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

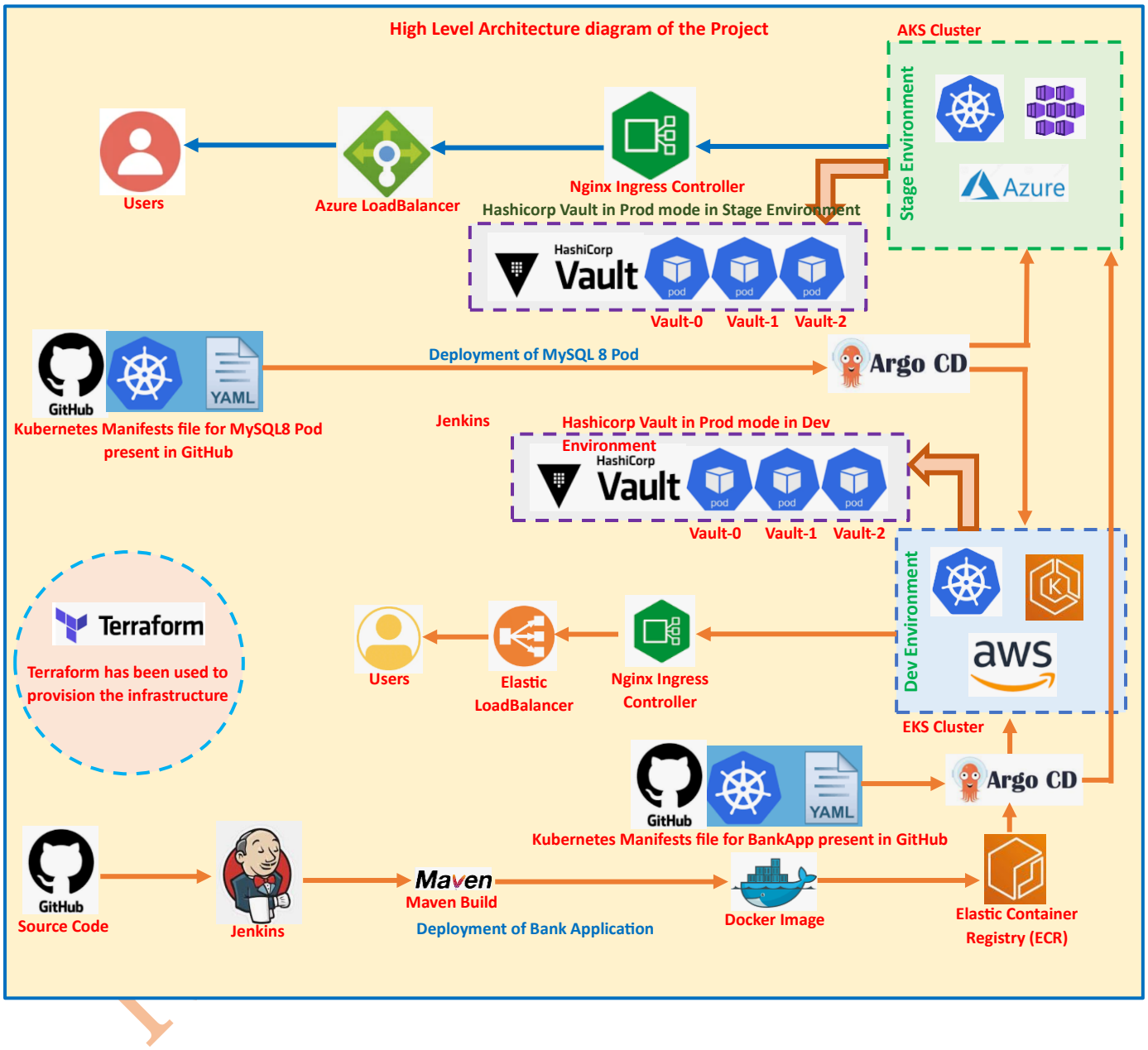
LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/

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

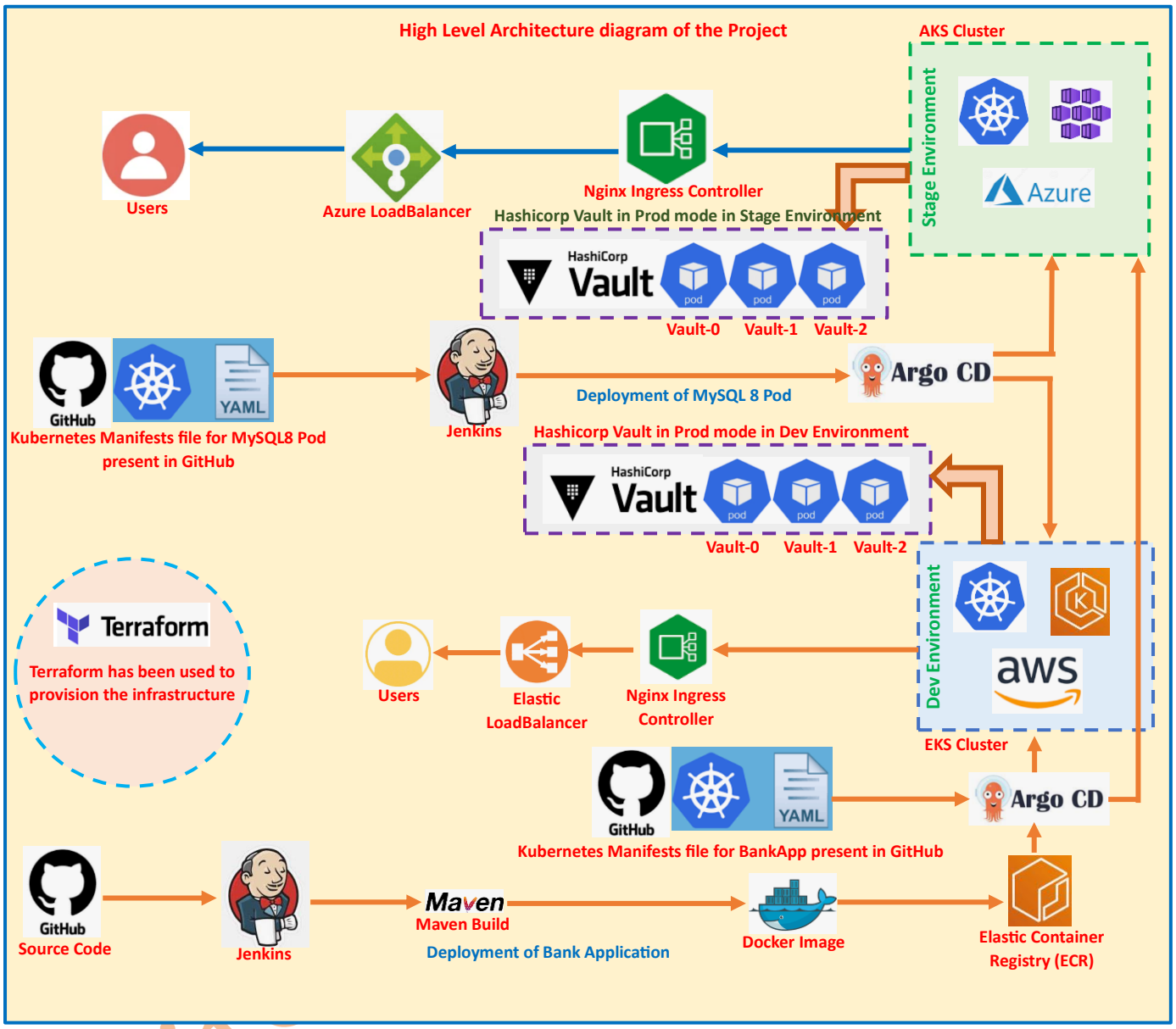
या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

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

Module-1: Manual Approach

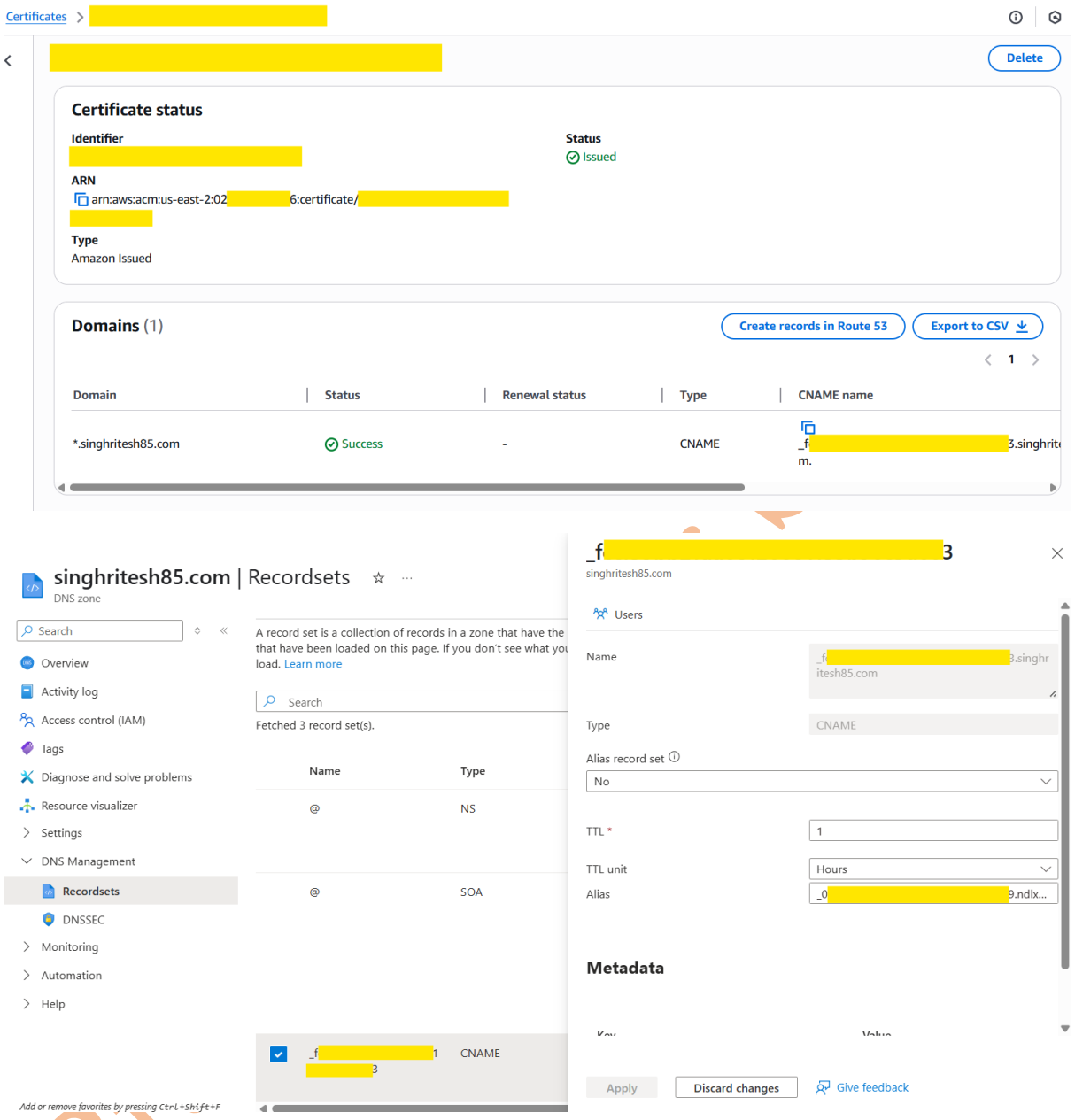


Module-2: Automation Approach



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",
  ]
}
```

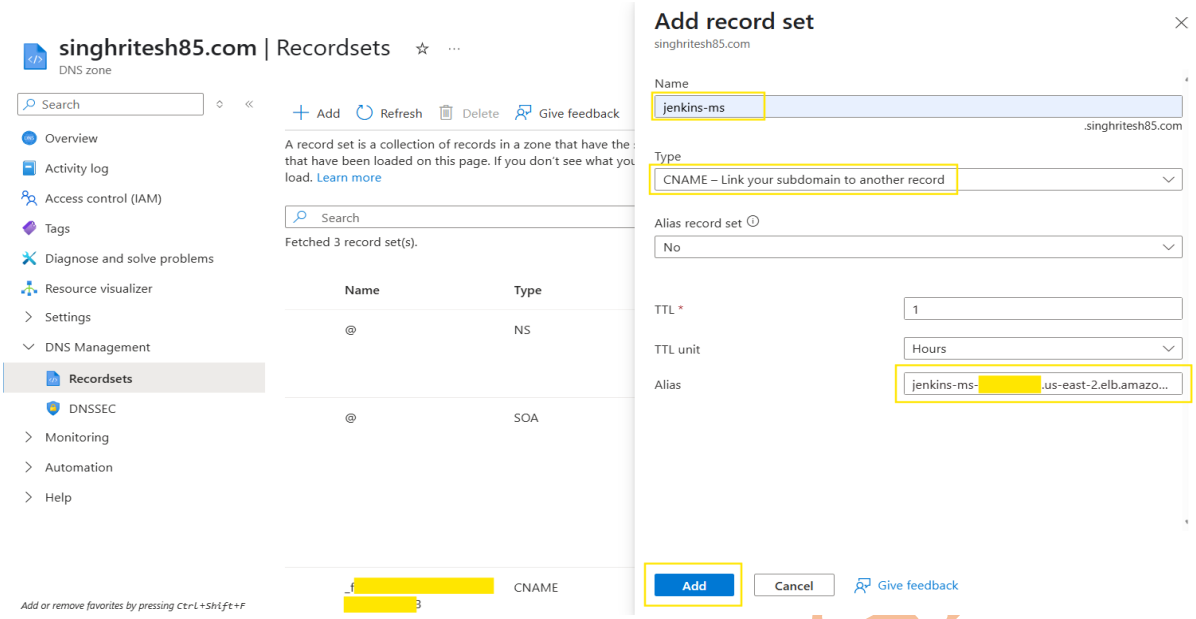
After creation of the resources a kubeconfig file was generated, I did below change in the kubeconfig file as shown in the screenshot attached below.

```
- cluster:
    certificate-authority-data: L[redacted]K
    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
```

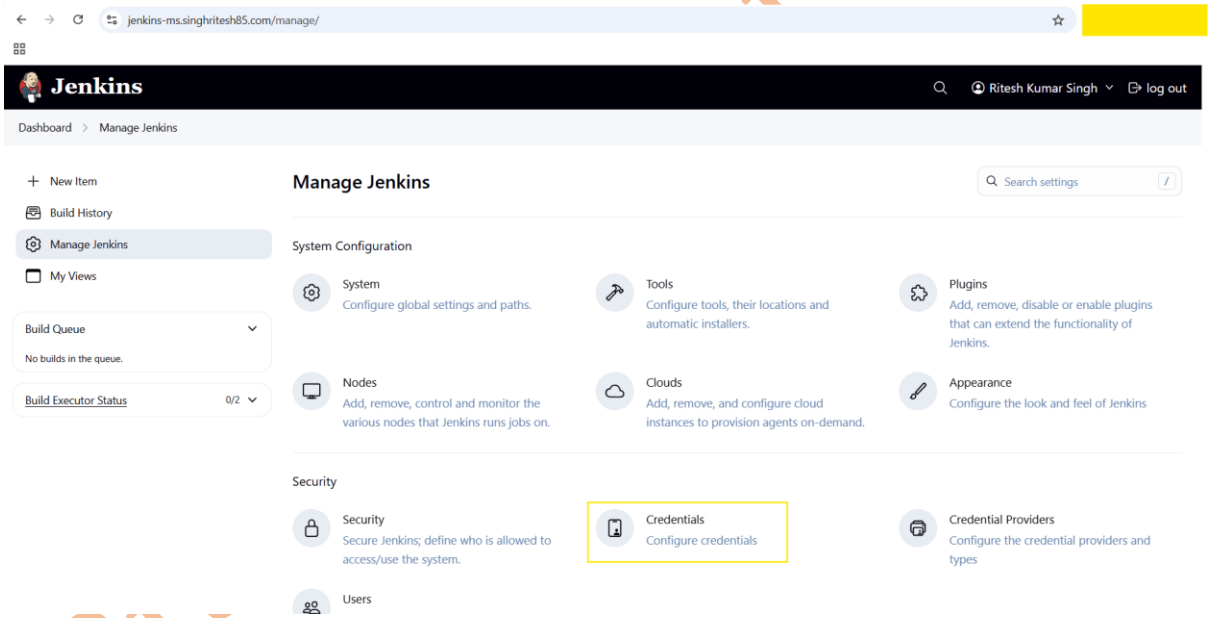
It is my suggestion before any change in the kubeconfig file take a backup of the original kubeconfig file. After creation of EKS and AKS Cluster the node in the Kubernetes Cluster can be seen as shown in the screenshot attached below.

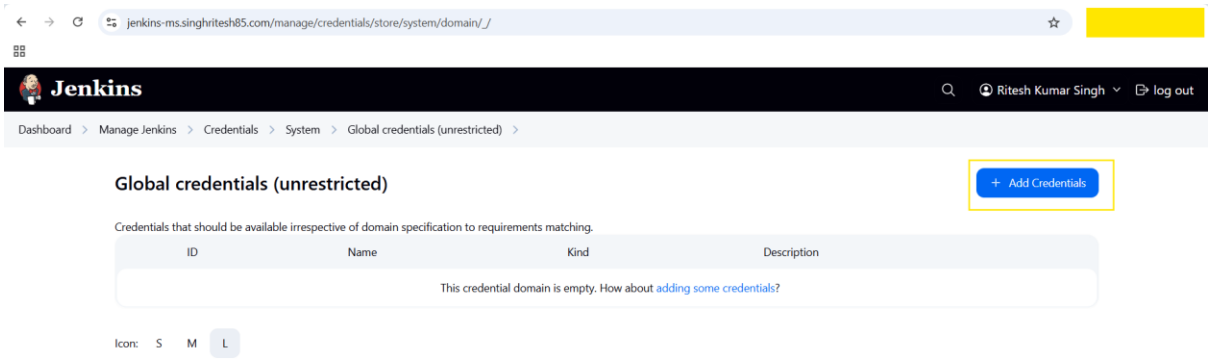
```
[root@[redacted] main]# kubectl get nodes --context=aks-cluster
NAME                                  STATUS   ROLES    AGE    VERSION
aks-agentpool-[redacted]-vmss000000   Ready    <none>   121m   v1.30.0
aks-userpool-[redacted]-vmss000000    Ready    <none>   116m   v1.30.0
aks-userpool-[redacted]-vmss000001    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]
```

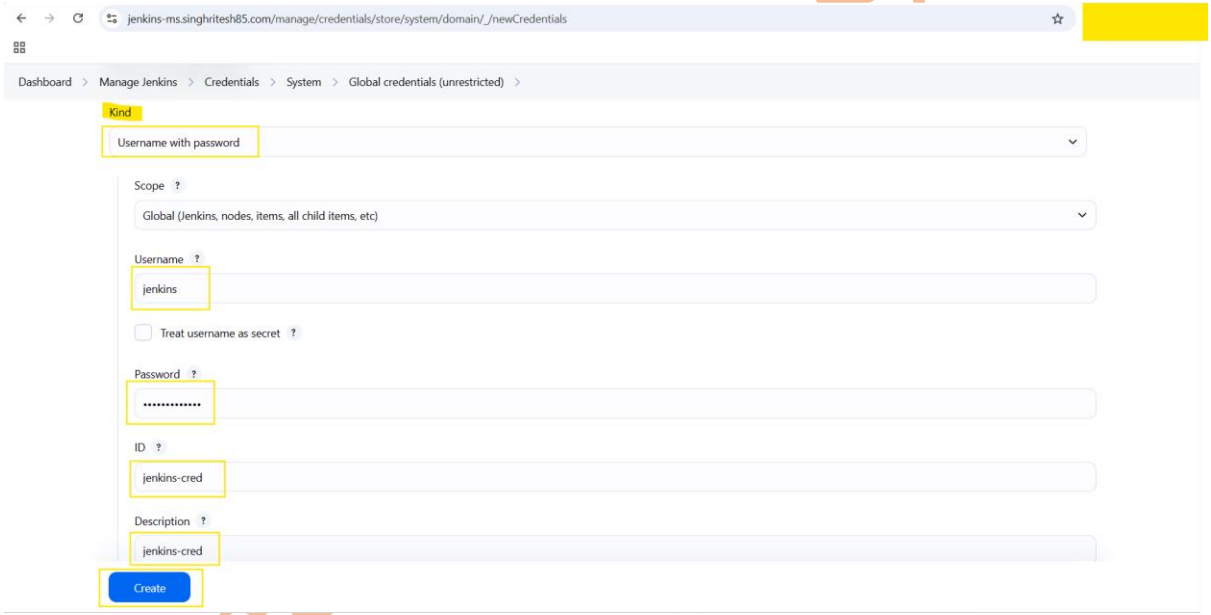
I had created the URL for Jenkins by creating the record set of Type CNAME in Azure DNS Zone using the Jenkins LoadBalancer DNS Name as shown in the screenshot attached below.

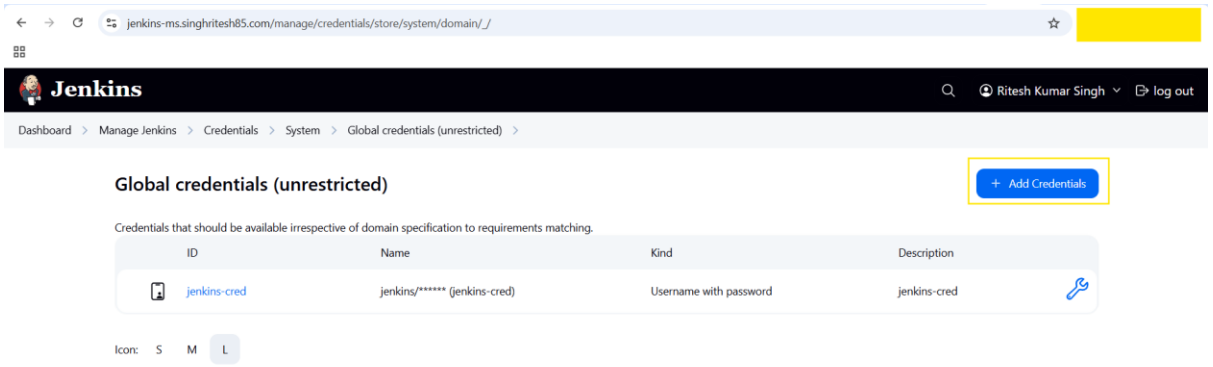


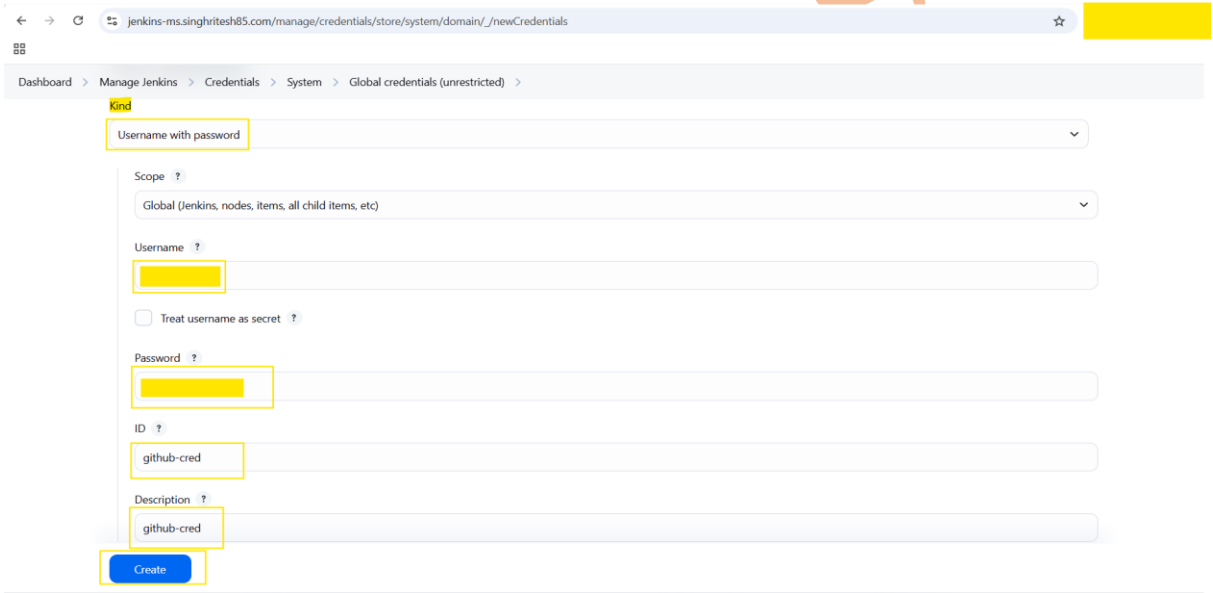
Finally, I logged-in to the Jenkins and created two credentials for Jenkins Slave and GitHub Credentials.



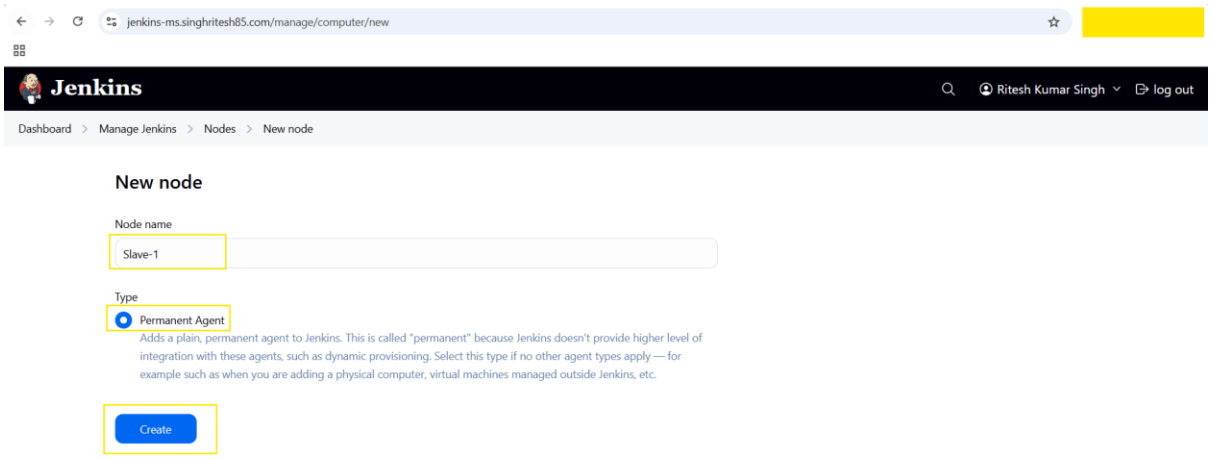

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
This credential domain is empty. How about adding some credentials?
Icon: S M L
REST API
Jenkins 2.504.1

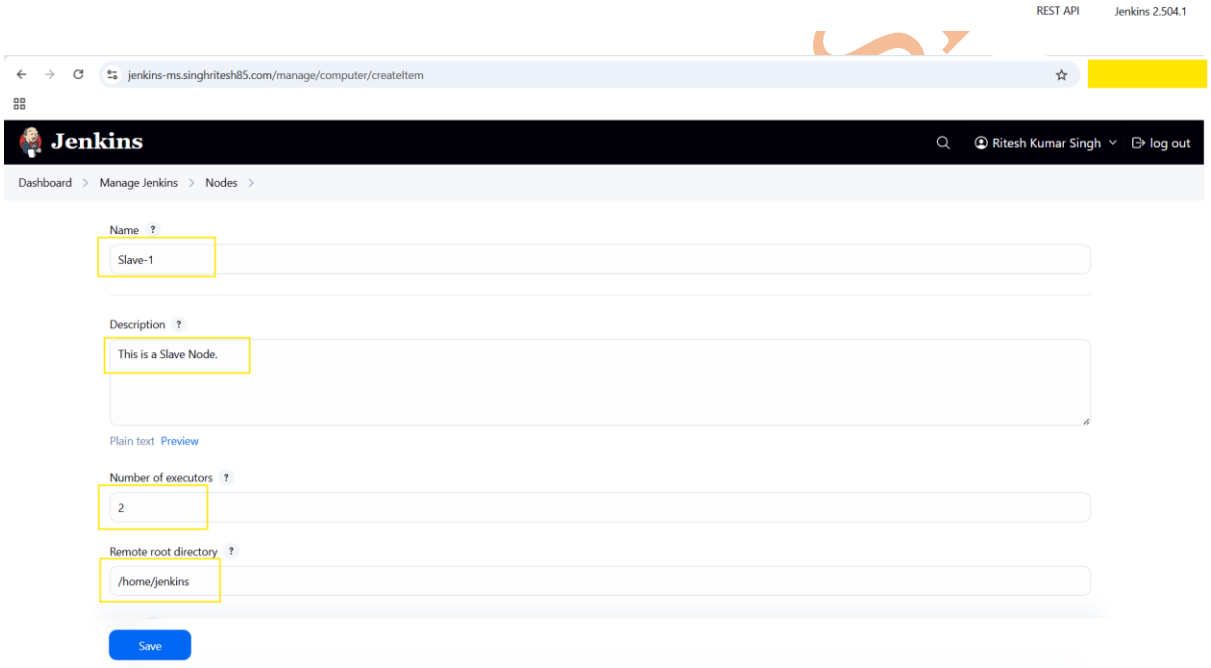

jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
jenkins
Treat username as secret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

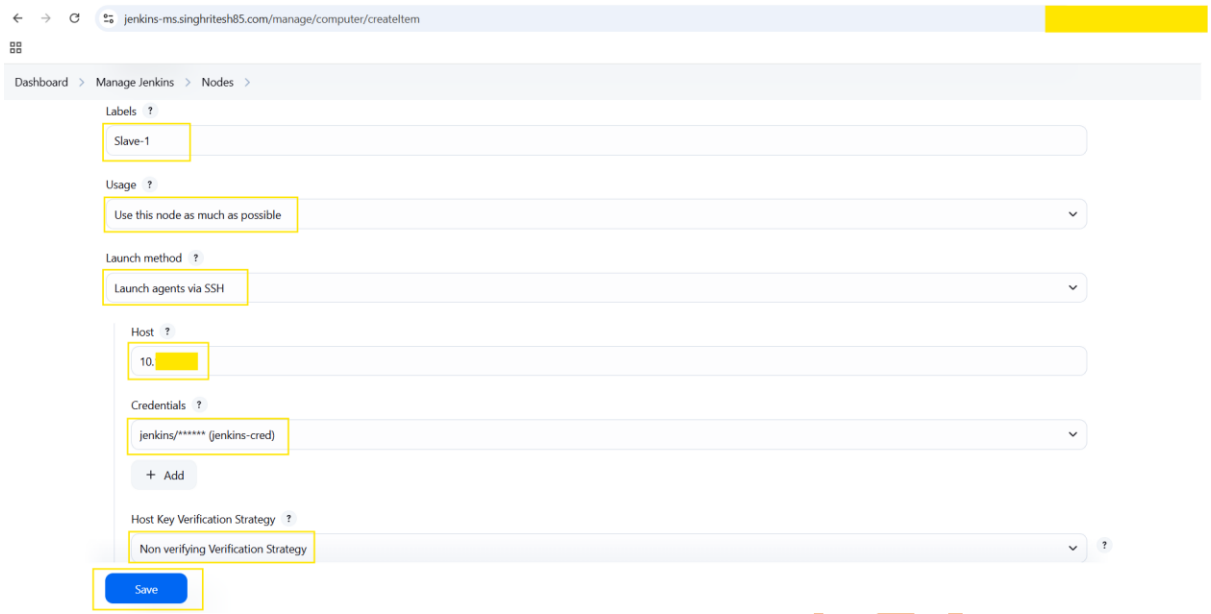


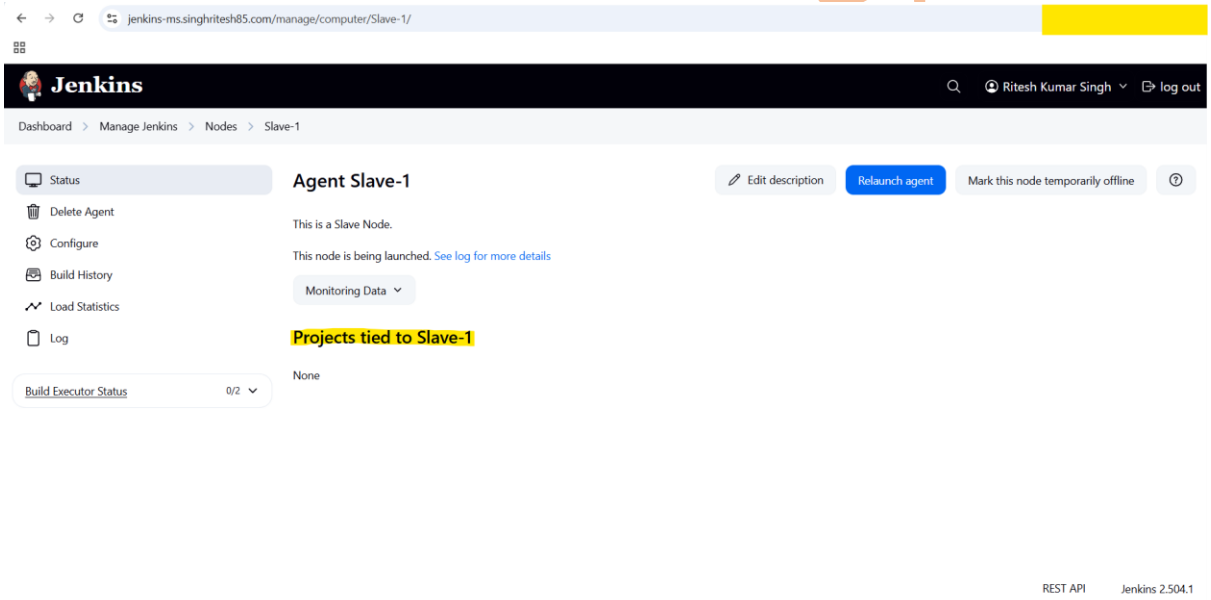


Then I had added Jenkins Slave Node as shown in the screenshot attached below. Go to Jenkins and then Manage Jenkins > Nodes > New Node.


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


REST API
Jenkins 2.504.1
jenkins-ms.singhritesh85.com/manage/computer/createItem
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Nodes >
Name
Slave-1
Description
This is a Slave Node.
Plain text Preview
Number of executors
2
Remote root directory
/home/jenkins
Save





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=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
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                                       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                  false
Total Shares            5
Threshold               3
Version                 1.19.0
Build Date              2025-
Storage Type            raft
Cluster Name            vault-cluster-
Cluster ID              6
Removed From Cluster    false
HA Enabled              true
HA Cluster              https://vault-0.vault-internal:8201
HA Mode                 active
Active Since            2025
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] ~]# 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] 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
```

