

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/>

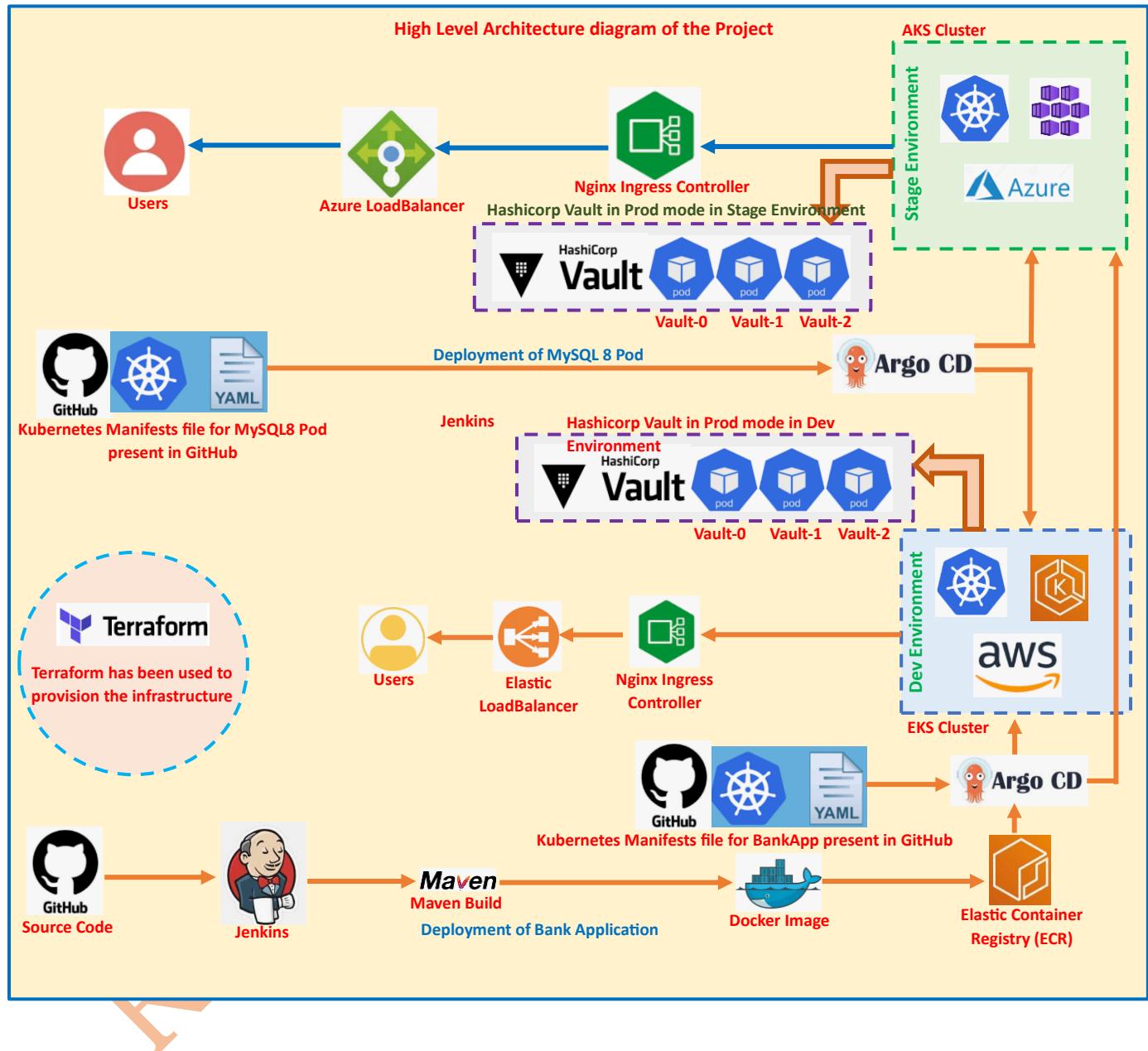
GitHub: - <https://github.com/singhritesh85>



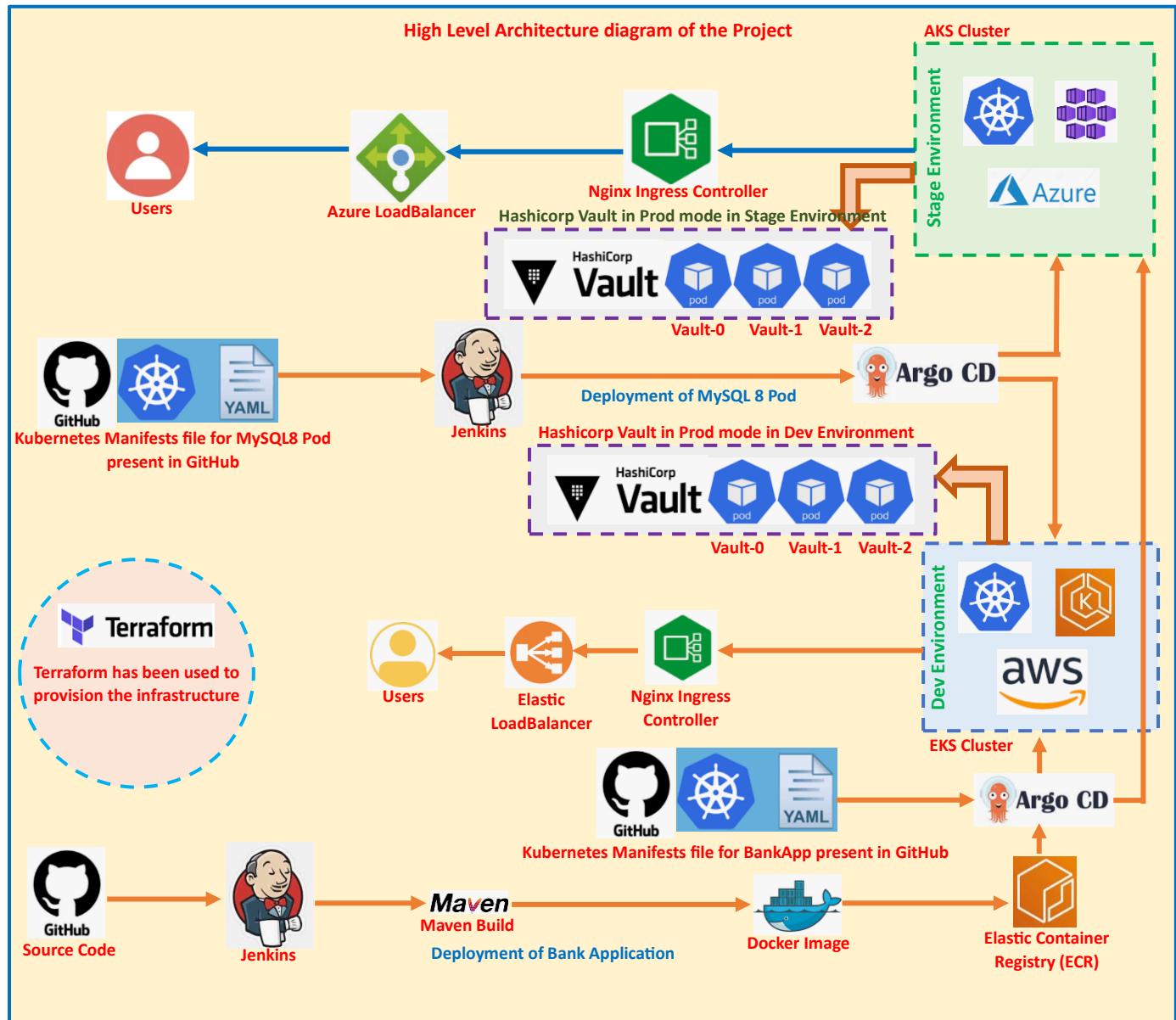
या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाङ्ग्यापहा ॥

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.

Certificate status

Identifier	Status
ARN	Issued
Type	Amazon Issued

Domains (1)

Domain	Status	Renewal status	Type	CNAME name
*.singhritesh85.com	Success	-	CNAME	f.singhritesh85.com

singhritesh85.com | Recordsets

Name	Type
@	NS
@	SOA
f	CNAME

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@[REDACTED] main]# aws configure
AWS Access Key ID [*****]: [REDACTED]
AWS Secret Access Key [*****]: [REDACTED]
Default region name [REDACTED]: [REDACTED]
Default output format [REDACTED]: [REDACTED]
[root@[REDACTED] main]#
```

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

```
[root@[REDACTED] main]# yum install -y https://packages.microsoft.com/config/rhel/8/packages-microsoft-prod.rpm
[root@[REDACTED] main]# yum install azure-cli -y
[root@[REDACTED] main]# az login
To sign in, use a web browser to open the page [REDACTED] and enter the code [REDACTED] 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.[REDACTED]"
    "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10.[REDACTED]"
    "Jenkins_ALB_DNS_Name" = "jenkins-ms-[REDACTED].us-east-2.elb.amazonaws.com"
    "registry_id" = [
        "02[REDACTED]6",
        "02[REDACTED]6",
    ]
    "repository_url" = [
        "02[REDACTED]6.ecr.us-east-2.amazonaws.com/bankapp-1",
        "02[REDACTED]6.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
  server: https://F                               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]: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]: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.

