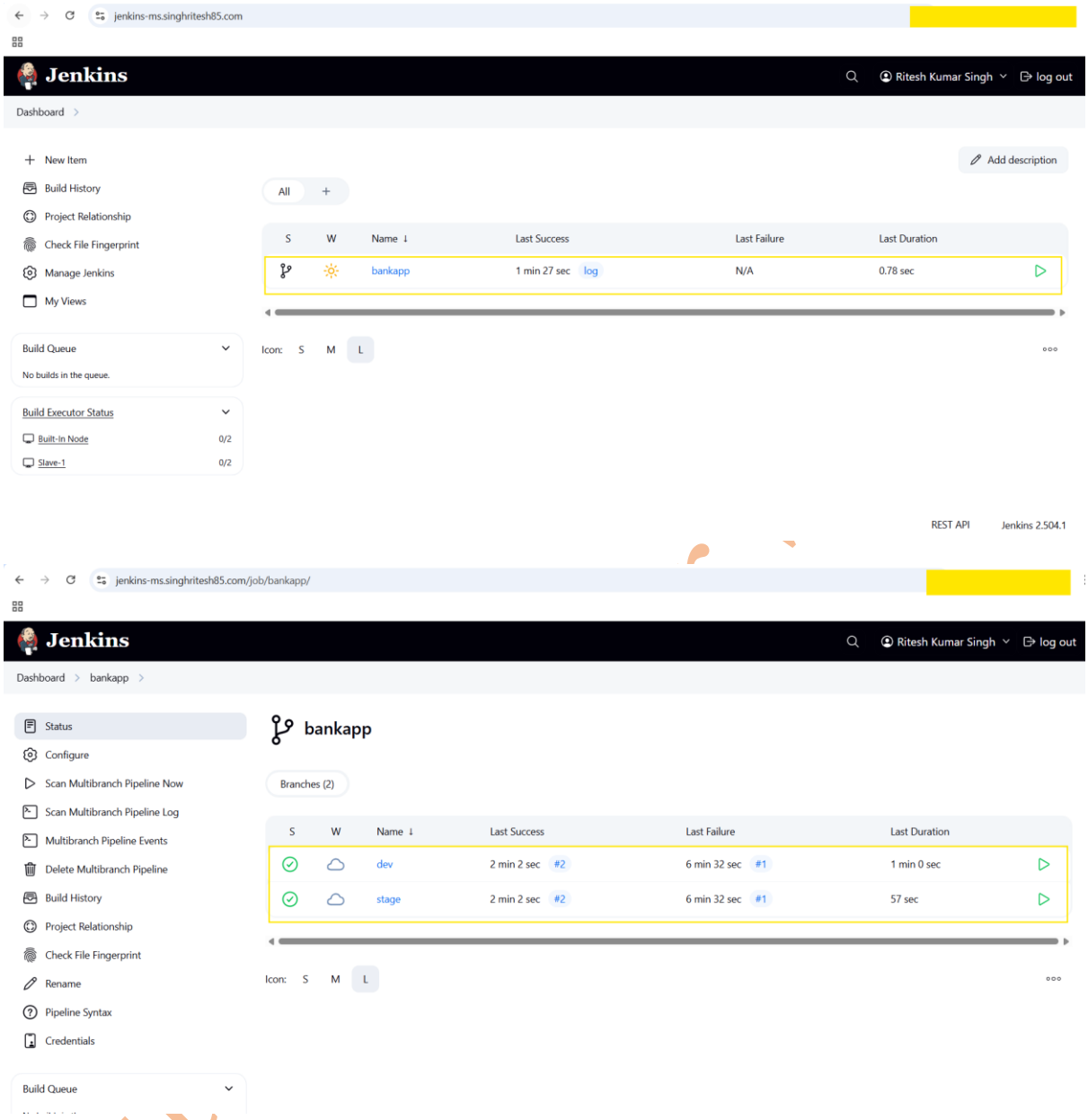
```
[root@                    ]# kubectl create secret docker-registry regcred --docker-server=02          6.dkr.ecr.us-east-2.amazonaws.com --docker-username=AWS
 --docker-password=$(aws ecr get-login-password) --namespace=bankapp --context=aks-cluster
secret/regcred created
```

However, I had Attached IAM Role to EKS Node Group and hence the EKS Nodes so that they can easily access the ECR Private Repository whenever needed.

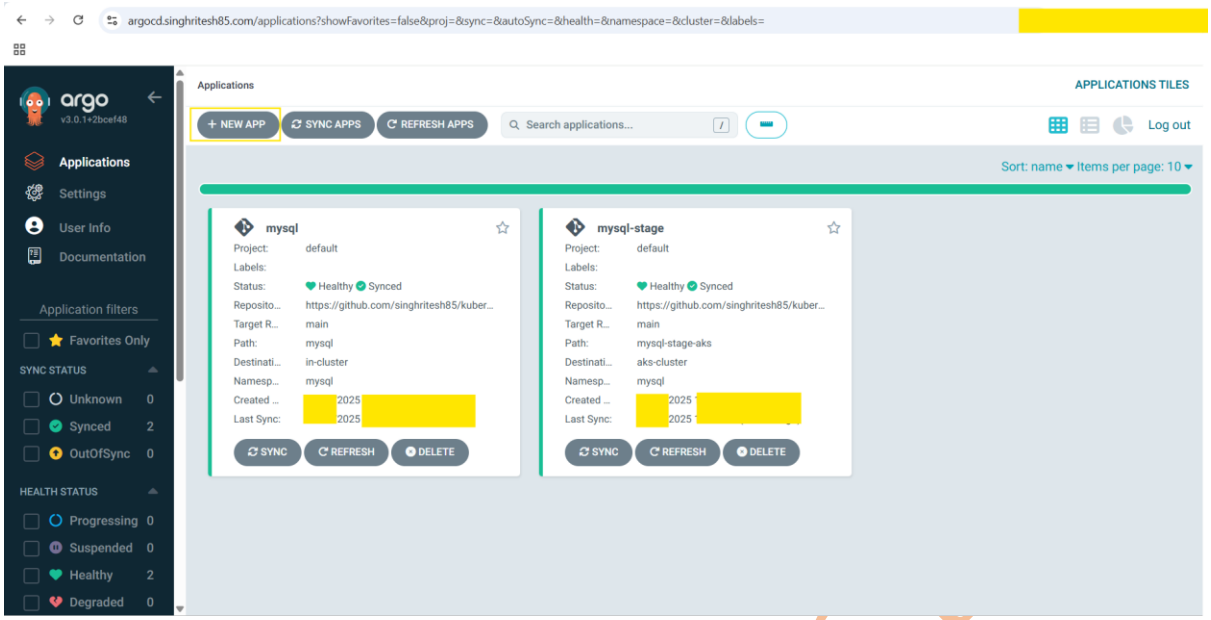
I had built the Bank Application code and created the Docker Image then pushed the Docker Image to ECR (Elastic Container Registry) using the Multibranch Jenkins Job as shown in the screenshot attached below.

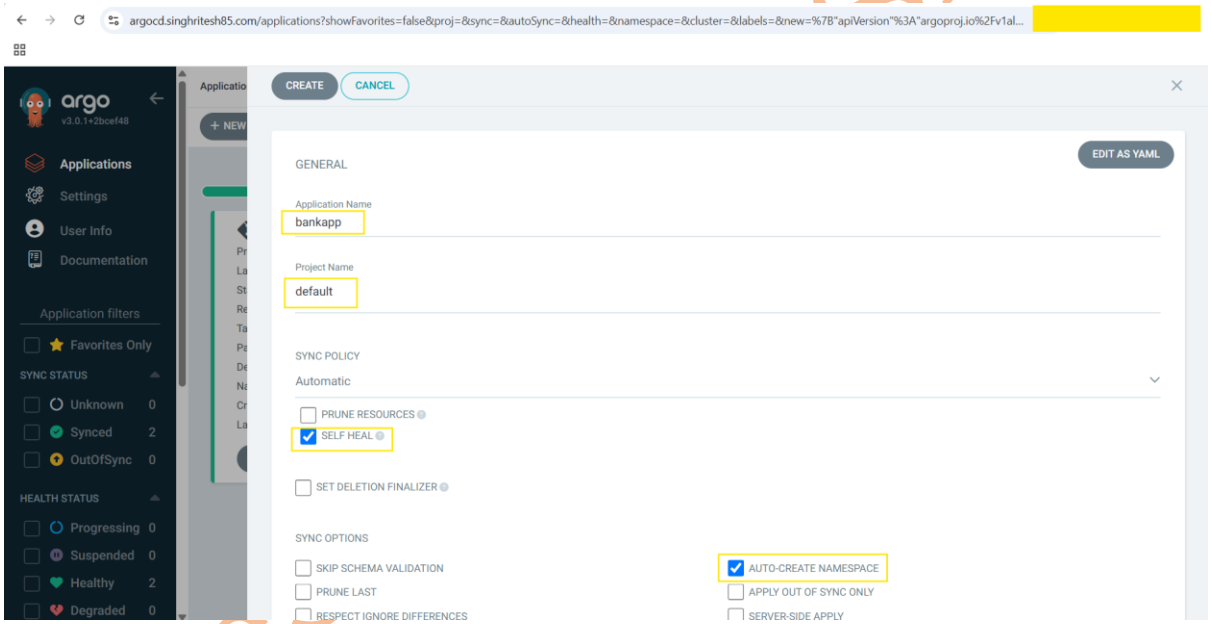

jenkins-ms.singhritesh85.com/job/bankapp/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > bankapp > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
bankapp
Description
Plain text Preview
Branch Sources
Git
Project Repository
Save
Apply
jenkins-ms.singhritesh85.com/job/bankapp/configure
Dashboard > bankapp > Configuration
Configuration
Branch Sources
Git
Project Repository
https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git
Credentials
singhritesh85/****** (github-cred)
+ Add
Behaviors
Discover branches
Filter by name (with wildcards)
Include
dev stage
Save
Apply
jenkins-ms.singhritesh85.com/job/bankapp/configure
Dashboard > bankapp > Configuration
Configuration
Build Configuration
Mode
by Jenkinsfile
Script Path
Jenkinsfile-manual-approach
Scan Multibranch Pipeline Triggers
Periodically if not otherwise run
Orphaned Item Strategy
Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.
Abort builds
Discard old items
Save
Apply