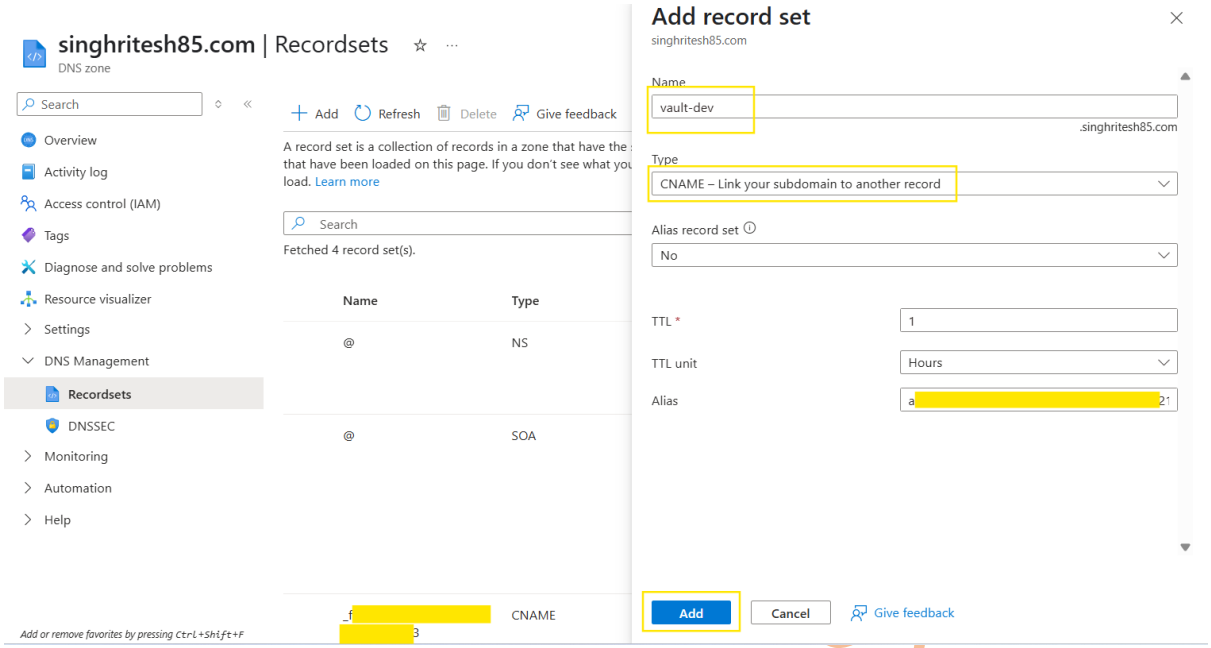
```
/ $ vault operator unseal
Unseal Key (will be hidden):
Key                       Value
---                       -----
Seal Type                 shamir
Initialized               true
Sealed                    true
Total Shares              5
Threshold                 3
Unseal Progress           0/3
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
```

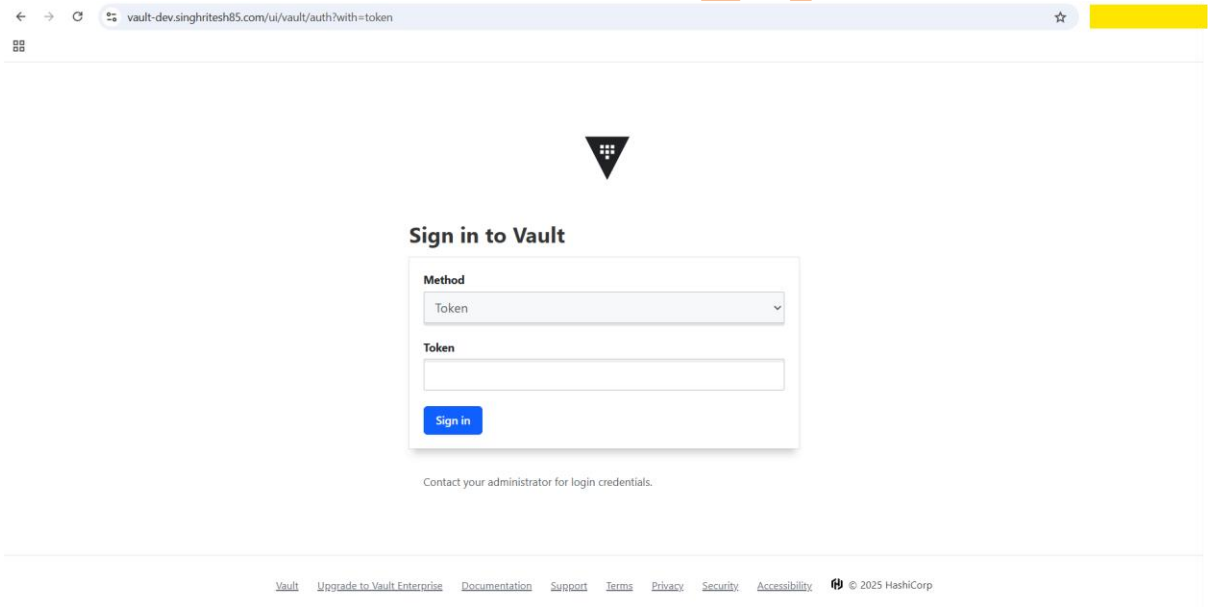
Then I created the Ingress Rule and finally a Record Set of Type CNAME in Azure DNS Zone to access the vault in prod mode in dev cluster.

```
[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.

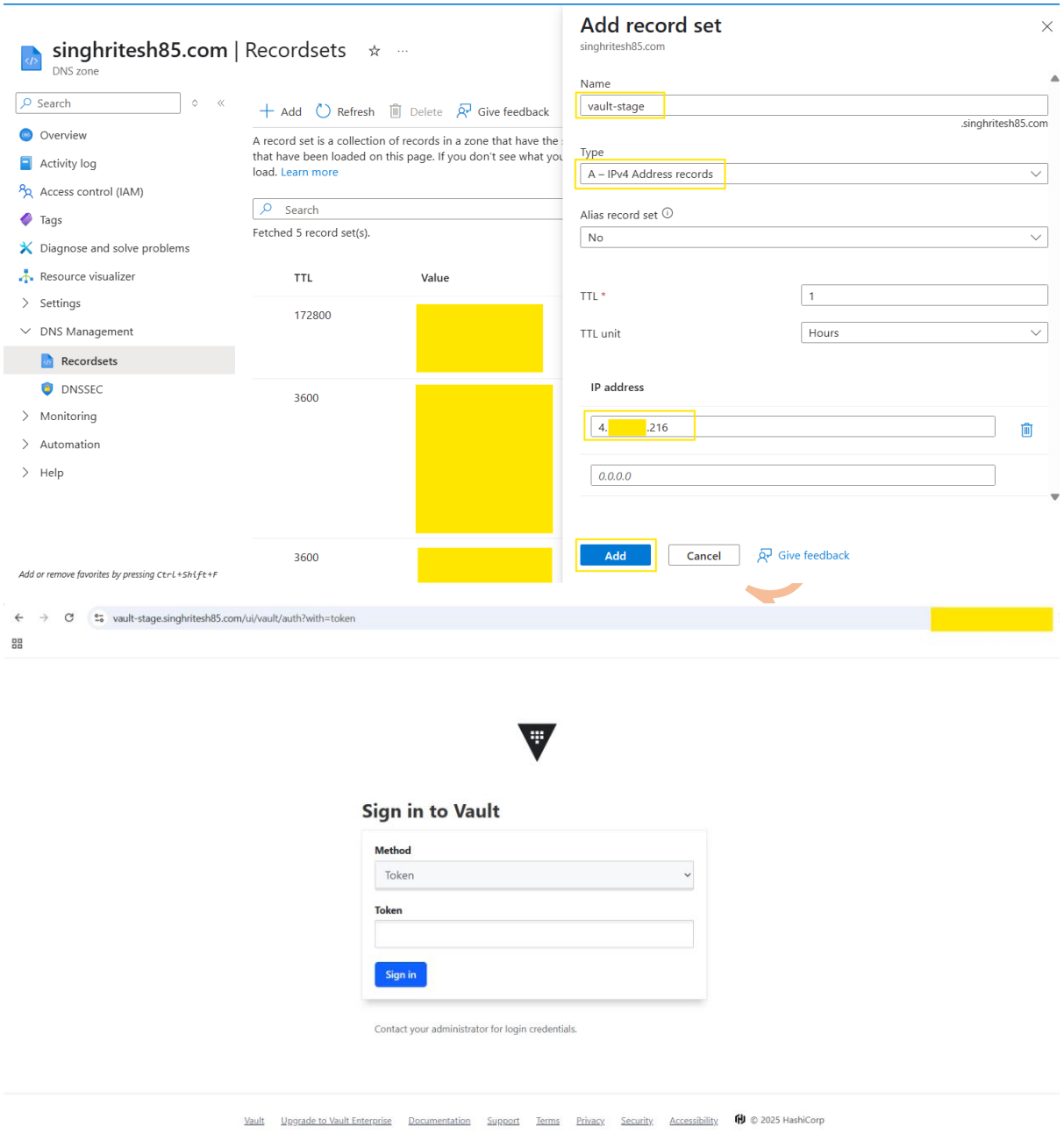
```
[root@                    ~]# kubectl apply -f vault-stage-ingress-rule.yaml --context=aks-cluster
ingress.networking.k8s.io/vault-ingress created
[root@                    ~]# kubectl get ing -A --watch --context=aks-cluster
NAMESPACE   NAME            CLASS   HOSTS                          ADDRESS   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
```

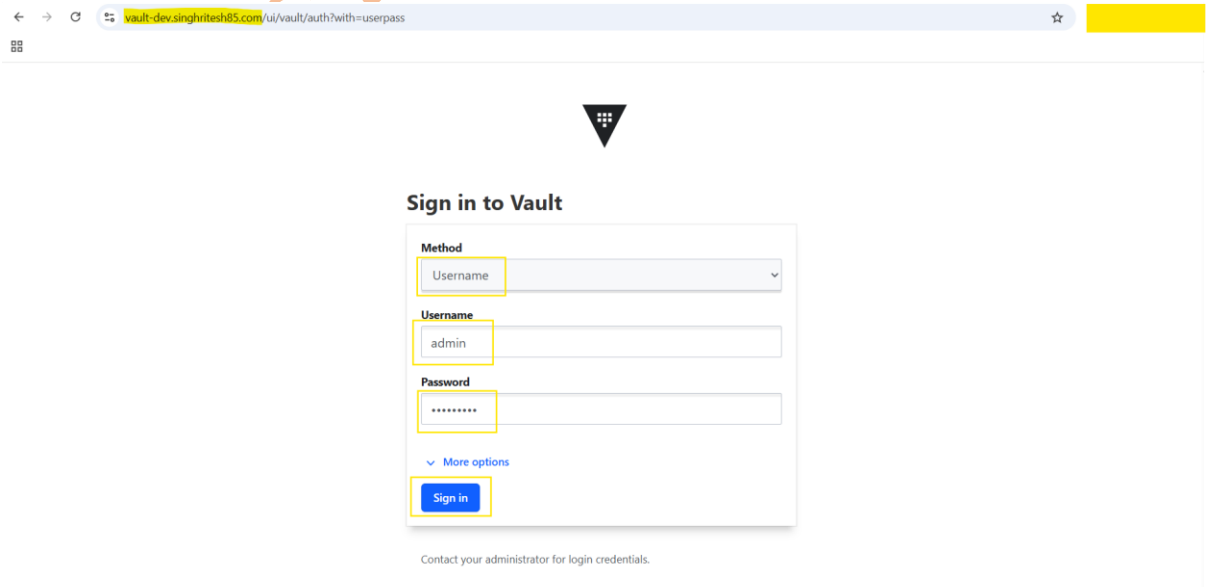


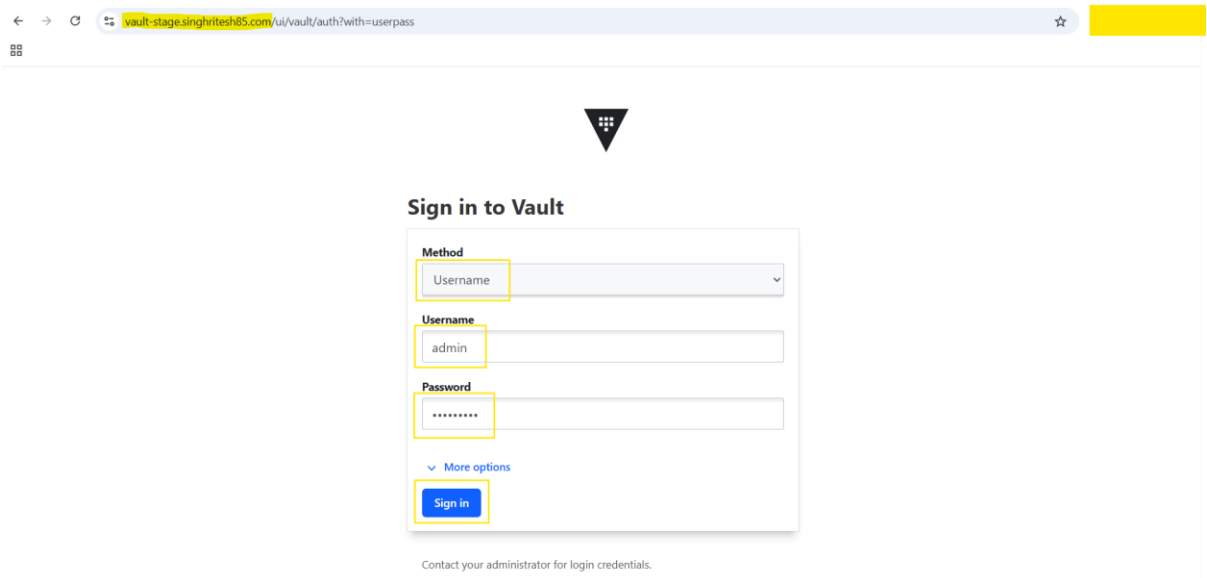
For the first time login method into the vault is Token. The Token which I got from the vault-0 pod in dev cluster and stage cluster was used to login into the Hashicorp vault prod mode in dev cluster and stage cluster respectively. Then I enabled the vault auth userpass and created a username with password 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.





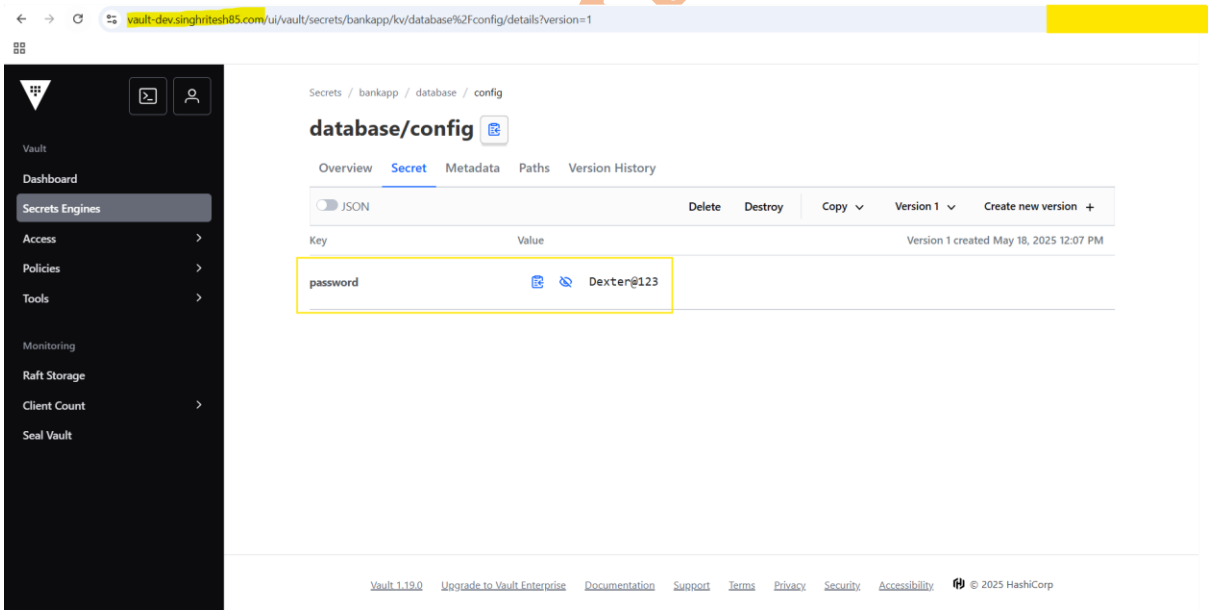
Then created the Hashicorp vault secrets in dev cluster (EKS Cluster) and stage cluster (AKS Cluster) as shown in the screenshot attached below.

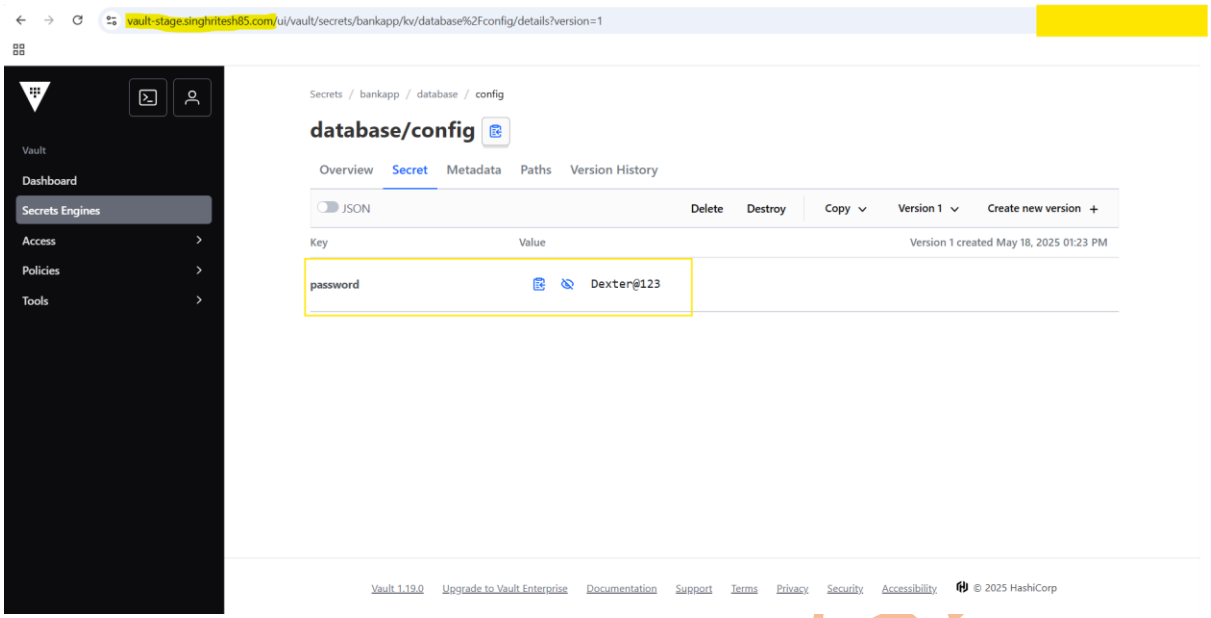
```
[root@                ~]# kubectl exec -it vault-0 -n vault sh --context=eks-demo-cluster-dev
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
```

```
/ $ 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        6: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

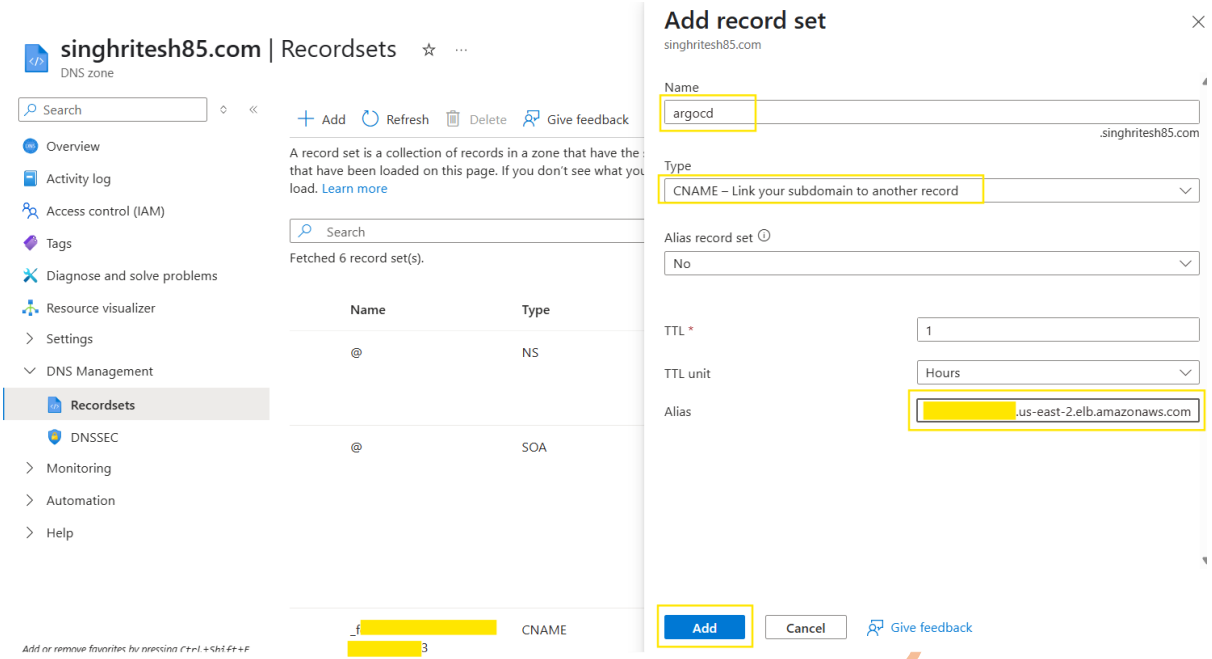
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d

```
[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@                ~]#
```

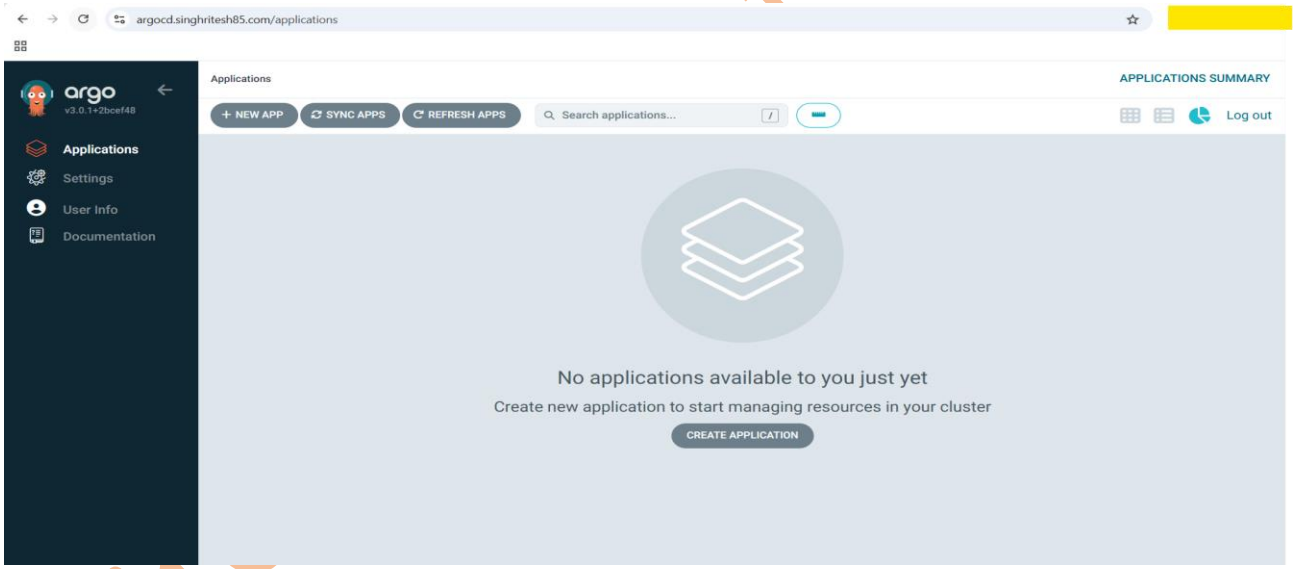
```
cat argocd-ingress-rule.yaml

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

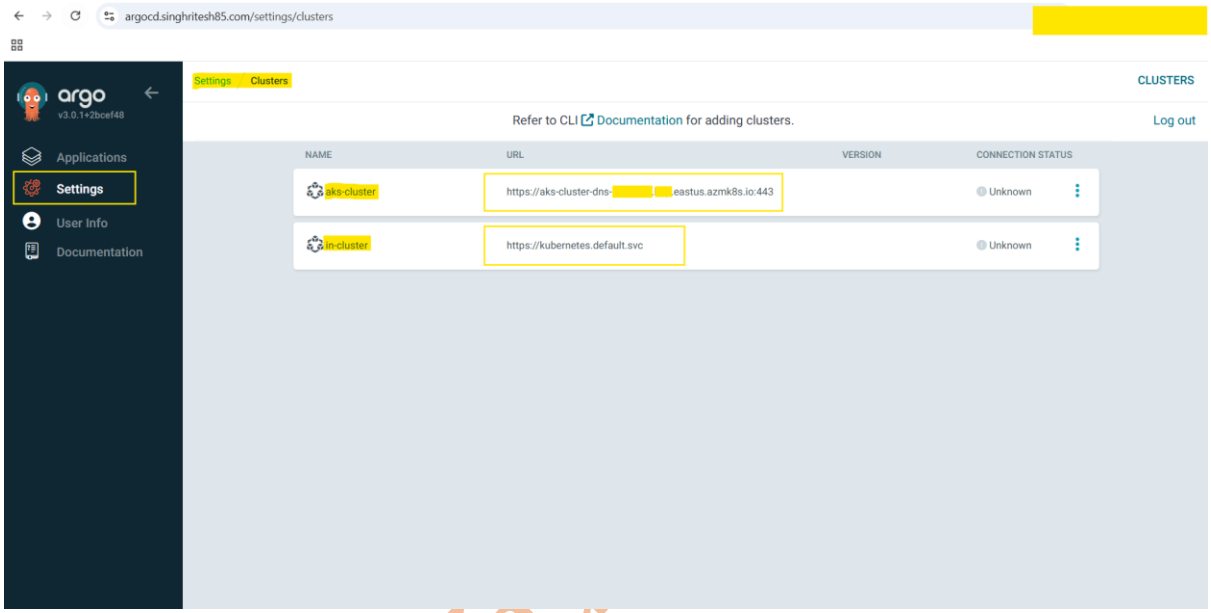
```
[jenkins@          ~]$ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
[jenkins@          ~]$ sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
[jenkins@          ~]$ rm argocd-linux-amd64
```

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-
```

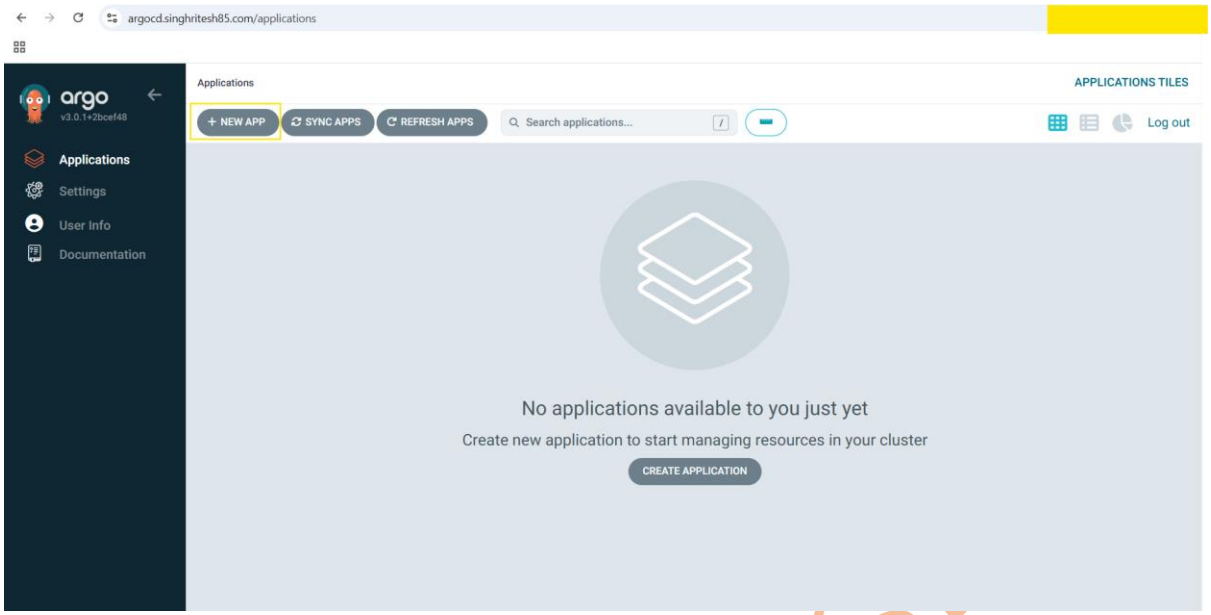


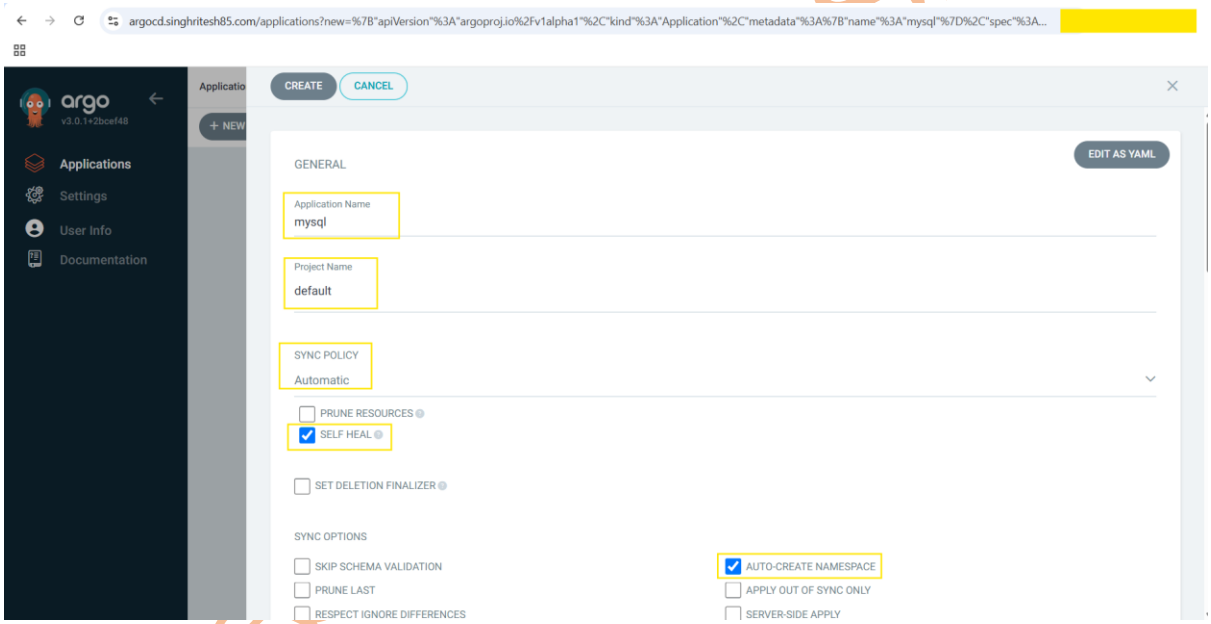
On Jenkins Slave Node I had provided Authentication and Authorization using Access Key and Secret Key as shown in the screenshot attached below.