The screenshot shows the Azure portal interface for managing DNS records. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Settings, DNS Management, Records, DNSSEC, Monitoring, Automation, Help, and Add or remove favorites (Ctrl+Shift+F). The 'Records' section is currently selected. On the right, a modal window titled 'Add record set' is open for the 'singhritesh85.com' DNS zone. The 'Name' field contains 'jenkins-ms'. The 'Type' dropdown is set to 'CNAME – Link your subdomain to another record'. Under 'Alias record set', the 'No' option is selected. The 'TTL' field is set to '1', 'TTL unit' is 'Hours', and the 'Alias' field contains 'jenkins-ms-[REDACTED].us-east-2.elb.amazon...'. At the bottom of the modal, there are 'Add', 'Cancel', and 'Give feedback' buttons.

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

The screenshot shows the Jenkins Manage Jenkins dashboard. On the left sidebar, there are links for New Item, Build History, Manage Jenkins (which is selected and highlighted with a yellow box), My Views, Build Queue (No builds in the queue), and Build Executor Status (0/2). The main content area is titled 'Manage Jenkins' and contains sections for System Configuration (System, Tools, Nodes, Clouds), Security (Security, Users), and Plugins (Plugins, Appearance). A 'Credentials' link under the Security section is also highlighted with a yellow box. At the bottom of the page, there is a navigation bar with links for REST API and Jenkins 2.504.1.

Global credentials (unrestricted)

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

[jenkins-ms.singhritesh85.com/manage/credentials/store/system/_/](#)

Jenkins

Ritesh Kumar Singh log out

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

ID	Name	Kind	Description
 jenkins-cred	jenkins/***** (jenkins-cred)	Username with password	jenkins-cred

Icon: S M L

REST API Jenkins 2.504.1

[jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/newCredentials](#)

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username:

Treat username as secret

Password:

ID: github-cred

Description: github-cred

Create

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

jenkins-ms.singhritesh85.com/manage/computer/new

Jenkins

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

jenkins-ms.singhritesh85.com/manage/computer/creatitem

Jenkins

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

The screenshot shows two Jenkins web pages. The top page is a 'Create Item' form for a new node named 'Slave-1'. It includes fields for Labels ('Slave-1'), Usage ('Use this node as much as possible'), Launch method ('Launch agents via SSH'), Host ('10.10.10.10'), Credentials ('jenkins/******** (jenkins-cred)'), and Host Key Verification Strategy ('Non verifying Verification Strategy'). A yellow box highlights the 'Save' button. The bottom page is the 'Manage Jenkins > Nodes > Slave-1' view for 'Agent Slave-1'. It shows a status bar with 'Status' (green), 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', and 'Log'. A 'Monitoring Data' section is present. Below it, a 'Projects tied to Slave-1' section shows 'None'. A 'Build Executor Status' table is at the bottom left. At the top right, there are links for 'Edit description', 'Relaunch agent', 'Mark this node temporarily offline', and a help icon. The Jenkins logo and user 'Ritesh Kumar Singh' are at the top.

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-2:02XXXXXXXXXX6:certificate/XXXXXXX-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@XXXXXXXXXX ~]# kubectl config get-contexts
CURRENT  NAME          CLUSTER      AUTHINFO
*       aks-cluster    aks-cluster   clusterUser_aks-rg_aks-cluster
[root@XXXXXXXXXX ~]# kubectl config use-context eks-demo-cluster-dev
Switched to context "eks-demo-cluster-dev".
[root@XXXXXXXXXX ~]# kubectl create ns ingress-nginx
namespace/ingress-nginx created
[root@XXXXXXXXXX ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
[root@XXXXXXXXXX ~]# 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 Helm-ing!
```

```
[root@XXXXXXXXXX ~]# 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-2:02XXXXXXXXXX6:certificate/XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX --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@XXXXXXXXXX ~]# kubectl get all -n ingress-nginx
NAME           READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-XXXXXX   1/1     Running   0          2m6s

NAME          AGE
service/ingress-nginx-controller   LoadBalancer  10.XXXXXX.92.a.XXXXXXX.us-east-2.elb.amazonaws.com  80:32585/
TCP,443:31666/TCP  2m6s
service/ingress-nginx-controller-admission ClusterIP   10.XXXXXX.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-XXXXXX  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-namespaces --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@[REDACTED] ~]# helm repo add hashicorp https://helm.releases.hashicorp.com
"hashicorp" has been added to your repositories
[root@[REDACTED] ~]# 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@[REDACTED] ~]# 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.replicas=3,server.dev.enabled=false,server.ui.enabled=true,server.ui.serviceType=ClusterIP,server.dataStorage.size=2Gi,eClass=gp2,server.dataStorage.size=2Gi
Release "vault" does not exist. Installing it now.
NAME: vault
LAST DEPLOYED: [REDACTED] 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
```

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-[REDACTED]	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-7XXXXXXXXXb-mXXXv	1/1	Running	0	3h46m

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

```
[root@yellow ~]# 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: 1yellow8
Unseal Key 2: Nyellowd
Unseal Key 3: Fyellowx
Unseal Key 4: 2yellowG
Unseal Key 5: Xyellowi

Initial Root Token: hyellow4

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  8yellowf
Version      1.19.0
Build Date   2025
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

Ritesh Kumar

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

```

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

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

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

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

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

```

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

```
[root@[REDACTED] 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
UnsealNonce  [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
UnsealNonce  [REDACTED]
Version      1.19.0
Build Date   2025[REDACTED]
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

R Singh

```
/ $ 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
UnsealNonce  n/a
Version      1.19.0
Build Date   2025-03-14T14:25:00Z
Storage Type raft
Removed From Cluster false
HA Enabled   true
/ $
```

```
[root@REDACTED main]# kubectl -n vault exec -ti vault-0 -- vault operator raft list-peers
Node          Address          State  Voter
---          -----
REDACTED        vault-0.vault-internal:8201  leader  true
REDACTED        vault-1.vault-internal:8201  follower  true
REDACTED        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@REDACTED ~]# kubectl apply -f vault-dev-ingress-rule.yaml --context=eks-demo-cluster-dev
ingress.networking.k8s.io/vault-ingress created
[root@REDACTED ~]# 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  aREDACTED.us-east-2.elb.amazonaws.com  80      15s
^C[root@REDACTED ~]#
[root@REDACTED ~]#
[root@REDACTED ~]# 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
```

The screenshot shows the Azure DNS Management portal for the domain singhritesh85.com. On the left, the 'Recordsets' section is selected. A modal window titled 'Add record set' is open, prompting for a 'Name' (set to 'vault-dev') and a 'Type' (set to 'CNAME – Link your subdomain to another record'). The 'Alias record set' dropdown is set to 'No'. The 'TTL' field is set to '1' and the 'TTL unit' is 'Hours'. The 'Alias' field contains 'vault-dev.singhritesh85.com'. At the bottom of the modal are 'Add' and 'Cancel' buttons, with 'Give feedback' link below.

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

The screenshot shows a web browser displaying the 'Sign in to Vault' page. The URL in the address bar is 'vault-dev.singhritesh85.com/ui/vault/auth?with=token'. The page features a logo at the top, followed by a form with 'Method' set to 'Token' and a 'Token' input field. A 'Sign in' button is at the bottom of the form. Below the form, a note says 'Contact your administrator for login credentials.' At the bottom of the page, there's a footer with links to 'Vault', 'Upgrade to Vault Enterprise', 'Documentation', 'Support', 'Terms', 'Privacy', 'Security', 'Accessibility', and a copyright notice for HashiCorp.

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

[root@yellow ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" already exists with the same configuration, skipping
[root@yellow ~]# 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 Helm-ing!
[root@yellow ~]# helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx

[root@yellow ~]# 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@yellow ~]# kubectl get all -n ingress-nginx
NAME                           READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-  1/1     Running   0          5m43s
                                          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/ingress-nginx-controller   LoadBalancer   10.0.0.227   4.0.0.216   80:32191/TCP,443:31275/TCP   5m43s
service/ingress-nginx-controller-admission   ClusterIP   10.0.0.213   <none>       443/TCP          5m43s
                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller  1/1     1          1          5m43s
                                          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: [REDACTED] 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-[REDACTED] 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@XXXXXXXXXX ~]# 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 XXXXXXXXXX F
Unseal Key 2: O XXXXXXXXXX J
Unseal Key 3: 8 XXXXXXXXXX Z
Unseal Key 4: 8 XXXXXXXXXX +
Unseal Key 5: o XXXXXXXXXX C

Initial Root Token: H XXXXXXXXXX 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 XXXXXXXXXX
Version      1.19.0
Build Date   2025-01-10T14:45:00Z
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

Ritesh Kumar