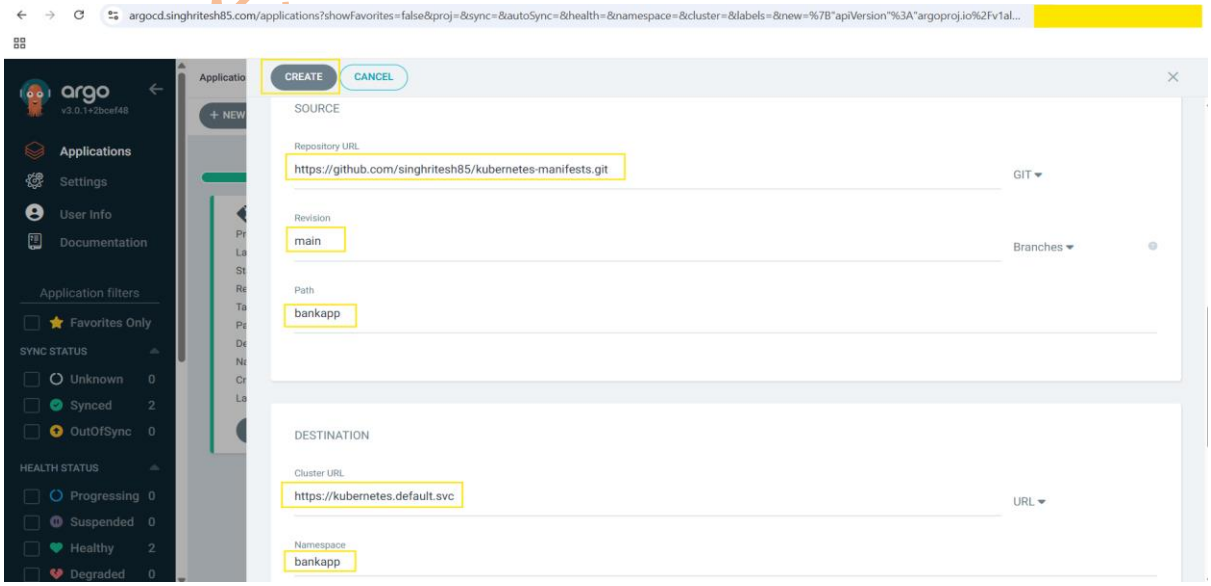
After doing this I had seen two Jenkins Job created with the name as that of the branch name in the multi branch Jenkins Job as shown in the screenshot attached below.

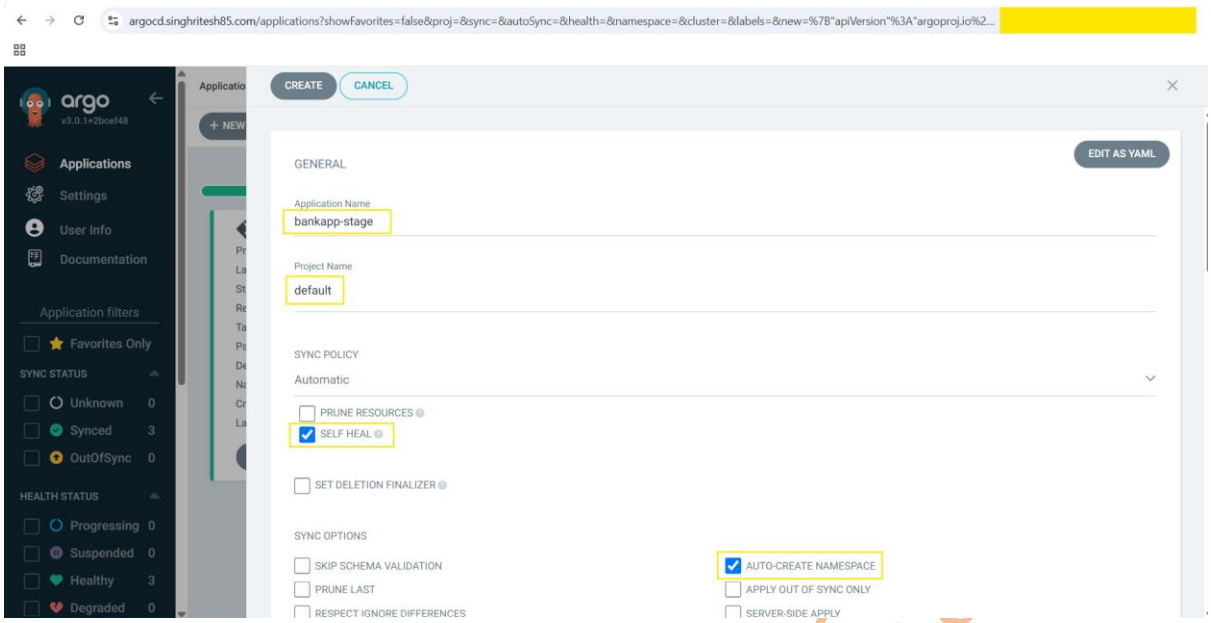


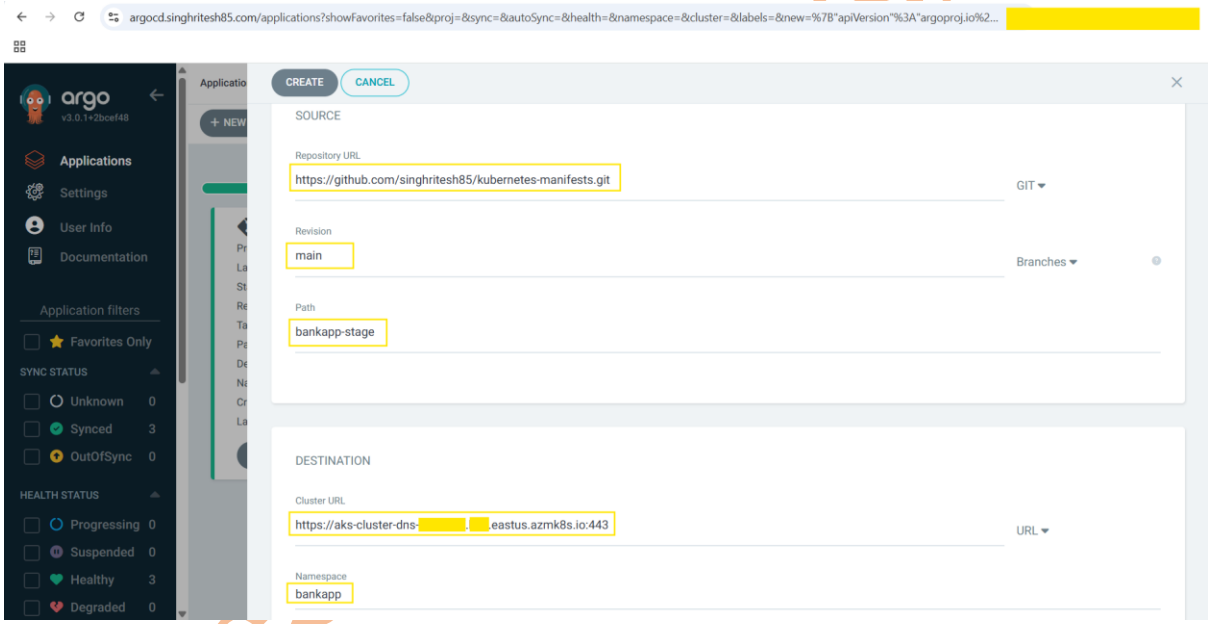
Now Docker Image is ready in ECR (Elastic Container Registry). I had created two ECRs bankapp-1 (for dev cluster) and bankapp-2 (for stage-cluster). Then I created the ArgoCD Application to deploy the pods on Kubernetes Clusters (in dev and stage environment) using these Docker Images as shown in the screenshot attached below.

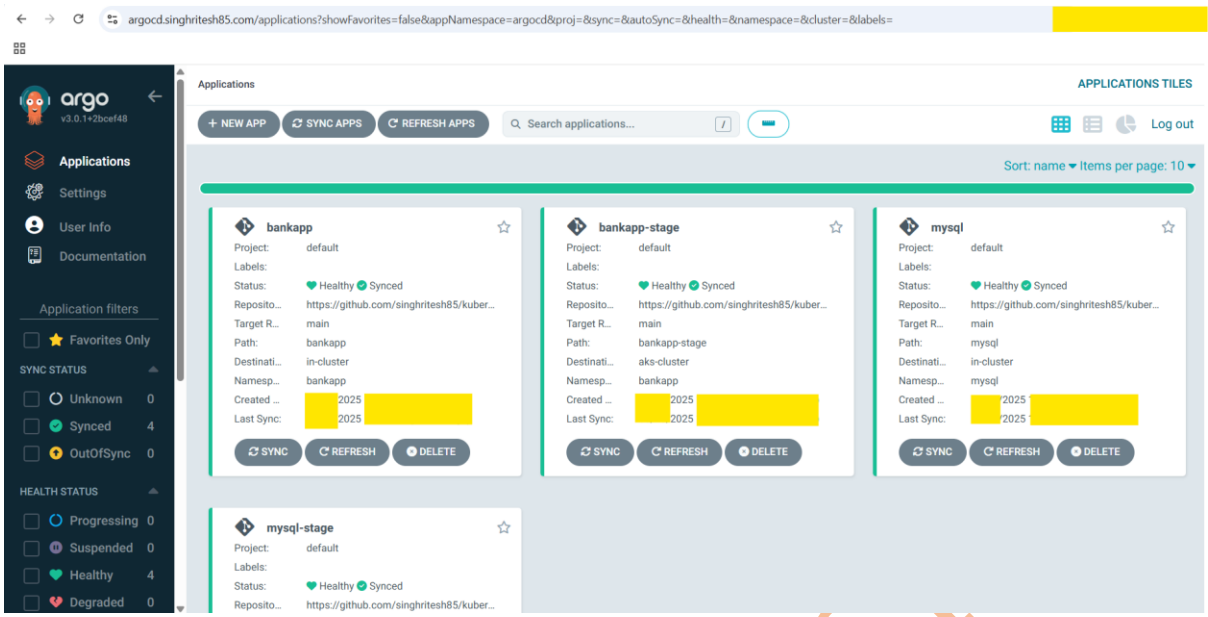

argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
Application filters
Favorites Only
SYNC STATUS
Unknown 0
Synced 2
OutOfSync 0
HEALTH STATUS
Progressing 0
Suspended 0
Healthy 2
Degraded 0
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
Sort: name
Items per page: 10
mysql
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/singhritesh85/kuber...
Target R... main
Path: mysql
Destinati... in-cluster
Namesp... mysql
Created ... 2025
Last Sync: 2025
SYNC
REFRESH
DELETE
mysql-stage
Project: default
Labels:
Status: Healthy Synced
Reposito... https://github.com/singhritesh85/kuber...
Target R... main
Path: mysql-stage-aks
Destinati... aks-cluster
Namesp... mysql
Created ... 2025
Last Sync: 2025
SYNC
REFRESH
DELETE


argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B"apiVersion"%3A"argoproj.io%2Fv1al...
CREATE
CANCEL
EDIT AS YAML
GENERAL
Application Name
bankapp
Project Name
default
SYNC POLICY
Automatic
PRUNE RESOURCES
SELF HEAL
SET DELETION FINALIZER
SYNC OPTIONS
SKIP SCHEMA VALIDATION
PRUNE LAST
RESPECT IGNORE DIFFERENCES
AUTO-CREATE NAMESPACE
APPLY OUT OF SYNC ONLY
SERVER-SIDE APPLY


argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B"apiVersion"%3A"argoproj.io%2Fv1al...
CREATE
CANCEL
SOURCE
Repository URL
https://github.com/singhritesh85/kubernetes-manifests.git
GIT
Revision
main
Branches
Path
bankapp
DESTINATION
Cluster URL
https://kubernetes.default.svc
URL
Namespace
bankapp


argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B"apiVersion"%3A"argoproj.io%2...
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
Application filters
Favorites Only
SYNC STATUS
Unknown 0
Synced 3
OutOfSync 0
HEALTH STATUS
Progressing 0
Suspended 0
Healthy 3
Degraded 0
CREATE
CANCEL
EDIT AS YAML
GENERAL
Application Name
bankapp-stage
Project Name
default
SYNC POLICY
Automatic
PRUNE RESOURCES
SELF HEAL
SET DELETION FINALIZER
SYNC OPTIONS
SKIP SCHEMA VALIDATION
PRUNE LAST
RESPECT IGNORE DIFFERENCES
AUTO-CREATE NAMESPACE
APPLY OUT OF SYNC ONLY
SERVER-SIDE APPLY


argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B"apiVersion"%3A"argoproj.io%2...
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
Application filters
Favorites Only
SYNC STATUS
Unknown 0
Synced 3
OutOfSync 0
HEALTH STATUS
Progressing 0
Suspended 0
Healthy 3
Degraded 0
CREATE
CANCEL
SOURCE
Repository URL
https://github.com/singhritesh85/kubernetes-manifests.git
GIT
Revision
main
Branches
Path
bankapp-stage
DESTINATION
Cluster URL
https://aks-cluster-dns-[redacted].eastus.azmk8s.io:443
URL
Namespace
bankapp



Finally, bankapp pods had been created in the EKS cluster (dev cluster) and AKS Cluster (Stage Cluster) as shown in the screenshot attached below.

```
[jenkins@ ~]$ kubectl get pods -n bankapp --watch --context=eks-demo-cluster-dev
NAME                      READY   STATUS    RESTARTS   AGE
bankapp-                  1/1     Running   0          
bankapp-                  1/1     Running   0          
^C[jenkins@ ~]$
[jenkins@ ~]$ kubectl get pods -n bankapp --watch --context=aks-cluster
NAME                      READY   STATUS    RESTARTS   AGE
bankapp-                  1/1     Running   0          
bankapp-                  1/1     Running   0          
```

```
[jenkins@ ~]$ kubectl get svc -n bankapp --watch --context=eks-demo-cluster-dev
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP   10.    .182     <none>        80/TCP    
[jenkins@ ~]$
[jenkins@ ~]$ kubectl get svc -n bankapp --watch --context=aks-cluster
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP   10.    .229    <none>        80/TCP    
```

Then I created Ingress Rule in dev cluster and stage cluster to access the Bank Application and did the entry for DNS Name and External IP Address of LoadBalancer in the Azure DNS Zone Record Set of Type CNAME and A Type respectively as shown in the screenshot attached below.

I had created the kubernetes secret in AKS Cluster for the Ingress Rule as shown in the screenshot attached below.

kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster

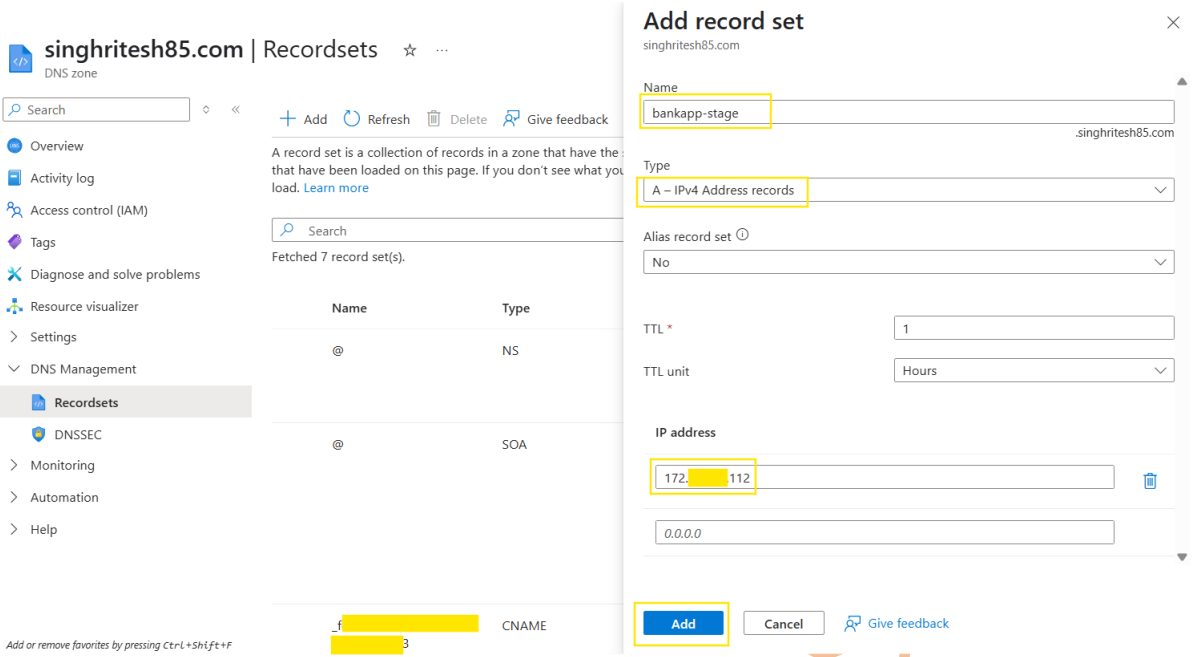
```
[root@ ~]# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster
secret/ingress-tls created
```

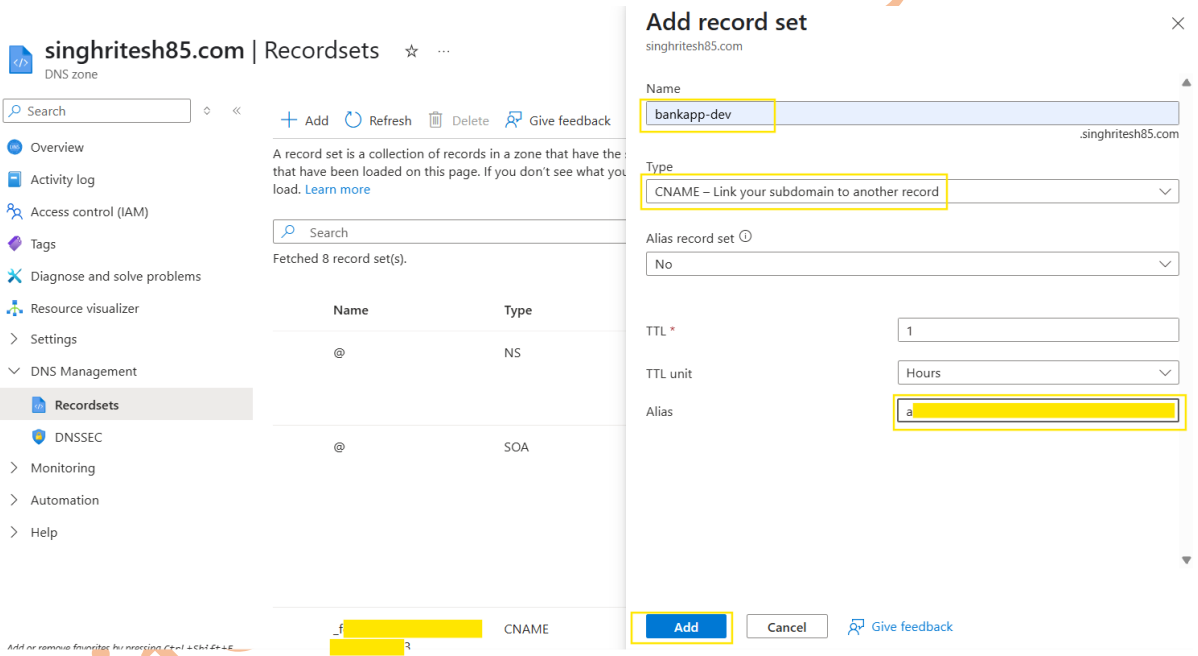
```
[root@ ~]# kubectl apply -f ingress-rule-stage.yaml --context=aks-cluster
ingress.networking.k8s.io/bankapp-ingress-stage created
[root@ ~]# kubectl apply -f ingress-rule-dev.yaml --context=eks-demo-cluster-dev
ingress.networking.k8s.io/bankapp-ingress-dev created
```

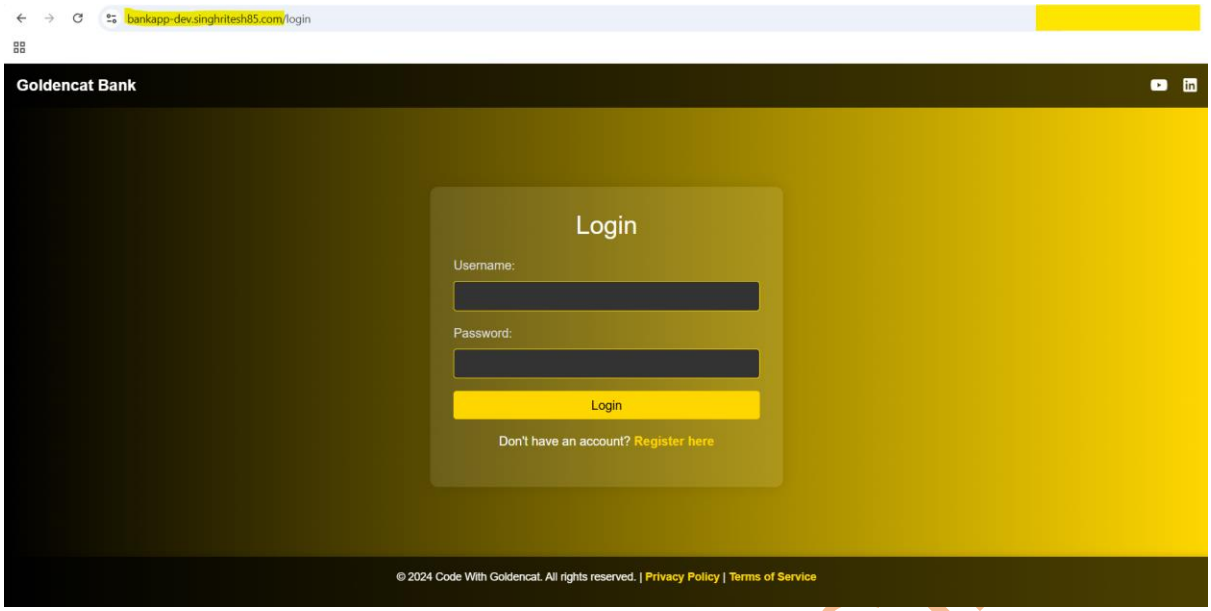
```
[root@ ~]# cat ingress-rule-dev.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress-dev
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: bankapp-dev.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80
```