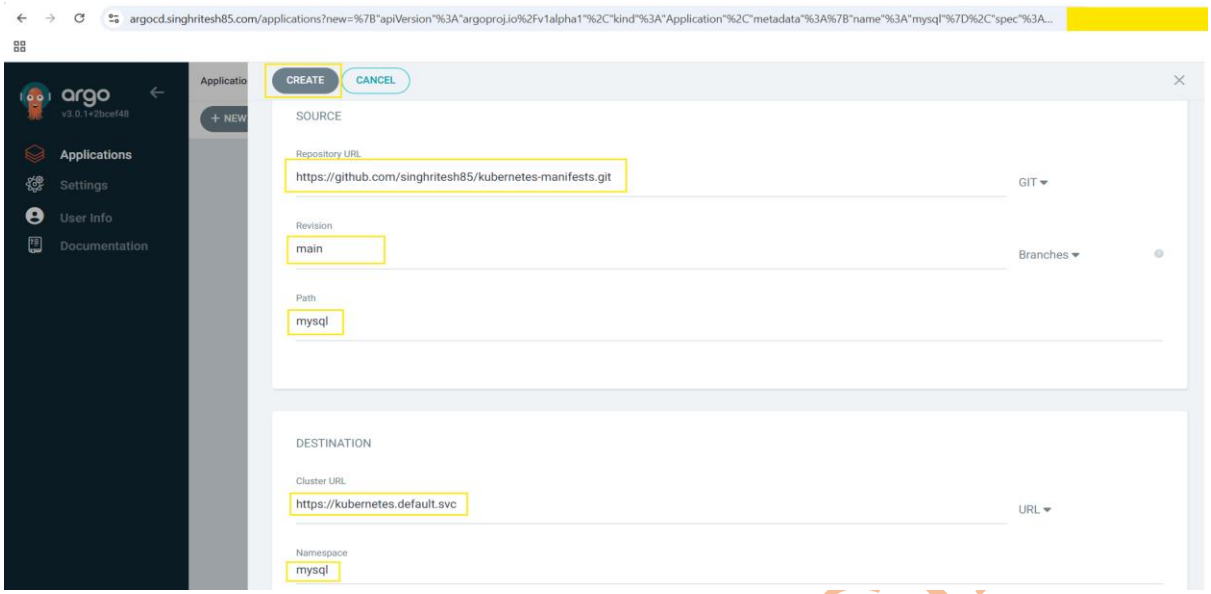
```
[jenkins@ ~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Module-1: Manual Approach

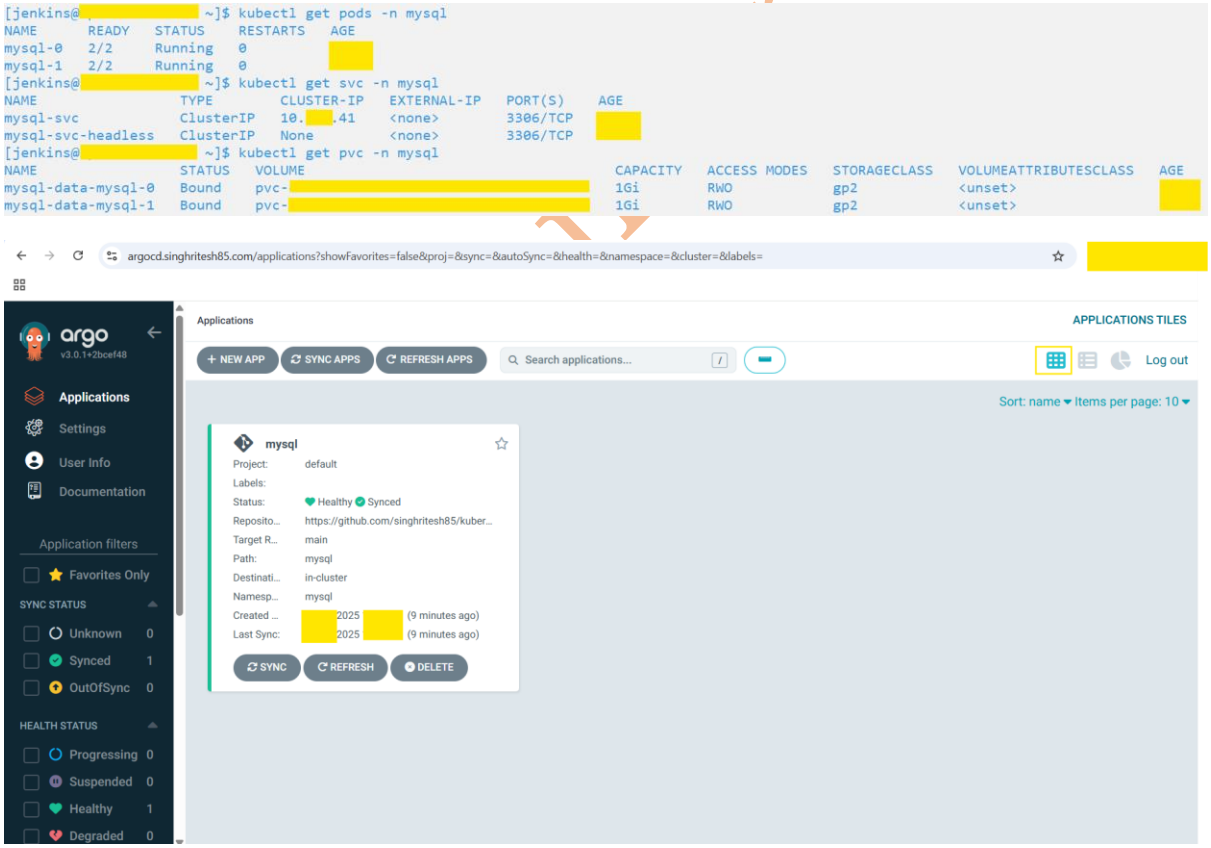
This project is basically a multibranch pipeline project in which only dev and stage environment exists. I had created mysql pod in dev and stage environment using the ArgoCD as shown in the screenshot attached below.

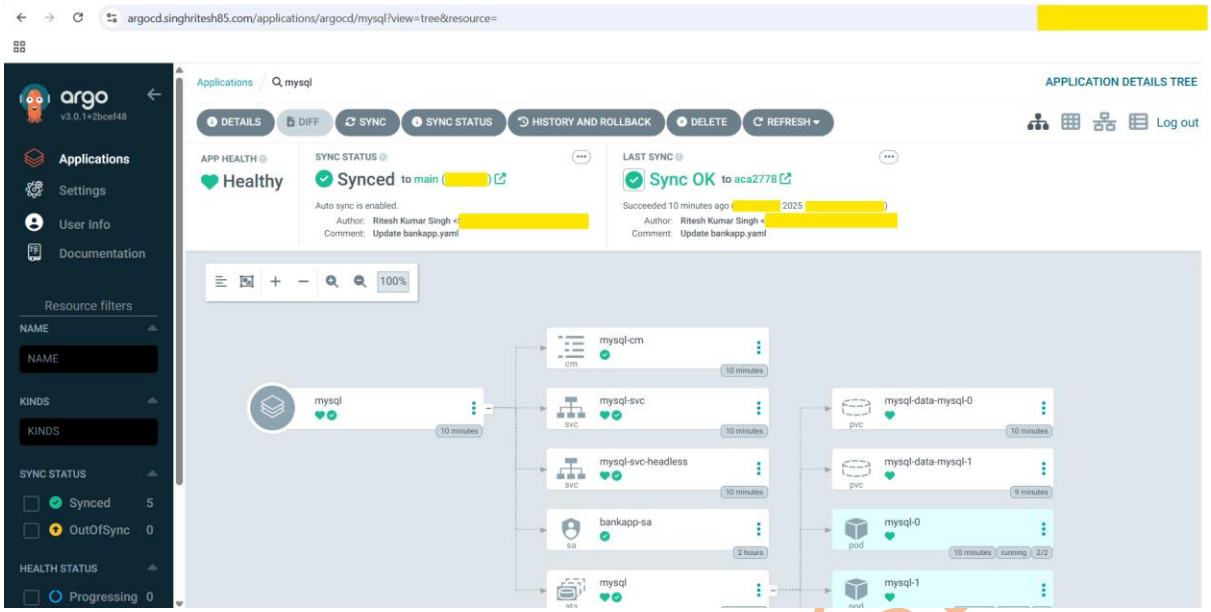

argocd.singhritesh85.com/applications
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
No applications available to you just yet
Create new application to start managing resources in your cluster
CREATE APPLICATION


argocd.singhritesh85.com/applications?new=%7B"apiVersion"%3A"argoproj.io%2Fv1alpha1"%2C"kind"%3A"Application"%2C"metadata"%3A%7B"name"%3A"mysql"%7D%2C"spec"%3A...
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
CREATE
CANCEL
EDIT AS YAML
GENERAL
Application Name
mysql
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

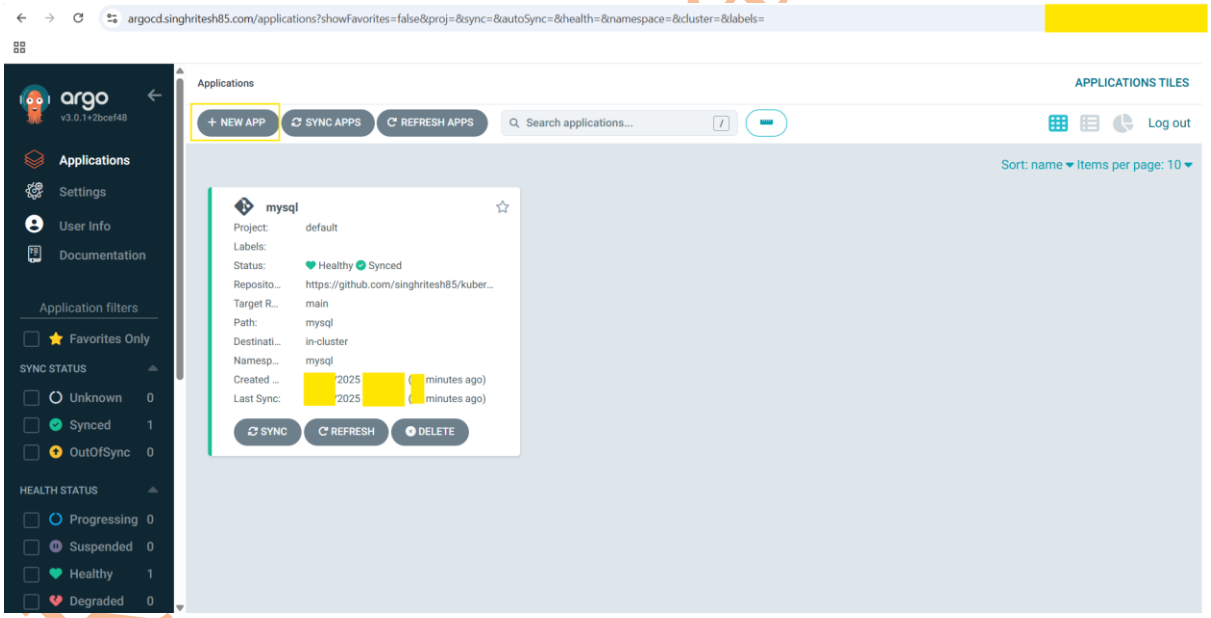


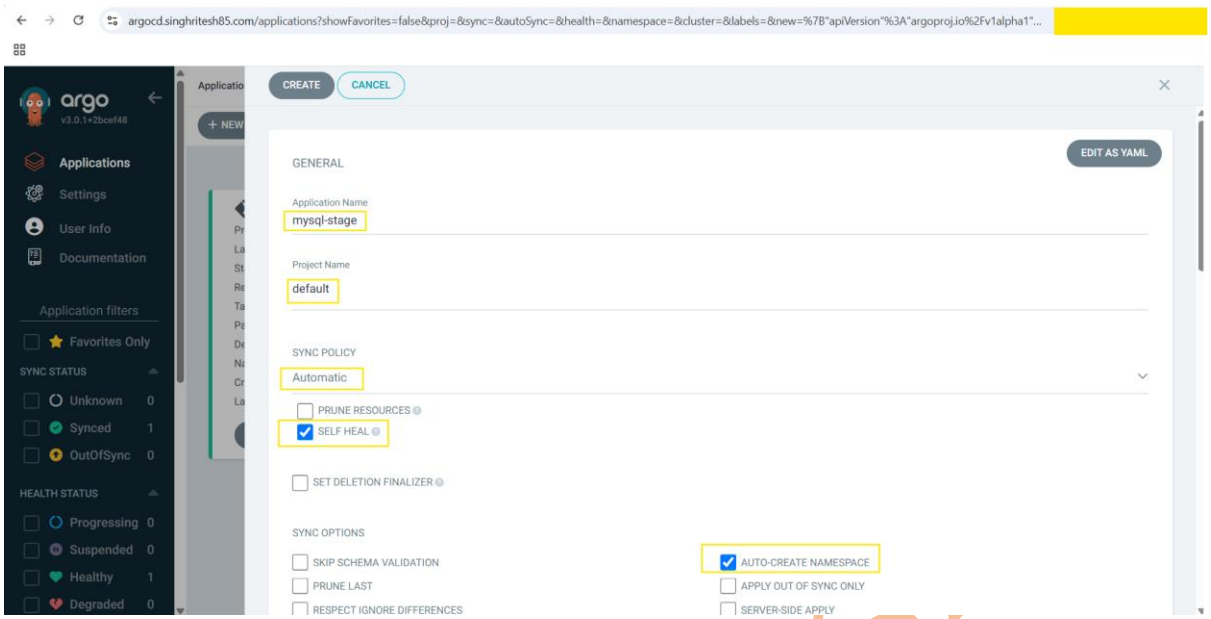
After creation of the Application mysql as shown in screenshot attached below the mysql pods had been created on dev cluster (EKS Cluster) as shown in the screenshot attached below.

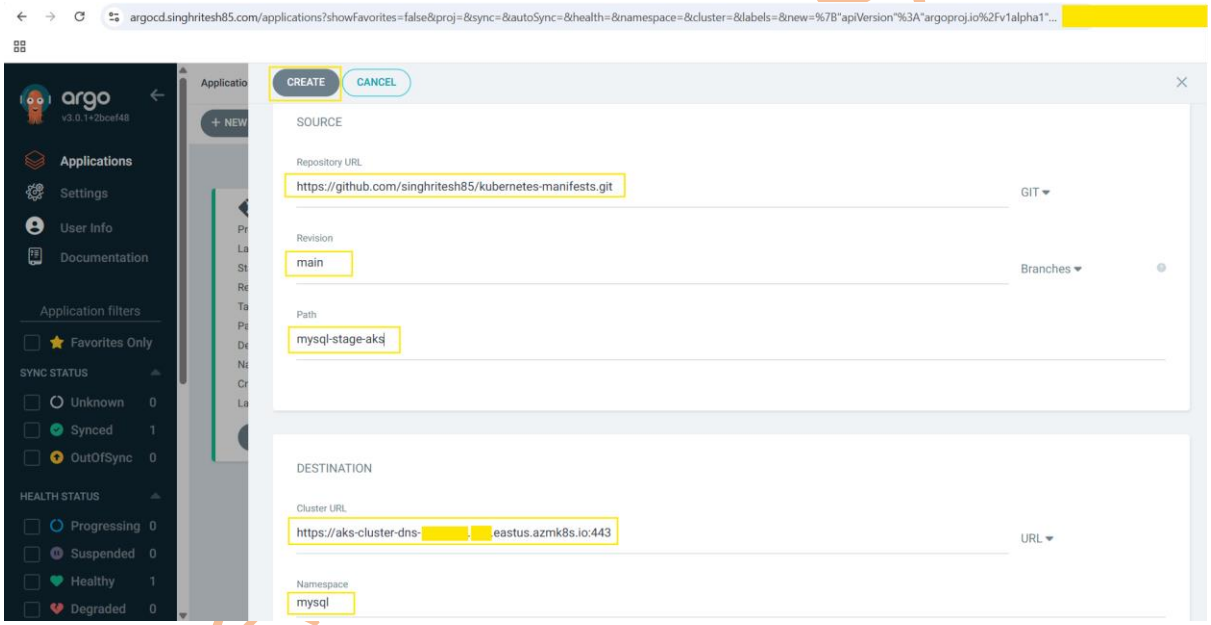




Then I had created **New App** in ArgoCD to deploy pods on AKS Cluster (Stage Environment) as shown in the screenshot attached below.

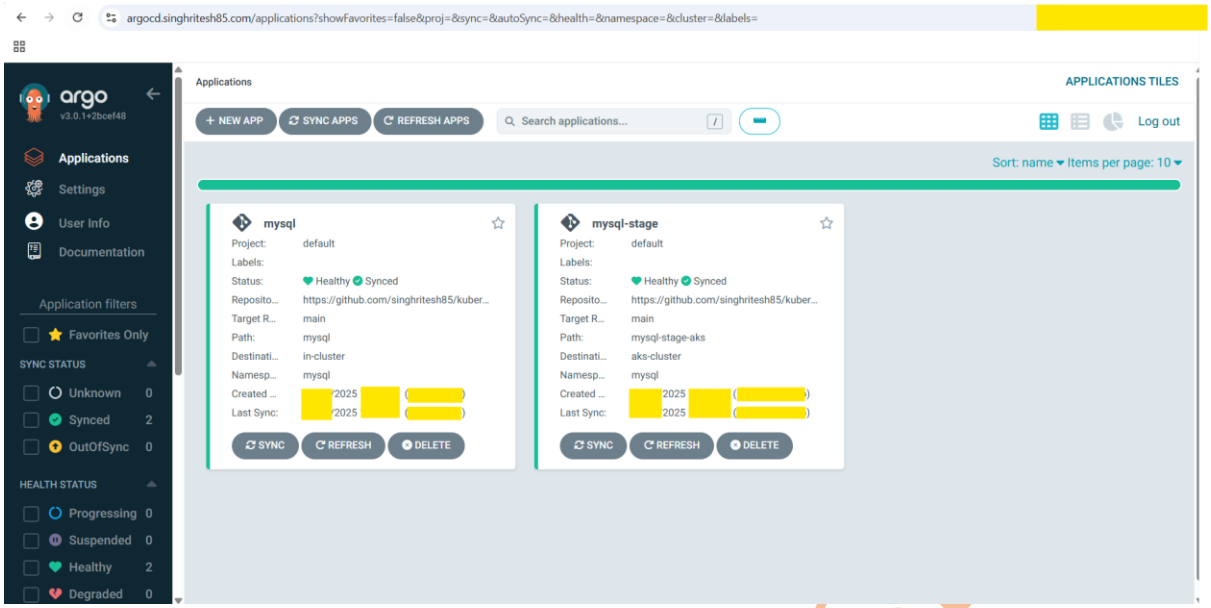






After creation of the Application **mysql-stage** as shown in screenshot attached below the mysql pods had been created on stage cluster (AKS Cluster) as shown in the screenshot attached below.

```
[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.

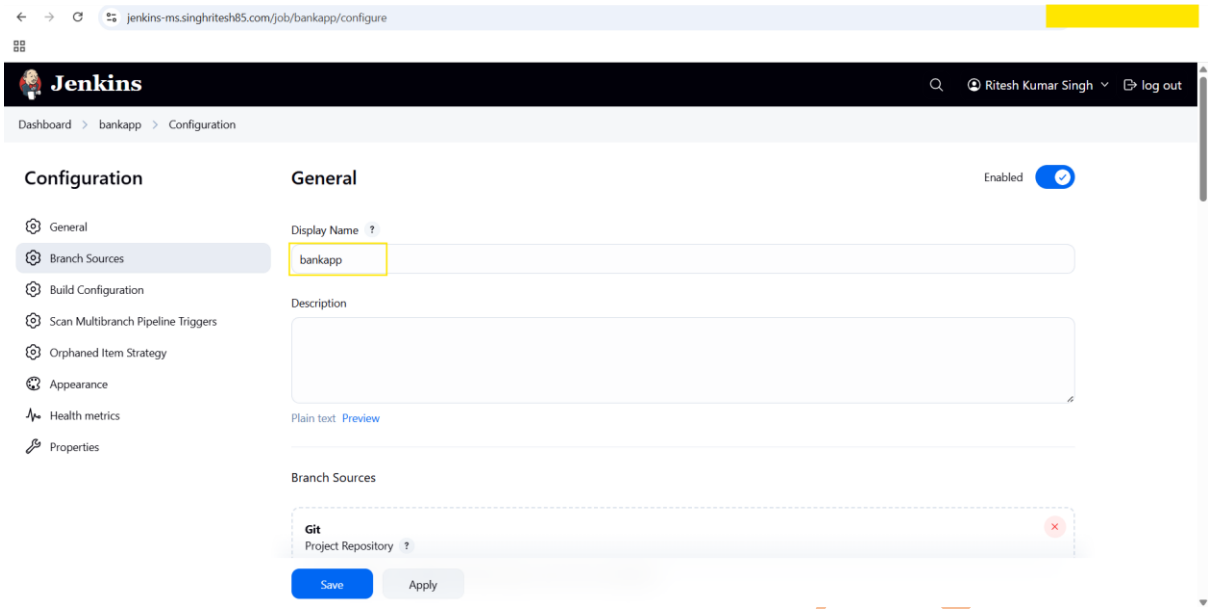
To access Docker Image from ECR (Elastic Container Registry) I created kubernetes secret in AKS Cluster as shown in the screenshot attached below.

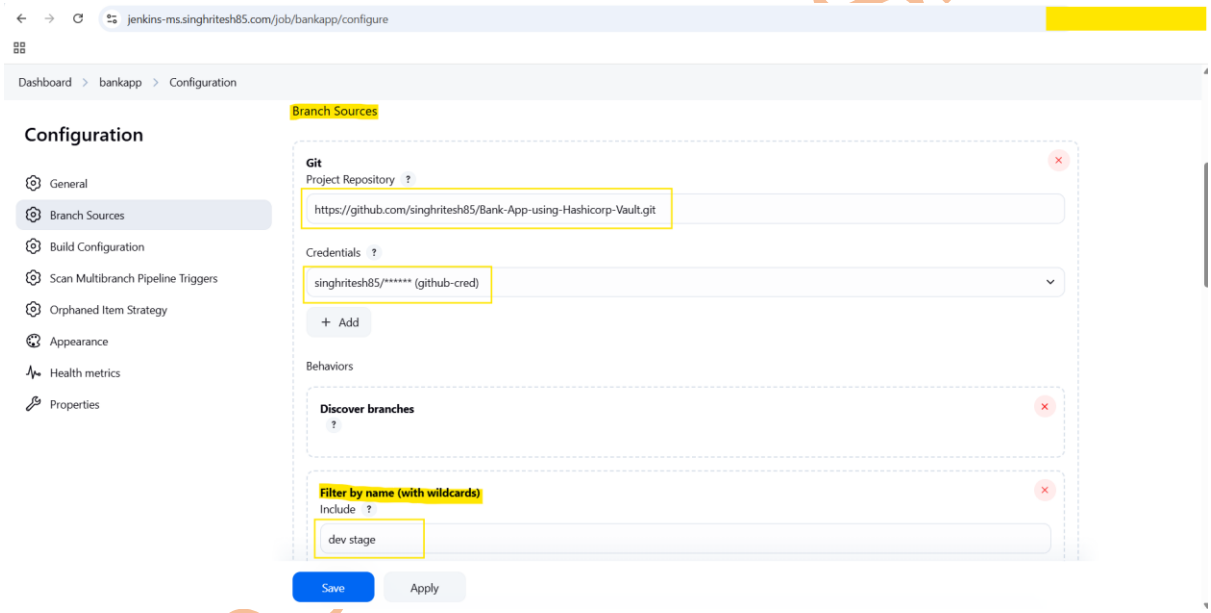
kubectl create secret docker-registry regcred --docker-server=02XXXXXXXXX6.dkr.ecr.us-east-2.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password) --namespace=bankapp --context=aks-cluster

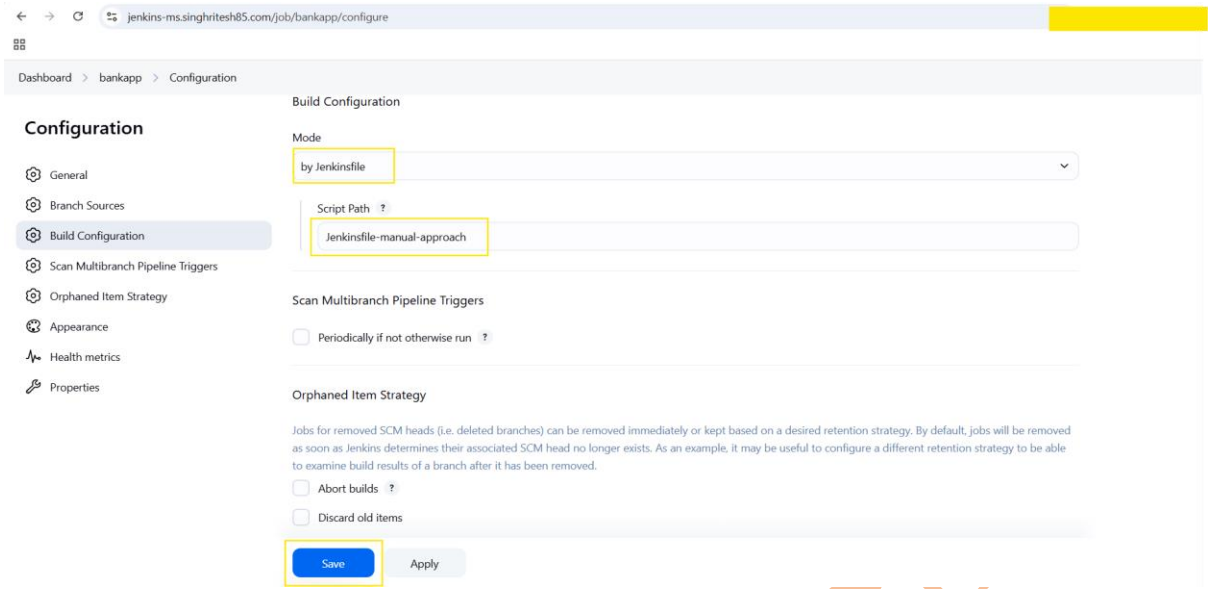
```
[root@                    ]# kubectl create secret docker-registry regcred --docker-server=02          6.dkr.ecr.us-east-2.amazonaws.com --docker-username=AWS
 --docker-password=$(aws ecr get-login-password) --namespace=bankapp --context=aks-cluster
secret/regcred created
```

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.

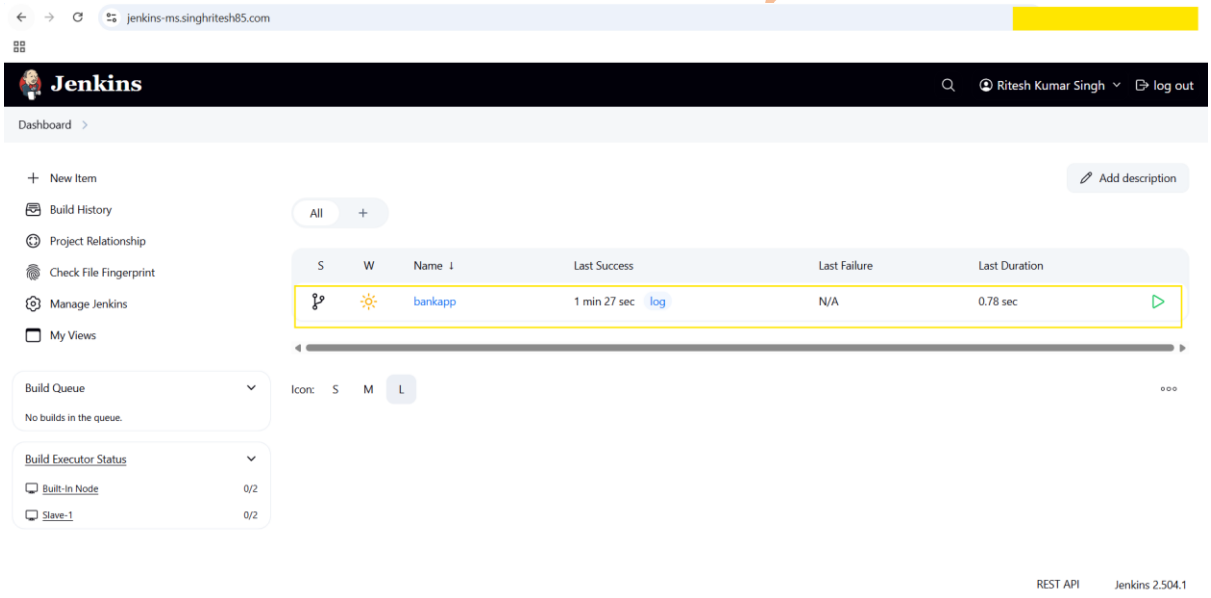
I had built the Bank Application code and created the Docker Image then pushed the Docker Image to ECR (Elastic Container Registry) using the Multibranch Jenkins Job as shown in the screenshot attached below.

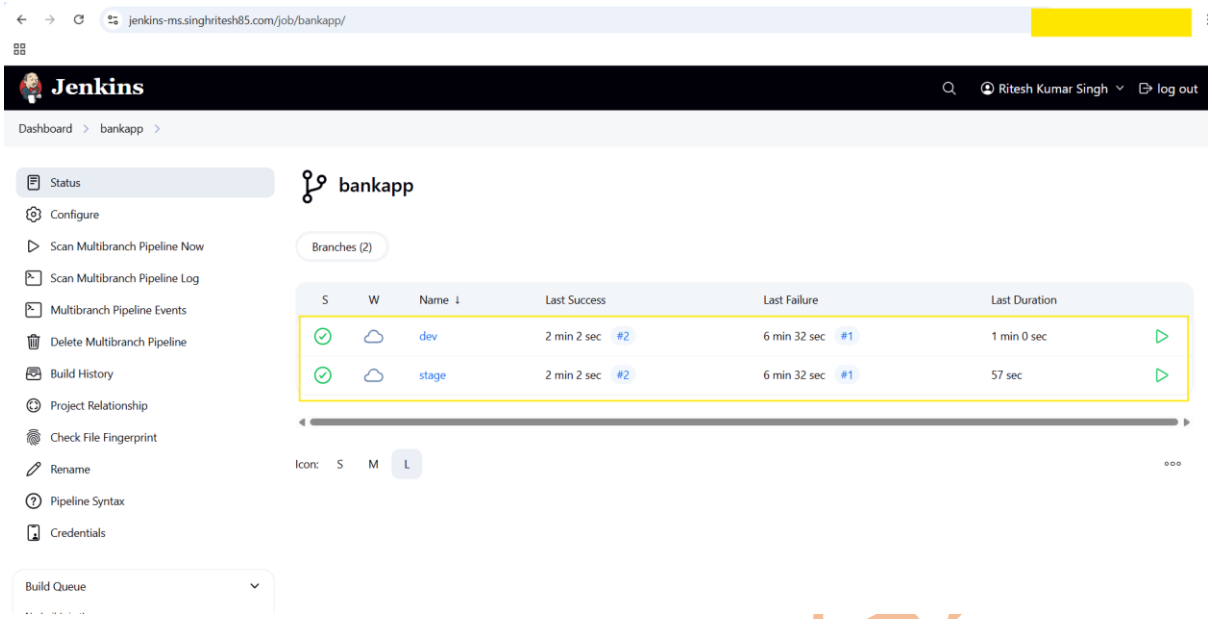

jenkins-ms.singhritesh85.com/job/bankapp/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > bankapp > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
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
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
singhritesh85/****** (github-cred)
+ Add
Behaviors
Discover branches
Filter by name (with wildcards)
Include
dev stage
Save
Apply

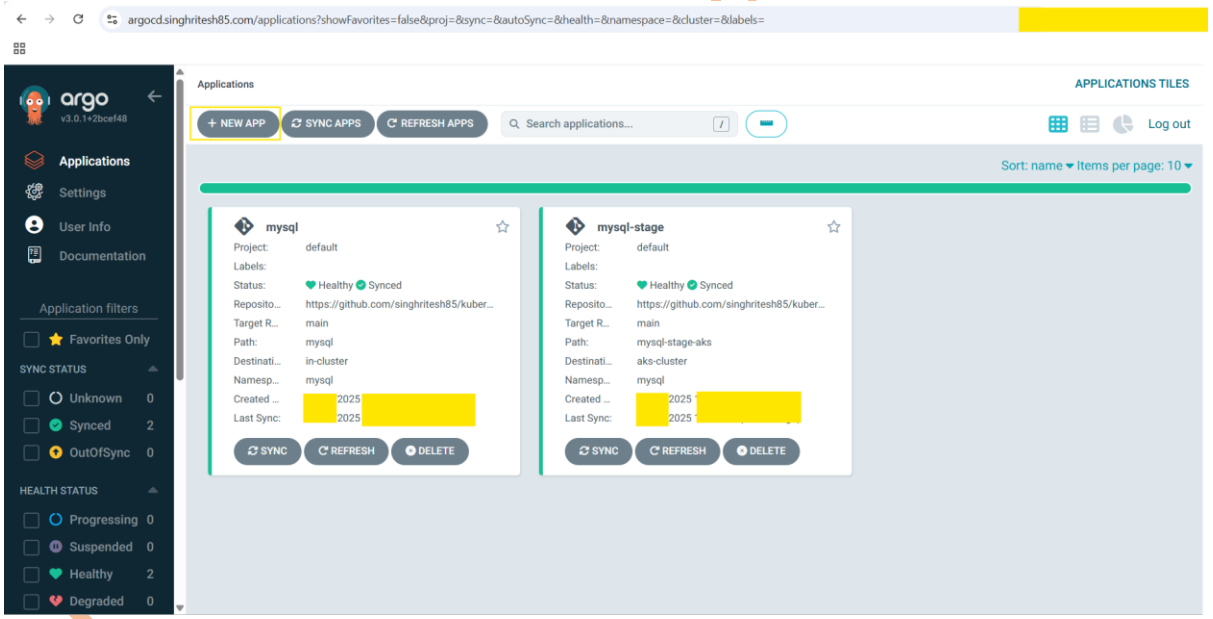


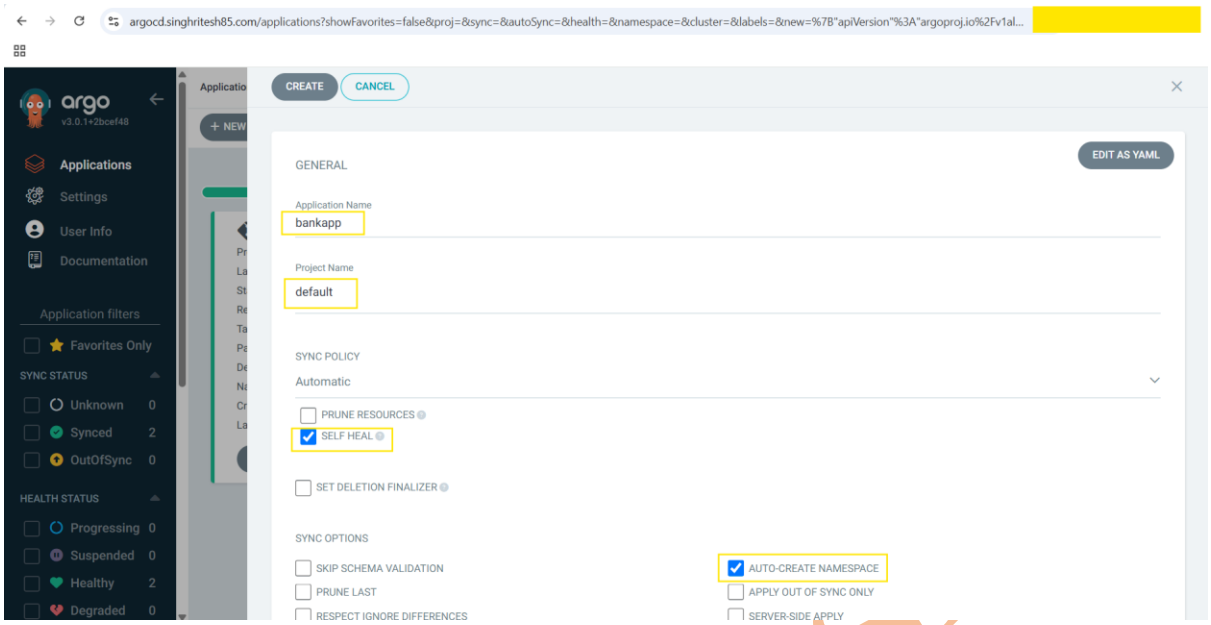
After doing this I had seen two Jenkins Job created with the name as that of the branch name in the multi branch Jenkins Job as shown in the screenshot attached below.