```
/ $ 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[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   infinity
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.
```

Ritesh Kumar

```
/ $ 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 [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
Version      1.19.0
Build Date   2025 [REDACTED]
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

Rite*

```
/ $ 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
UnsealNonce  n/a
Version      1.19.0
Build Date   2025[REDACTED]
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

```
[root@[REDACTED] ~]# 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
UnsealNonce  [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
UnsealNonce  [REDACTED]
Version      1.19.0
Build Date   2025[REDACTED]
Storage Type raft
Removed From Cluster false
HA Enabled   true
```

RIT
SINGH

```
/ $ 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
UnsealNonce  n/a
Version      1.19.0
Build Date   2025-04-18T14:45:00Z
Storage Type raft
Removed From Cluster false
HA Enabled   true
/ $
[root@[REDACTED] ~]#
```

```
[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
[REDACTED]           vault-1.vault-internal:8201  follower  true
[REDACTED]           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@[REDACTED] ~]# 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@[REDACTED] ~]# kubectl apply -f vault-stage-ingress-rule.yaml --context=aks-cluster
ingress.networking.k8s.io/vault-ingress created
[root@[REDACTED] ~]# 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.[REDACTED].216  80, 443  10s
^C[root@[REDACTED] ~]#
[root@[REDACTED] ~]# 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
```

Ritesh Kumar

```
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
```

singhritesh85.com | Records

A record set is a collection of records in a zone that have been loaded on this page. If you don't see what you load. [Learn more](#)

Fetched 5 record set(s).

TTL	Value
172800	[REDACTED]
3600	[REDACTED]
3600	[REDACTED]

Add or remove favorites by pressing **Ctrl+Shift+F**

Add record set

Name: **vault-stage**

Type: **A – IPv4 Address records**

Alias record set: **No**

TTL *: **1**

TTL unit: **Hours**

IP address: **4.192.216.0.0.0**

Add **Cancel** **Give feedback**

Sign in to Vault

Method: **Token**

Token:

Sign in

Contact your administrator for login credentials.

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
```

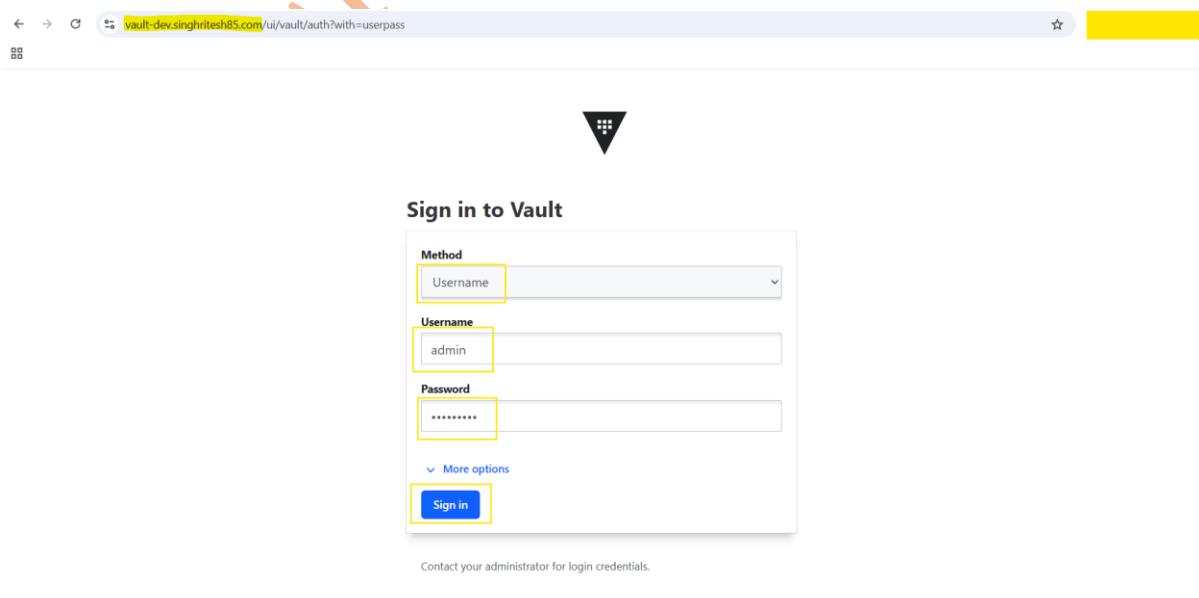
```
$ vault policy write admin -<<EOF
```

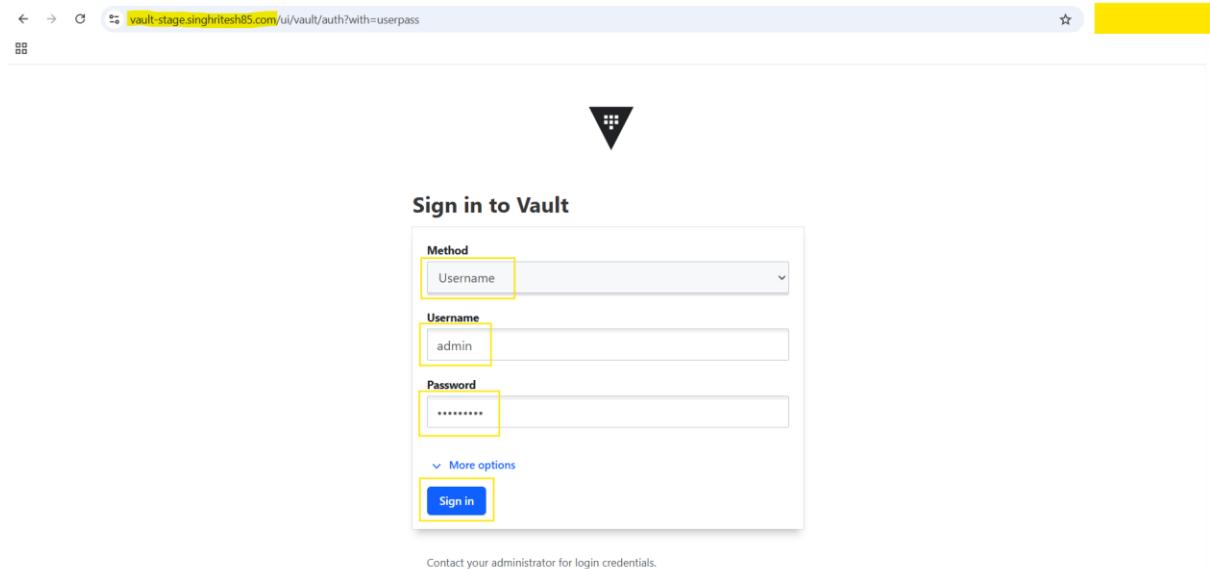
```
path "*" {
  capabilities = ["create", "read", "update", "delete", "list", "sudo"]
}
EOF
```

```
$ 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@yellow ~]# 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_ea37fc08  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.

The screenshot shows the Hashicorp Vault UI interface. On the left is a sidebar with navigation links like 'Vault', 'Dashboard', 'Secrets Engines', 'Access', 'Policies', 'Tools', 'Monitoring', 'Raft Storage', 'Client Count', and 'Seal Vault'. The main area shows a breadcrumb path 'Secrets / bankapp / database / config' and a title 'database/config'. Below the title are tabs for 'Overview', 'Secret' (which is selected), 'Metadata', 'Paths', and 'Version History'. Under the 'Secret' tab, there's a 'JSON' button, a 'Delete' button, a 'Destroy' button, a 'Copy' dropdown, a 'Version 1' dropdown, and a 'Create new version +' button. A table lists a single secret entry: 'password' with a value of 'Dexter@123'. At the bottom of the page, there are links for 'Vault 1.19.0', 'Upgrade to Vault Enterprise', 'Documentation', 'Support', 'Terms', 'Privacy', 'Security', 'Accessibility', and a copyright notice '© 2025 HashiCorp'.

The screenshot shows the Hashicorp Vault UI interface. On the left is a sidebar with 'Vault' and 'Dashboard' sections, and 'Secrets Engines' highlighted. The main area shows a breadcrumb path: Secrets / bankapp / database / config. Below this is a table with one row, where 'password' is the key and 'Dexter@123' is the value. A yellow box highlights the 'password' key and its value. At the bottom right of the table, it says 'Version 1 created May 18, 2025 01:23 PM'. The footer includes links for Vault 1.19.0, Upgrade to Vault Enterprise, Documentation, Support, Terms, Privacy, Security, Accessibility, and © 2025 Hashicorp.

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@yellow ~]# 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@yellow ~]# 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@yellow ~]# 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@yellow ~]# 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@yellow ~]# kubectl config get-contexts
CURRENT  NAME          CLUSTER      AUTHINFO                                     NAMESPACES
*        aks-cluster    aks-cluster   clusterUser_aks-rg_aks-cluster
*        eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:02[REDACTED]:cluster/eks-demo-cluster-dev
[root@yellow ~]# kubectl create ns mysql
namespace/mysql created
[root@yellow ~]# kubectl create sa bankapp-sa -n mysql
serviceaccount/bankapp-sa created
[root@yellow ~]# kubectl config use-context aks-cluster
Switched to context "aks-cluster".
[root@yellow ~]# kubectl create ns mysql
namespace/mysql created
[root@yellow ~]# 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@yellow ~]# kubectl config get-contexts
CURRENT   NAME          CLUSTER           AUTHINFO
*         aks-cluster    aks-cluster        clusterUser_aks_rg_aks-cluster
          eks-demo-cluster-dev  eks-demo-cluster-dev  arn:aws:eks:us-east-2:02yellow:6:cluster/eks-demo-cluster-dev
[root@yellow ~]# kubectl config use-context eks-demo-cluster-dev
Switched to context "eks-demo-cluster-dev".
[root@yellow ~]# kubectl create namespace argocd
namespace/argocd created
[root@yellow ~]# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
[root@yellow ~]# kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
k[redacted][root@yellow ~]#
```

```
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@singhritesh85 ~]# kubectl apply -f argocd-ingress-rule.yaml
ingress.networking.k8s.io/minimal-ingress created
[root@singhritesh85 ~]# 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
```