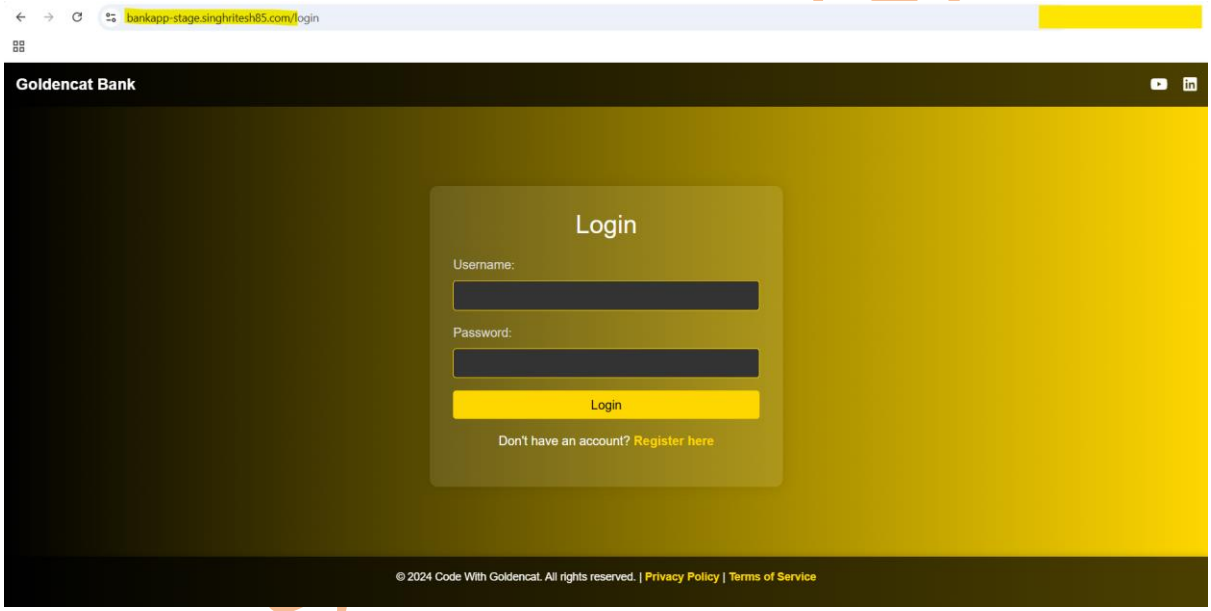
```
[root@ ~]# cat ingress-rule-stage.yaml
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress-stage
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - bankapp-stage.singhritesh85.com
    secretName: ingress-tls
  rules:
  - host: bankapp-stage.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80
```

```
[root@ ~]# kubectl get ing -A --watch --context=aks-cluster
NAMESPACE   NAME                CLASS   HOSTS                          ADDRESS          PORTS     AGE
bankapp     bankapp-ingress-stage   nginx   bankapp-stage.singhritesh85.com   172.  .112   80, 443
vault       vault-ingress           nginx   vault-stage.singhritesh85.com     172.  .112   80, 443
[root@ ~]#
[root@ ~]#  kubectl get ing -A --watch --context=eks-demo-cluster-dev
NAMESPACE   NAME                CLASS   HOSTS                      ADDRESS                                                                  PORTS   AG
E
argocd      minimal-ingress     nginx   argocd.singhritesh85.com   a                                                .us-east-2.elb.amazonaws.com   80
14m
bankapp     bankapp-ingress-dev nginx   bankapp-dev.singhritesh85.com  a                                                .us-east-2.elb.amazonaws.com   80
s
vault       vault-ingress       nginx   vault-dev.singhritesh85.com    a                                                .us-east-2.elb.amazonaws.com   80
```


singhritesh85.com | Recordsets
DNS zone
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
DNS Management
Recordsets
DNSSEC
Monitoring
Automation
Help
Add
Refresh
Delete
Give feedback
A record set is a collection of records in a zone that have the
that have been loaded on this page. If you don't see what you
load. Learn more
Search
Fetched 7 record set(s).
Name
Type
@
NS
@
SOA
CNAME
Add or remove favorites by pressing Ctrl+Shift+F
Add record set
singhritesh85.com
Name
bankapp-stage
.singhritesh85.com
Type
A – IPv4 Address records
Alias record set
No
TTL *
1
TTL unit
Hours
IP address
172. 112
0.0.0.0
Add
Cancel
Give feedback


singhritesh85.com | Recordsets
DNS zone
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
DNS Management
Recordsets
DNSSEC
Monitoring
Automation
Help
Add
Refresh
Delete
Give feedback
A record set is a collection of records in a zone that have the
that have been loaded on this page. If you don't see what you
load. Learn more
Search
Fetched 8 record set(s).
Name
Type
@
NS
@
SOA
CNAME
Add record set
singhritesh85.com
Name
bankapp-dev
.singhritesh85.com
Type
CNAME – Link your subdomain to another record
Alias record set
No
TTL *
1
TTL unit
Hours
Alias
a
Add
Cancel
Give feedback

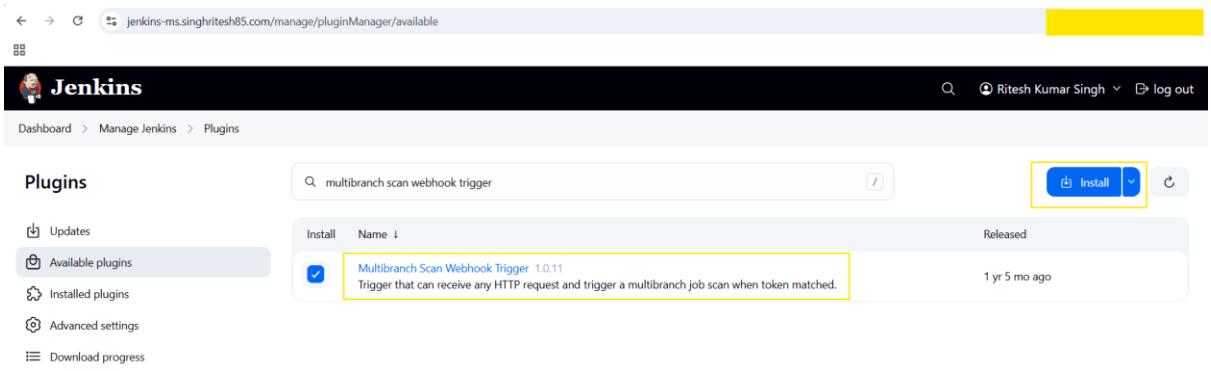


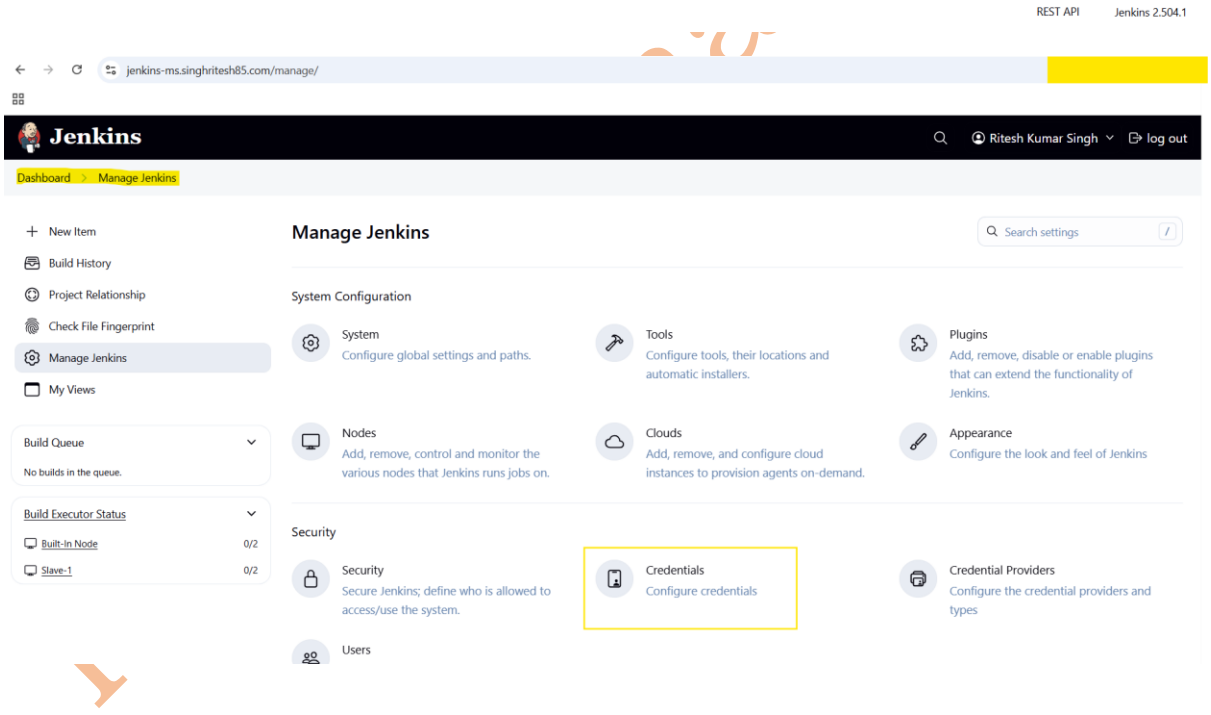


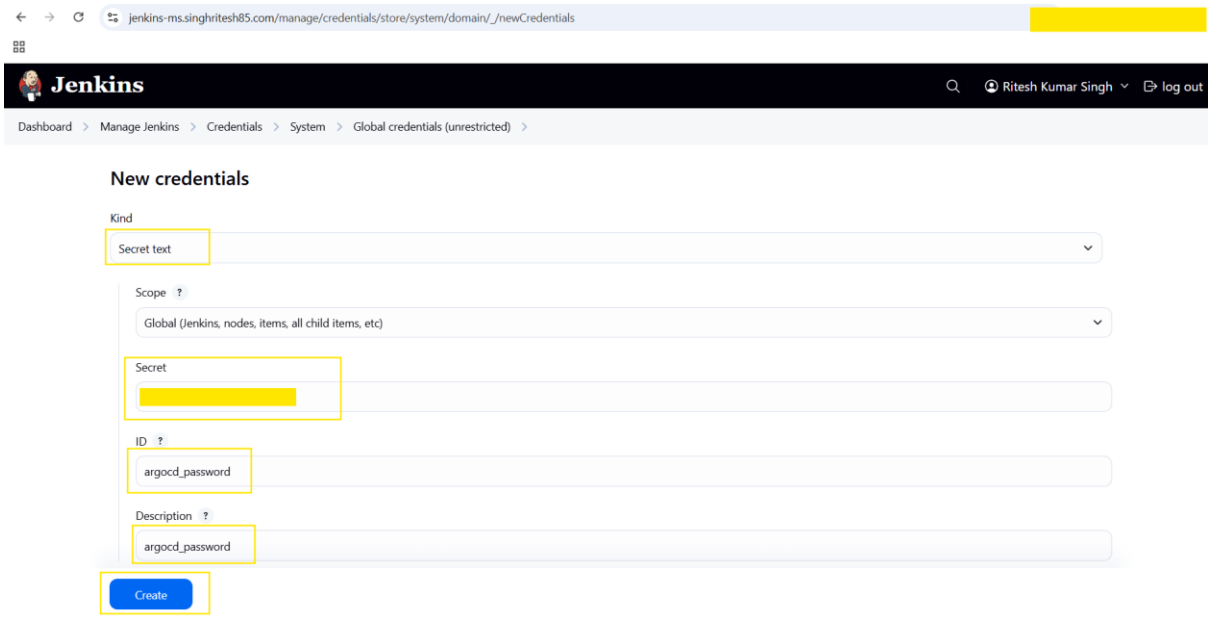
In Manual Approach the disadvantage is for a new Release, manually you need to Run the Jenkins Job then delete the corresponding Application in the ArgoCD then recreate it manually.