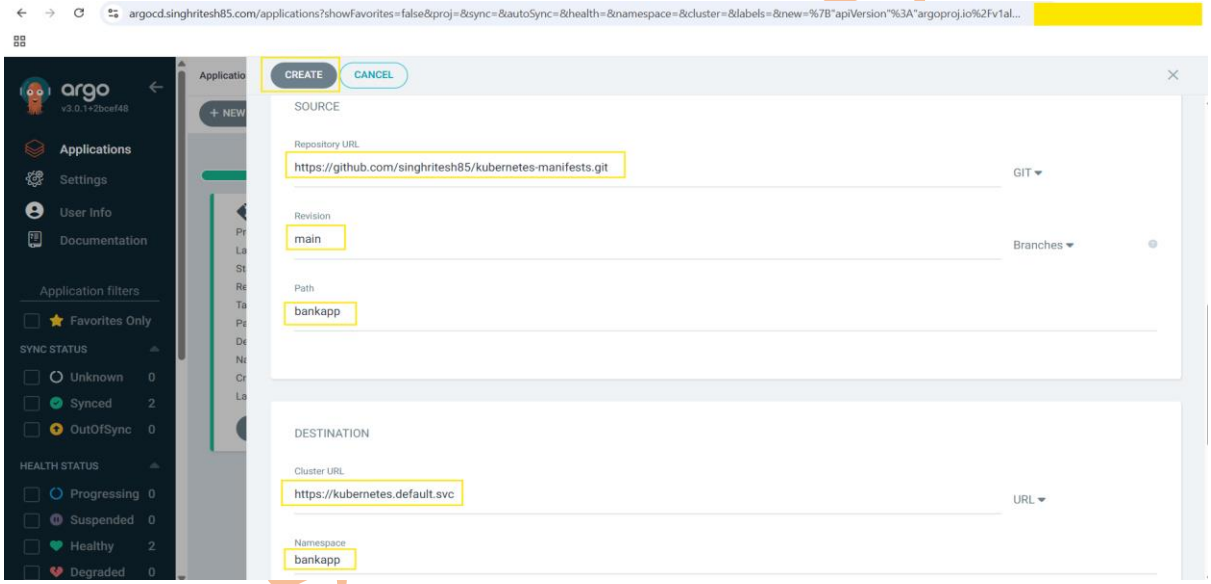


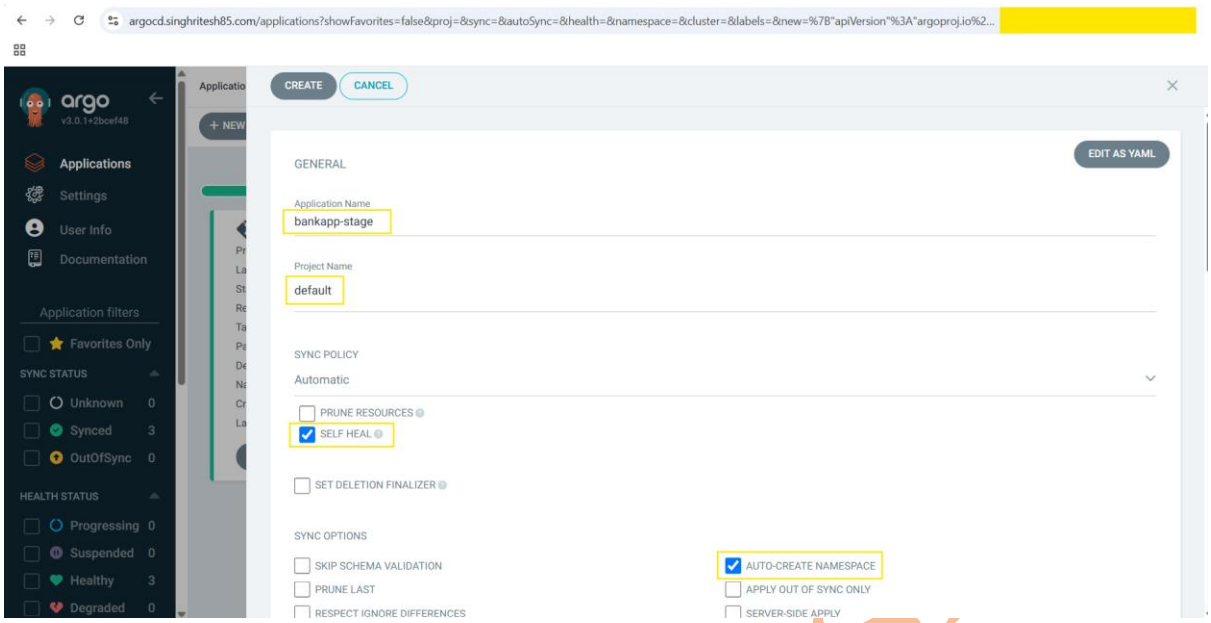


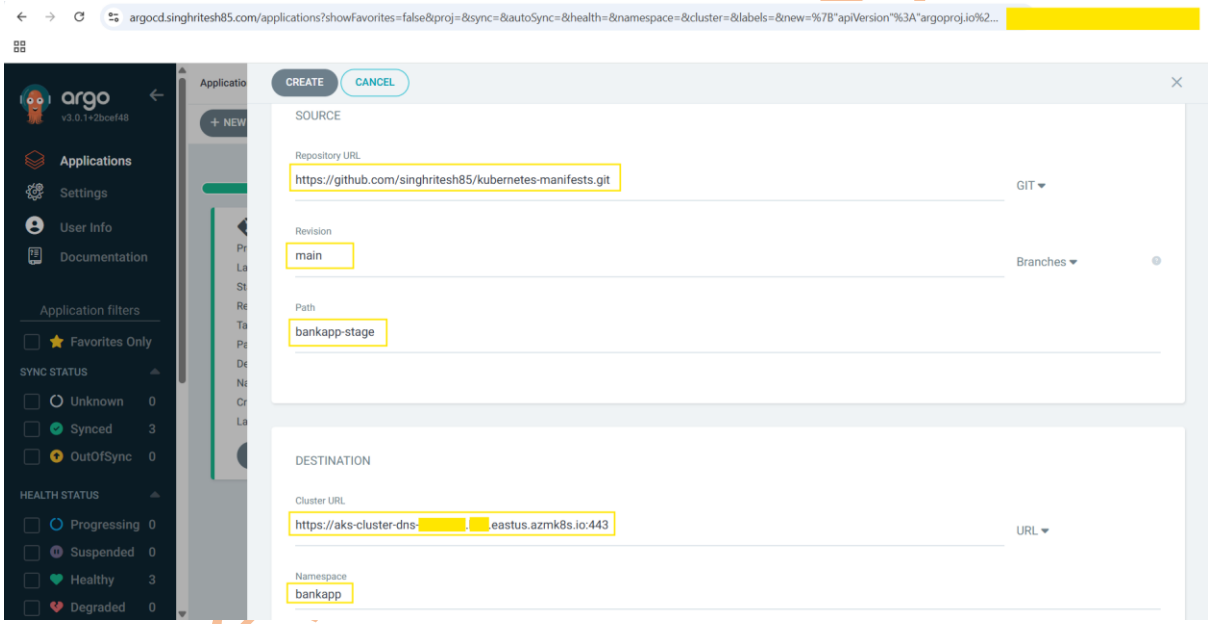
Now Docker Image is ready in ECR (Elastic Container Registry). I had created two ECRs bankapp-1 (for dev cluster) and bankapp-2 (for stage-cluster). Then I created the ArgoCD Application to deploy the pods on Kubernetes Clusters (in dev and stage environment) using these Docker Images as shown in the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

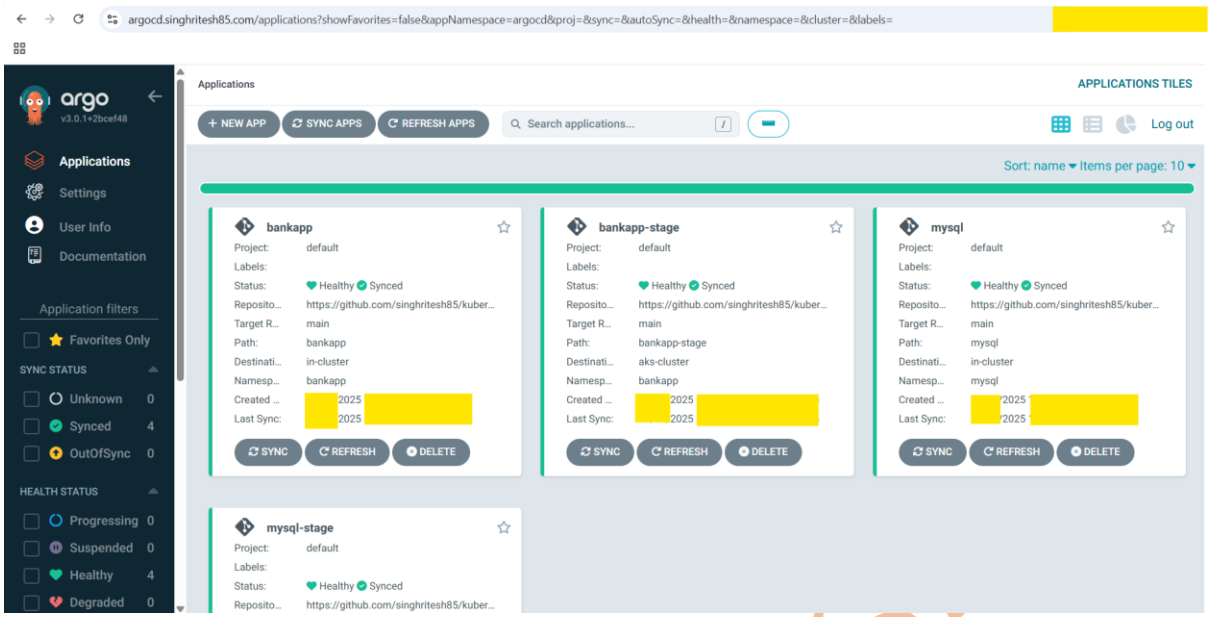



argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B"apiVersion"%3A"argoproj.io%2Fv1al...
argo
v3.0.1+2bcef48
Applications
Settings
User Info
Documentation
Application filters
Favorites Only
SYNC STATUS
Unknown 0
Synced 2
OutOfSync 0
HEALTH STATUS
Progressing 0
Suspended 0
Healthy 2
Degraded 0
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...
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
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

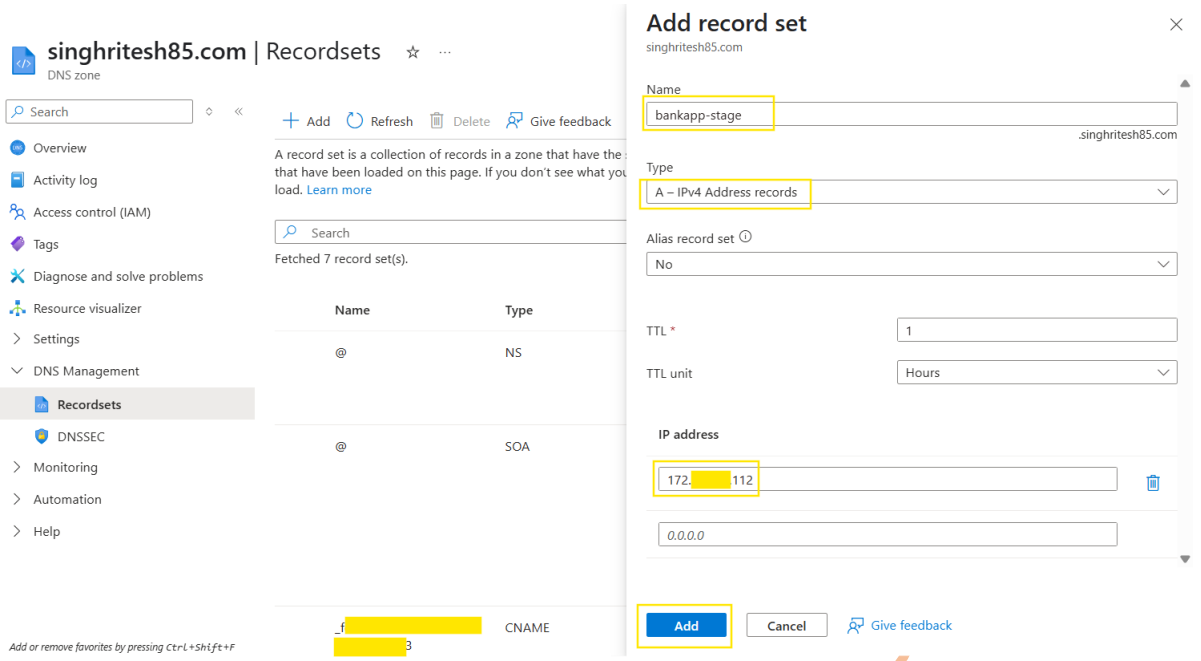
```
[root@ ~]# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster
secret/ingress-tls created
```

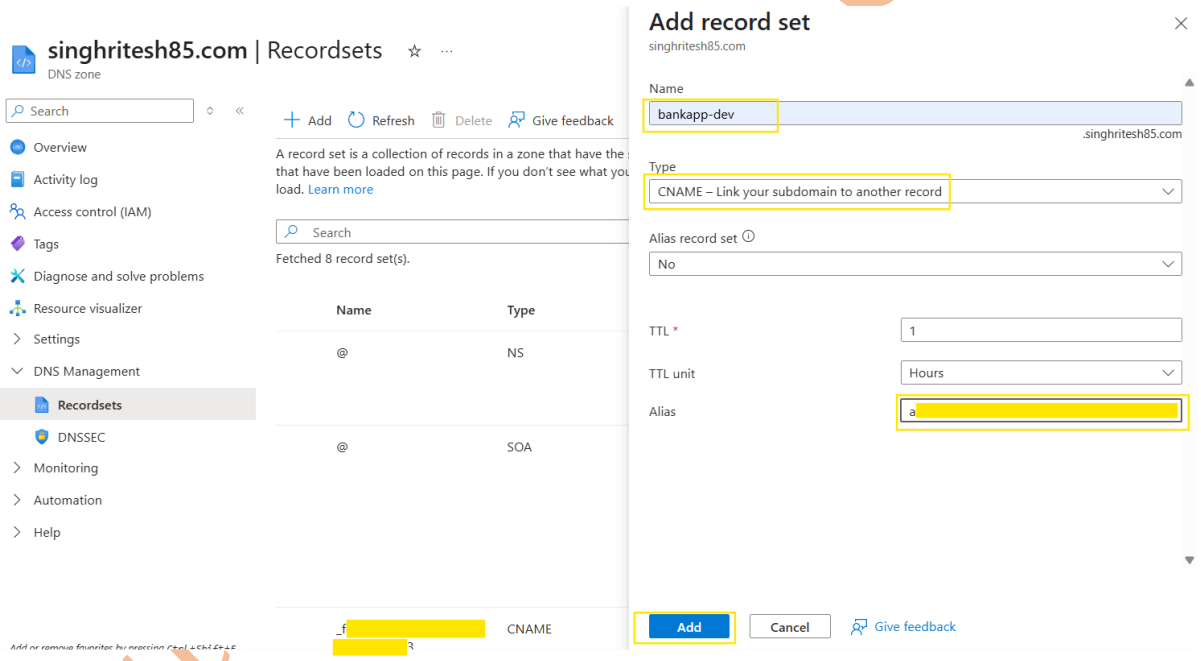
```
[root@ ~]# kubectl apply -f ingress-rule-stage.yaml --context=aks-cluster
ingress.networking.k8s.io/bankapp-ingress-stage created
[root@ ~]# kubectl apply -f ingress-rule-dev.yaml --context=eks-demo-cluster-dev
ingress.networking.k8s.io/bankapp-ingress-dev created
```

```
[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
```

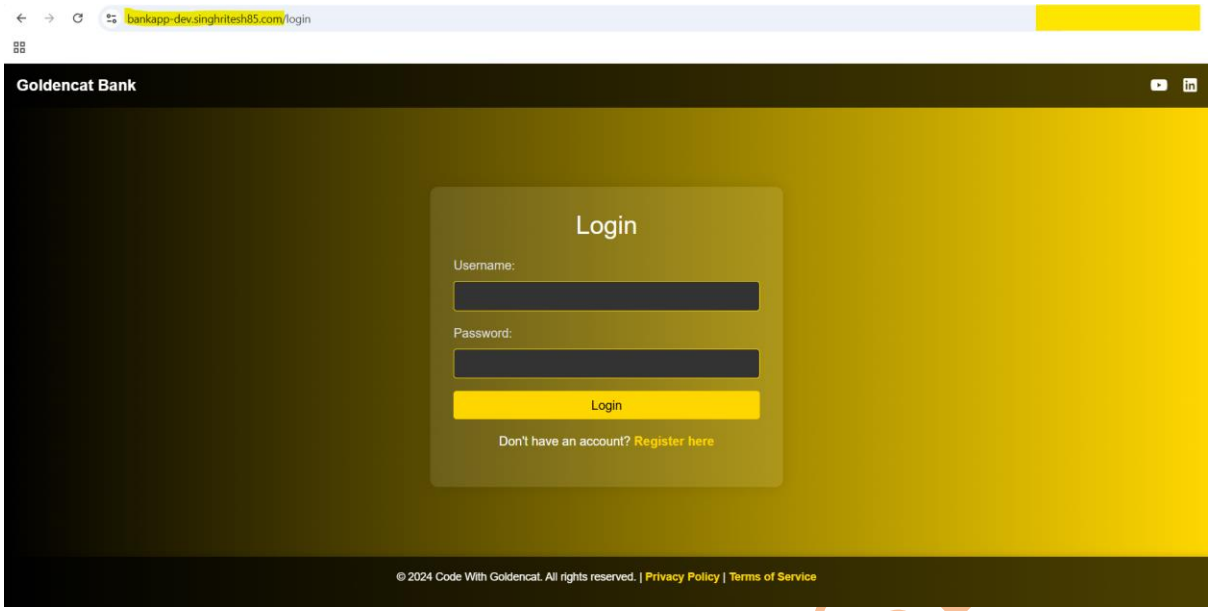
```
[root@ ~]# cat ingress-rule-stage.yaml
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=aks-cluster
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress-stage
  namespace: bankapp
  annotations:
    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
```

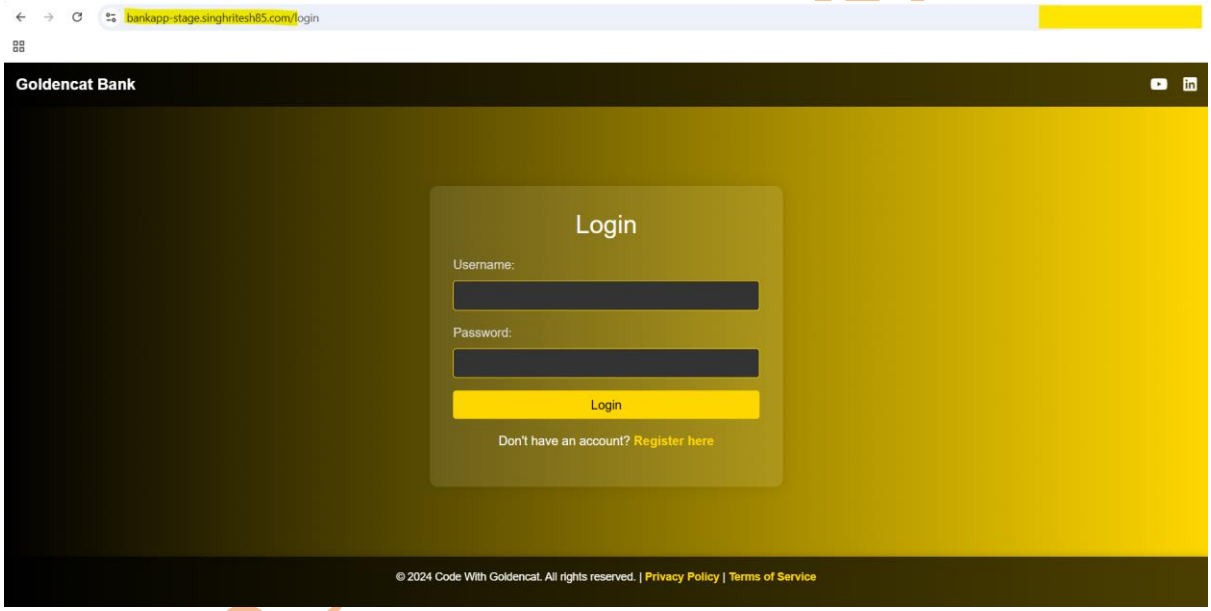
```
[root@ ~]# kubectl get ing -A --watch --context=aks-cluster
NAMESPACE   NAME                CLASS   HOSTS                          ADDRESS          PORTS     AGE
bankapp     bankapp-ingress-stage   nginx   bankapp-stage.singhritesh85.com   172.  .112   80, 443
vault       vault-ingress           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   AGE
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
```





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

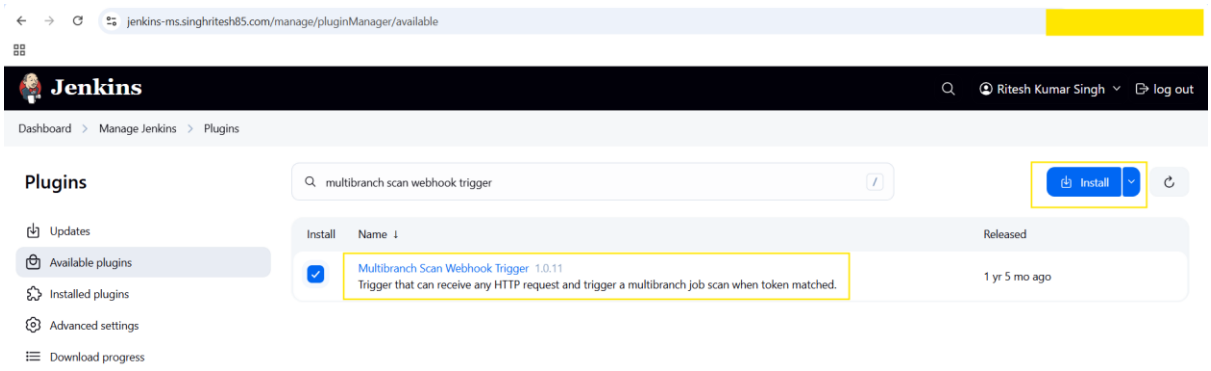


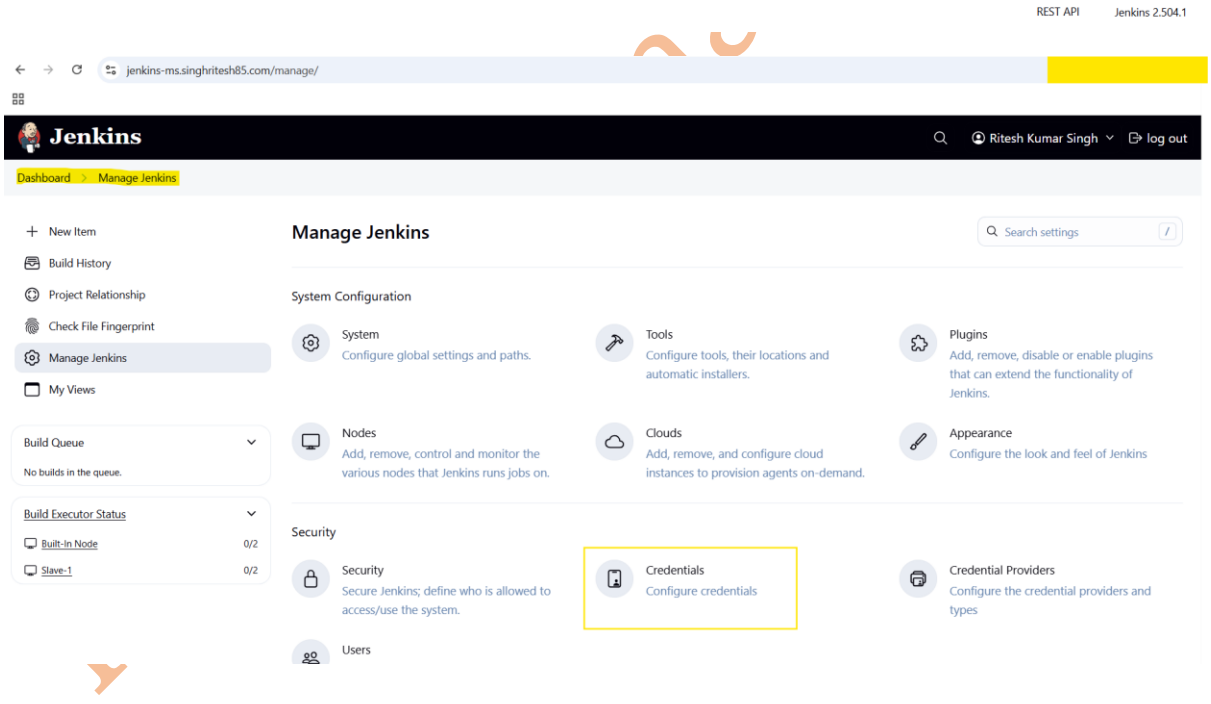


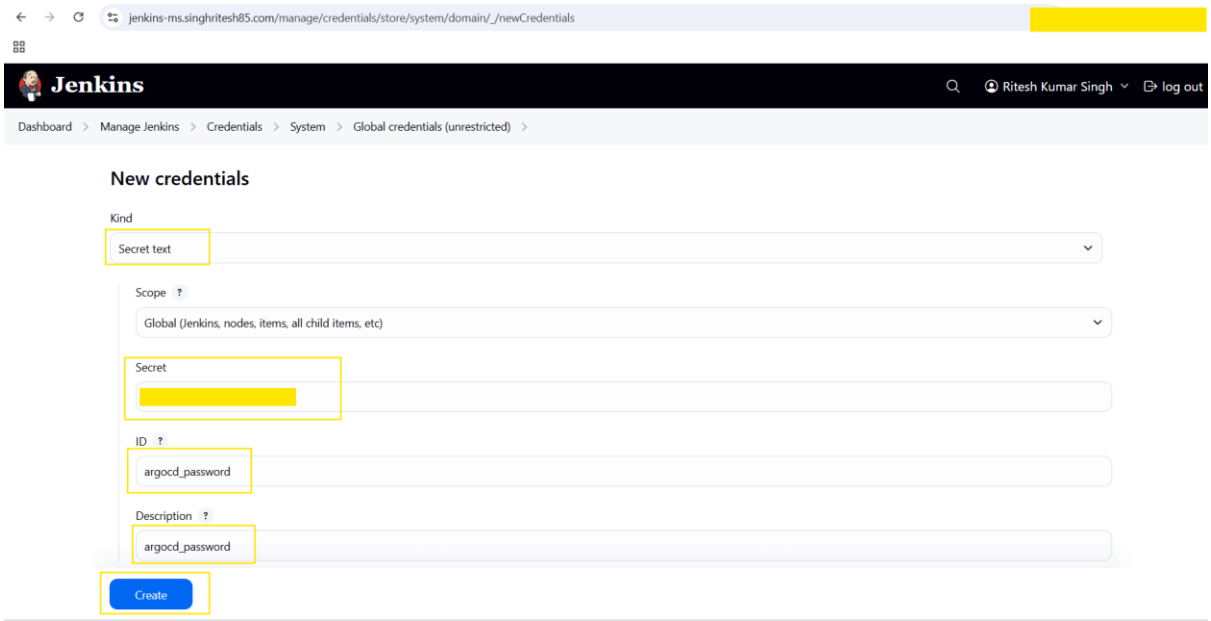
In Manual Approach the disadvantage is for a new Release, manually you need to Run the Jenkins Job then delete the corresponding Application in the ArgoCD then recreate it manually.

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.

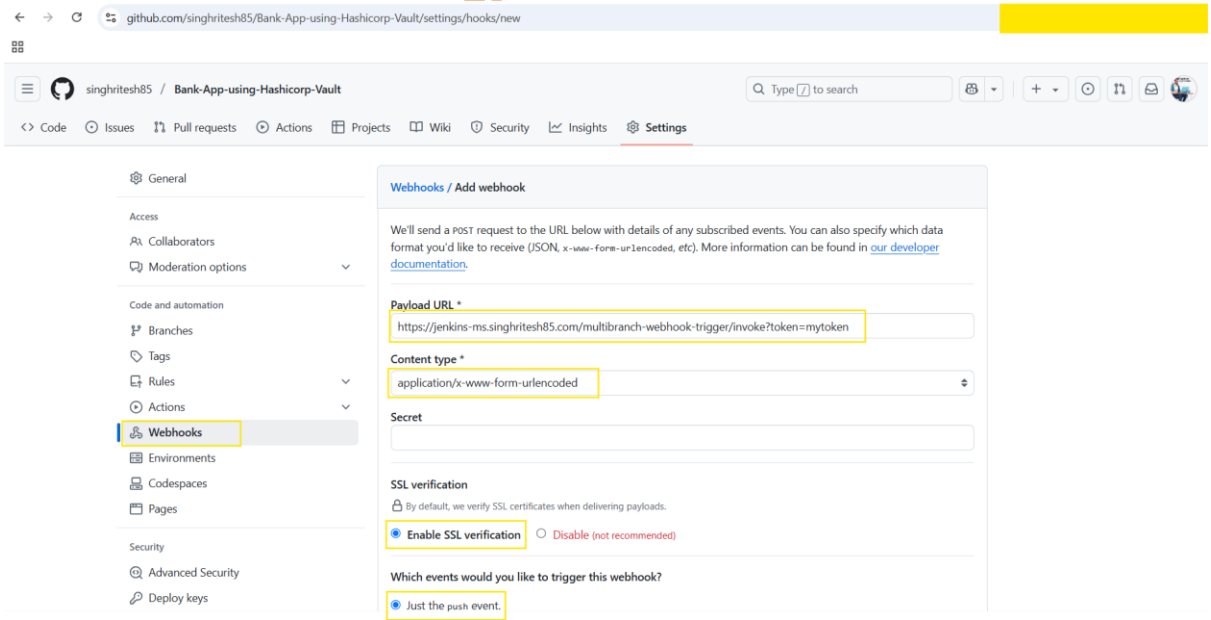


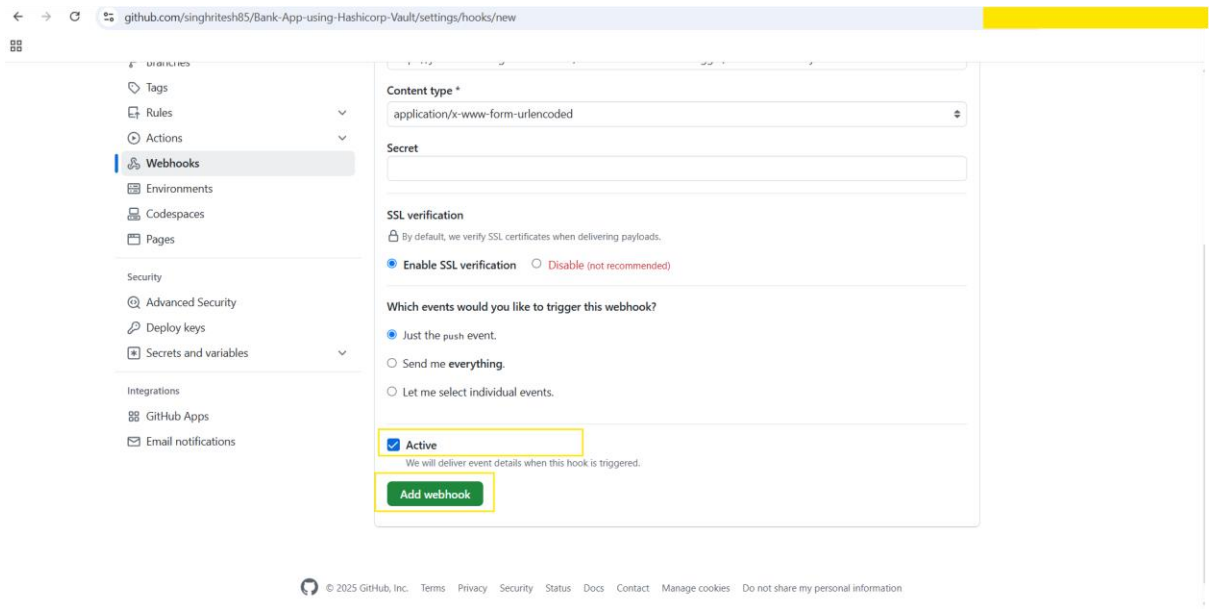


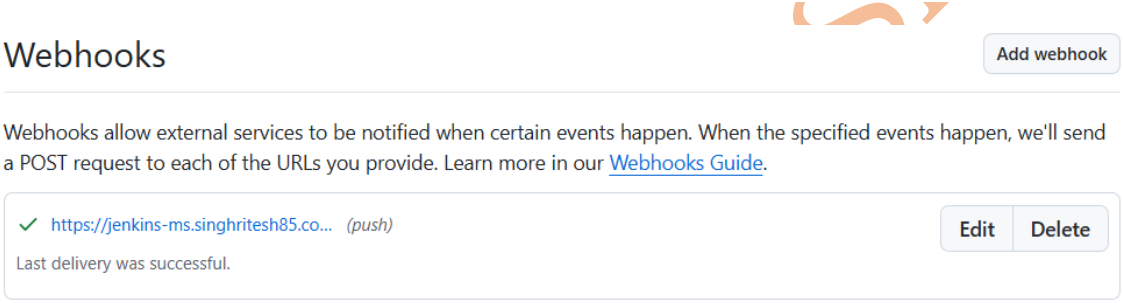


In Manual Approach (Module-1) I had already shown how to create the Jenkins credentials for jenkins slave and GitHub Account which was jenkins-cred and github-cred respectively. In the similar way create the Jenkins Credentials here. Create the kubernetes secrets **ingress-tls**, **vault-tls-secret** 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).

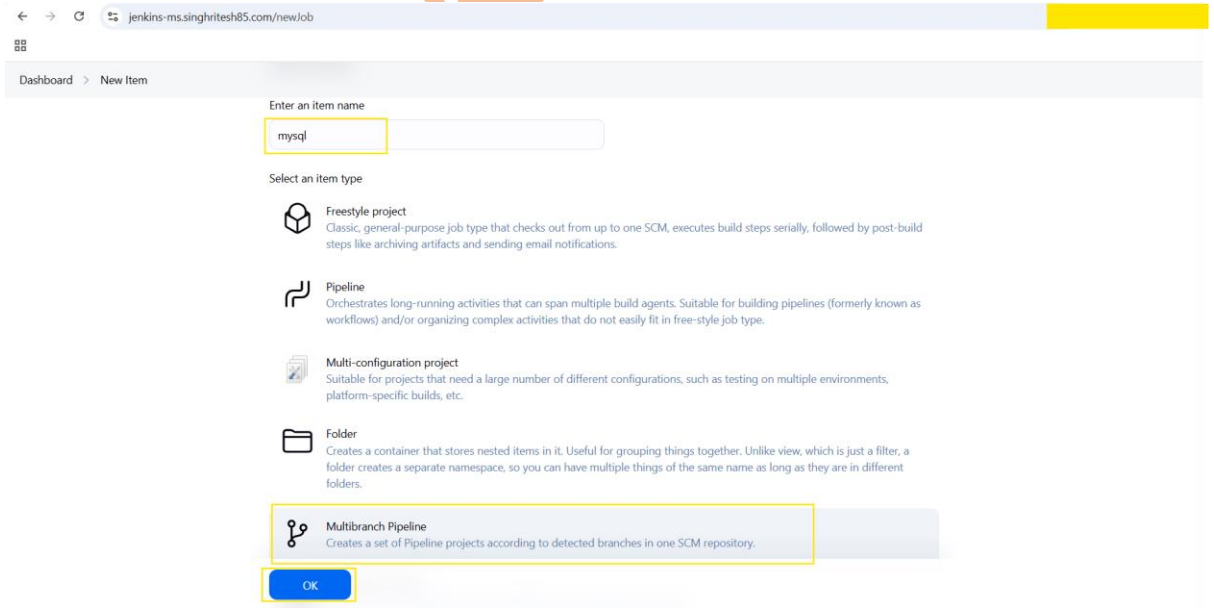
Then go to Github Repo for your project in this case I use the GitHub Repository https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git for source code, and then go to Repository Settings > Webhooks and provide the Payload URL **https://jenkins-ms.singhritesh85.com/multibranch-webhook-trigger/invoke?token=mytoken**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

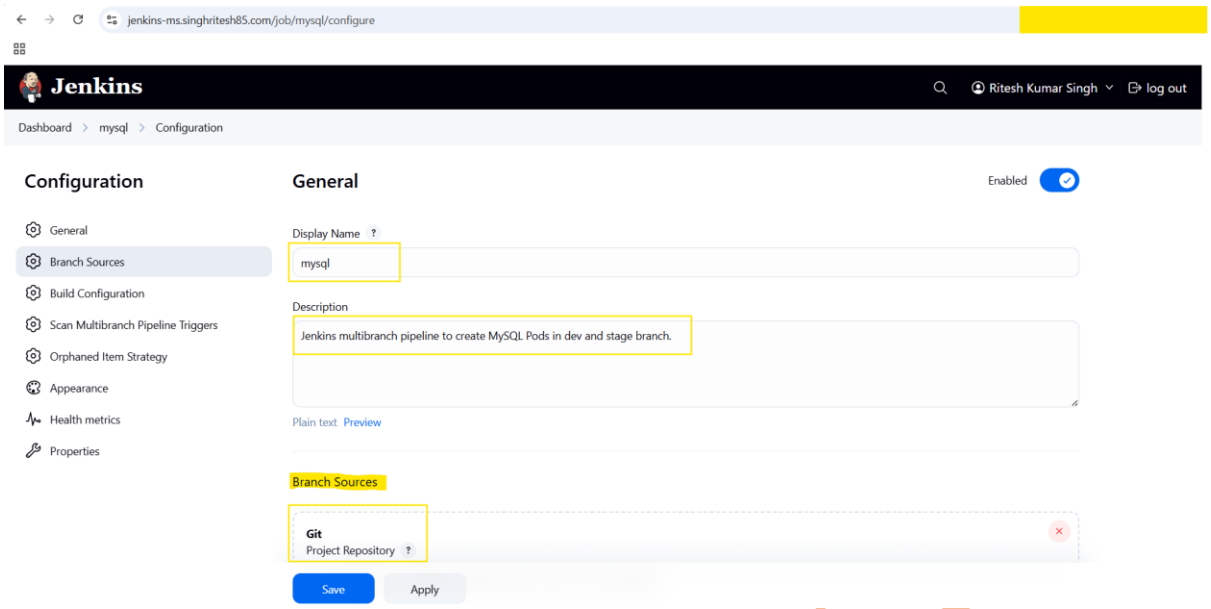


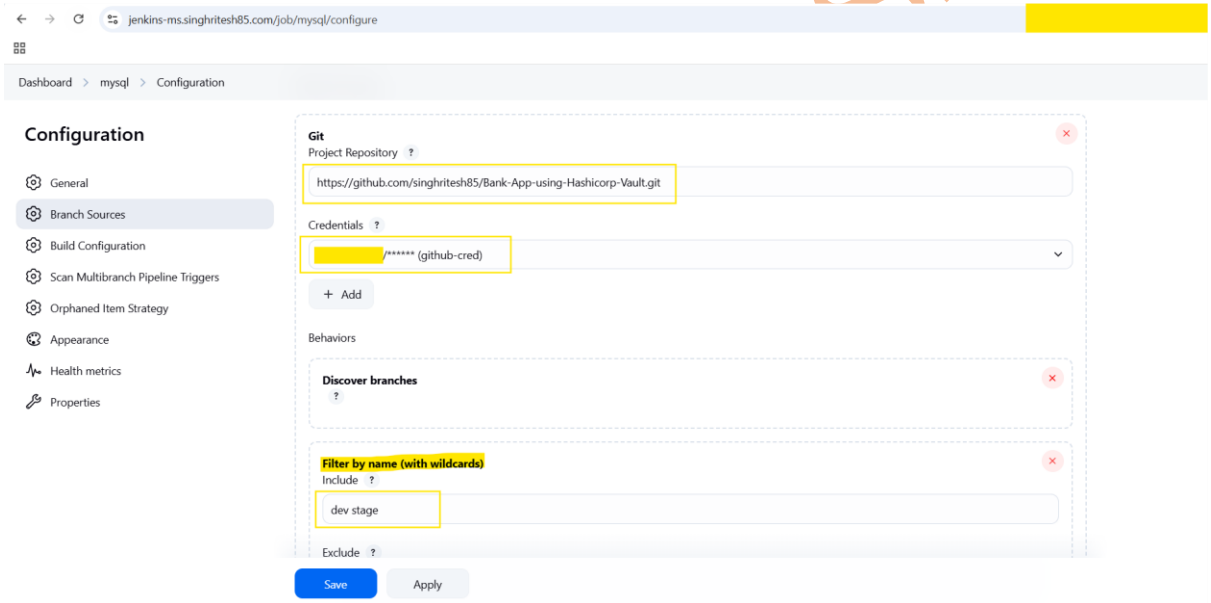


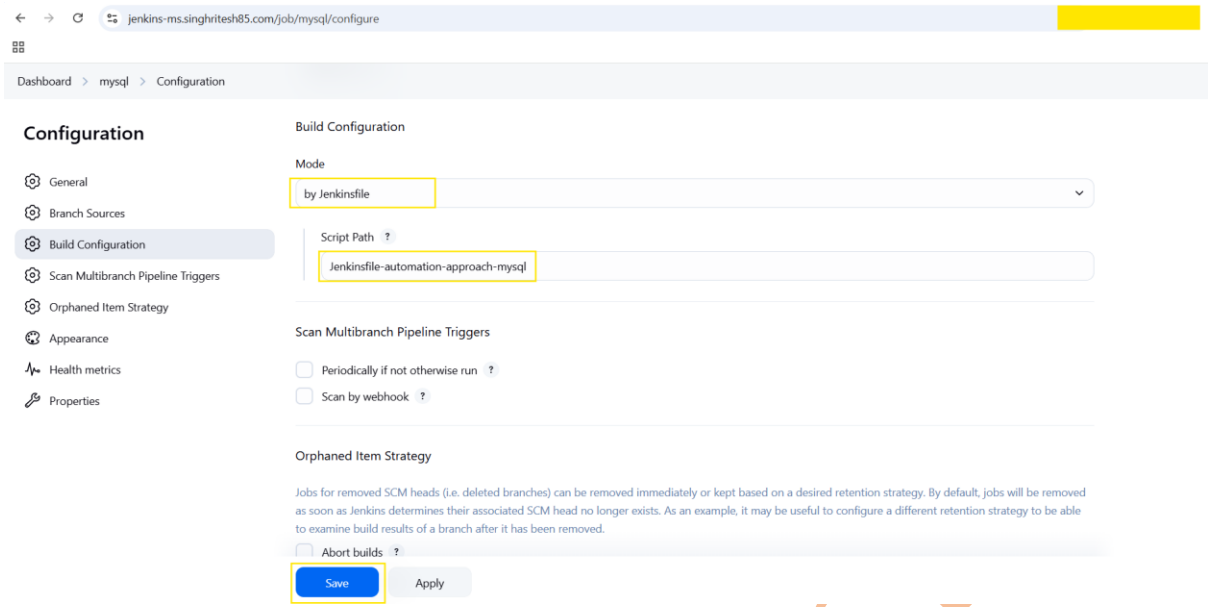


Then create the multibranch pipeline for mysql and then for bankapp (with webhook) as shown in the screenshot attached below.