NAMESPACE	NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
argocd	minimal-ingress	nginx	argocd.singhritesh85.com	a [REDACTED]	80	85s
vault	vault-ingress	nginx	vault-dev.singhritesh85.com	a [REDACTED]	80	140m

singhritesh85.com | Recordsets

DNS zone

Search

[+ Add](#) [Refresh](#) [Delete](#) [Give feedback](#)

- [Overview](#)
- [Activity log](#)
- [Access control \(IAM\)](#)
- [Tags](#)
- [Diagnose and solve problems](#)
- [Resource visualizer](#)
- [Settings](#)
- [DNS Management](#)
- Records**
- [DNSSEC](#)
- [Monitoring](#)
- [Automation](#)
- [Help](#)

A record set is a collection of records in a zone that have been loaded on this page. If you don't see what you load. [Learn more](#)

Search

Fetched 6 record set(s).

Name	Type
@	NS
@	SOA

Add or remove favorites by pressing **Ctrl + Shift + F**

Add record set

singhritesh85.com

Name

Type

Alias record set

TTL *

TTL unit

Alias

[Add](#) [Cancel](#) [Give feedback](#)

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

argocd.singhritesh85.com/applications

argo v3.0.1+2bcdf48

Applications

[+ NEW APP](#) [SYNC APPS](#) [REFRESH APPS](#) [Log out](#)

No applications available to you just yet

Create new application to start managing resources in your cluster

[CREATE APPLICATION](#)

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@[REDACTED] ~]$ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
[jenkins@[REDACTED] ~]$ sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
[jenkins@[REDACTED] ~]$ 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@[REDACTED] ~]$ 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@[REDACTED] ~]$ 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 want to continue [y/N]? y
{"level":"info","msg":"ServiceAccount `\"argocd-manager\"` created in namespace `\"kube-system\"`","time":"2025-01-11T10:45:25.000Z"}
{"level":"info","msg":"ClusterRole `\"argocd-manager-role\"` created","time":"2025-01-11T10:45:25.000Z"}
{"level":"info","msg":"ClusterRoleBinding `\"argocd-manager-role-binding\"` created","time":"2025-01-11T10:45:25.000Z"}
{"level":"info","msg":"Created bearer token secret for ServiceAccount `\"argocd-manager\"`","time":"2025-01-11T10:45:25.000Z"}
Cluster 'https://aks-cluster-dns-[REDACTED].eastus.azurek8s.io:443' added
```



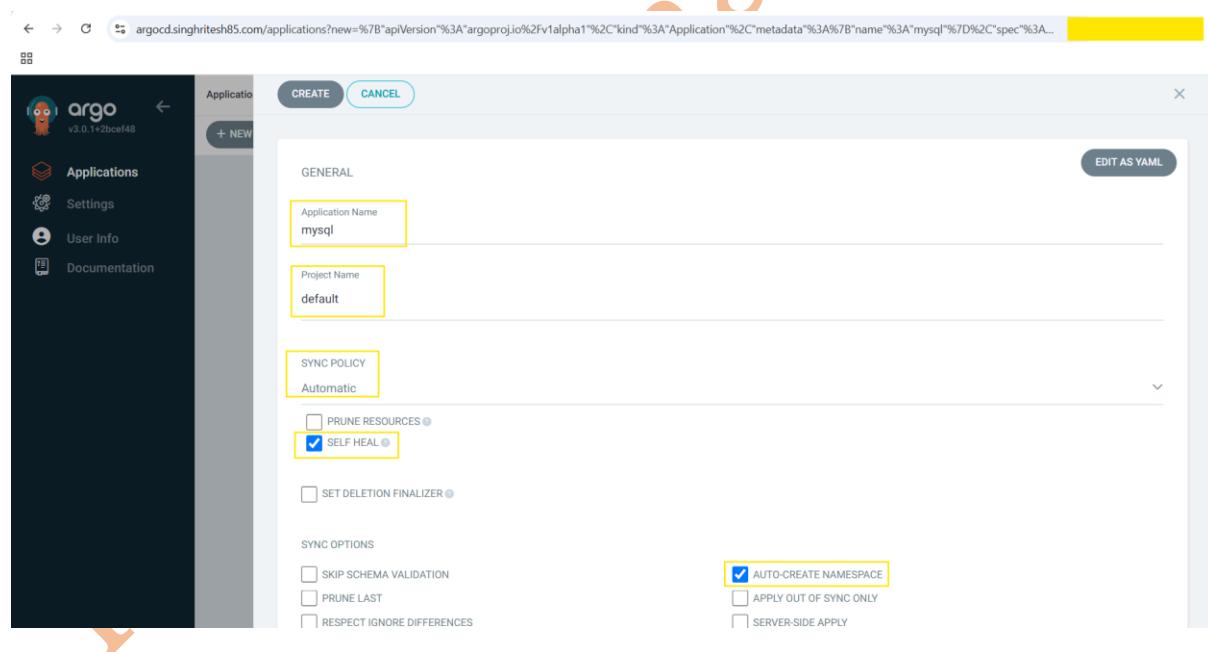
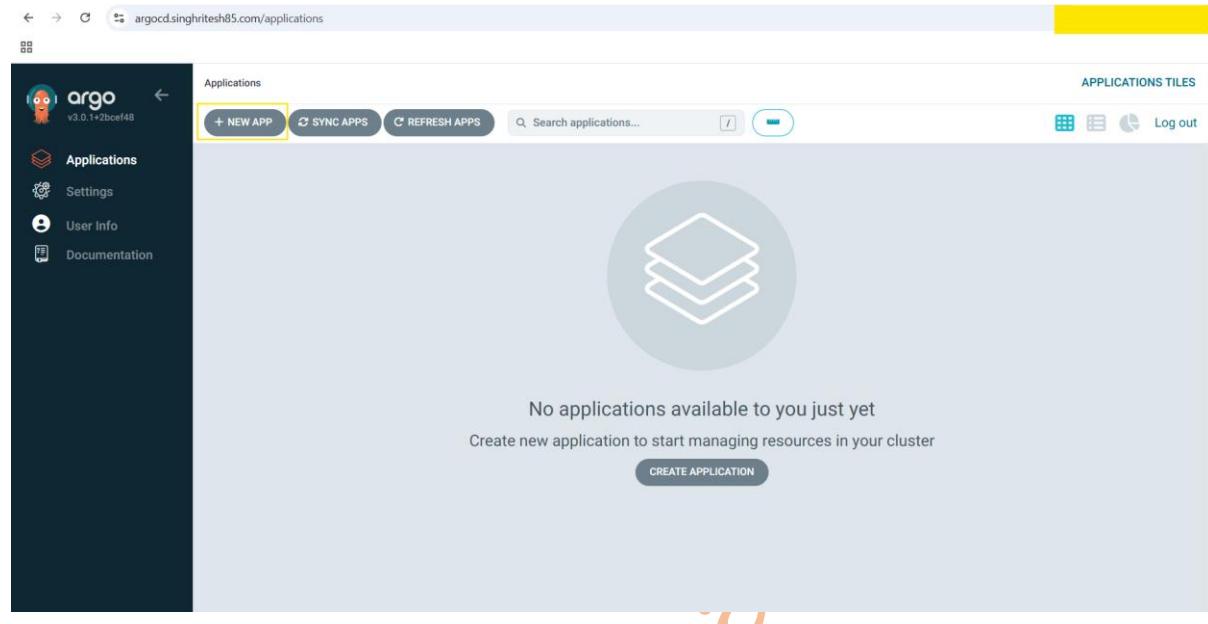
NAME	URL	VERSION	CONNECTION STATUS
aks-cluster	https://aks-cluster-dns-[REDACTED].eastus.azurek8s.io:443	v1.0.0	Unknown
in-cluster	https://kubernetes.default.svc	v1.0.0	Unknown

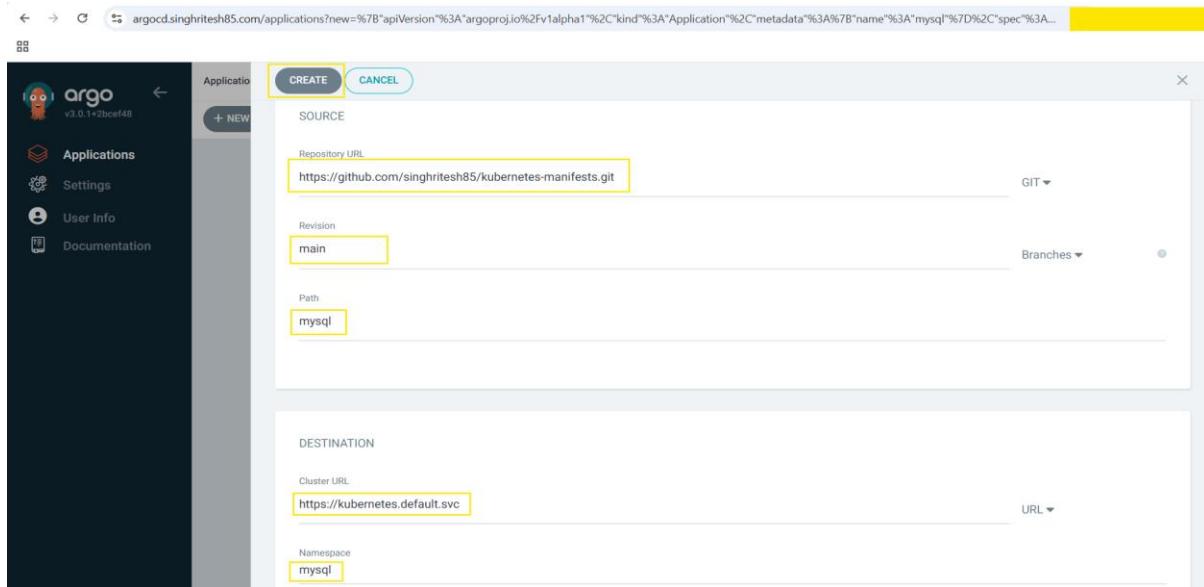
On Jenkins Slave Node I had provided Authentication and Authorization using Access Key and Secret Key as shown in the screenshot attached below.

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

Module-1: Manual Approach

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





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