Module-2: Automation Approach

In Automation Approach I triggered the Jenkins Job using the webhook and for that I installed the Jenkins Plugin **multibranch scan webhook trigger** and then created a Jenkins credentials of kind secret text **argocd_password** as shown in the screenshot attached below.

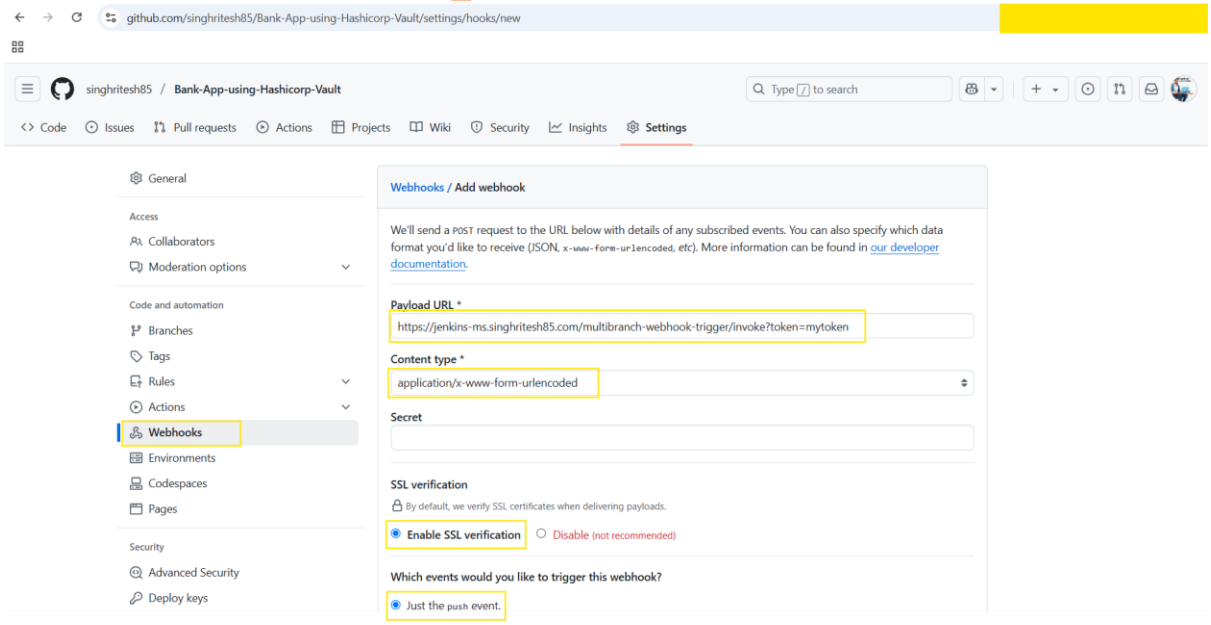


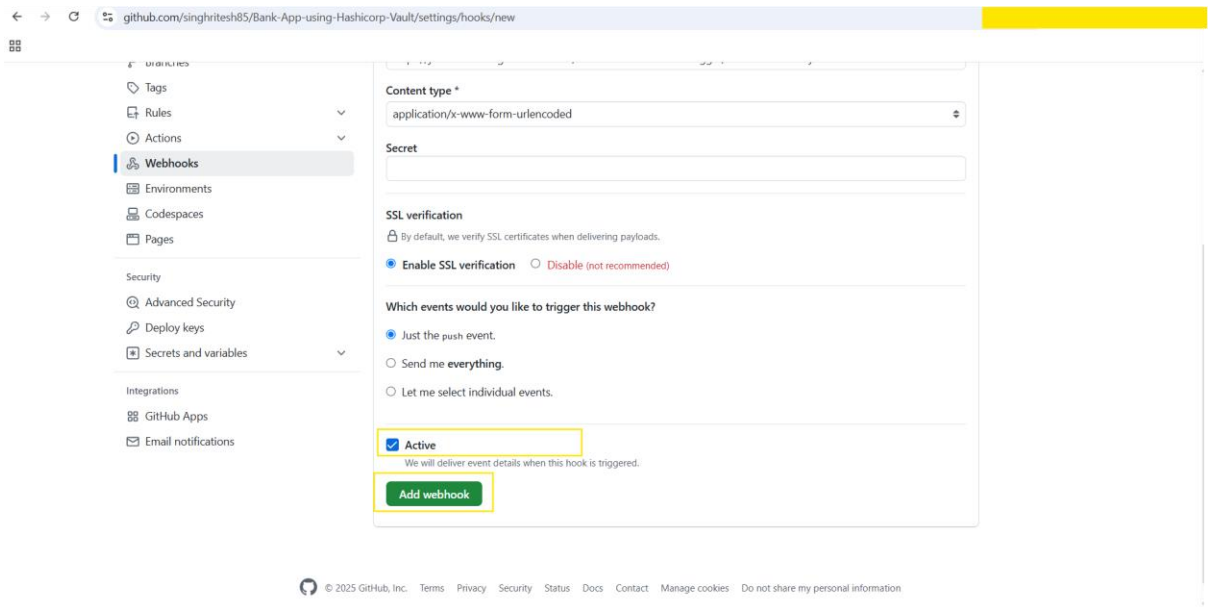


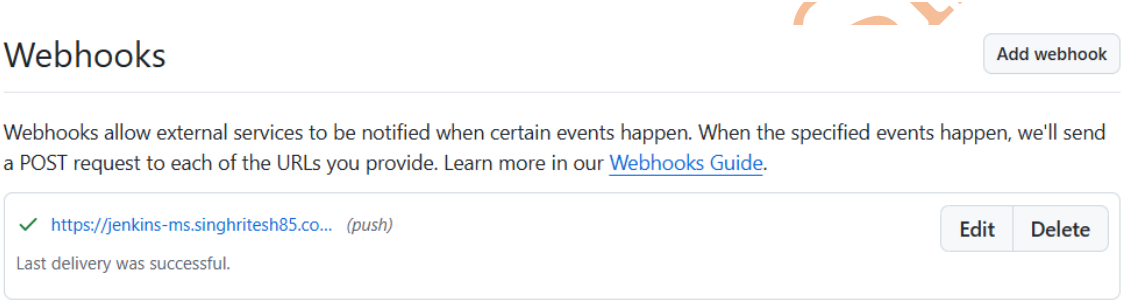


In Manual Approach (Module-1) I had already shown how to create the Jenkins credentials for jenkins slave and GitHub Account which was jenkins-cred and github-cred respectively. In the similar way create the Jenkins Credentials here. Create the kubernetes secrets **ingress-tls**, **vault-tls-secret** and kubernetes secret **regcred** to pull docker image present in ECR (Elastic Container Registry) **bankapp-2** for stage cluster (AKS Cluster) which I already discussed in Module-1 (Manual Approach).

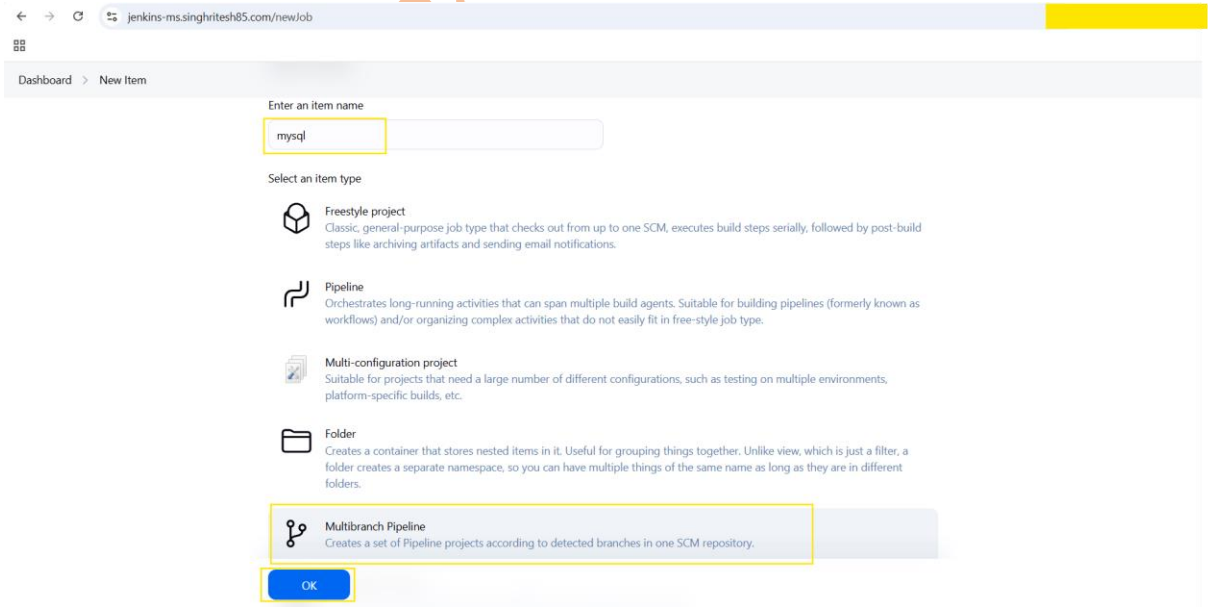
Then go to Github Repo for your project in this case I use the GitHub Repository https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git for source code, and then go to Repository Settings > Webhooks and provide the Payload URL **https://jenkins-ms.singhritesh85.com/multibranch-webhook-trigger/invoke?token=mytoken** as shown in the screenshot attached below.