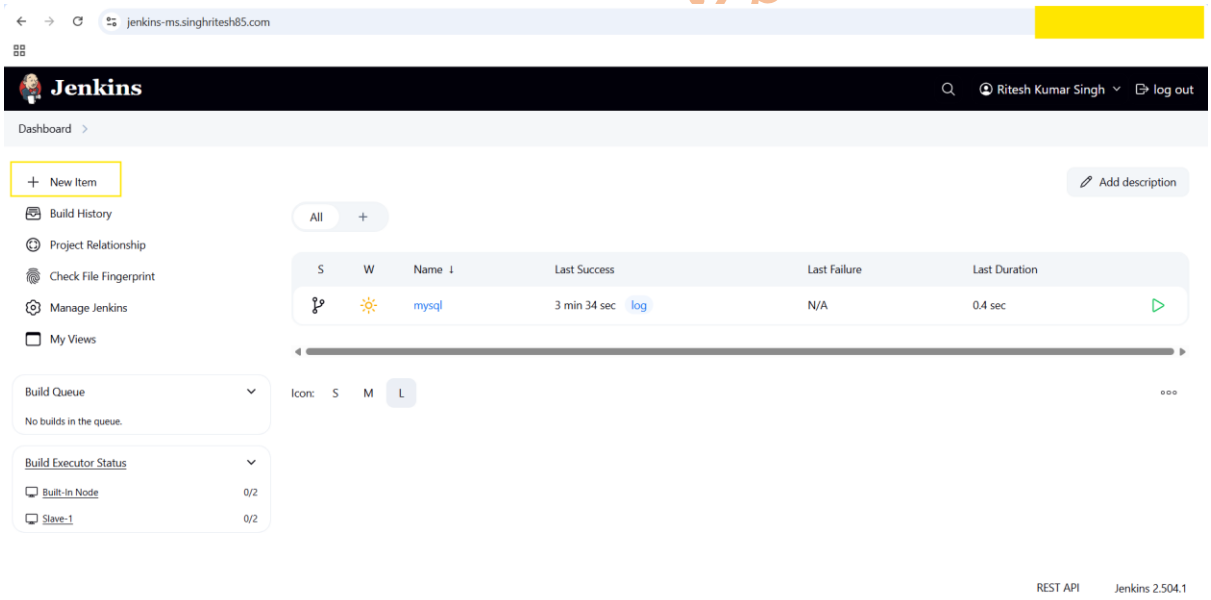


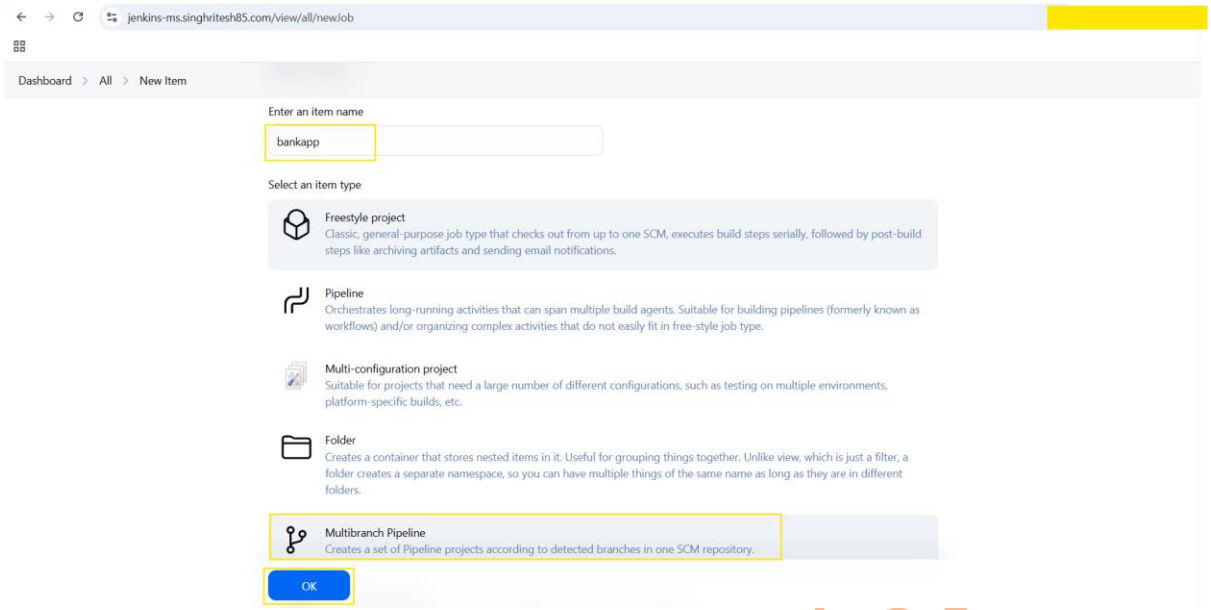

jenkins-ms.singhritesh85.com/job/mysql/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > mysql > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
mysql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

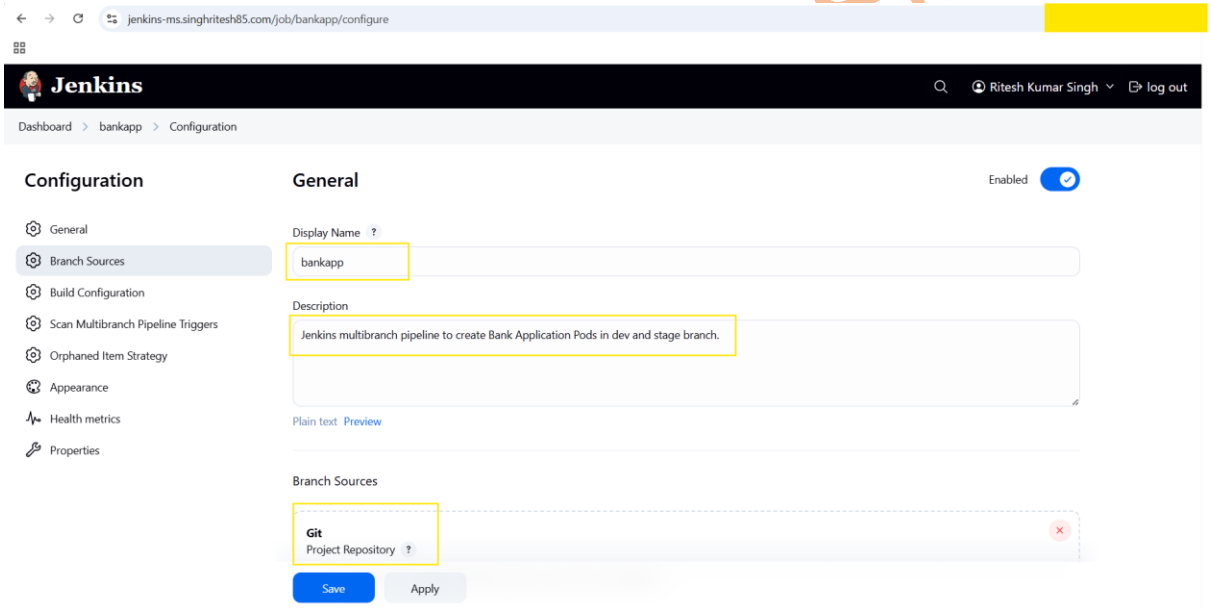

jenkins-ms.singhritesh85.com/job/mysql/configure
Dashboard > mysql > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
Git
Project Repository
https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git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

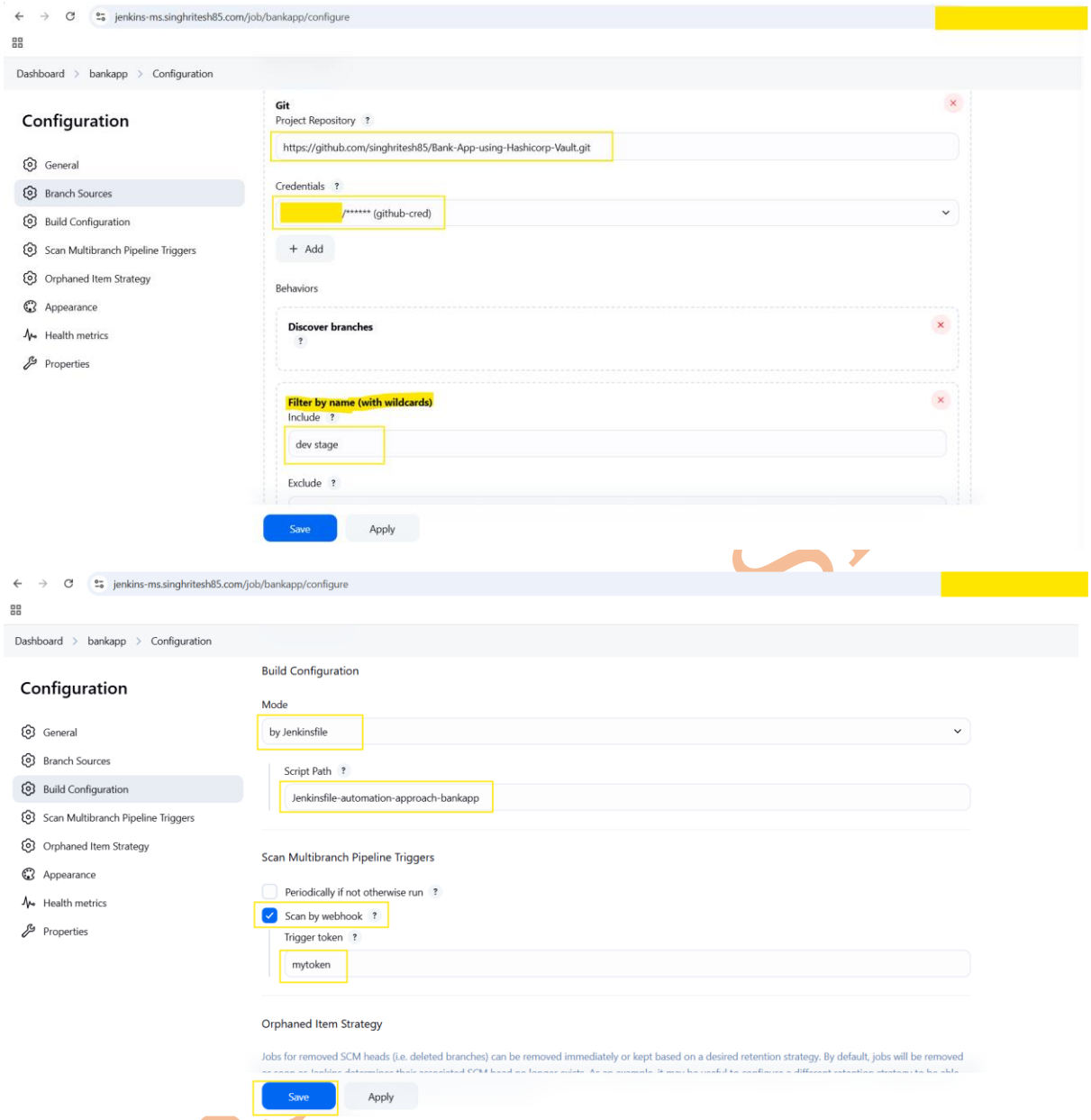


I did not apply the webhook to deploy the MySQL 8 Pods but I used the webhook to deploy bankapp application pods as shown in the screenshot attached below. Below screenshot shows how I had created the Multibranch Pipeline for Jenkins to create bankapp application pods in EKS and AKS Clusters.

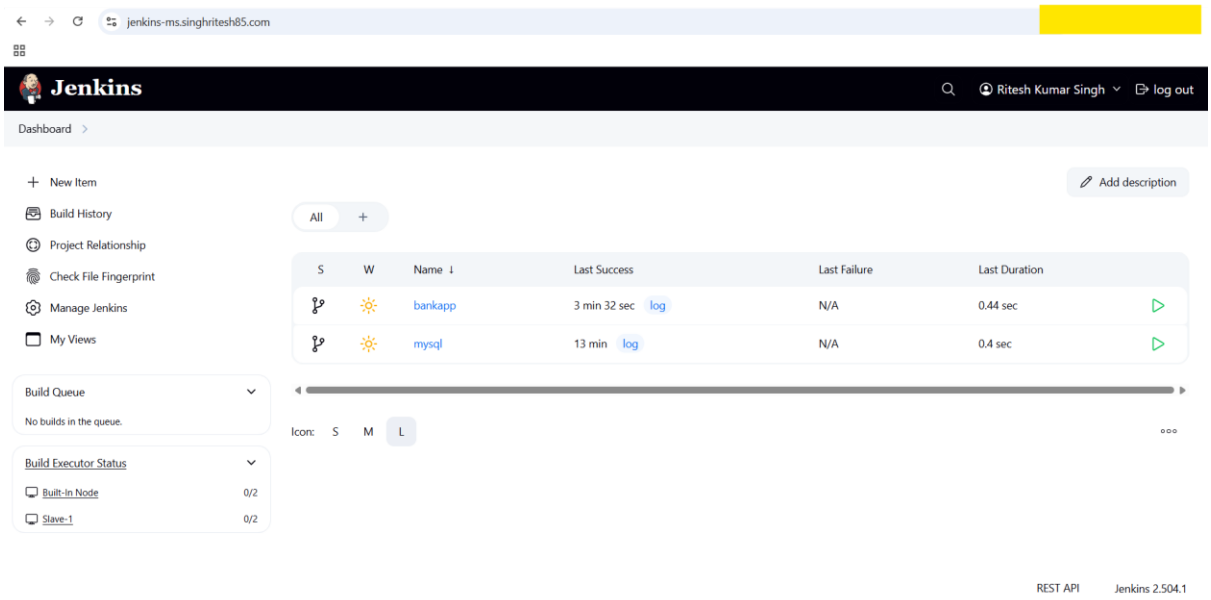


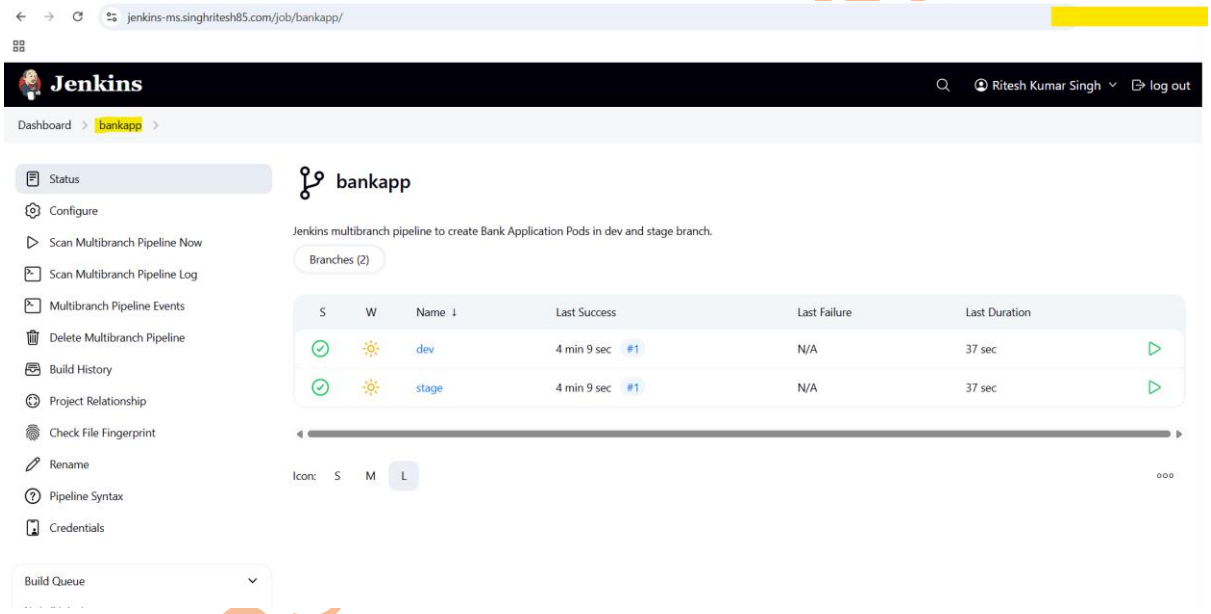

jenkins-ms.singhritesh85.com/view/all/newJob
Dashboard > All > New Item
Enter an item name
bankapp
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.
OK

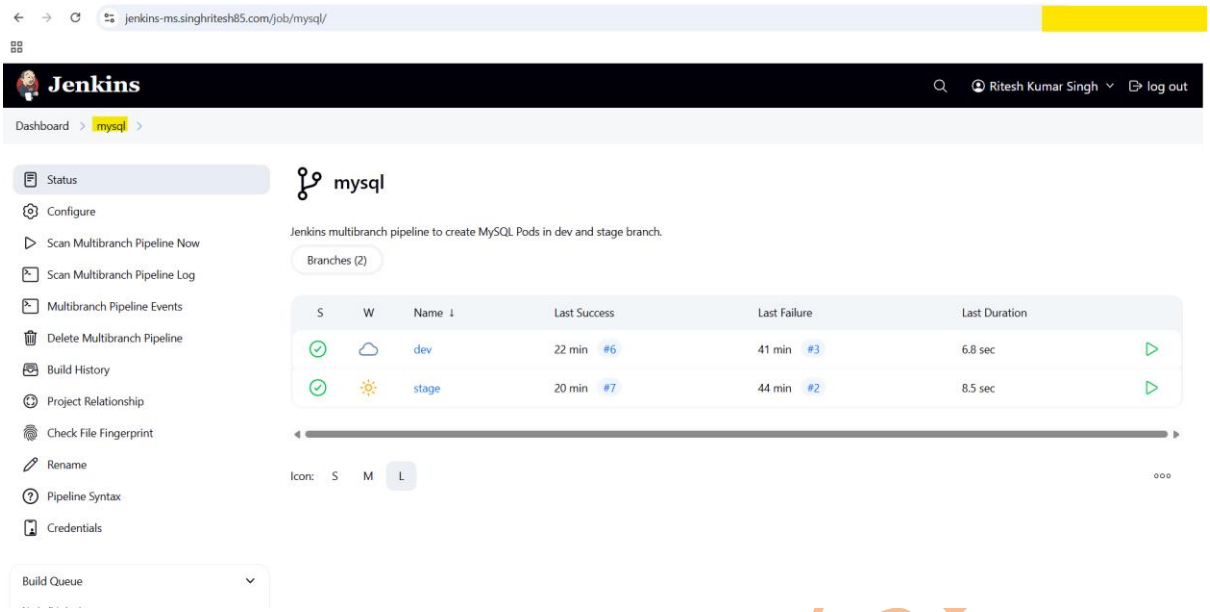

jenkins-ms.singhritesh85.com/job/bankapp/configure
Jenkins
Ritesh Kumar Singh
log out
Dashboard > bankapp > Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
General
Enabled
Display Name
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



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


jenkins-ms.singhritesh85.com
Jenkins
Ritesh Kumar Singh
log out
Dashboard
New Item
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
bankapp
3 min 32 sec
log
N/A
0.44 sec
mysql
13 min
log
N/A
0.4 sec
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node
0/2
Slave-1
0/2
Icon: S M L
REST API
Jenkins 2.504.1


jenkins-ms.singhritesh85.com/job/bankapp/
Jenkins
Ritesh Kumar Singh
log out
Dashboard
bankapp
Status
Configure
Scan Multibranch Pipeline Now
Scan Multibranch Pipeline Log
Multibranch Pipeline Events
Delete Multibranch Pipeline
Build History
Project Relationship
Check File Fingerprint
Rename
Pipeline Syntax
Credentials
bankapp
Jenkins multibranch pipeline to create Bank Application Pods in dev and stage branch.
Branches (2)
S
W
Name
Last Success
Last Failure
Last Duration
dev
4 min 9 sec
#1
N/A
37 sec
stage
4 min 9 sec
#1
N/A
37 sec
Icon: S M L
Build Queue



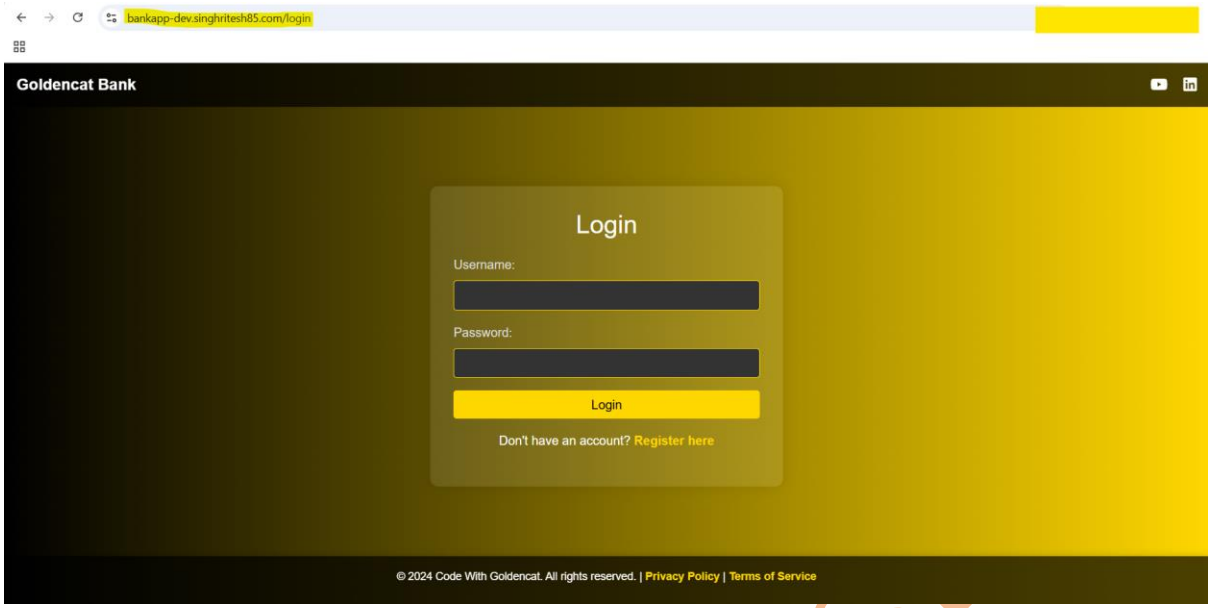
Then create the ingress rule and do the entry for LoadBalancer DNS Name and LoadBalancer External IP in Azure DNS Zone to create the Record Set of type CNAME and A Type for EKS and AKS Cluster respectively as I discussed in Module-1.

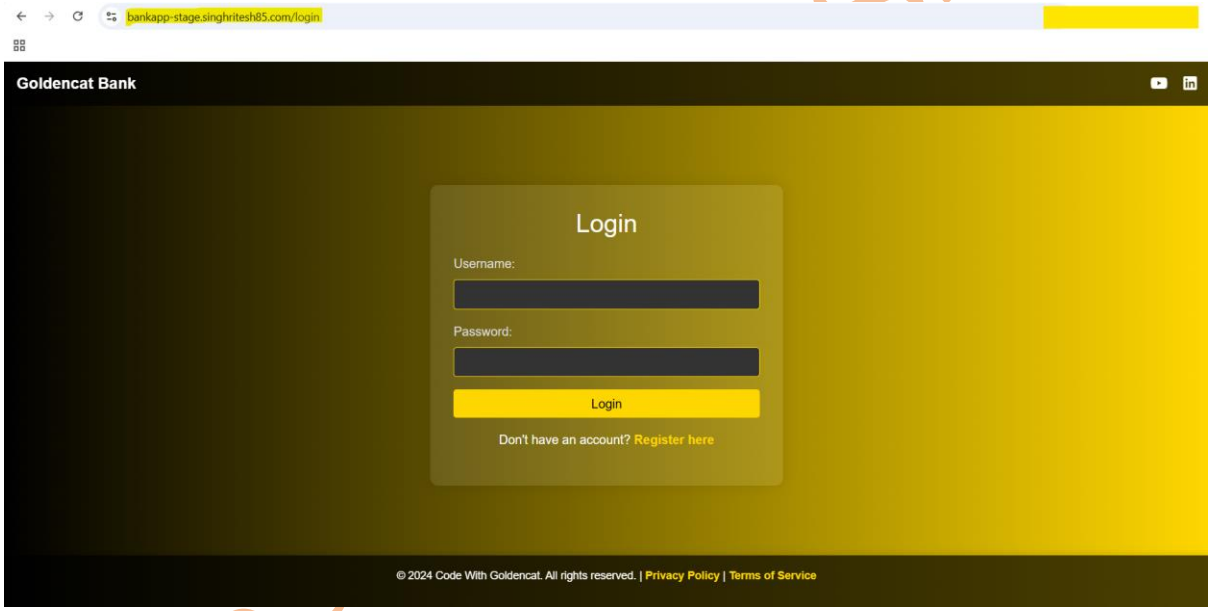
```
cat ingress-rule-dev.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress-dev
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: bankapp-dev.singhritesh85.com
    http:
      paths:
  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

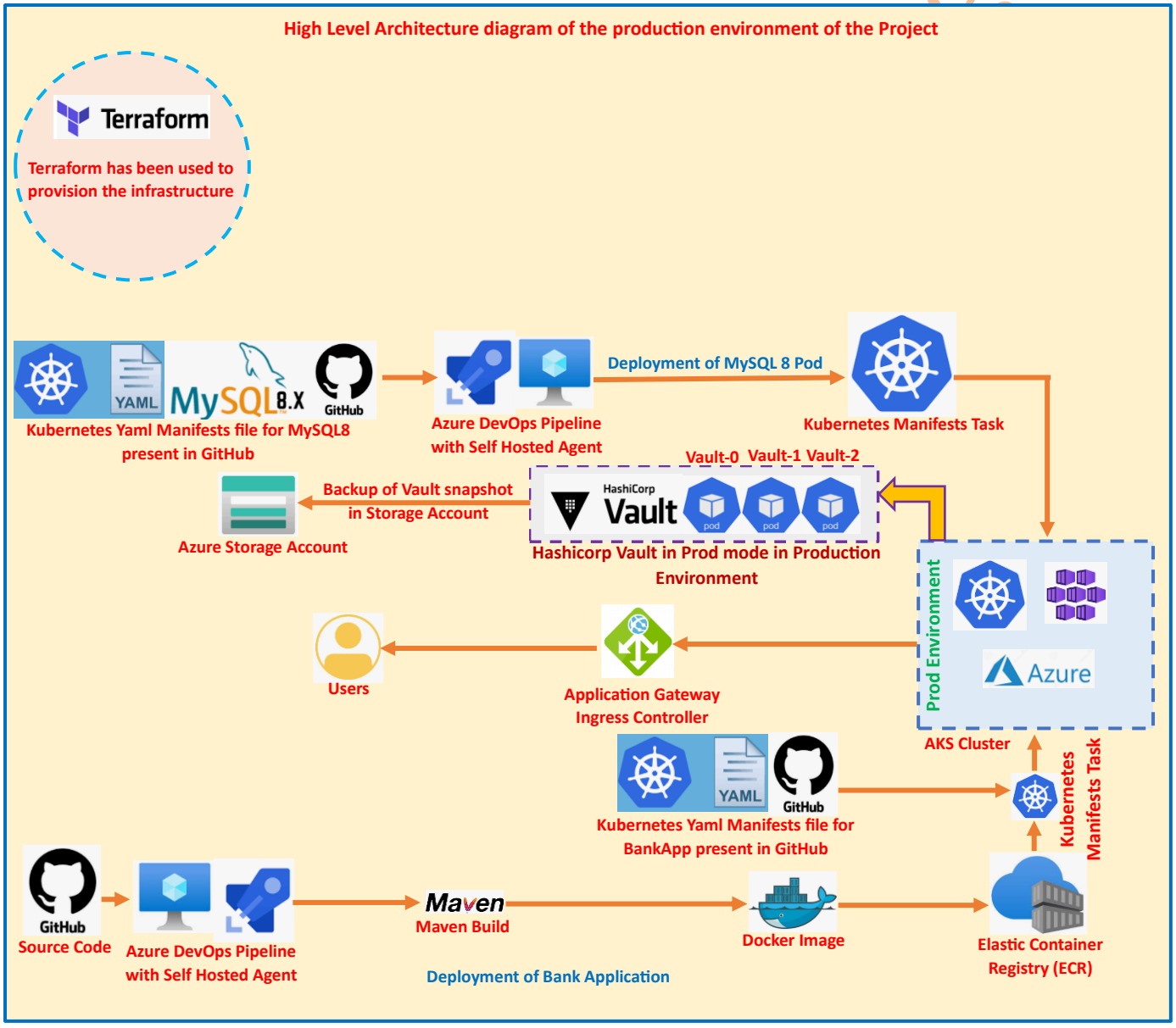
Finally, I was able to access the Bank Application as shown in the screenshot attached below.


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


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

Module-3: [Production Environment]

The production environment completely existed in the Azure and I used Azure DevOps Pipeline as CI/CD Tool for this project. I had not discussed here about the Integration of SonarQube, Artifactory and Trivy Image Scan. If you are interested in the same the please refer the project present in GitHub Repository https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git, there I also discussed about Monitoring using Prometheus and Grafana, Log Aggregation using Loki and GitHub Branching Strategies. In **Module-3** of this project I discussed mainly about the Deployment in Production environment and storage of MySQL credentials in Hashicorp vault. The Hashicorp Vault Backup I took using the Cron Job which I will be discussed later in this module.



I had created the Azure Resources first 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-prod**. Before running the terraform script I ran the shell script present in the directory **terraform-bankapp-multibranch-multicloud-prod** as shown in the screenshot attached below. This shell script will install the aws cli, kubectl and helm.

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

Then I logged-out and login again and after that 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.aks.null_resource.kubectl: Creation complete after 3s [id=                    ]

Apply complete! Resources: 33 added, 0 changed, 0 destroyed.

Outputs:

acr_azure_loki_vm_private_ip_container_sas_url = <sensitive>
[root@                main]#
```

```
[root@              main]# terraform output -json acr_azure_loki_vm_private_ip_container_sas_url
{"acr_login_server":"bankappprodcontainer24registry.azurecr.io","azure_devopsagent_vm":"10.      .4","sas_url_query_string":"https://vaultbackup24prod.blob.cor
e.windows.net/dexter?sv=
        }
```

After execution of the Terraform script, you can get the Blob SAS URL using the command **terraform output -json acr_azure_loki_vm_private_ip_container_sas_url** as shown in the screenshot attached above. After creation of the resources a kubeconfig file was generated.

I had installed Hashicorp Vault Prod in AKS Cluster using the commands as shown in the screenshot attached 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

```
[root@                ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...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: Wed May 21 08:49:54 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
NAME                                    READY   STATUS    RESTARTS   AGE
vault-0                                 0/1     Running   0          63s
vault-1                                 0/1     Running   0          62s
vault-2                                 0/1     Running   0          62s
vault-agent-injector-                   1/1     Running   0          63s
```

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
error: you must specify at least one command for the container
[root@Terraform-Server ~]# 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:
Unseal Key 2:
Unseal Key 3:
Unseal Key 4:
Unseal Key 5:

Initial Root Token: h                          A

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]A
token_accessor       [redacted]
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@[redacted] ~]# kubectl exec -ti vault-2 -n vault -- vault operator raft join http://vault-0.vault-internal:8200
Key       Value
---       -----
Joined    true
[root@[redacted] ~]#
```

```
[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@ ~]# 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 A Type in Azure DNS Zone to access the vault in prod mode in production cluster.