```
[jenkins@... ~]$ kubectl get pods -n mysql
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   2/2     Running   0          1d
mysql-1   2/2     Running   0          1d
[jenkins@... ~]$ kubectl get svc -n mysql
NAME        TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
mysql-svc   ClusterIP  10.41.41.41 <none>       3306/TCP  1d
mysql-svc-headless   ClusterIP  None         <none>       3306/TCP  1d
[jenkins@... ~]$ kubectl get pvc -n mysql
NAME        STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-data-mysql-0   Bound   pvc-...   1Gi        RWO          gp2           <unset>          <unset>          1d
mysql-data-mysql-1   Bound   pvc-...   1Gi        RWO          gp2           <unset>          <unset>          1d
```

The screenshot shows the ArgoCD interface with the URL argocd.singhritesh85.com/applications/argocd/mysql?view=tree&resource=.... The left sidebar has sections for Applications, Settings, User Info, and Documentation. Resource filters include NAME, KINDS, SYNC STATUS (Synced: 5, OutOfSync: 0), and HEALTH STATUS (Progressing: 0). The main area displays the application tree for 'mysql'. It includes a 'mysql' icon, a 'mysql' service (Synced, 10 minutes ago), a 'mysql-svc' service (Synced, 10 minutes ago), a 'mysql-svc-headless' service (Synced, 10 minutes ago), a 'bankapp-sa' service (Synced, 2 hours ago), and a 'mysql' pod (Running, 10 minutes ago). A 'LAST SYNC' section shows a successful sync to 'aca2778' at 10 minutes ago. Application details are shown in a tree view on the right.

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

The screenshot shows the ArgoCD interface with the URL argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=.... The left sidebar has sections for Applications, Settings, User Info, and Documentation. Application filters include Favorites Only, SYNC STATUS (Unknown: 0, Synced: 1, OutOfSync: 0), and HEALTH STATUS (Progressing: 0, Suspended: 0, Healthy: 1, Degraded: 0). The main area shows a single application entry for 'mysql'. It has a 'NEW APP' button highlighted with a yellow box. The application details are as follows:

- mysql**
- Project: default
- Labels: healthy, synced
- Status: healthy, Synced
- Repository: <https://github.com/singhritesh85/kuber...>
- Target R...: main
- Path: mysql
- Destinati...: in-cluster
- Namespac...: mysql
- Created ...: 2025-07-10 10:25:22 (10 minutes ago)
- Last Sync: 2025-07-10 10:25:22 (10 minutes ago)

Buttons at the bottom include SYNC, REFRESH, and DELETE. The right side shows application tiles and a log out link.

The screenshots show the Argo UI interface for creating a new application named 'mysql-stage'. The first screenshot displays the 'GENERAL' configuration tab, where the application name is set to 'mysql-stage', the project is 'default', and the sync policy is 'Automatic'. The 'SELF HEAL' option is checked under 'SYNC POLICY'. The second screenshot shows the 'SOURCE' configuration tab, with the repository URL set to 'https://github.com/singhritesh85/kubernetes-manifests.git', the revision set to 'main', and the path set to 'mysql-stage-aks'. The 'DESTINATION' tab shows the cluster URL as 'https://aks-cluster-dns-.eastus.azurek8s.io:443' and the namespace set to 'mysql'.

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@.47   <none>       3306/TCP  2m13s
mysql-svc-headless   ClusterIP  None        <none>       3306/TCP  2m13s
^C[jenkins@   1Gi        Rwo        managed-csi   <unset>   2m21s
mysql-data-mysql-1   Bound   pvc-   1Gi        Rwo        managed-csi   <unset>   86s
```

The screenshot shows the Argo CD interface with two MySQL applications listed:

- mysql**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber..., Target R.: main, Path: mysql, Destination: in-cluster, Namespace: mysql. Created: 2025-01-19, Last Sync: 2025-01-19.
- mysql-stage**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber..., Target R.: main, Path: mysql-stage-aks, Destination: aks-cluster, Namespace: mysql. Created: 2025-01-19, Last Sync: 2025-01-19.