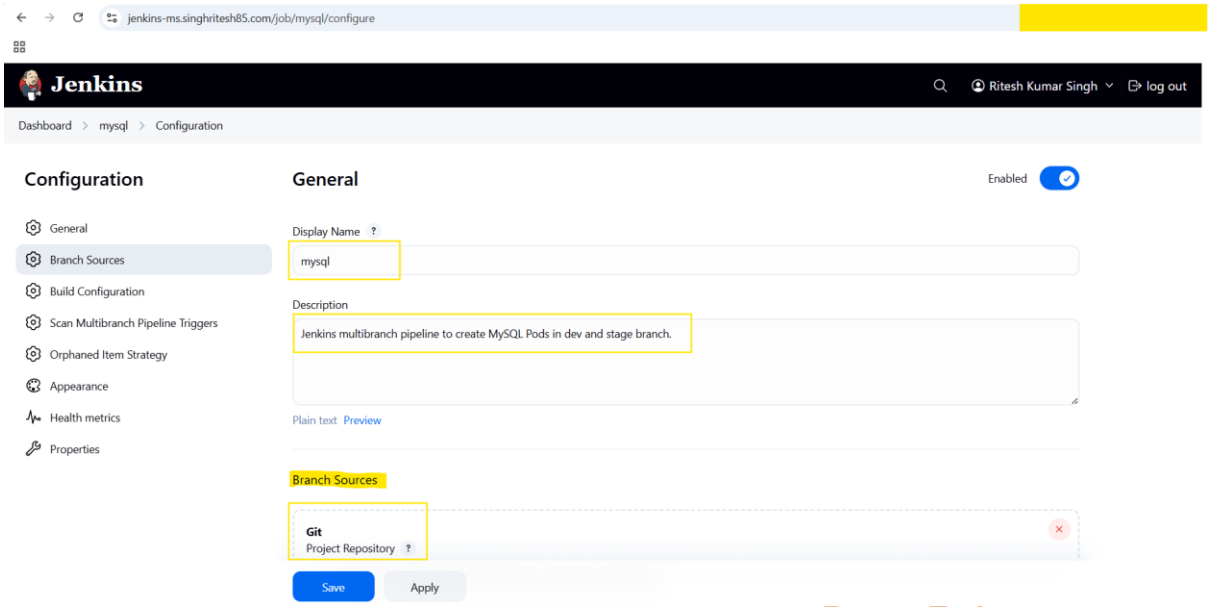


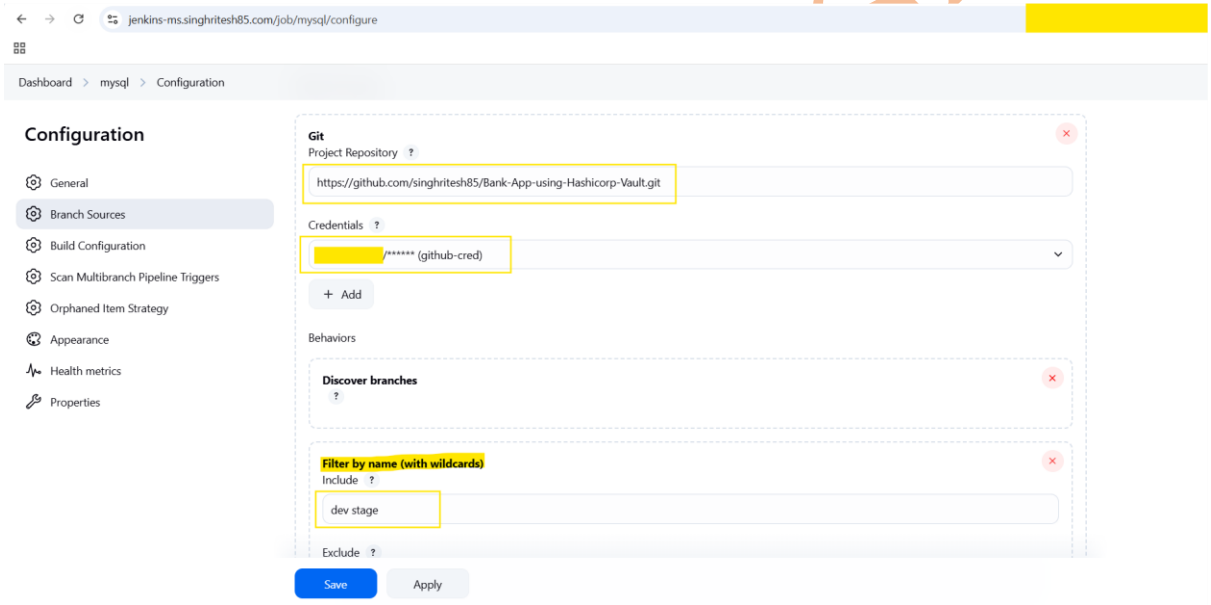


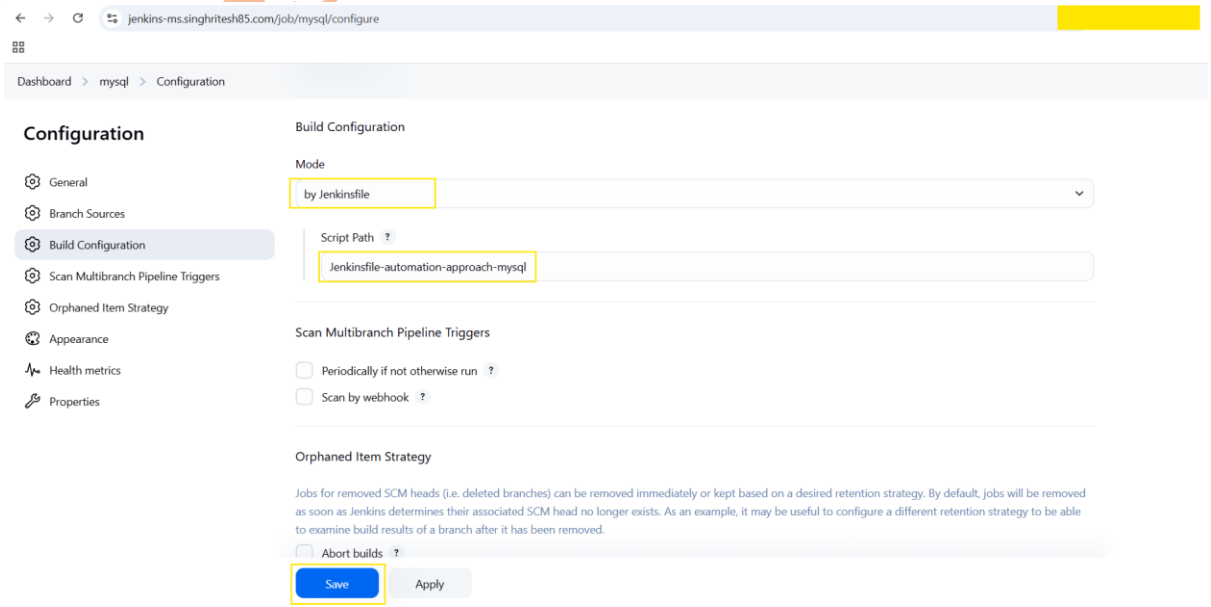


Then create the multibranch pipeline for mysql and then for bankapp (with webhook) as shown in the screenshot attached below.