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

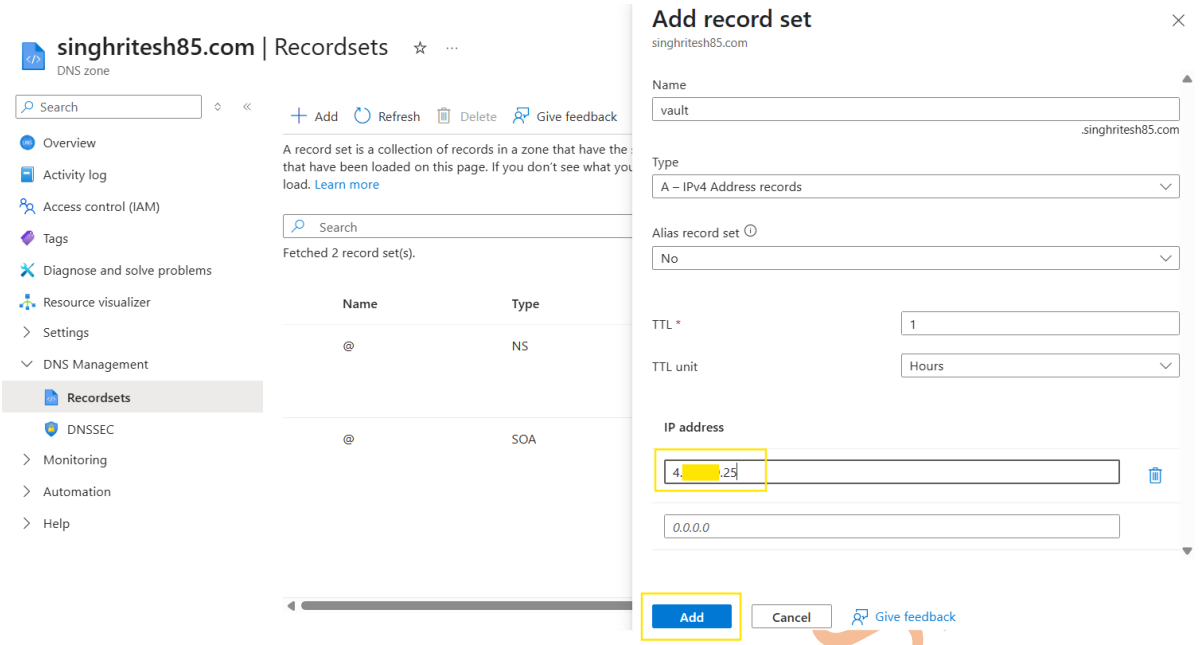
```
[root@ ~]# kubectl apply -f vault-ingress-rule-prod.yaml
ingress.networking.k8s.io/vault-ingress created
[root@ ~]# cat vault-ingress-rule-prod.yaml
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace vault
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-tls
  rules:
  - host: vault.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: vault-active
            port:
              number: 8200

[root@ ~]# kubectl get ing -A --watch
NAMESPACE   NAME            CLASS                       HOSTS                   ADDRESS        PORTS     AGE
vault       vault-ingress   azure-application-gateway   vault.singhritesh85.com   4.    .25   80, 443   13s
```

cat vault-ingress-rule-prod.yaml

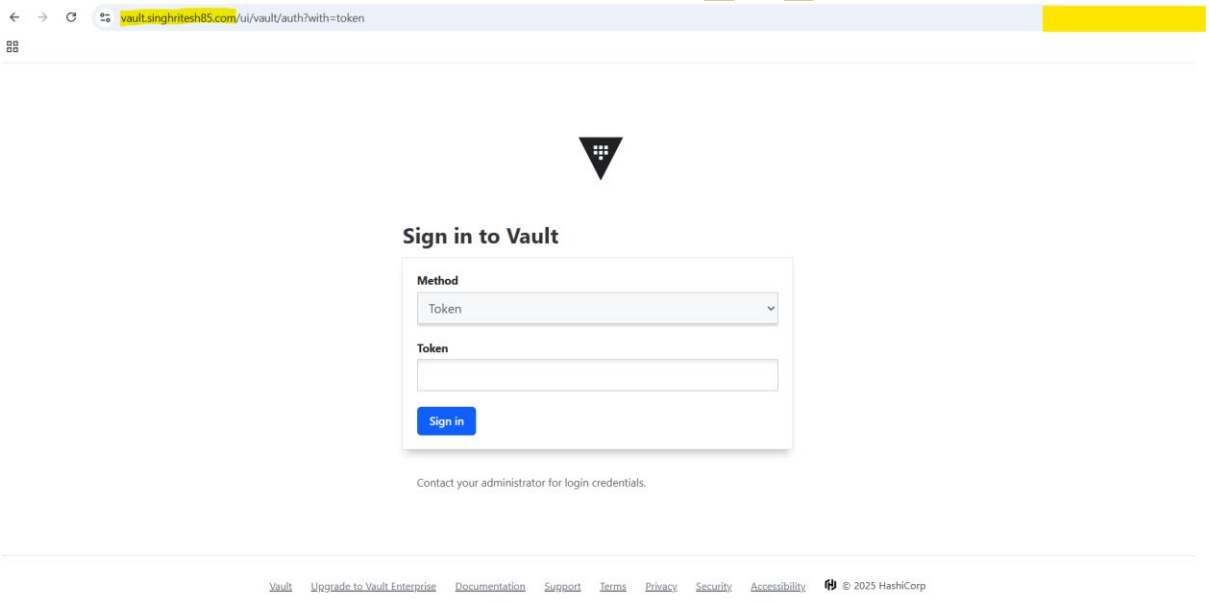
kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace vault

---

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: vault-ingress
  namespace: vault
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-tls
  rules:
  - host: vault.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 production cluster (Private AKS Cluster) as shown in the screenshot attached below.

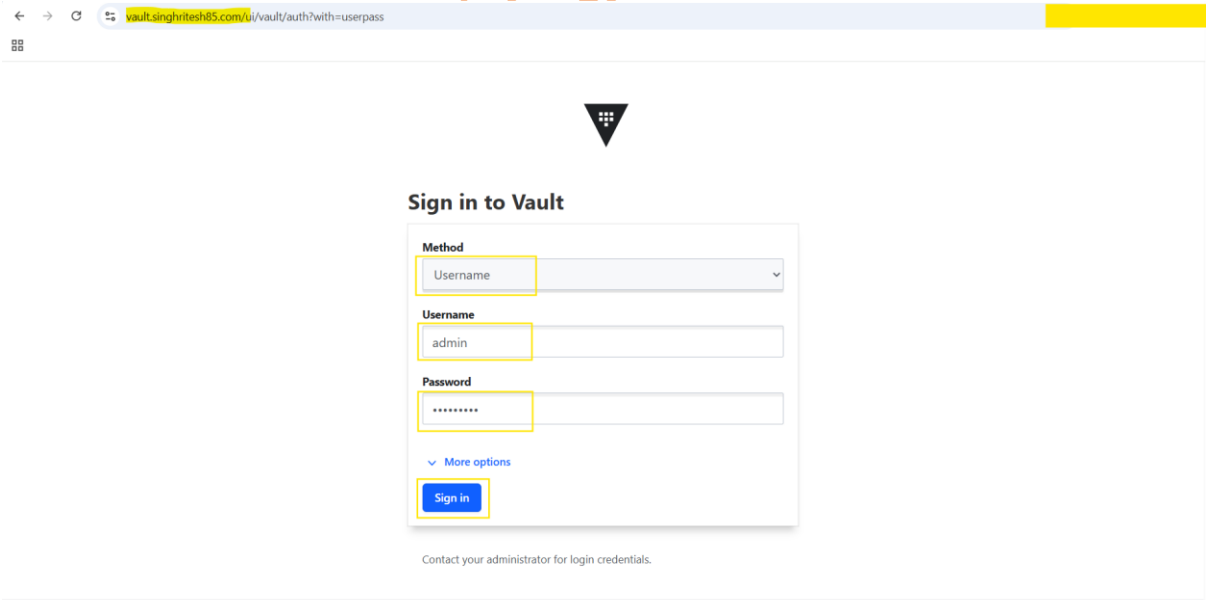


For the first time login method into the vault is Token. The Token which I got from the vault-0 pod in production cluster was used to login into the Hashicorp vault prod mode in production cluster. Then I enabled the vault auth userpass and created a username with password with Administrator privileges which I will use to login in to the Hashicorp Vault prod mode in production 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
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
/ $
```

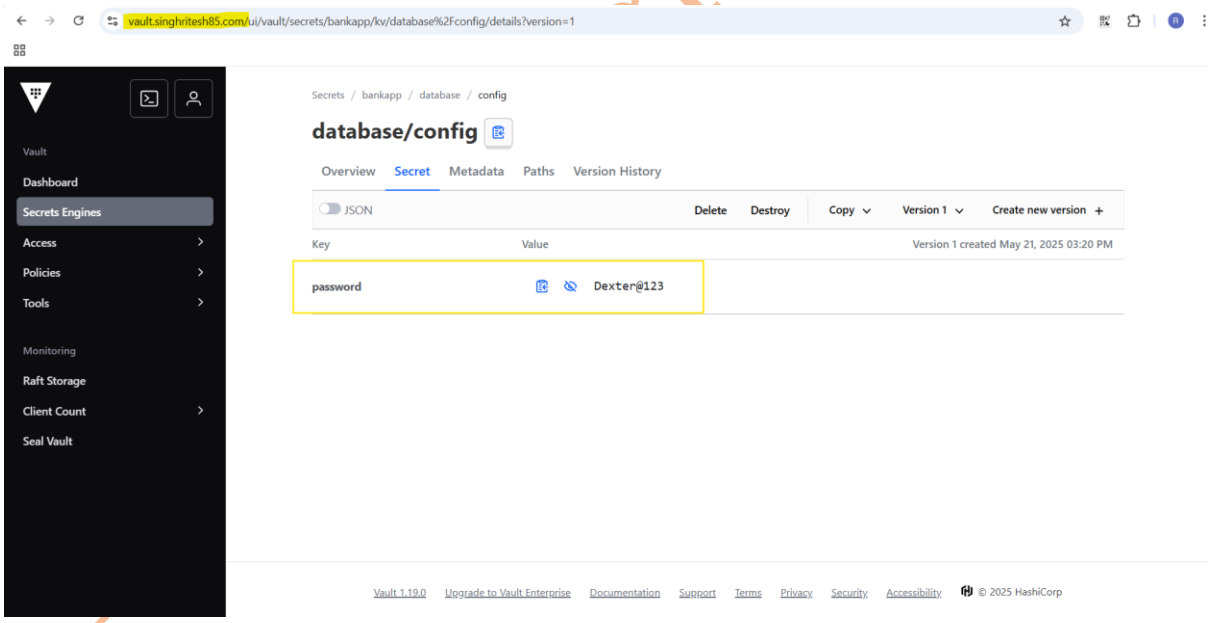
Then I logged-in with the created username and password as shown in the screenshot attached below.



Then created the Hashicorp vault secrets in production cluster (AKS Cluster) as shown in the screenshot attached below.

```
[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 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 production 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 Private 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 within a namespace. This namespace should be the same in which the pod exists which tries to get secrets from vault.

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

```
[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 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>

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
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
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

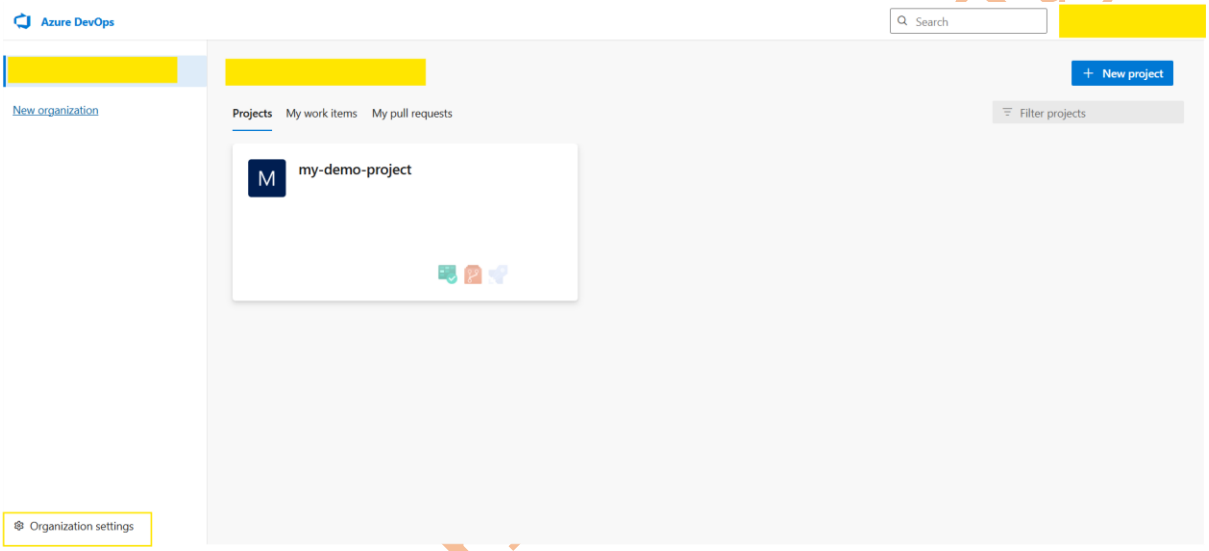
vault write auth/kubernetes/role/bankapp bound_service_account_names=bankapp-sa bound_service_account_namespaces=mysql policies=bankapp-policy ttl=1h

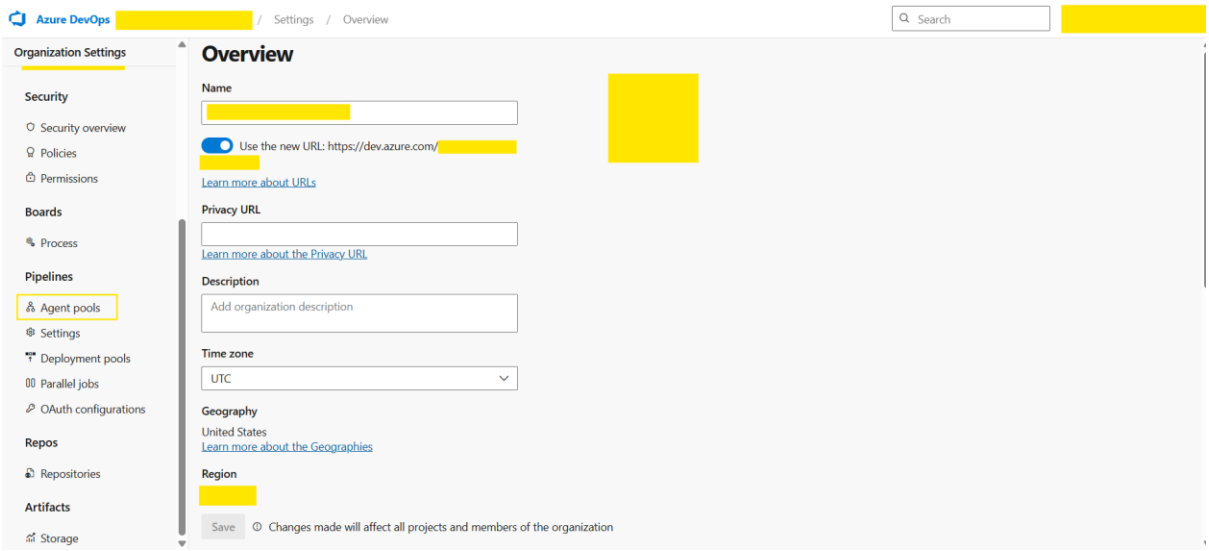
```
[root@                    ~]# kubectl exec -it vault-0 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault write auth/kubernetes/role/bankapp bound_service_account_names=bankapp-sa bound_service_account_namespaces=mysql policies=bankapp-policy ttl=1h
Success! Data written to: auth/kubernetes/role/bankapp
/ $
```

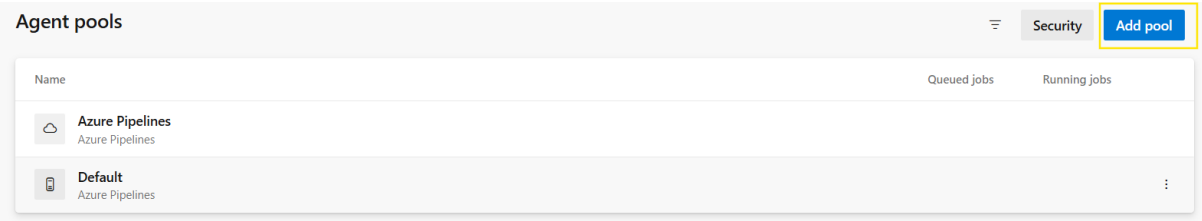
Installation of Azure DevOps Agent of Azure DevOps Agent Azure VM

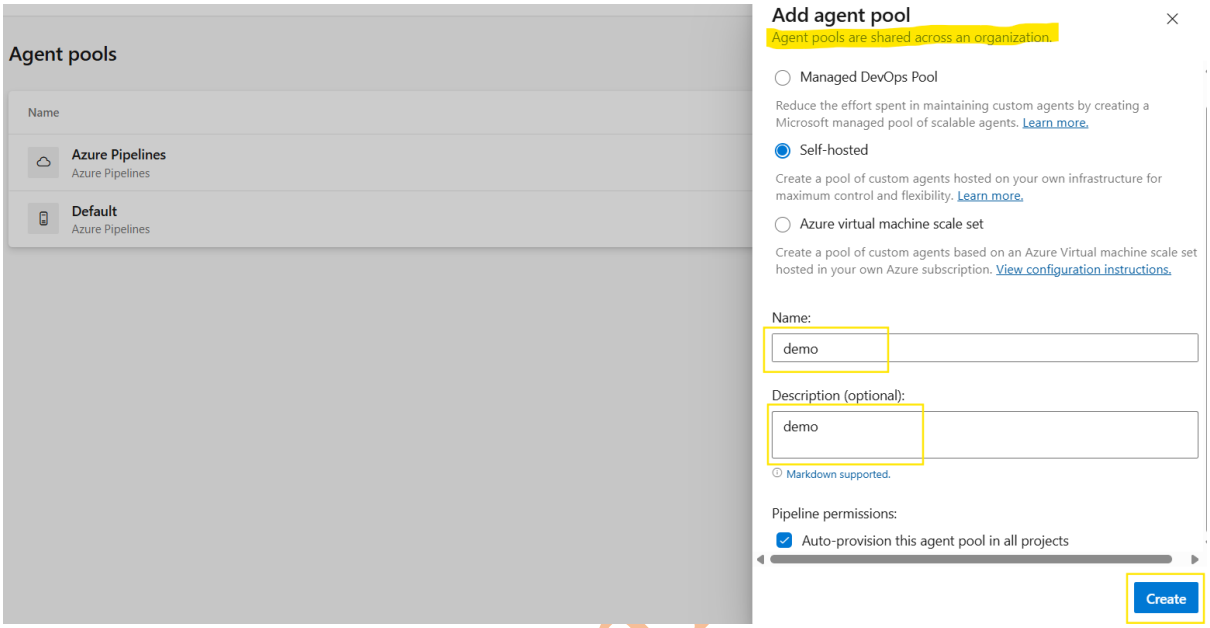
In production for CI/CD pipeline I used Azure DevOps Pipeline. For Azure DevOps Pipeline I had used Self-hosted-Agent and followed the below procedure to install it.

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

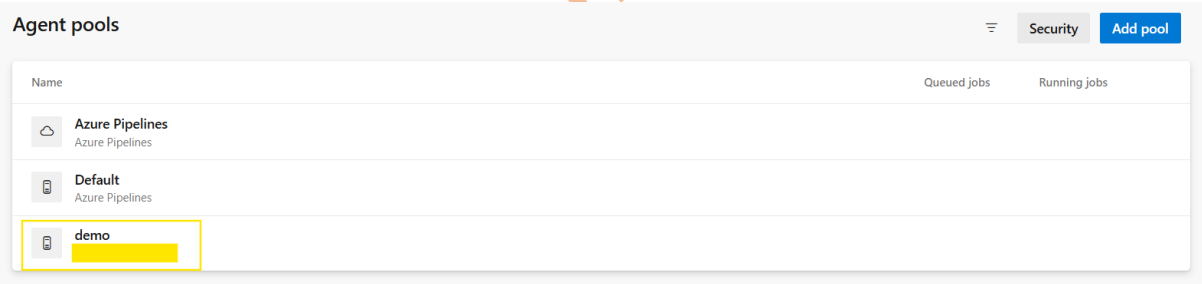




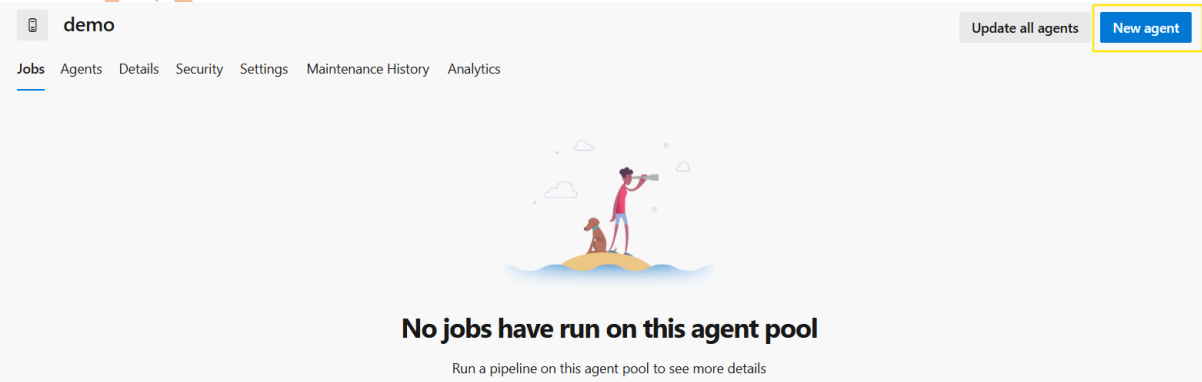




New Agent Pool had been added as shown in the screenshot attached below.



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



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

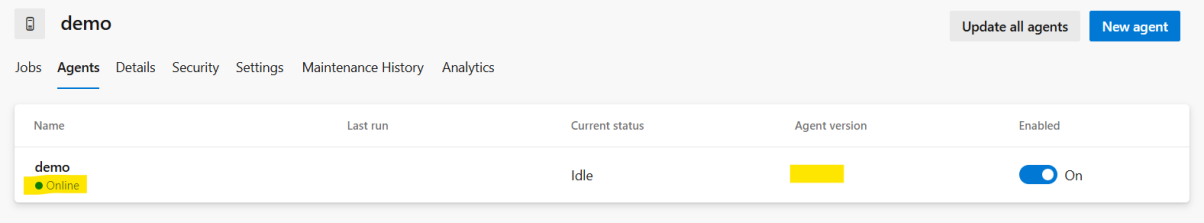
```
[root@devopsagent-vm ~]# cd /opt && mkdir myagent && cd myagent
[root@devopsagent-vm myagent]# wget https://vstsagentpackage.azureedge.net/agent/[redacted]/vsts-agent-linux-x64-[redacted].tar.gz
[root@devopsagent-vm myagent]# tar -xvf vsts-agent-linux-x64-[redacted].tar.gz
[root@devopsagent-vm myagent]#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

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

  ___                    ____ _                 _
 / _ \                  / ___ (_)               | (_)
/ /_\ \____  _ _ __ ___ | |_/ /_ _ __ ___  ___| |_ _ __   ___
|  _  |_  / | | '__/ _ \|  __/| | '_ ` _ \/ _ \ | | '_ \ / _ \
| | | |/ /| |_| | |  __/| |   | | | | | |  __/ | | | | |  __/
\_| |_/___|\__,_|_|  \___|\_|   |_|_| |_| |_\___|_|_|_| |_|\___|
                                  | |
       agent [redacted]           |_|          [redacted]

>> End User License Agreements:

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

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

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

>> Connect:

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

>> Register Agent:

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

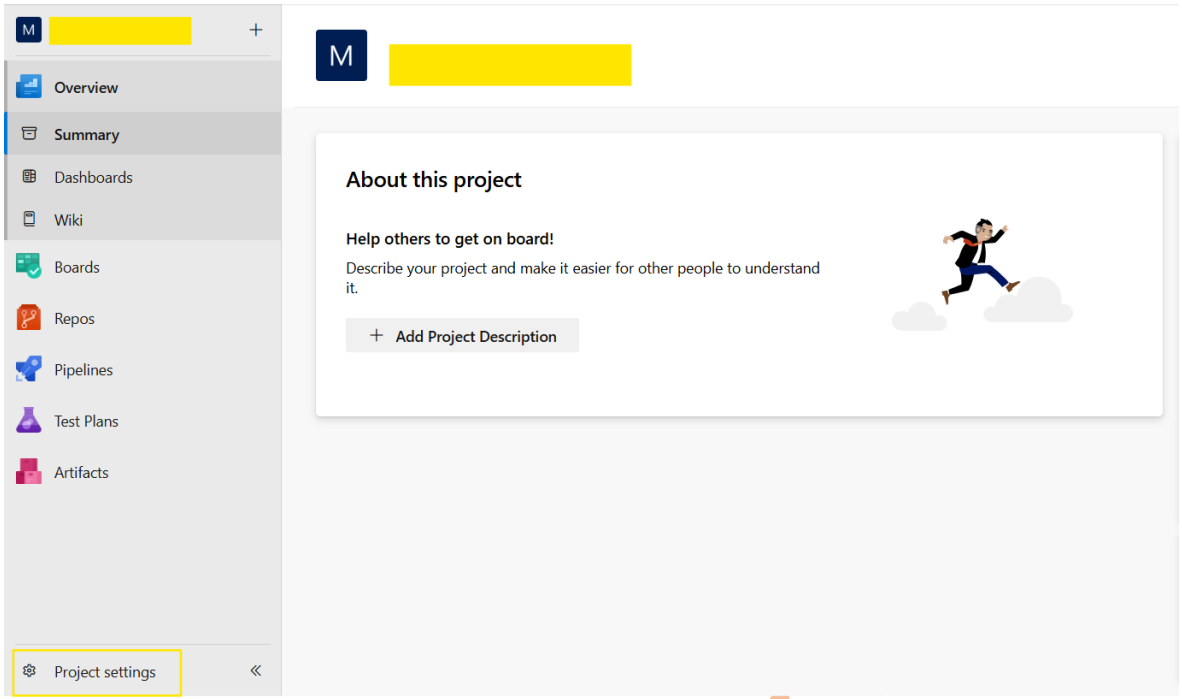
```
[demo@devopsagent-vm myagent]$ ./env.sh
[demo@devopsagent-vm myagent]$
[demo@devopsagent-vm myagent]$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/opt/sonar-scanner/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/opt/dependency-check/bin:/usr/local
/bin
```

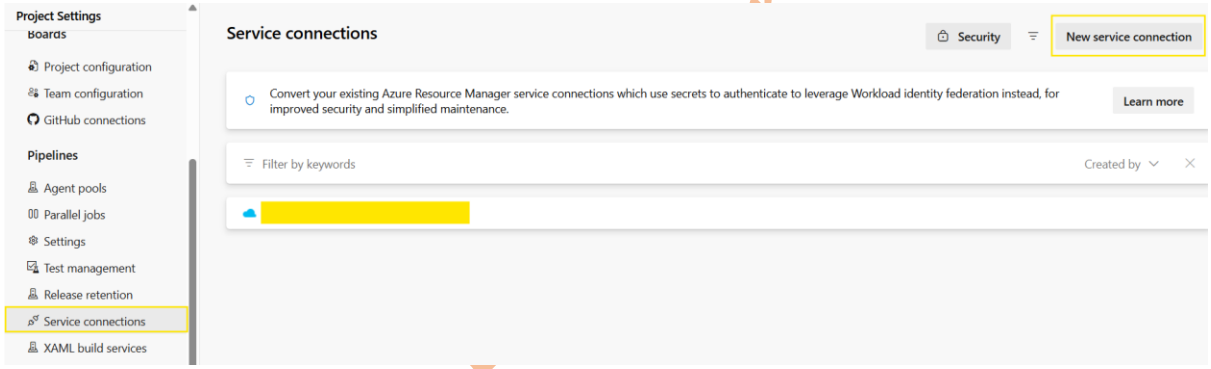
```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh install
[demo@devopsagent-vm myagent]$ sudo ./svc.sh start
```

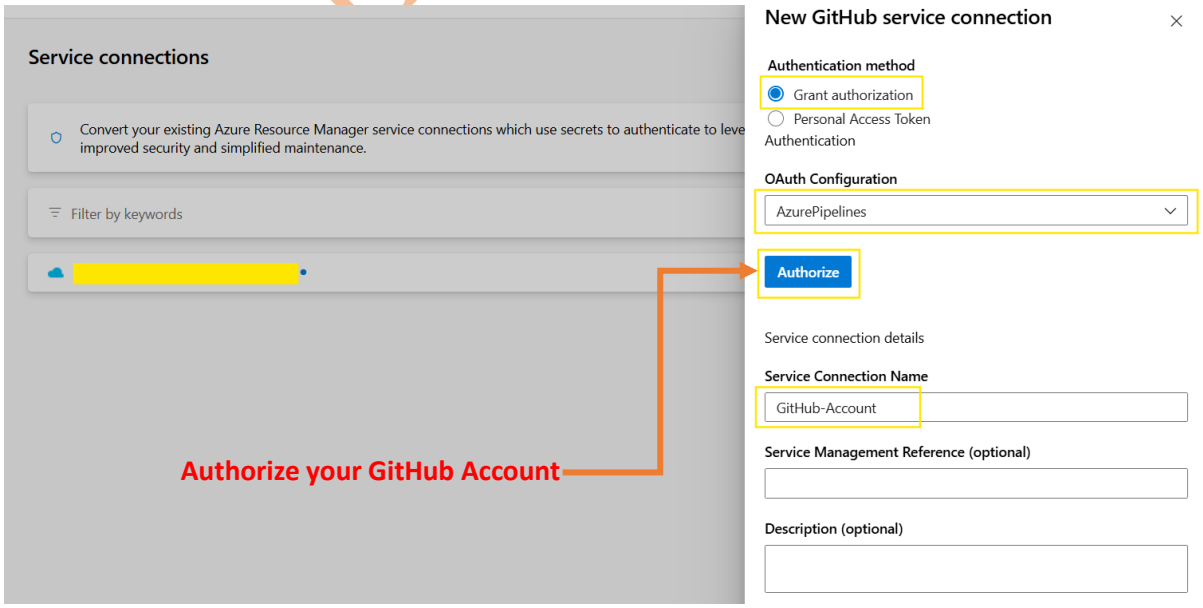
Now the Agent came in online state as shown in the screenshot attached below.

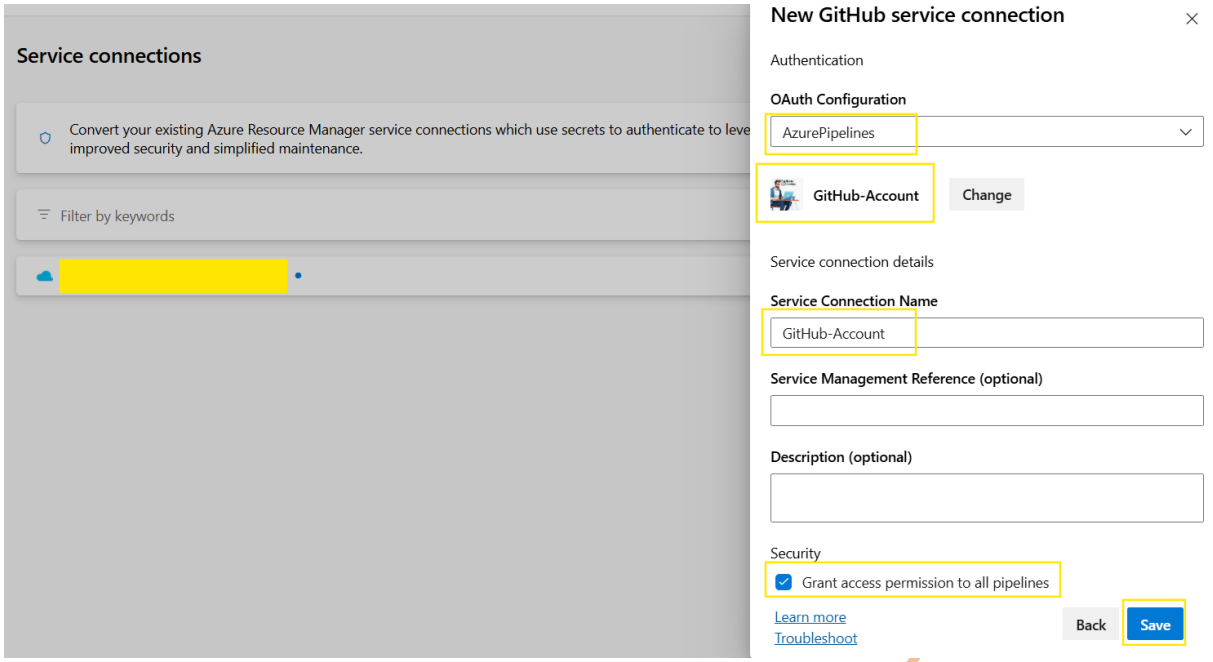


Then I went into the **Azure DevOps project > Project Settings** and created the **Service connections** for GitHub Account, Docker Registry as shown in the screenshot attached blow.


Overview
Summary
Dashboards
Wiki
Boards
Repos
Pipelines
Test Plans
Artifacts
Project settings
About this project
Help others to get on board!
Describe your project and make it easier for other people to understand it.
Add Project Description

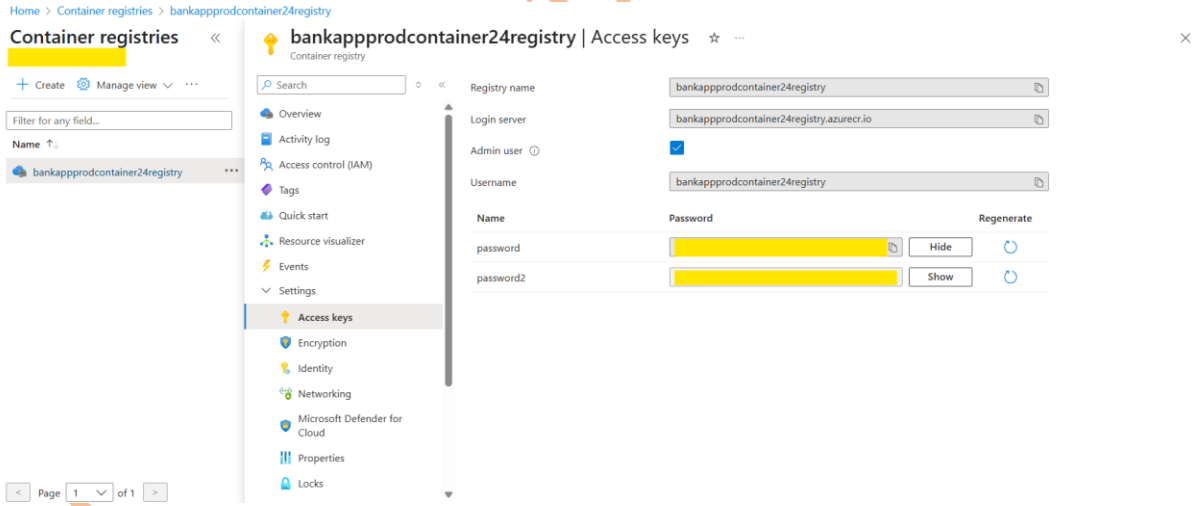

Project Settings
Project configuration
Team configuration
GitHub connections
Pipelines
Agent pools
Parallel jobs
Settings
Test management
Release retention
Service connections
XAML build services
Service connections
Security
New service connection
Convert your existing Azure Resource Manager service connections which use secrets to authenticate to leverage Workload identity federation instead, for improved security and simplified maintenance.
Learn more
Filter by keywords
Created by

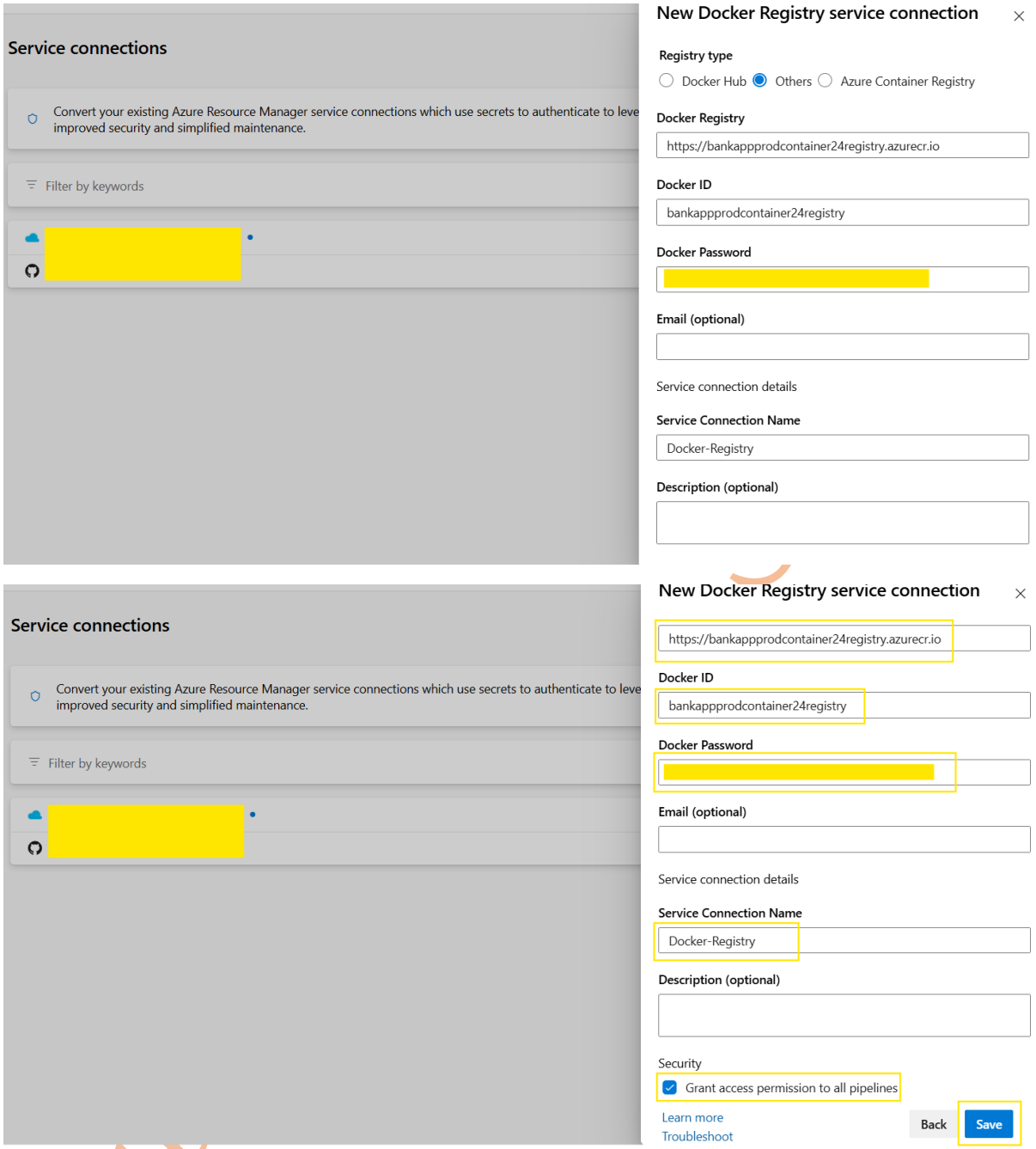

Service connections
Convert your existing Azure Resource Manager service connections which use secrets to authenticate to leve
improved security and simplified maintenance.
Filter by keywords
Authorize your GitHub Account
New GitHub service connection
Authentication method
Grant authorization
Personal Access Token
Authentication
OAuth Configuration
AzurePipelines
Authorize
Service connection details
Service Connection Name
GitHub-Account
Service Management Reference (optional)
Description (optional)



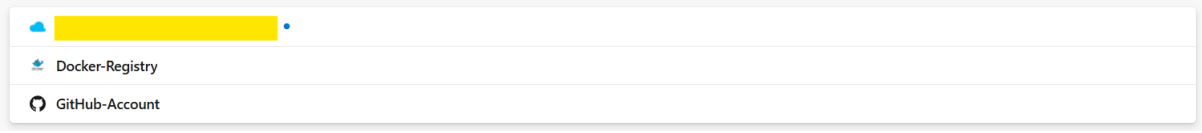
The Azure Container Registry which I had created for this project is as shown in the screenshot attached below.