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=02XXXXXXXXXX6.dkr.ecr.us-east-2.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password) --namespace=bankapp --context=aks-cluster
```

```
[root@ip-172-31-10-10 ~]# kubectl create secret docker-registry regcred --docker-server=02XXXXXXXXXX6.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

Configuration

General

Display Name: bankapp

Description:

Plain text [Preview](#)

Branch Sources

Git
Project Repository: https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git

Credentials: singhritesh85/******** (github-cred)

+ Add

Behaviors

Discover branches

Filter by name (with wildcards)
Include: dev stage

Save Apply

Branch Sources

Git
Project Repository: https://github.com/singhritesh85/Bank-App-using-Hashicorp-Vault.git

Credentials: singhritesh85/******** (github-cred)

+ Add

Behaviors

Discover branches

Filter by name (with wildcards)
Include: dev stage

Save Apply

Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile-manual-approach

Scan Multibranch Pipeline Triggers

Periodically if not otherwise run

Orphaned Item Strategy

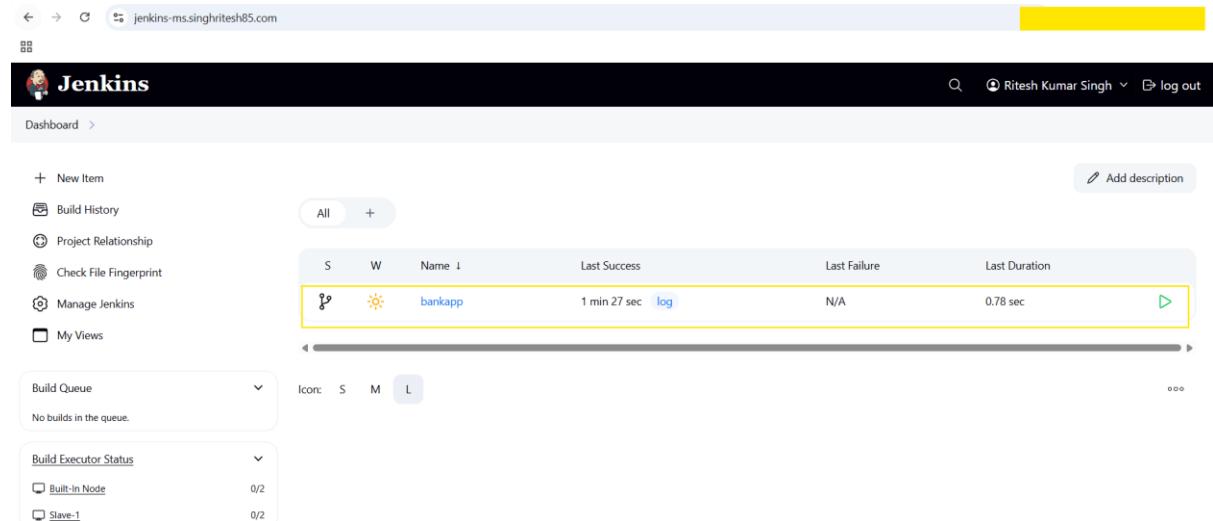
Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.

Abort builds

Discard old items

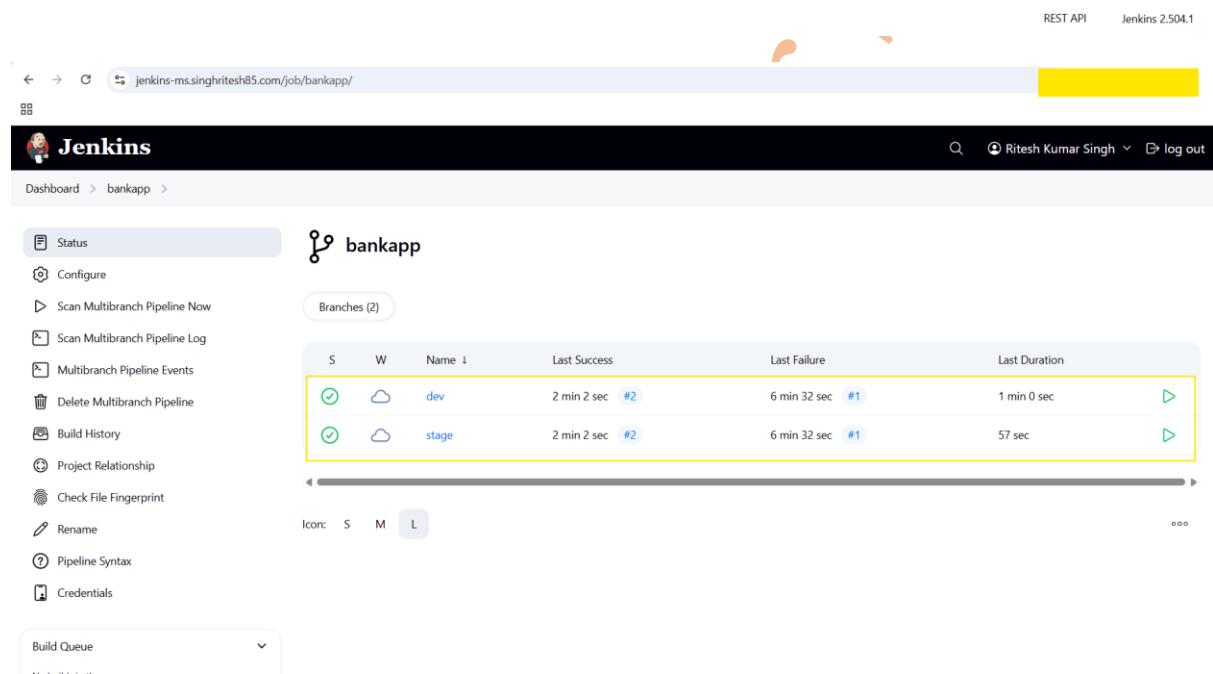
Save Apply

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



The screenshot shows the Jenkins dashboard. A multi-branch job named "bankapp" is highlighted with a yellow border. The table below shows the branches: dev (Last Success: 2 min 2 sec, Last Failure: 6 min 32 sec, Last Duration: 1 min 0 sec) and stage (Last Success: 2 min 2 sec, Last Failure: 6 min 32 sec, Last Duration: 57 sec). Other sections visible include Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Build Queue (empty), Build Executor Status (Built-in Node 0/2, Slave-1 0/2), and a REST API endpoint.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		bankapp	1 min 27 sec log	N/A	0.78 sec Duration



The screenshot shows the details for the "bankapp" job. It lists two branches: "dev" and "stage". Both branches have a green checkmark icon and a blue cloud icon. The "dev" branch has a last success of 2 min 2 sec, a last failure of 6 min 32 sec, and a last duration of 1 min 0 sec. The "stage" branch has a last success of 2 min 2 sec, a last failure of 6 min 32 sec, and a last duration of 57 sec. The left sidebar contains options like 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, and Credentials.

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%2F%2F%20%22new%22%20%3D%20%22true%22%7D

The screenshot shows the Argo CD interface with two application cards displayed:

- mysql**: Project: default, Status: Healthy Synced, Repository: https://github.com/singhritesh85/kubernetes-manifests.git, Target R.: main, Path: mysql, Destination: in-cluster, Namespace: mysql, Created: 2025-01-20, Last Sync: 2025-01-20.
- mysql-stage**: Project: default, Status: Healthy Synced, Repository: https://github.com/singhritesh85/kubernetes-manifests.git, Target R.: main, Path: mysql-stage-aks, Destination: aks-cluster, Namespace: mysql, Created: 2025-01-20, Last Sync: 2025-01-20.

argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B%2F%2F%20%22new%22%20%3D%20%22true%22%7D

The screenshot shows the "CREATE" dialog for a new application:

- GENERAL** tab: Application Name: bankapp, Project Name: default.
- SYNC POLICY** tab: Automatic, PRUNE RESOURCES (unchecked), SELF HEAL (checked), SET DELETION FINALIZER (unchecked).
- SYNC OPTIONS** tab: SKIP SCHEMA VALIDATION (unchecked), PRUNE LAST (unchecked), RESPECT IGNORE DIFFERENCES (unchecked), AUTO-CREATE NAMESPACE (checked), APPLY OUT OF SYNC ONLY (unchecked), SERVER-SIDE APPLY (unchecked).

argocd.singhritesh85.com/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=&new=%7B%2F%2F%20%22new%22%20%3D%20%22true%22%7D

The screenshot shows the "CREATE" dialog for a new application, focusing on the SOURCE section:

- SOURCE** tab: Repository URL: https://github.com/singhritesh85/kubernetes-manifests.git, Revision: main, Path: bankapp.
- DESTINATION** tab: Cluster URL: https://kubernetes.default.svc, Namespace: bankapp.

The image displays two screenshots of the Argo CD web interface, showing the creation of a new application named "bankapp-stage".

Screenshot 1: Application Creation - General Tab

- Application Name:** bankapp-stage
- Project Name:** default
- Sync Policy:** Automatic
- Sync Options:**
 - PRUNE RESOURCES
 - SELF HEAL
 - SET DELETION FINALIZER
- Sync Options (continued):**
 - SKIP SCHEMA VALIDATION
 - PRUNE LAST
 - RESPECT IGNORE DIFFERENCES
- Advanced Options:**
 - AUTO-CREATE NAMESPACE
 - APPLY OUT OF SYNC ONLY
 - SERVER-SIDE APPLY

Screenshot 2: Application Creation - Source Tab

- Source:**
 - Repository URL:** https://github.com/singhritesh85/kubernetes-manifests.git
 - Revision:** main
 - Path:** bankapp-stage
- Destination:**
 - Cluster URL:** https://aks-cluster-dns-eastus.azureaks.io:443
 - Namespace:** bankapp

The screenshot shows the Argo UI interface. On the left, there's a sidebar with 'Applications' selected. The main area displays four application cards:

- bankapp**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber... Target R...: main, Path: bankapp, Destinati...: in-cluster, Namesp...: bankapp, Created ...: 2025, Last Sync: 2025.
- bankapp-stage**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber... Target R...: main, Path: bankapp-stage, Destinati...: aks-cluster, Namesp...: bankapp, Created ...: 2025, Last Sync: 2025.
- mysql**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber... Target R...: main, Path: mysql, Destinati...: in-cluster, Namesp...: mysql, Created ...: 2025, Last Sync: 2025.
- mysql-stage**: Project: default, Status: Healthy Synced, Repo: https://github.com/singhritesh85/kuber... Target R...: main, Path: mysql-stage, Destinati...: aks-cluster, Namesp...: mysql, Created ...: 2025, Last Sync: 2025.