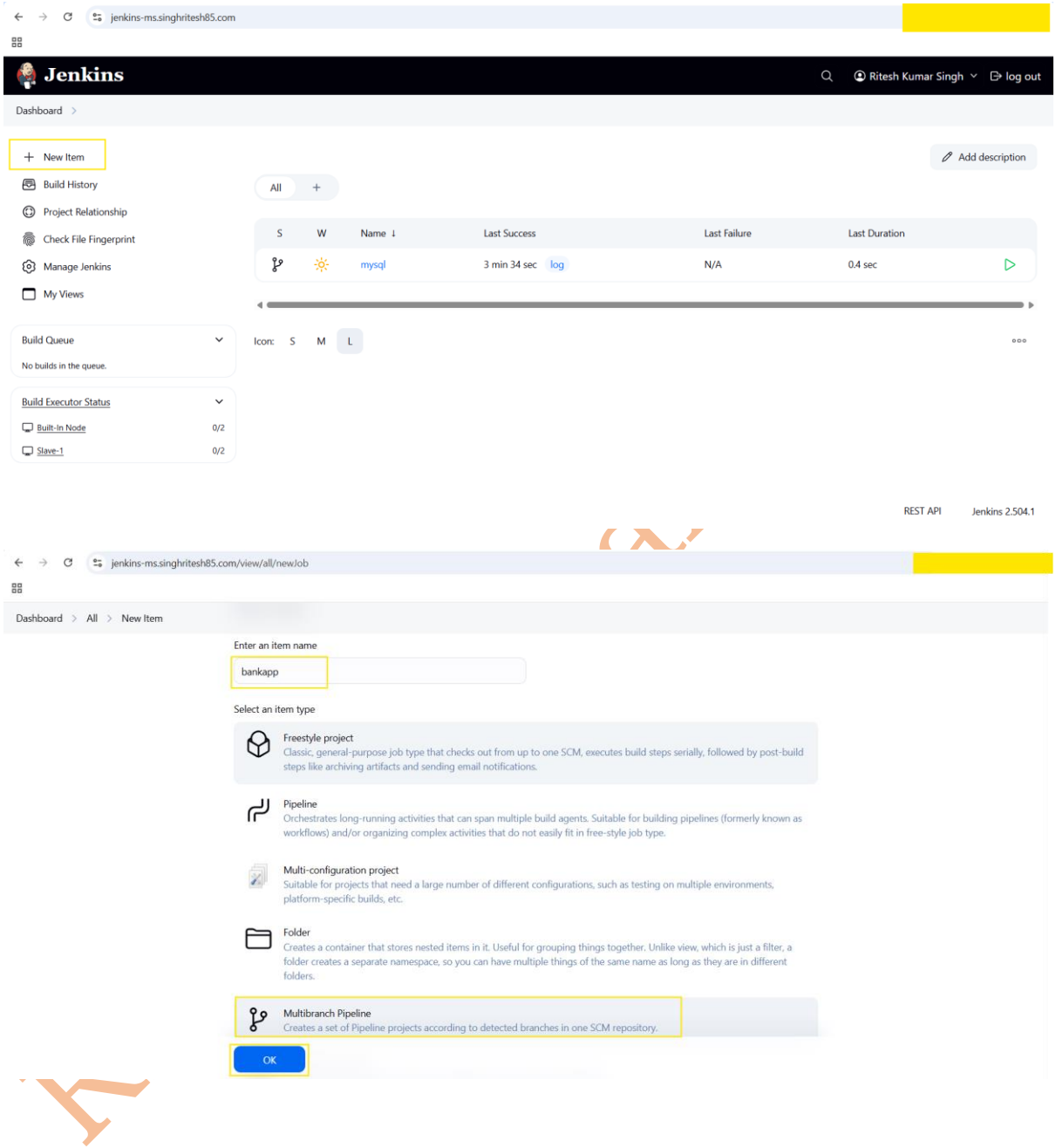


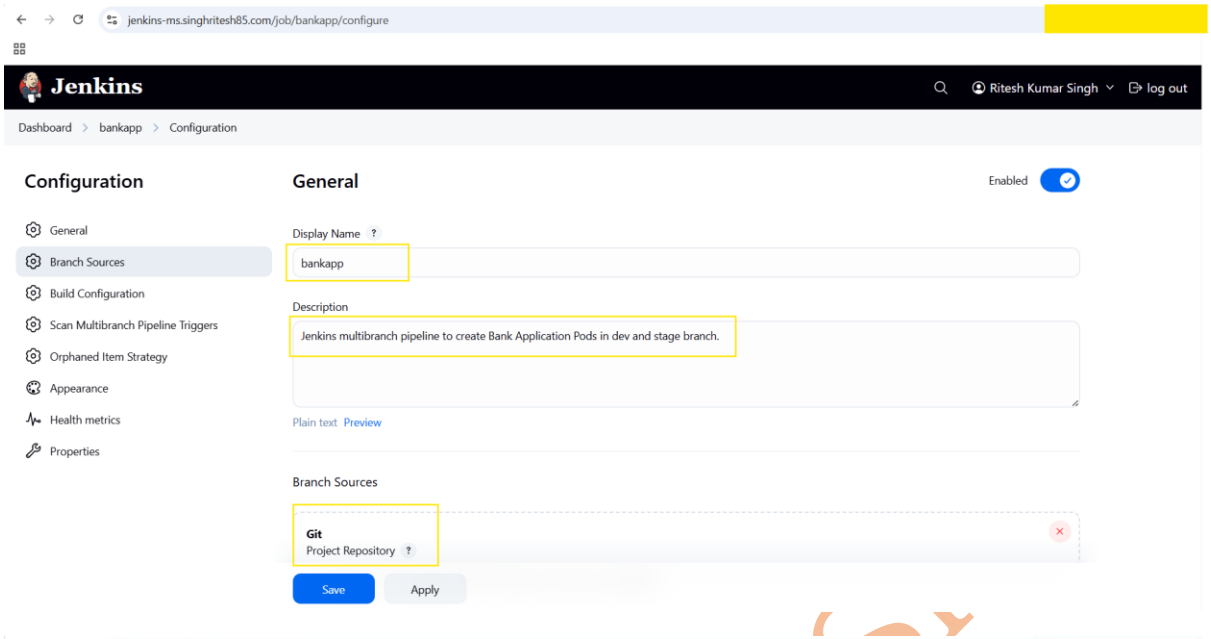

jenkins-ms.singhritesh85.com/job/mysql/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > mysql > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
mysql
Description
Jenkins multibranch pipeline to create MySQL Pods in dev and stage branch.
Plain text Preview
Branch Sources
Git
Project Repository
Save
Apply

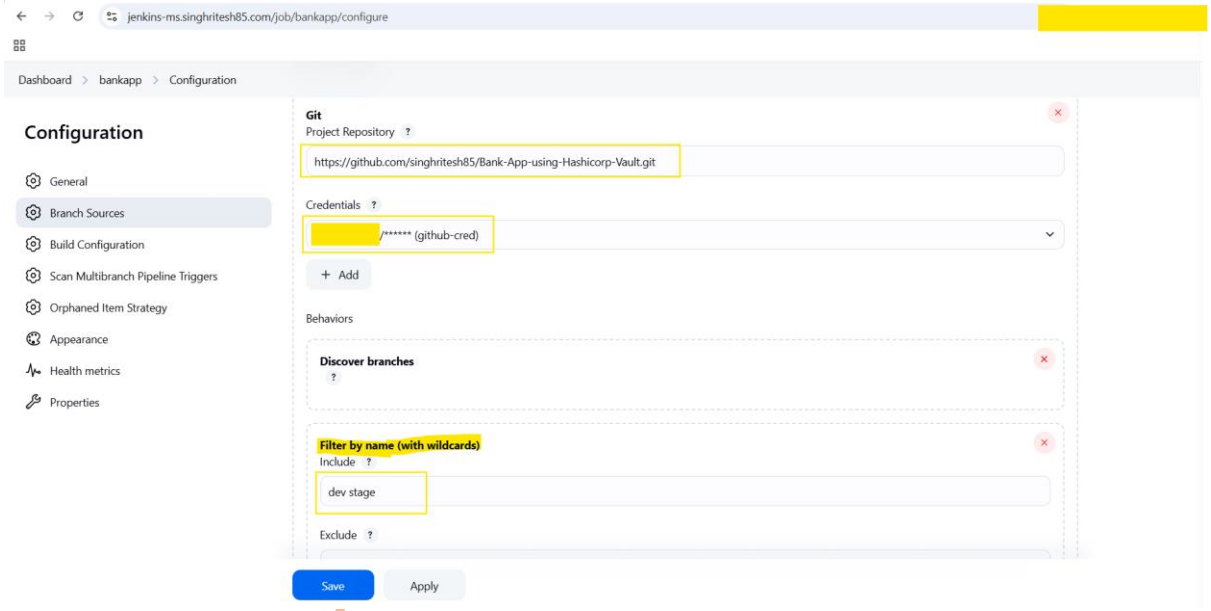

jenkins-ms.singhritesh85.com/job/mysql/configure
Dashboard > mysql > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
Git
Project Repository
https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git
Credentials
/****** (github-cred)
+ Add
Behaviors
Discover branches
Filter by name (with wildcards)
Include
dev stage
Exclude
Save
Apply

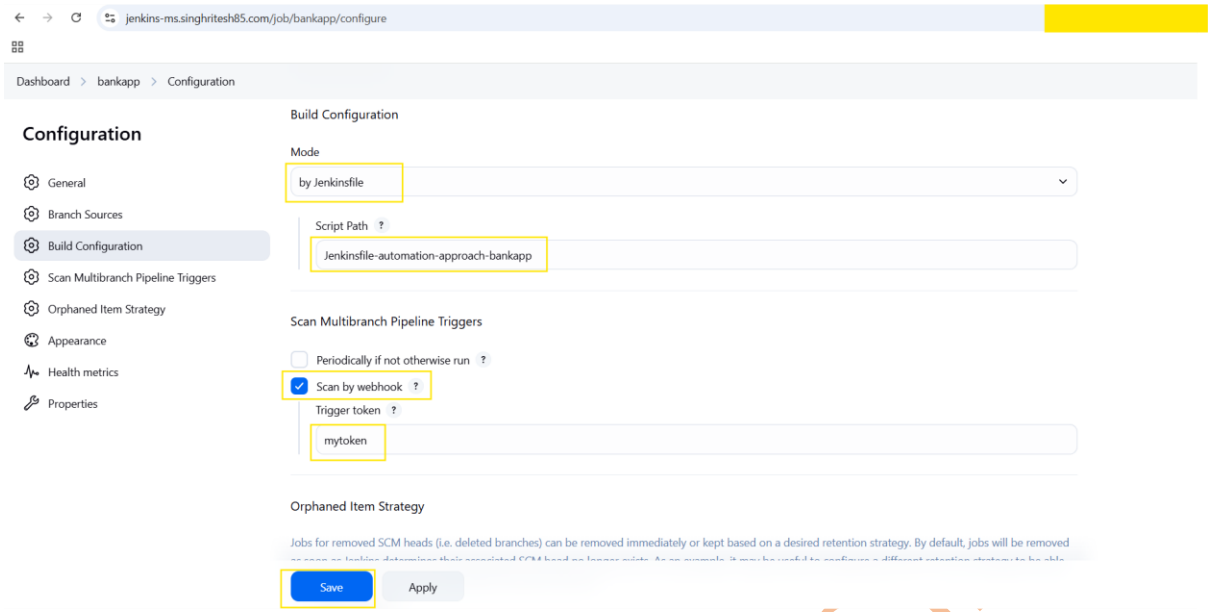

jenkins-ms.singhritesh85.com/job/mysql/configure
Dashboard > mysql > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
Build Configuration
Mode
by Jenkinsfile
Script Path
Jenkinsfile-automation-approach-mysql
Scan Multibranch Pipeline Triggers
Periodically if not otherwise run
Scan by webhook
Orphaned Item Strategy
Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.
Abort builds
Save
Apply

I did not apply the webhook to deploy the MySQL 8 Pods but I used the webhook to deploy bankapp application pods as shown in the screenshot attached below. Below screenshot shows how I had created the Multibranch Pipeline for Jenkins to create bankapp application pods in EKS and AKS Clusters.