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

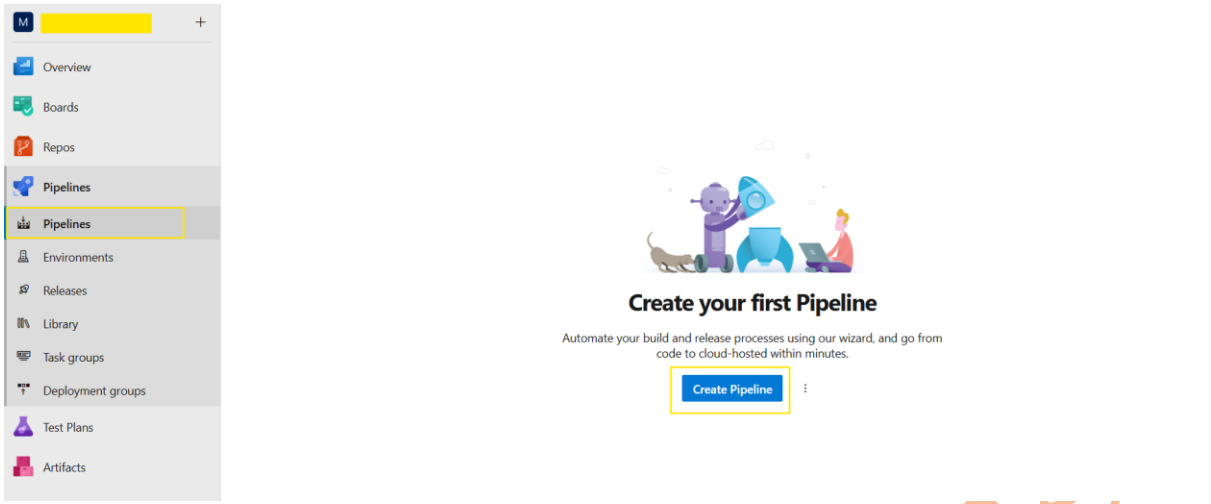


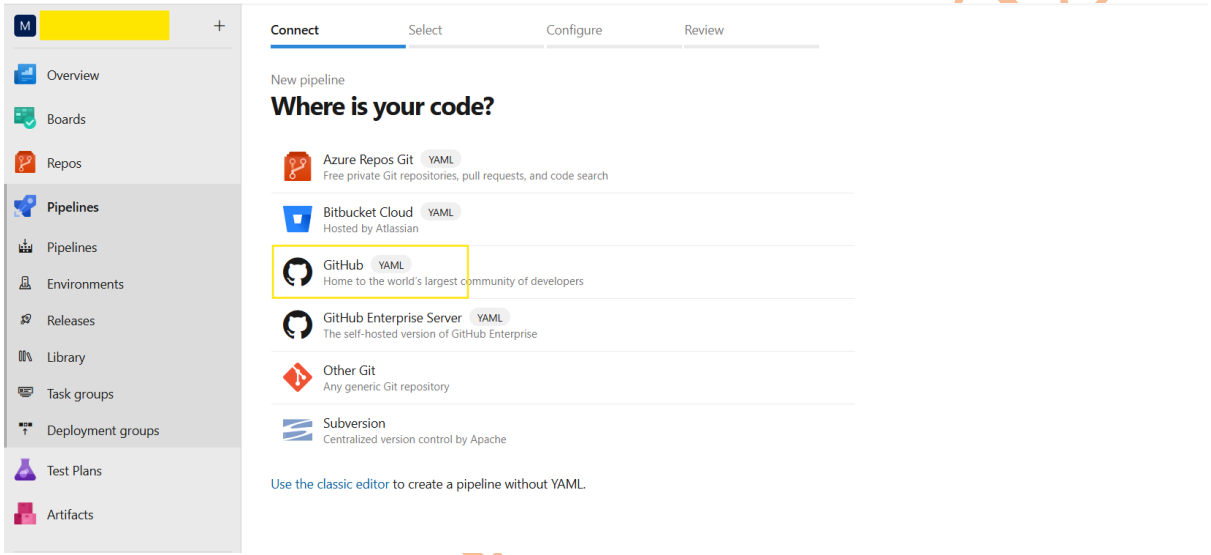


Finaly the service connections had been created as shown in the screenshot attached below.

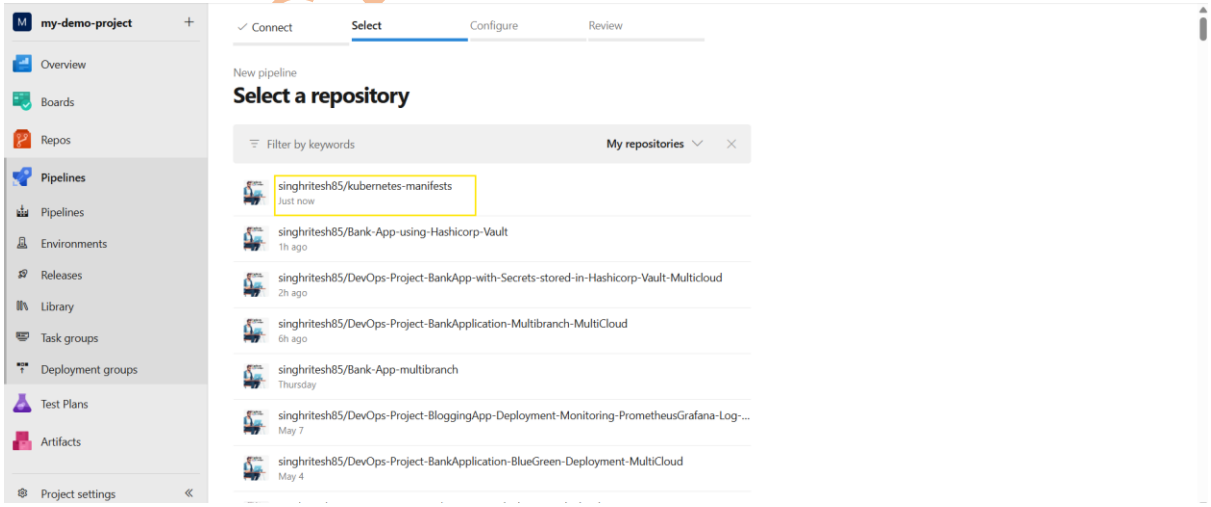


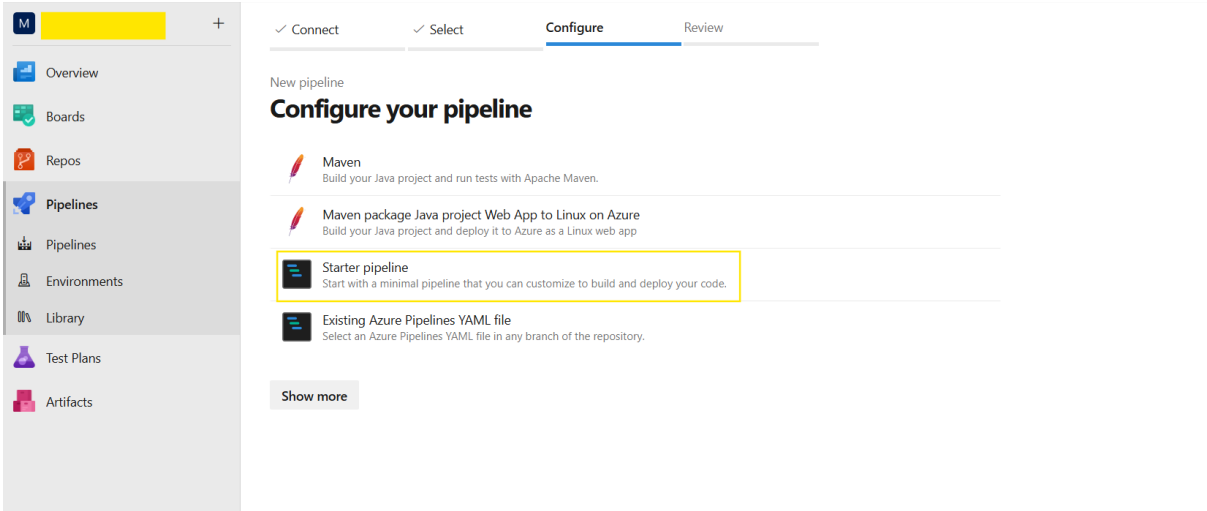
Then I created the Azure DevOps CI/CD pipeline as shown in the screenshot attached below.





At this stage it will ask to authorize your GitHub Account and after authorizing your GitHub Account select the desired GitHub Repo as shown in the screenshot attached below.





If you need to apply webhook in Azure DevOps Pipeline then in **azure-pipelines.yaml** trigger: provide the branch name as shown in the screenshot attached below.

```
trigger:
- main
```

This pipeline triggers for main branch only if you need to apply for other branches then provide the branch names as shown in the screenshot attached below.

```
trigger:
- main
- stage
- dev
```

For this Project I had created two Azure DevOps pipelines named as **mysql** and **bankapp.** The **mysql** pipeline will be created first and then **bankapp** pipeline. As I had not applied webhook for **mysql** pipeline and so **mysql** pipeline needs to run manually. If you want to apply the webhook for **bankapp** pipeline then use the method which I used to apply the webhook in production for Azure DevOps Pipeline (with the help of **trigger:**). Before running the Azure DevOps Pipeline for **mysql** create a kubernetes namespace with the name of mysql using the command **kubectl create ns mysql**.

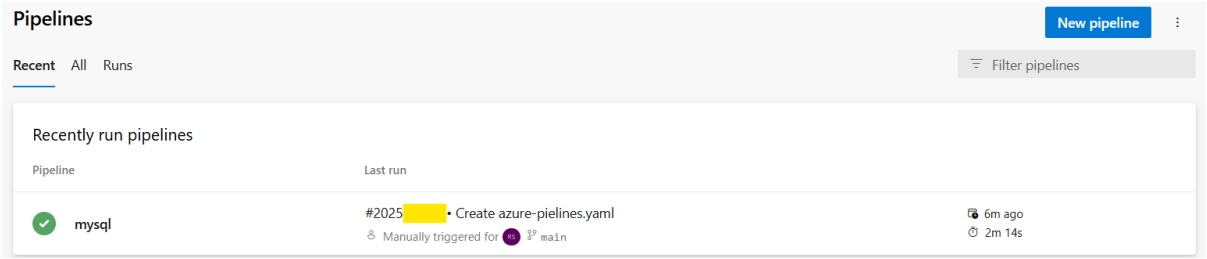
The mysql pods got the mysql root password from Hashicorp vault.

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

After successfully running the mysql pipeline pods, pvc and services had been created as shown in the screenshot attached below.

```
[root@                ~]# kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   2/2     Running   0          3m13s
mysql-1   2/2     Running   0          2m16s
[root@                ~]#
[root@                ~]# kubectl get pvc -n mysql --watch
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-data-mysql-0  Bound    pvc-                                       1Gi        RWO            managed-csi    <unset>                 3m20s
mysql-data-mysql-1  Bound    pvc-                                       1Gi        RWO            managed-csi    <unset>                 2m22s
[root@                ~]#
[root@                ~]# kubectl get svc -n mysql --watch
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql-svc         ClusterIP   10.0.237.106   <none>        3306/TCP   3m26s
mysql-svc-headless ClusterIP  None           <none>        3306/TCP   3m25s
```

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



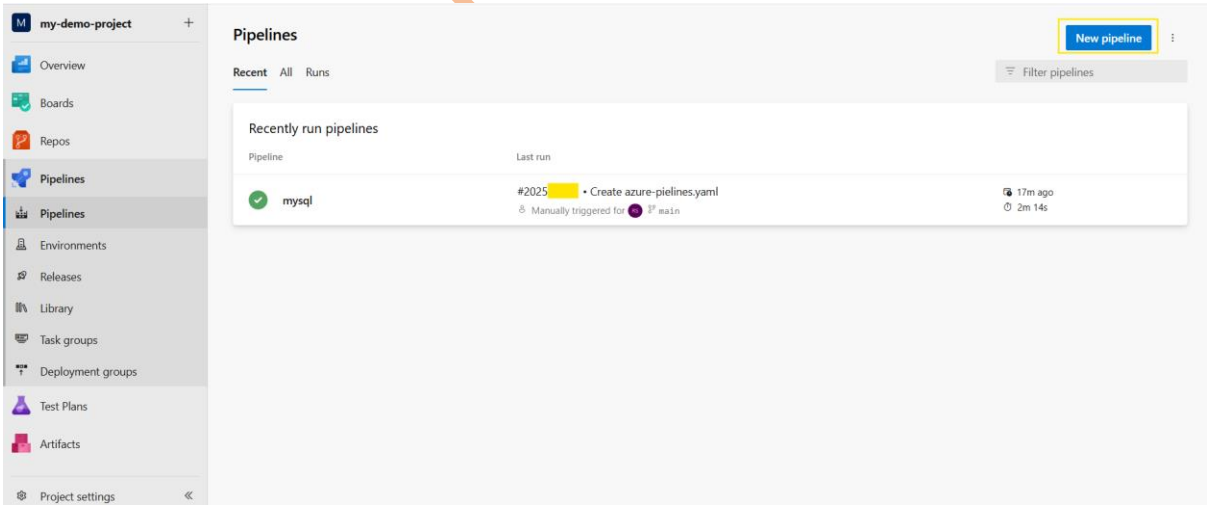
Then I had created the pipeline for bankapp. I had created the Docker Image for bankapp deployment in that the base image was used as eclipse-temurin. **OpenJDK Docker Image is deprecated so in this project eclipse-temurin docker image has been used as a base image in the Dockerfile**.

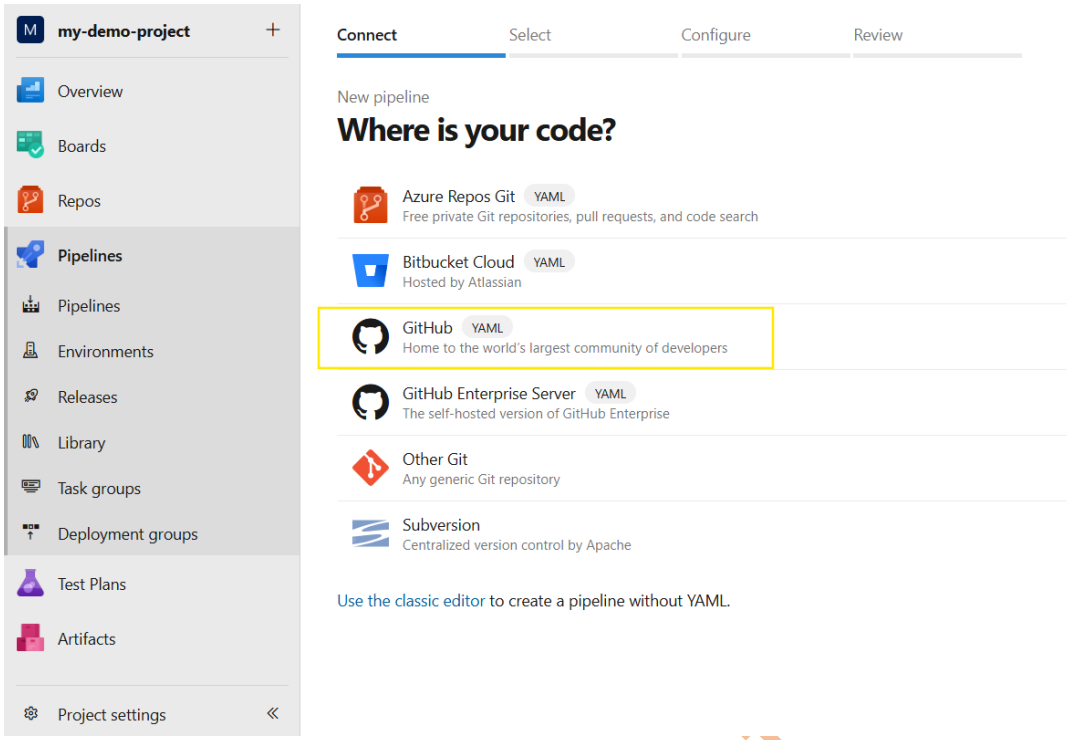
To access Docker Image from the Azure Container Registry during Kubernetes Deployment I created a kubernetes Secrets as shown in the screenshot attached below.

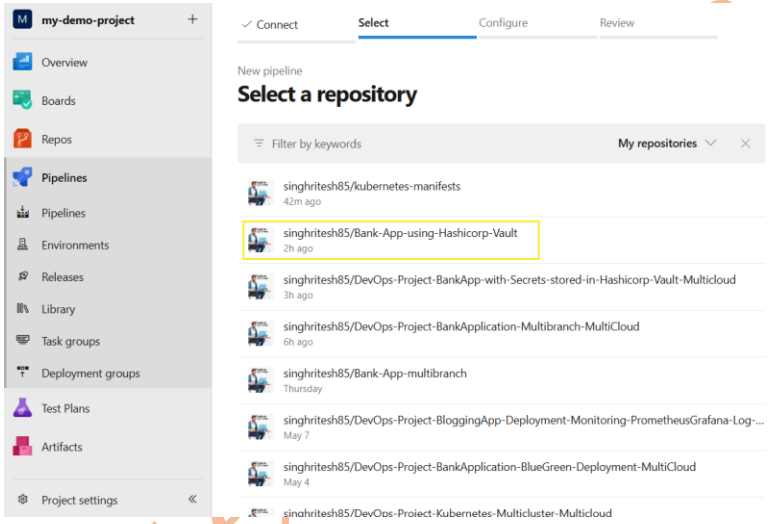
kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappprodcontainer24registry.azurecr.io --docker-username=bankappprodcontainer24registry --docker-password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -n bankapp

```
[root@                ~]# kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappprodcontainer24registry.azurecr.io --docker-usern
ame=bankappprodcontainer24registry --docker-password=                                                                                    -n bankapp
secret/bankapp-auth created
```

The below screenshot shows how I created the pipeline for **bankapp**.

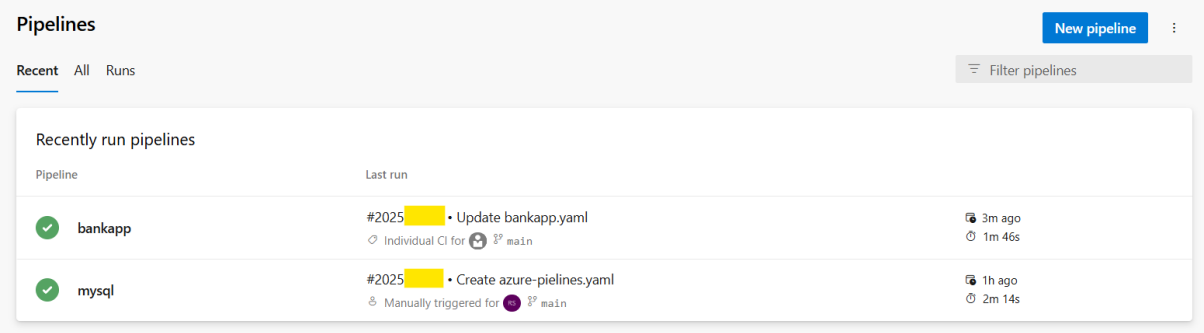






After success execution of bankapp pipeline the pods and service had been created as shown in the screenshot attached below.

```
[root@                    ~]# kubectl get all -n bankapp
NAME                          READY   STATUS    RESTARTS   AGE
pod/bankapp-                  1/1     Running   0          2m33s

NAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/bankapp-service       ClusterIP   10.      .114  <none>        80/TCP    2m33s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp       1/1     1            1           2m33s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-                 1         1         1       2m33s
```



Below screenshot shows the ingress rule which I used to access the Bank Application using the URL.

You can use the Ingress rule which I provided in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-with-Secrets-stored-in-Hashicorp-Vault-Multicloud.git with the file name **ingress-rule-prod.yaml**. First you need to create the kubernetes secrets as shown in the screenshot attached below.

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

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

```
[root@ ~]# kubectl apply -f ingress-rule-prod.yaml
ingress.networking.k8s.io/bankapp-ingress created
[root@ ~]# cat ingress-rule-prod.yaml
# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-tls
  rules:
  - host: bankapp.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80

[root@ ~]# kubectl get ing -A --watch
NAMESPACE   NAME              CLASS                       HOSTS                     ADDRESS       PORTS     AGE
bankapp     bankapp-ingress   azure-application-gateway   bankapp.singhritesh85.com   4.  .25   80, 443   11s
vault       vault-ingress     azure-application-gateway   vault.singhritesh85.com     4.  .25   80, 443   4h48m
```

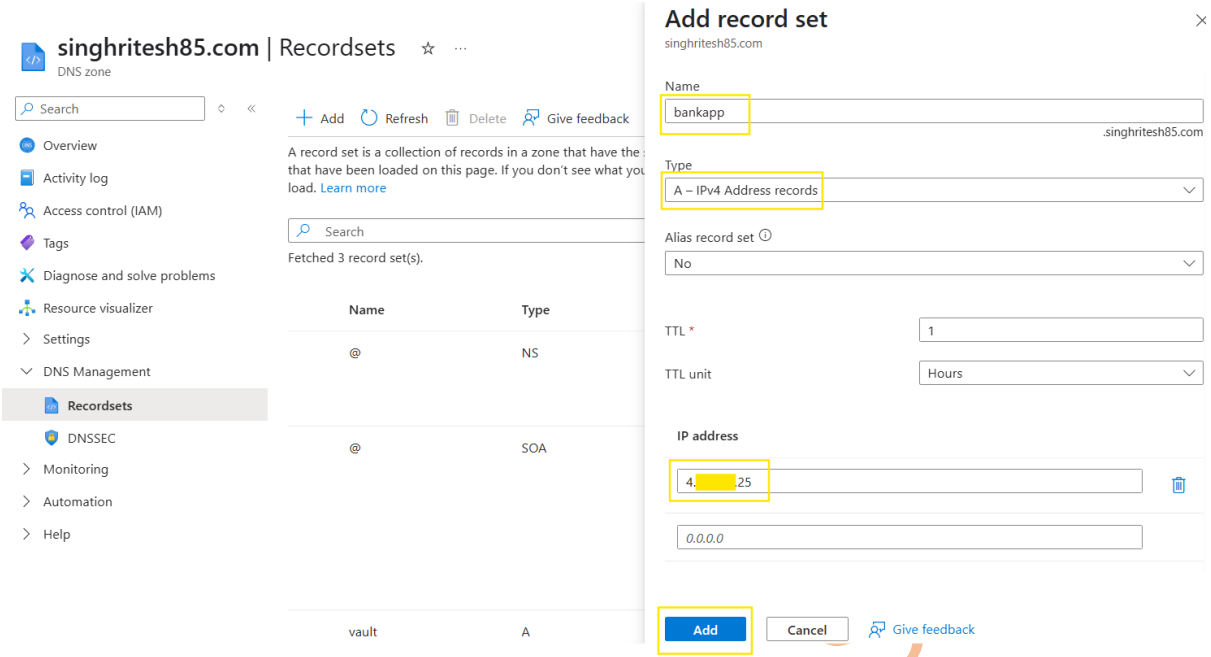
cat ingress-rule-prod.yaml

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

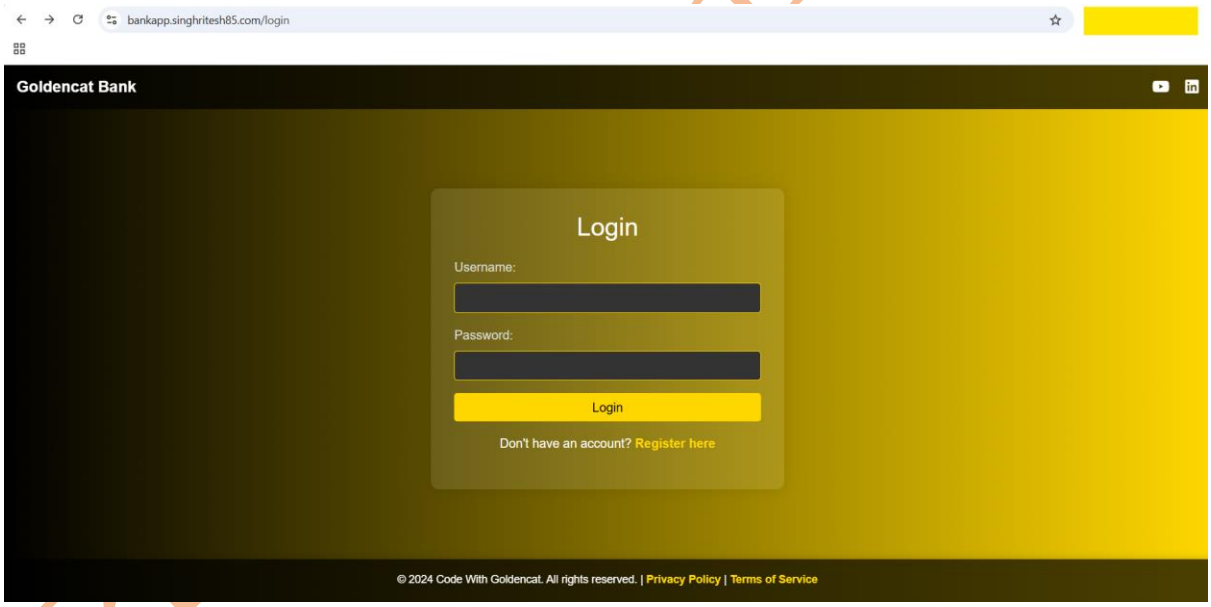
---

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

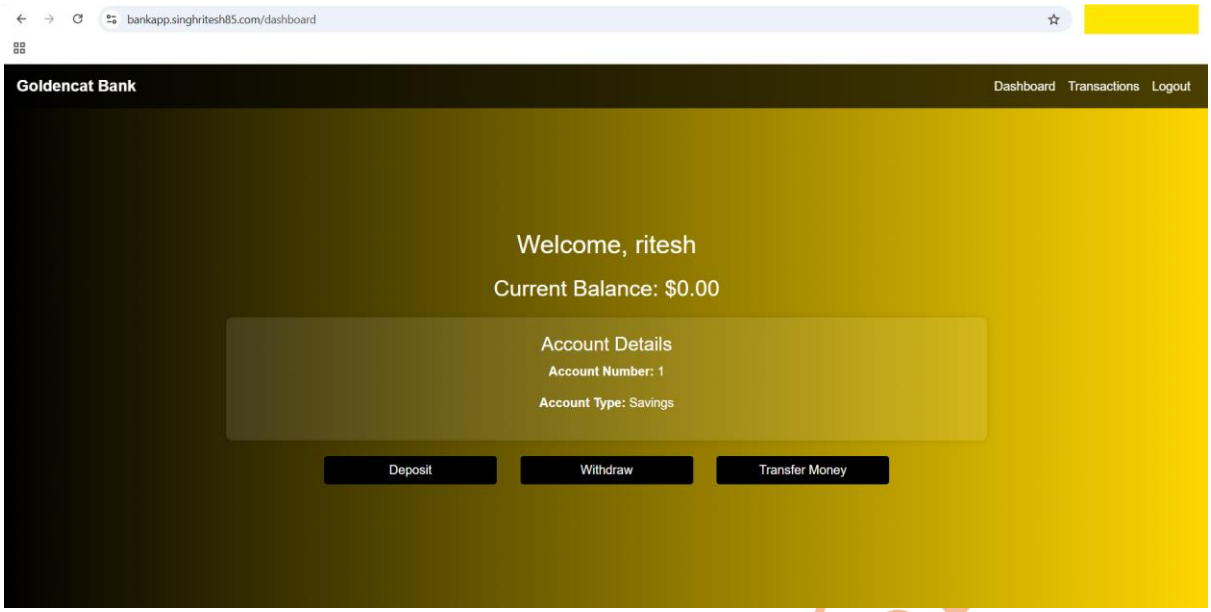
I did the entry for HOST **bankapp.singhritesh85.com** with Public IP Address as shown in the screenshot attached above in the Azure DNS Zone to create the Record Set of A Type.



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



I registered a user ritesh in the Bank Application as discussed in the production environment and then logged-in with the same user and I had also checked its entry in the MySQL database and found the same as shown in the screenshot attached below.




```
[root@                    ~]# kubectl exec -it mysql-0 -n mysql sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
Defaulted container "mysql" out of: mysql, vault-agent, vault-agent-init (init)
# mysql -h localhost -u root --password
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 103
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| #mysql50#lost+found |
| bankappdb          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.000 sec)

MariaDB [(none)]> use bankappdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [bankappdb]> show tables;
+--------------------+
| Tables_in_bankappdb |
+--------------------+
| account            |
| transaction        |
+--------------------+
2 rows in set (0.000 sec)
```

```
MariaDB [bankappdb]> select * from account;
+----+---------+--------------------------------------------------------------+----------+
| id | balance | password                                                     | username |
+----+---------+--------------------------------------------------------------+----------+
|  1 |    0.00 |                                                              | ritesh   |
+----+---------+--------------------------------------------------------------+----------+
1 row in set (0.000 sec)

MariaDB [bankappdb]>
```

Create snapshot of Hashicorp Vault and store it in Azure Storage Account Container

To create snapshot of Hashicorp vault and store it in Azure Storage Account Container I used the kubernetes CronJob which is present in the Github Repo https://github.com/singhritesh85/kubernetes-manifests.git at the path **snapshot-store**. I deployed this CronJob using Azure DevOps Pipeline. To run this pipeline, I used the VAULT_TOKEN and SAS Token for Azure Storage Account Container. I created an approle in Hashicorp Vault and its Token I used for VAULT_TOKEN and If you noticed in initial stage of the project when I created resources using Terraform script then I got the SAS Token for Azure Storage Account Container which you can get printed using the command **terraform output -json acr_azure_loki_vm_private_ip_container_sas_url**. The vault token and SAS Token I provided as Pipeline variable to run pipeline successfully. For this demonstration I ran this CronJob per minute but in your organisation, you need to run it periodically in non-production hours. In this CronJob I used a Docker Image **peterdavehello/azcopy:10** form Docker Hub Public Repo to run the **azcopy** command successfully. For your organisation, you can use your own Docker Image. Below screenshot shows the Image layers for this Docker Image. I would like to clearly mention here that the Vault Token which I used here was not the Root Token but the **Approle** based vault token.

Image Layers

1	ADD alpine-minirootfs-3.19.4-x86.tar.gz / # buildkit	3.1 MB
2	CMD ["/bin/sh"]	0 B
3	ARG AZCOPY_VERSION=10.27.1	0 B
4	LABEL name=docker-azcopy	0 B
5	LABEL version=10.27.1	0 B
6	LABEL maintainer=Peter Dave Hello <hsu@peterdavehello.org>	0 B
7	COPY /azcopy/azcopy /usr/local/bin # buildkit	21.01 MB
8	WORKDIR /WORKDIR	97 B
9	CMD ["azcopy"]	0 B

It is not suggestable to hardcode any Token or SAS Token in kubernetes manifests file directly So I used the Azure DevOps Pipeline variable.