Buttons for SYNC, REFRESH, and DELETE are at the bottom of each card.

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          2m
bankapp-...     1/1     Running   0          2m
^C[jenkins@... ~]$ 
[jenkins@... ~]$ kubectl get pods -n bankapp --watch --context=aks-cluster
NAME           READY   STATUS    RESTARTS   AGE
bankapp-...     1/1     Running   0          2m
bankapp-...     1/1     Running   0          2m
[jenkins@... ~]$ kubectl get svc -n bankapp --watch --context=eks-demo-cluster-dev
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP  10.10.182   <none>        80/TCP   2m
[jenkins@... ~]$ 
[jenkins@... ~]$ kubectl get svc -n bankapp --watch --context=aks-cluster
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP  10.10.229   <none>        80/TCP   2m
```

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@192.168.1.101 ~]# 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
```

high

```
[root@192.168.1.101 ~]# 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
```

low

```
[root@192.168.1.101 ~]# kubectl get ing -A --watch --context=aks-cluster
NAMESPACE   NAME           CLASS      HOSTS           ADDRESS        PORTS      AGE
bankapp     bankapp-ingress-stage   nginx    bankapp-stage.singhritesh85.com  172.10.1.112  80, 443  14m
vault       vault-ingress        nginx    vault-stage.singhritesh85.com   172.10.1.112  80, 443  14m
[root@192.168.1.101 ~]
[root@192.168.1.101 ~]# kubectl get ing -A --watch --context=eks-demo-cluster-dev
NAMESPACE   NAME           CLASS      HOSTS           ADDRESS        PORTS      AGE
E
argocd     minimal-ingress    nginx    argocd.singhritesh85.com   a[REDACTED].us-east-2.elb.amazonaws.com  80
14m
bankapp     bankapp-ingress-dev   nginx   bankapp-dev.singhritesh85.com  a[REDACTED].us-east-2.elb.amazonaws.com  80
s
vault       vault-ingress        nginx   vault-dev.singhritesh85.com   a[REDACTED].us-east-2.elb.amazonaws.com  80
```

singhritesh85.com | Recordsets

Add record set

Name: bankapp-stage

Type: A – IPv4 Address records

Alias record set: No

TTL: 1

TTL unit: Hours

IP address: 172.17.0.112

0.0.0.0

Add Cancel Give feedback

singhritesh85.com | Recordsets

Add record set

Name: bankapp-dev

Type: CNAME – Link your subdomain to another record

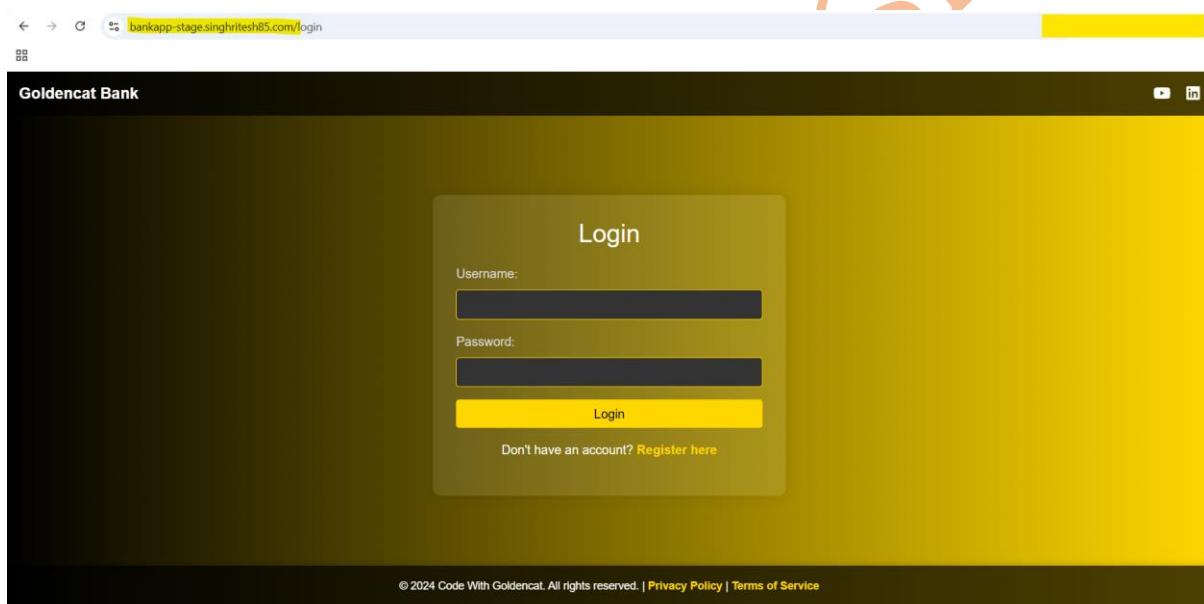
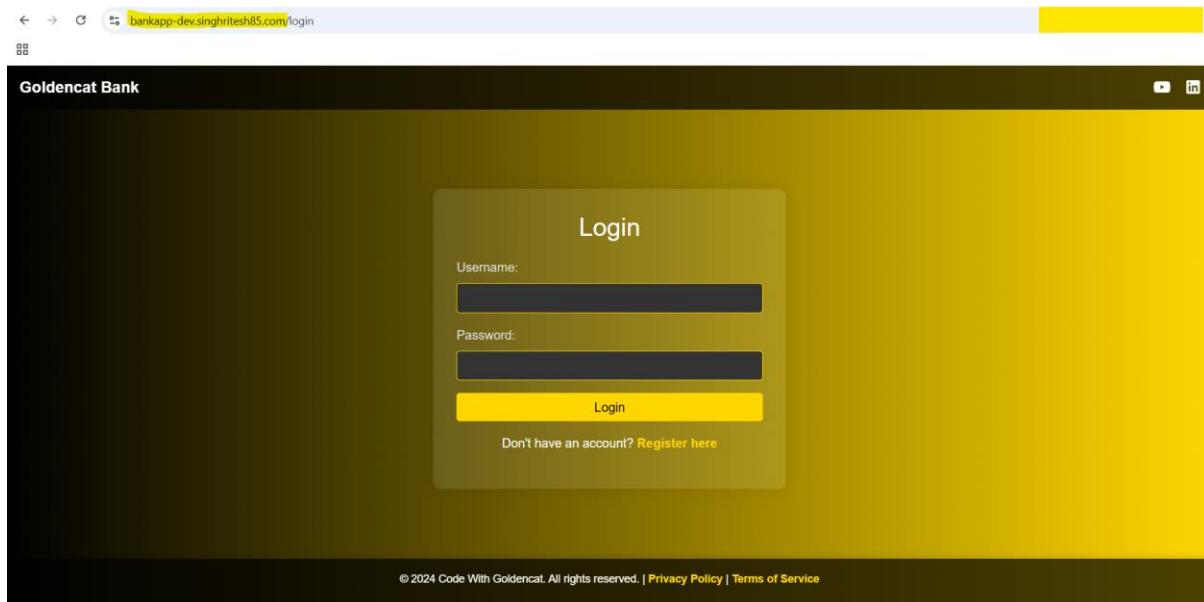
Alias record set: No