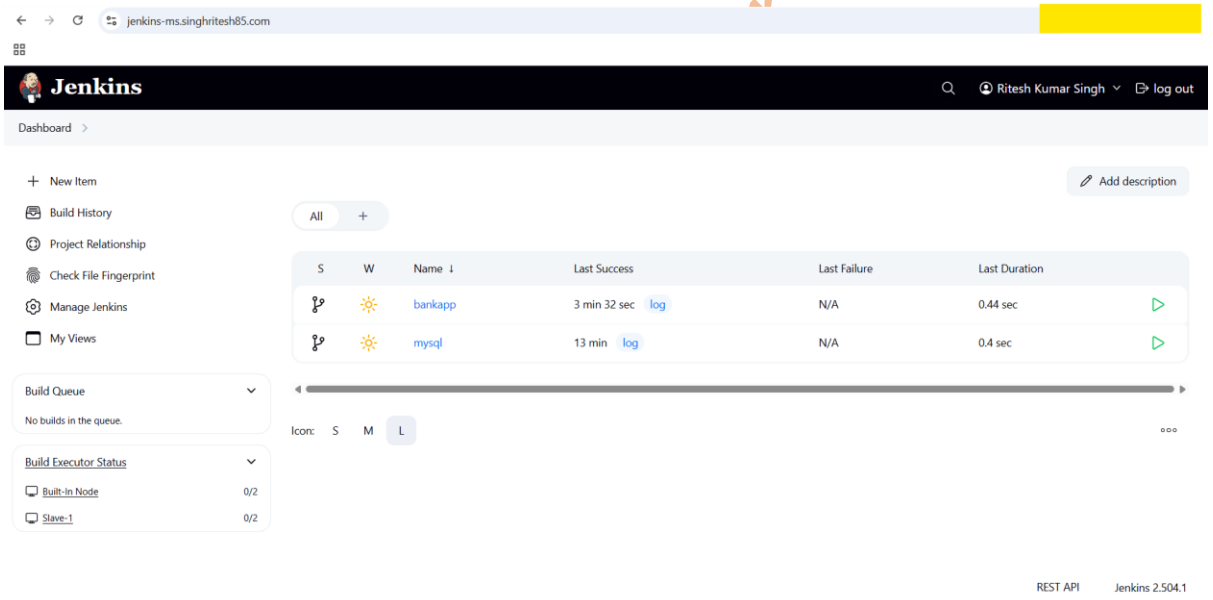


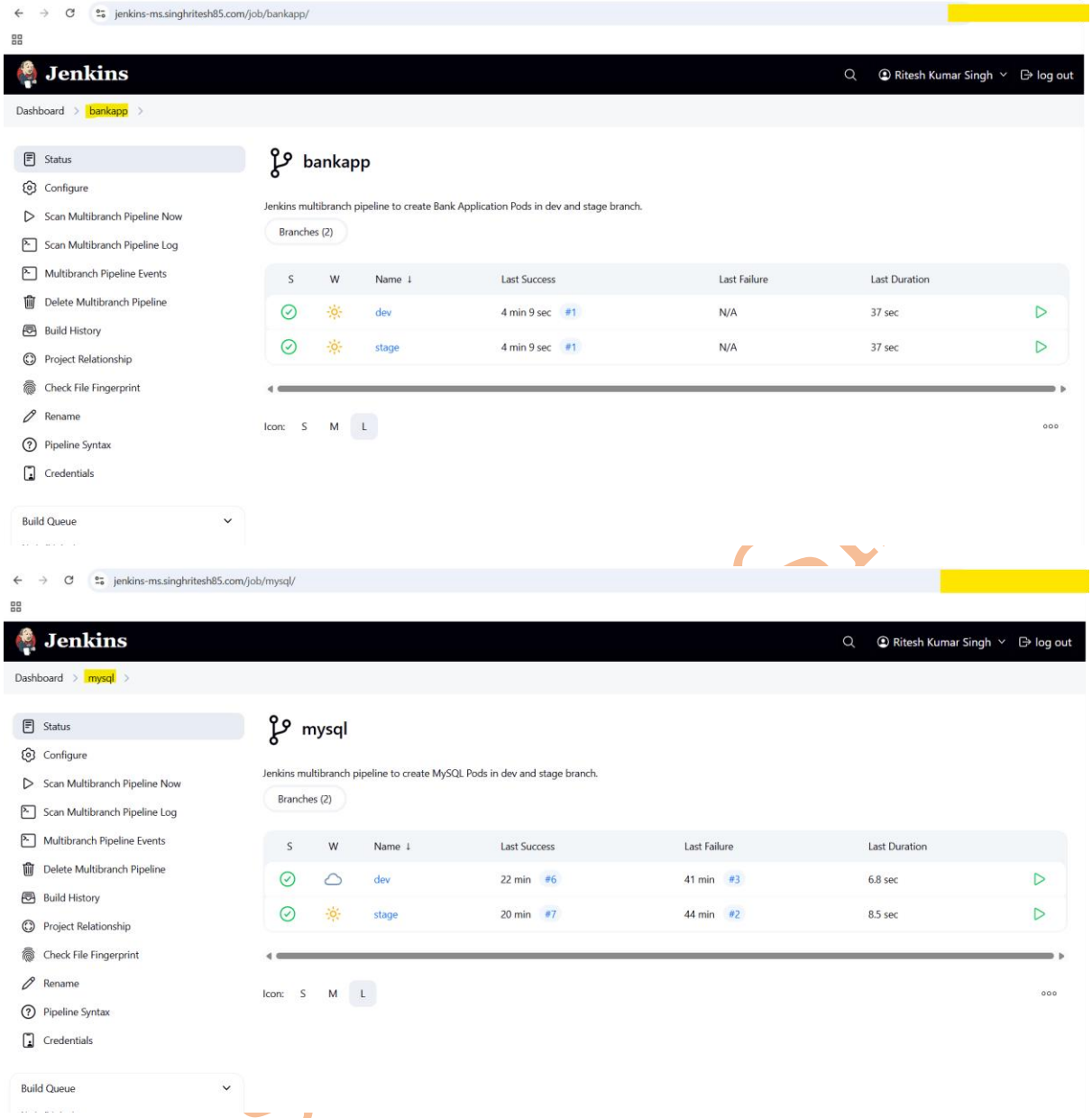

jenkins-ms.singhritesh85.com/job/bankapp/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > bankapp > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
bankapp
Description
Jenkins multibranch pipeline to create Bank Application Pods in dev and stage branch.
Plain text Preview
Branch Sources
Git
Project Repository
Save
Apply


jenkins-ms.singhritesh85.com/job/bankapp/configure
Dashboard > bankapp > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
Git
Project Repository
https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git
Credentials
/****** (github-cred)
+ Add
Behaviors
Discover branches
Filter by name (with wildcards)
Include
dev stage
Exclude
Save
Apply



The screenshot for both the Multibranch Jenkins for MySQL and Bank Application Pods are as shown in the screenshot attached below.





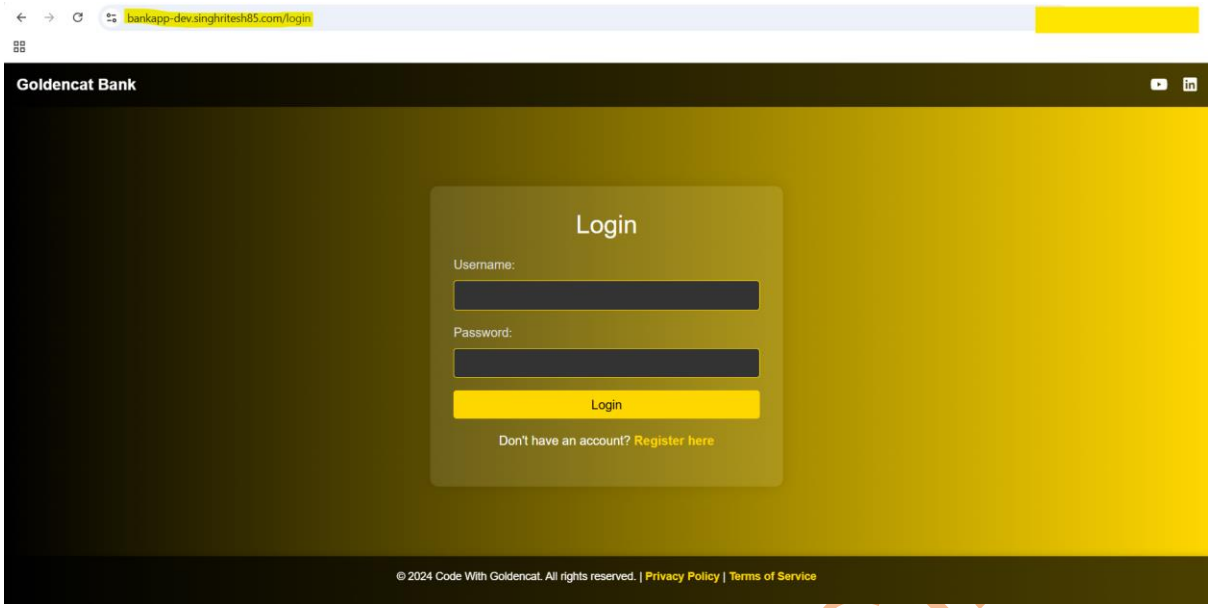
Then create the ingress rule and do the entry for LoadBalancer DNS Name and LoadBalancer External IP in Azure DNS Zone to create the Record Set of type CNAME and A Type for EKS and AKS Cluster respectively as I discussed in Module-1.

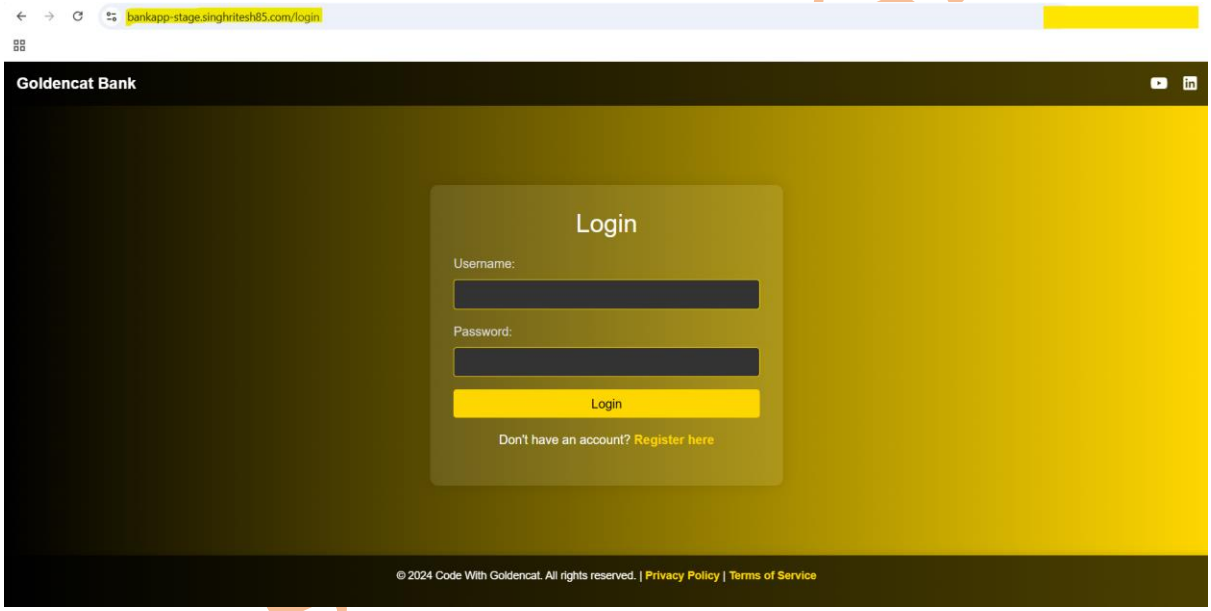
```
cat ingress-rule-dev.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress-dev
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: bankapp-dev.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80
```

cat ingress-rule-stage.yaml

```
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

 name: bankapp-ingress-stage

 namespace: bankapp

 annotations:

  kubernetes.io/ingress.class: nginx

spec:

 ingressClassName: nginx

 tls:

 - hosts:

  - bankapp-stage.singhritesh85.com

  secretName: ingress-tls

 rules:

 - host: bankapp-stage.singhritesh85.com

  http:

   paths:

   - path: /

    pathType: Prefix

    backend:

     service:

      name: bankapp-service

      port:

       number: 80
```

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





Source Code: - https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git

GitHub Repo: - https://github.com/singhritesh85/DevOps-Project-BankApp-with-Secrets-stored-in-Hashicorp-Vault-Multicloud.git

Kubernetes Manifests File: - https://github.com/singhritesh85/kubernetes-manifests.git

Terraform Script: - https://github.com/singhritesh85/DevOps-Project-BankApp-with-Secrets-stored-in-Hashicorp-Vault-Multicloud.git

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