First, I create the Policy and then the approle by attaching the policy to the approle as shown in the screenshot attached below.

```
vault policy write backup_policy - <<EOF
path "sys/storage/raft/snapshot" {
  capabilities = ["read"]
}
EOF
```

```
vault auth enable approle
```

```
vault write auth/approle/role/backup-role policies=backup_policy
```

```
vault read auth/approle/role/backup-role/role-id
```

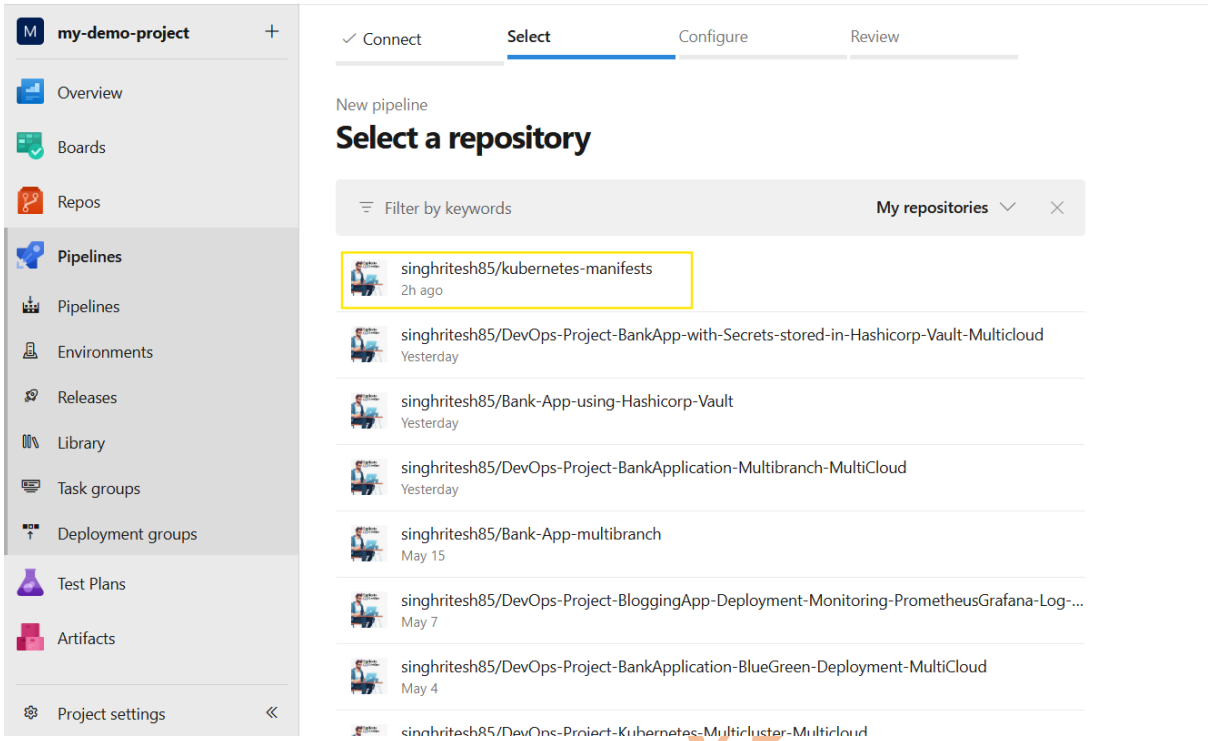
```
vault write -f auth/approle/role/backup-role/secret-id
```

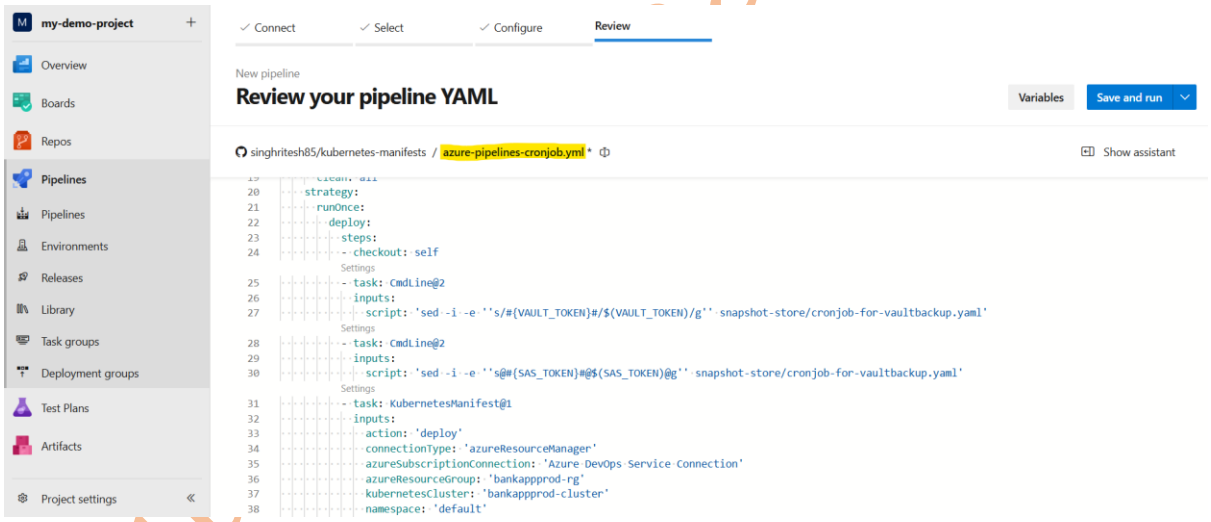
Finally get the Token using the command as written below **(don't run this command second time otherwise the older secret_id will be lost)**

```
vault write auth/approle/login role_id=<role-id> secret_id=<secret-id>
```

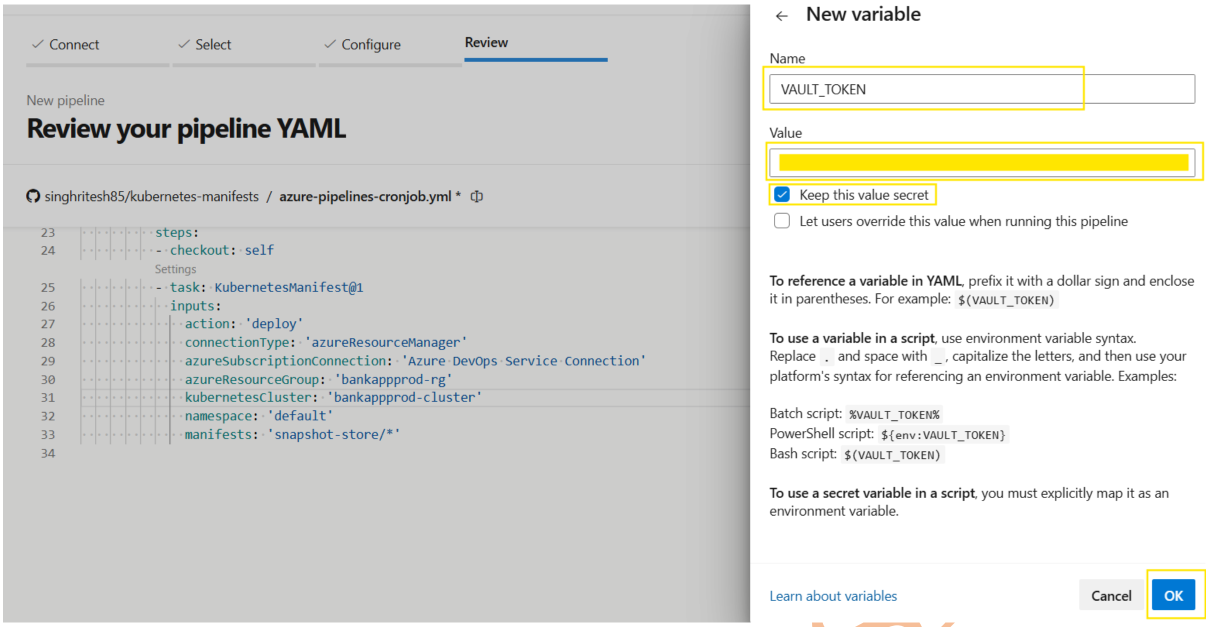
```
[root@                ~]# kubectl exec -it vault-0 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault policy write backup_policy - <<EOF
> path "sys/storage/raft/snapshot" {
>   capabilities = ["read"]
> }
> EOF
Success! Uploaded policy: backup_policy
/ $
/ $ vault auth enable approle
Success! Enabled approle auth method at: approle/
/ $
/ $ vault write auth/approle/role/backup-role policies=backup_policy
Success! Data written to: auth/approle/role/backup-role
/ $
/ $ vault read auth/approle/role/backup-role/role-id
Key        Value
---        -----
role_id
/ $
/ $ vault write -f auth/approle/role/backup-role/secret-id
Key                   Value
---                   -----
secret_id
secret_id_accessor
secret_id_num_uses    0
secret_id_ttl         0s
/ $ vault write auth/approle/login role_id=                                  secret_id=
Key                     Value
---                     -----
token
token_accessor
token_duration          768h
token_renewable         true
token_policies          ["backup_policy" "default"]
identity_policies       []
policies                ["backup_policy" "default"]
token_meta_role_name    backup-role
/ $
```

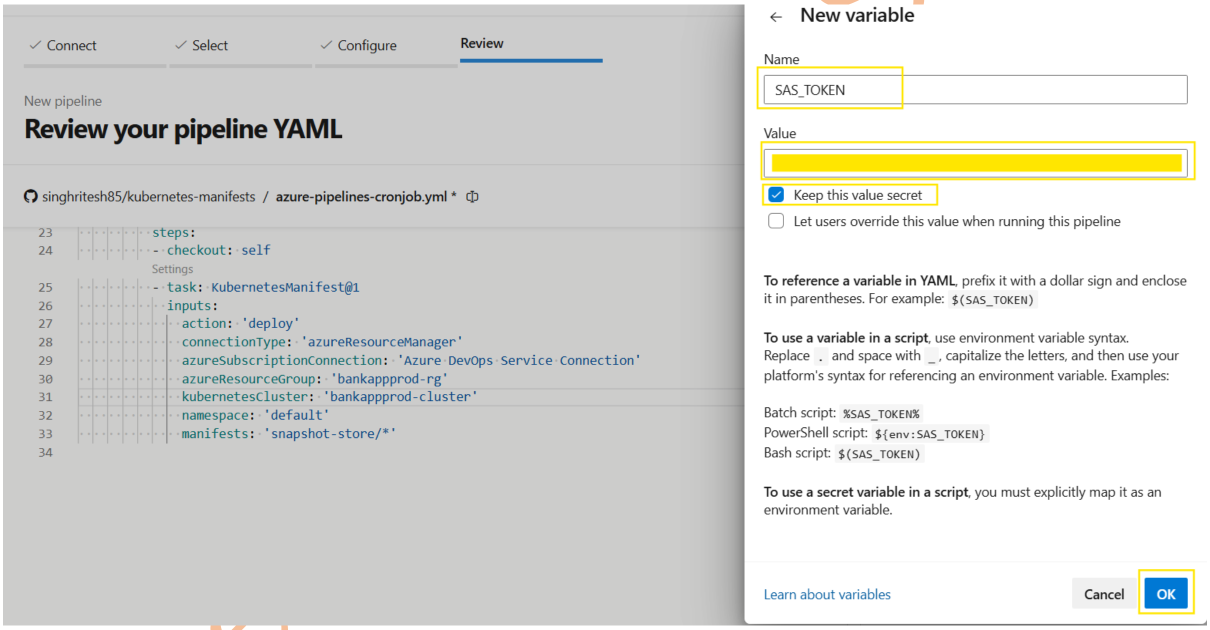
Below screenshots show how I had created the Azure DevOps Pipeline for cronjob to run.





Below screenshot show how I provided vault token and SAS Token in Azure DevOps Pipeline.





It is important to mention here how I provided variables for SAS Token in Azure DevOps Pipeline variable. Below is the original SAS Token which I got from **terraform output -json acr_azure_loki_vm_private_ip_container_sas_url** command.

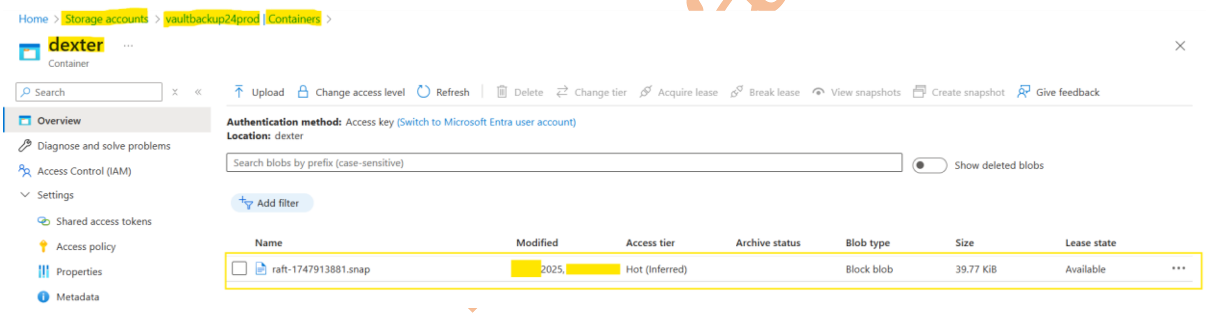
"https://vaultbackup24prod.blob.core.windows.net/dexter?XX=2018-11-09\XXXXXXX=X\XXXXXXX=2025-03-21\XXXXXXX=2029-03-21\XXXXXXXXXrawl\XXXXXXXr=https\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

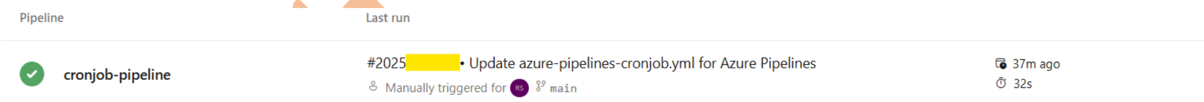
Below variable's value I provided in the Pipeline variable. Please see how I provided / (forward slash) and \ (backslash) otherwise Pod created after CronJob runs will through Error.

\\"https://vaultbackup24prod.blob.core.windows.net/dexter?XX=2018-11-09\\XXXXXXX=X\\XXXXXXX=2025-03-21\\XXXXXXX=2029-03-21\XXXXXXXXXrawl\\XXXXXXXr=https\\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\\"

Below screenshot shows that CronJob ran successfully and snapshot had been copied to the Asure Storage Account container.

```
[root@[redacted] ~]# kubectl get all
NAME                                   READY   STATUS      RESTARTS   AGE
pod/vault-backup-[redacted]            0/1     Completed   0          11s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.0.0.1     <none>        443/TCP   4h7m

NAME                         SCHEDULE    TIMEZONE   SUSPEND   ACTIVE   LAST SCHEDULE   AGE
cronjob.batch/vault-backup   * * * * *   <none>     False     1        11s             40s

NAME                               STATUS    COMPLETIONS   DURATION   AGE
job.batch/vault-backup-[redacted]  Running   0/1           11s        11s
```

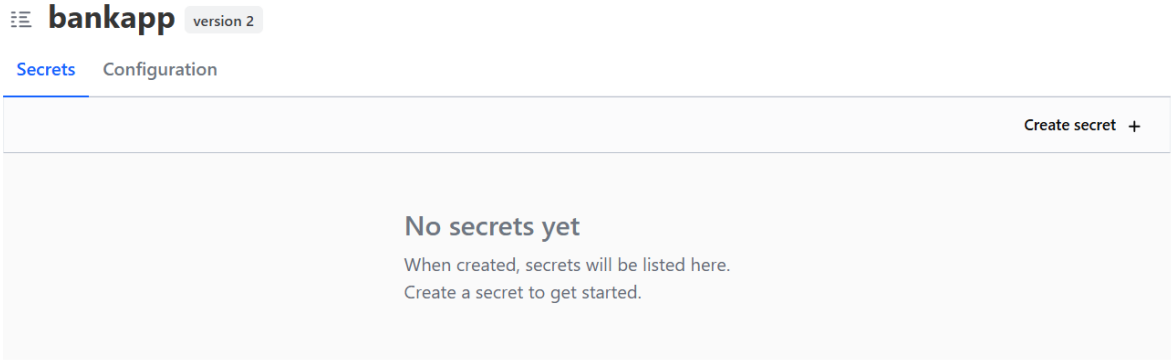




By default, Approle exists for 32 days if you want it for a greater number of days then you need to mention the same with **token_ttl** and **token_max_ttl**.

Restore the Hashicorp Vault Backup

Consider the situation if the secrets from the vault had been deleted and you want to restore the secrets from the backup you had taken. As mentioned in the above section I took the backup of vault and stored it in Azure Storage Account Container. I had created a secret at the path **bankapp/database/config** for demonstration purpose I had deleted it as shown in the screenshot attached below.



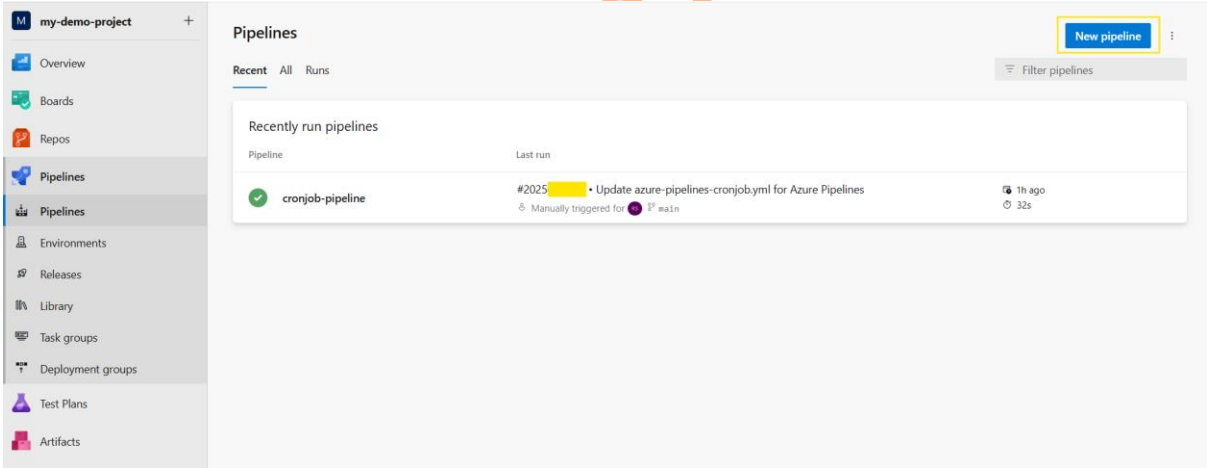
In backup section I had created a policy named as backup-policy which had only read access which is not sufficient to restore backup So I had created a policy with the privileges as shown below.

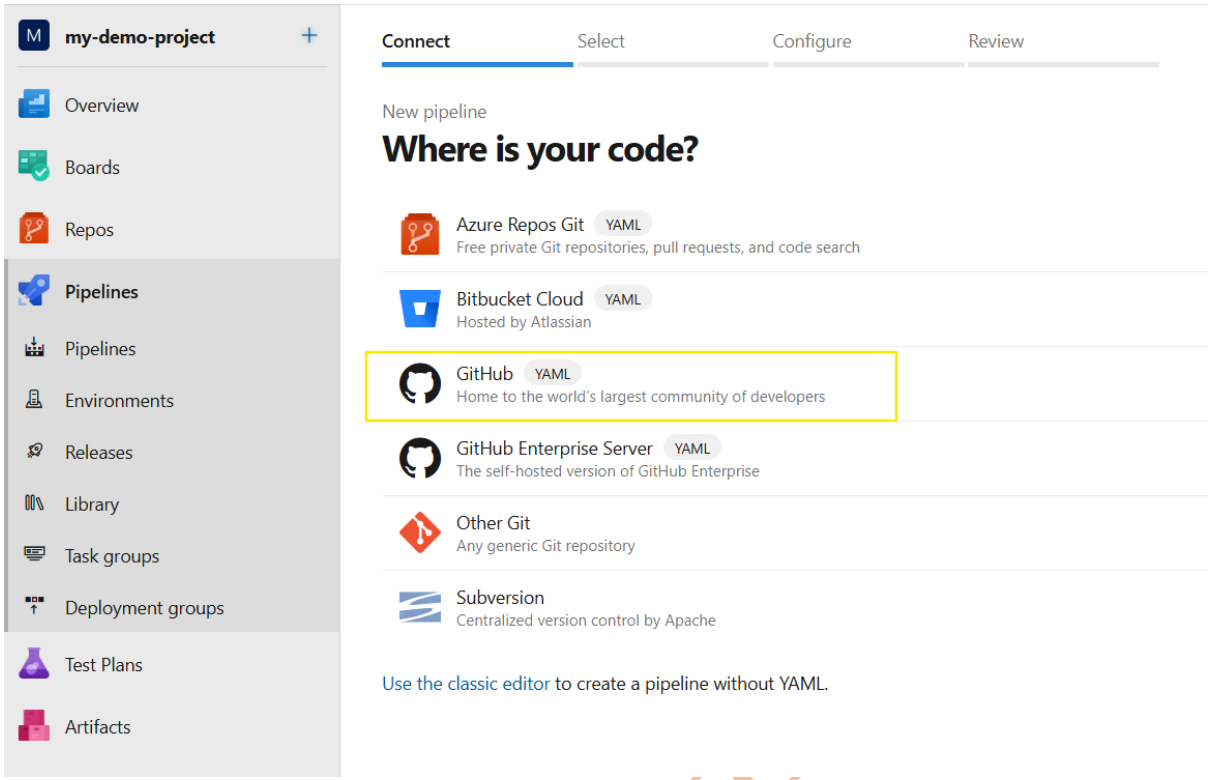
```
vault policy write backup_policy - <<EOF
path "sys/storage/raft/snapshot" {
 capabilities = ["update"]
}
path "sys/storage/raft/snapshot-force" {
 capabilities = ["update"]
}
EOF
vault write auth/approle/role/backup-role policies=backup_policy


vault read auth/approle/role/backup-role/role-id


vault write -f auth/approle/role/backup-role/secret-id


vault write auth/approle/login role_id=<provide role id> secret_id=<provide secret id>
```

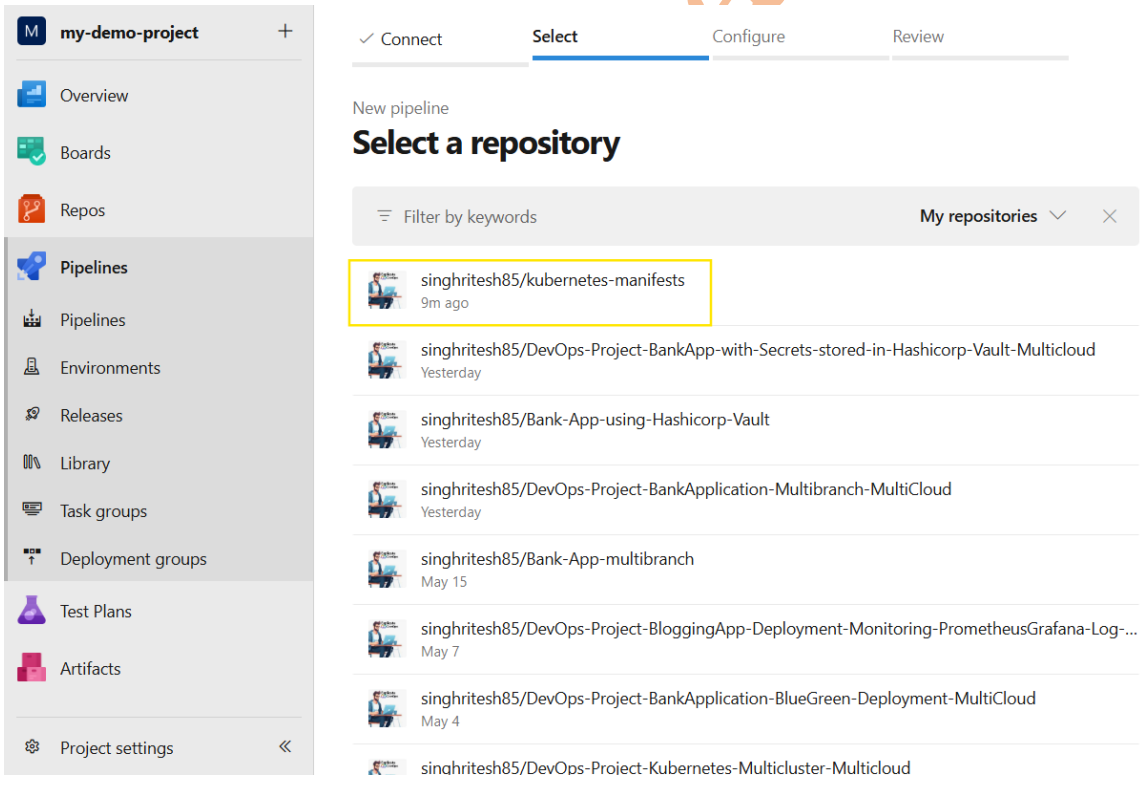
```
[root@                    ~]# kubectl exec -it vault-0 -n vault sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ $ vault policy delete backup_policy
Success! Deleted policy: backup_policy
/ $ vault policy write backup_policy - <<EOF
> path "sys/storage/raft/snapshot" {
>   capabilities = ["update"]
> }
> path "sys/storage/raft/snapshot-force" {
>   capabilities = ["update"]
> }
> EOF
Success! Uploaded policy: backup_policy
/ $ vault write auth/approle/role/backup-role policies=backup_policy
Success! Data written to: auth/approle/role/backup-role
/ $ vault read auth/approle/role/backup-role/role-id
Key        Value
---        -----
role_id
/ $ vault write -f auth/approle/role/backup-role/secret-id
Key                   Value
---                   -----
secret_id
secret_id_accessor
secret_id_num_uses    0
secret_id_ttl         0s
```

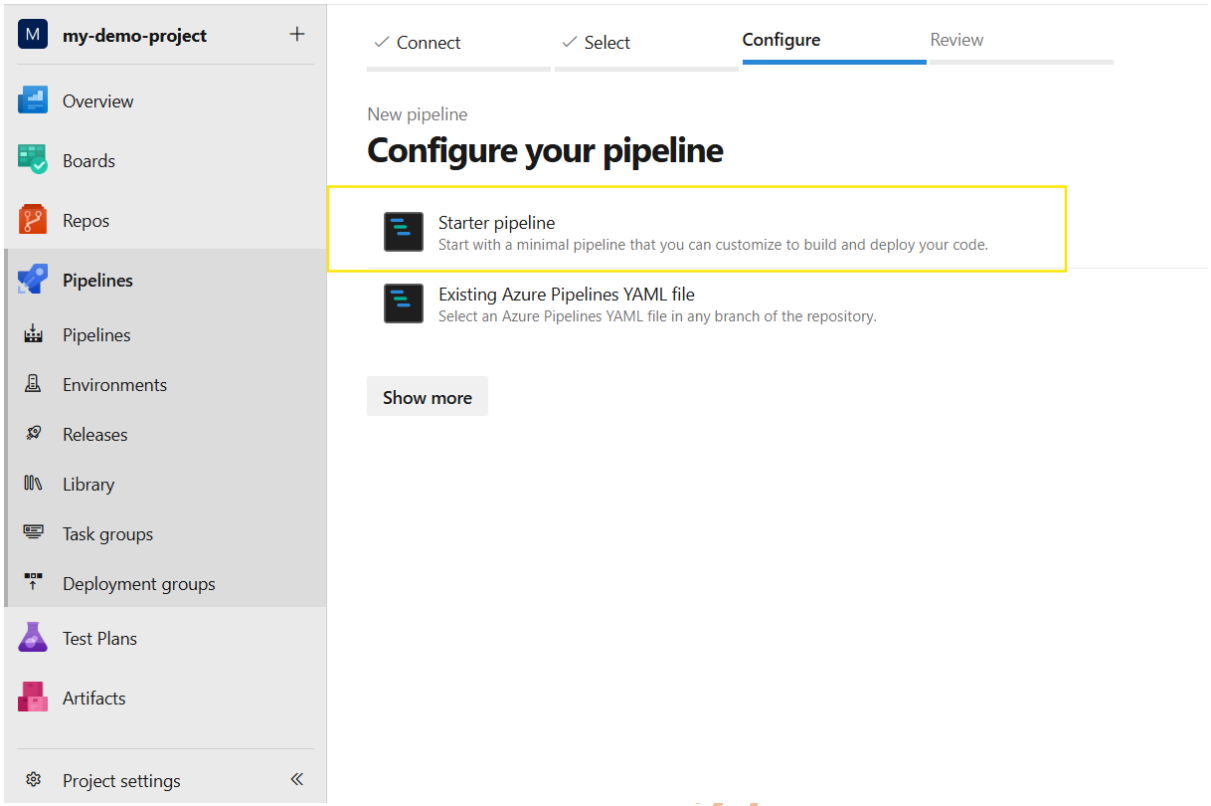
```
/ $ vault write auth/approle/login role_id=                    secret_id=
Key                     Value
---                     -----
token
token_accessor
token_duration          768h
token_renewable         true
token_policies          ["backup_policy" "default"]
identity_policies       []
policies                ["backup_policy" "default"]
token_meta_role_name    backup-role
/ $
```

Created the Approle and got the token in the same way which I discussed above in the snapshot.

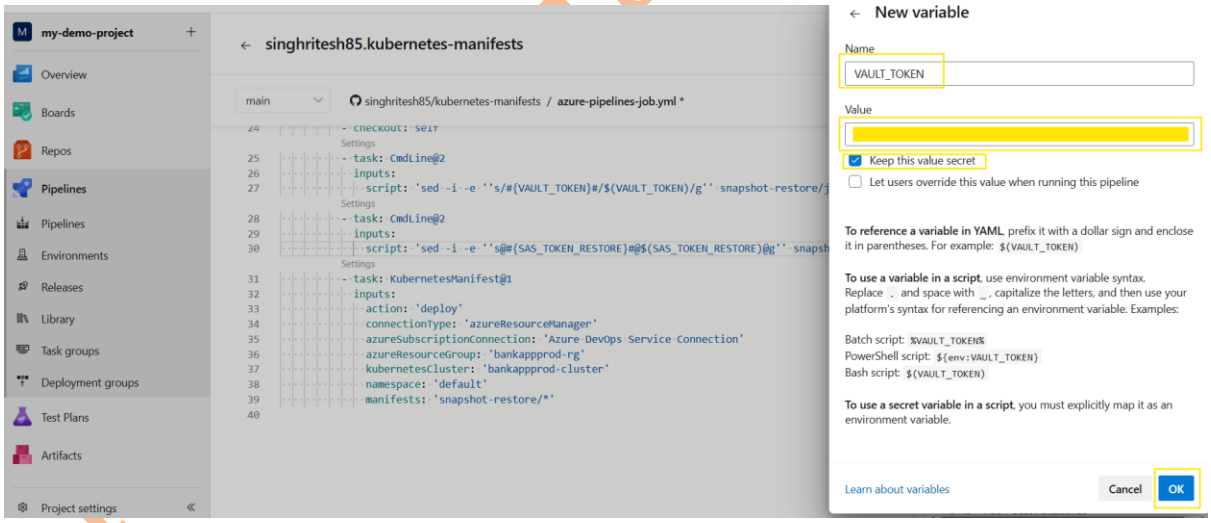


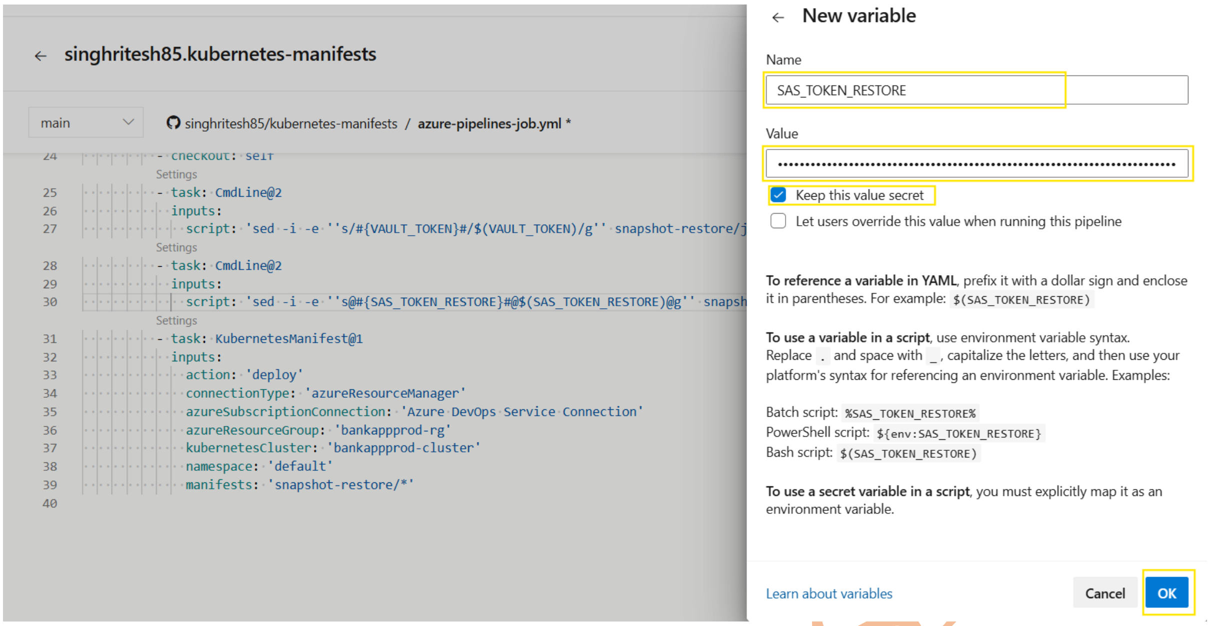

my-demo-project
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts
Connect
Select
Configure
Review
New pipeline
Where is your code?
Azure Repos Git YAML
Free private Git repositories, pull requests, and code search
Bitbucket Cloud YAML
Hosted by Atlassian
GitHub YAML
Home to the world's largest community of developers
GitHub Enterprise Server YAML
The self-hosted version of GitHub Enterprise
Other Git
Any generic Git repository
Subversion
Centralized version control by Apache
Use the classic editor to create a pipeline without YAML.


my-demo-project
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts
Project settings
Connect
Select
Configure
Review
New pipeline
Select a repository
Filter by keywords
My repositories
singhritesh85/kubernetes-manifests
9m ago
singhritesh85/DevOps-Project-BankApp-with-Secrets-stored-in-Hashicorp-Vault-Multicloud
Yesterday
singhritesh85/Bank-App-using-Hashicorp-Vault
Yesterday
singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud
Yesterday
singhritesh85/Bank-App-multibranch
May 15
singhritesh85/DevOps-Project-BloggingApp-Deployment-Monitoring-PrometheusGrafana-Log-...
May 7
singhritesh85/DevOps-Project-BankApplication-BlueGreen-Deployment-MultiCloud
May 4
singhritesh85/DevOps-Project-Kubernetes-Multicluster-Multicloud



Then I created an Azure DevOps Pipeline and ran it as shown in the screenshot attached below. Variable VAULT_TOKEN and SAS_TOKEN_RESTORE was created as discussed below.





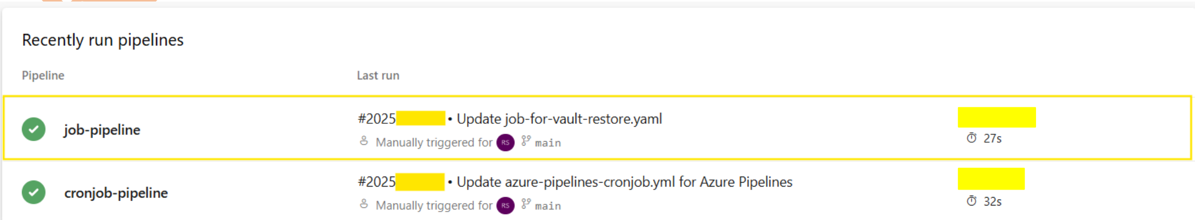
It is important to mention here how I provided variables for SAS Token in Azure DevOps Pipeline variable. Below is the original SAS Token which I got from **terraform output -json acr_azure_loki_vm_private_ip_container_sas_url** command.

"https://vaultbackup24prod.blob.core.windows.net/dexter?XX=2018-11-09\XXXXXXX=X\XXXXXXX=2025-03-21\XXXXXXX=2029-03-21\XXXXXXXXXrawl\XXXXXXXr=https\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

Below variable's value I provided in the Pipeline variable. Please see how I provided / (forward slash) and \ (backslash) otherwise Pod created after CronJob runs will through Error.

\\"https://vaultbackup24prod.blob.core.windows.net/dexter/<name-of-the-vault-snapshot-file-which-needs-to-be-restored>?XX=2018-11-09\\XXXXXXX=X\\XXXXXXX=2025-03-21\\XXXXXXX=2029-03-21\XXXXXXXXXrawl\\XXXXXXXr=https\\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\\"

Below screenshot shows that Kubernetes Job ran successfully and snapshot had been restored.



Before running the Azure DevOps Pipeline **job-pipeline** as shown in the screenshot attached above you need to be assured that the name of the snapshot file given in the variable **SNAPSHOT_TOKEN_RESTORE** in pipeline's variable and in the file **job-for-vault-restore.yaml** present

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

at the path **snapshot-restore** in GitHub Repo https://github.com/singhritesh85/kubernetes-manifests.git should be the one which we wanted to be restored.

```
[root@                    ~]# kubectl get all
NAME                         READY   STATUS      RESTARTS   AGE
pod/vault-restore-           0/1     Completed   0          29s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.0.0.1     <none>        443/TCP   8h

NAME                         STATUS     COMPLETIONS   DURATION   AGE
job.batch/vault-restore      Complete   1/1           12s        30s
```

Then I checked the vault UI and found that secrets had been restored successfully 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