TTL: 1

TTL unit: Hours

Alias: a

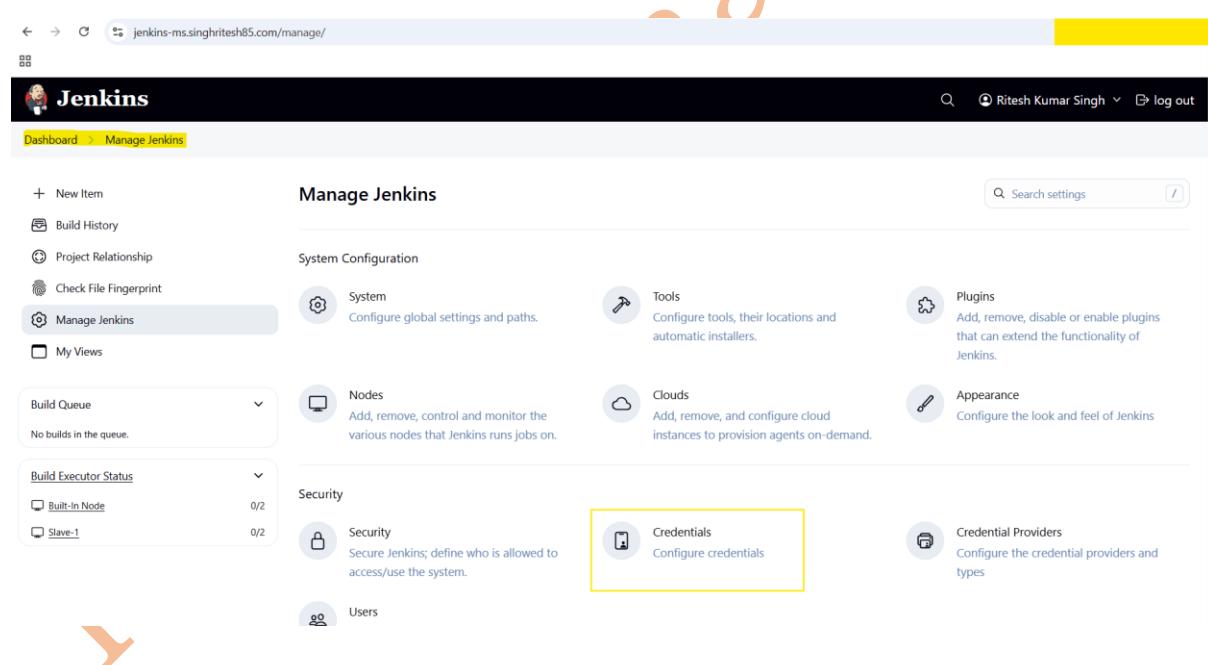
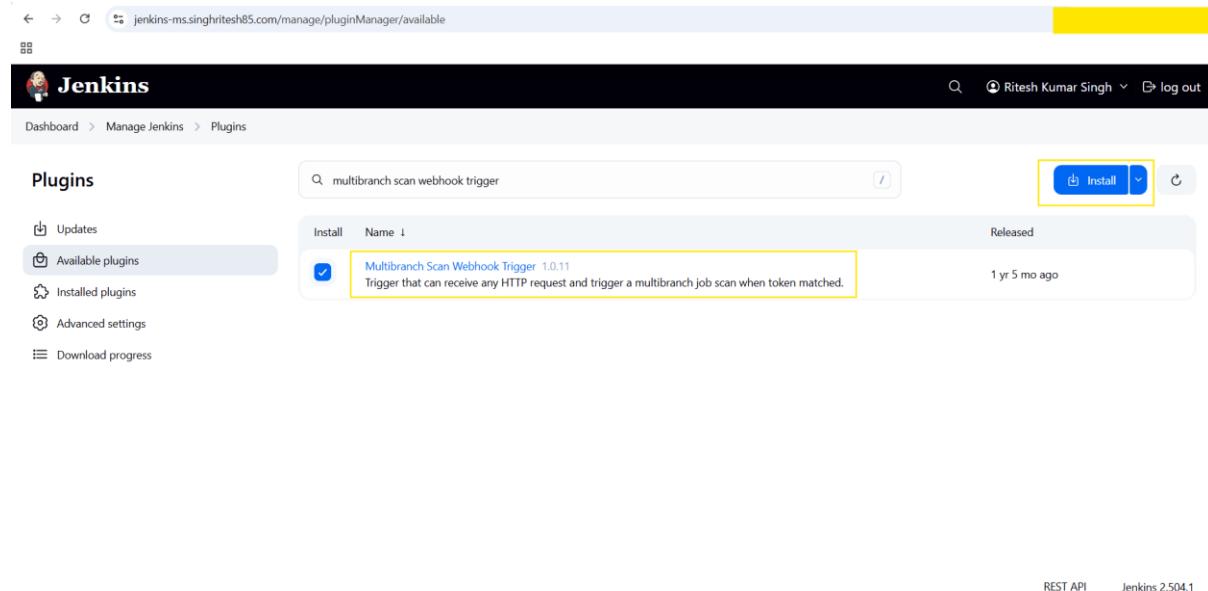
Add Cancel Give feedback



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

Module-2: Automation Approach

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



Jenkins credentials creation screenshot:

The screenshot shows the Jenkins 'New credentials' configuration page. The 'Kind' field is set to 'Secret text'. The 'Scope' is 'Global'. The 'Secret' field contains 'argocd_password'. The 'ID' field is also 'argocd_password'. The 'Description' field is 'argocd_password'. A blue 'Create' button is at the bottom.

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.

Github Repository settings - Webhooks screenshot:

The screenshot shows the 'Webhooks / Add webhook' configuration in a GitHub repository settings. The 'Payload URL' is set to 'https://jenkins-ms.singhritesh85.com/multibranch-webhook-trigger/invoke?token=mytoken'. The 'Content type' is 'application/x-www-form-urlencoded'. The 'SSL verification' option is 'Enable SSL verification'. The 'Events' section is set to 'Just the push event.'

The screenshot shows the GitHub 'Webhooks' settings page for a repository. The left sidebar includes 'Tags', 'Rules', 'Actions', 'Webhooks' (which is selected and highlighted in blue), 'Environments', 'Codespaces', and 'Pages'. Under 'Security', there are 'Advanced Security', 'Deploy keys', and 'Secrets and variables'. Under 'Integrations', there are 'GitHub Apps' and 'Email notifications'. The main right panel shows a form for creating a new webhook:

- Content type ***: application/x-www-form-urlencoded
- Secret**: (empty input field)
- SSL verification**: Enable SSL verification Disable (not recommended)
- Which events would you like to trigger this webhook?**
 - Just the push event.
 - Send me everything.
 - Let me select individual events.
- Active**: (with explanatory text below: "We will deliver event details when this hook is triggered.")
- Add webhook** button

Webhooks

[Add webhook](#)

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <https://jenkins-ms.singhritesh85.co...> (push)

[Edit](#) [Delete](#)

Last delivery was successful.

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

The screenshot shows the Jenkins 'New Item' creation page. The URL is jenkins-ms.singhritesh85.com/newJob. The page has a 'Dashboard > New Item' header. It asks for an item name ('mysql') and item type. The 'Multibranch Pipeline' option is highlighted with a yellow box and described as 'Creates a set of Pipeline projects according to detected branches in one SCM repository.' A large orange hand-drawn style arrow points from the 'Webhooks' section above to this 'Multibranch Pipeline' selection. At the bottom is an 'OK' button.

[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

Display Name: mysql

Description: Jenkins multibranch pipeline to create MySQL Pods in dev and stage branch.

Plain text Preview

Enabled:

Branch Sources

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): dev stage

Include: ?

Exclude: ?

Save Apply

[jenkins-ms.singhritesh85.com/job/mysql/configure](#)

Configuration

- General
- Branch Sources
- Build Configuration
- Scan Multibranch Pipeline Triggers
- Orphaned Item Strategy
- Appearance
- Health metrics
- Properties

Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile-automation-approach-mysql

Scan Multibranch Pipeline Triggers

Periodically if not otherwise run

Scan by webhook

Orphaned Item Strategy

Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.

Abort builds

Save Apply

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

The screenshot displays two screenshots of the Jenkins interface. The top screenshot shows the Jenkins dashboard with a table of jobs. One job, named 'mysql', is highlighted with a yellow border. The bottom screenshot shows a 'New Item' creation dialog where 'bankapp' has been typed into the item name field. A 'Multibranch Pipeline' option is selected and highlighted with a yellow border. The 'OK' button at the bottom of the dialog is also highlighted with a yellow border.

jenkins-ms.singhrithesh85.com/job/bankapp/configure

Jenkins

Ritesh Kumar Singh log out

Dashboard > bankapp > Configuration

Configuration

General

Enabled

Display Name ?
bankapp

Description
Jenkins multibranch pipeline to create Bank Application Pods in dev and stage branch.

Plain text [Preview](#)

Branch Sources

Git Project Repository ?

Save Apply

jenkins-ms.singhrithesh85.com/job/bankapp/configure

Configuration

Branch Sources

Project Repository ?
<https://github.com/singhrithesh85/Bank-App-using-Hashicorp-Vault.git>

Credentials ?
***** (github-cred)

+ Add

Behaviors

Discover branches ?

Filter by name (with wildcards)
Include ?
dev stage

Exclude ?

Save Apply

Ritesh

Configuration

Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile-automation-approach-bankapp

Scan Multibranch Pipeline Triggers

Periodically if not otherwise run ?

Scan by webhook ?
Trigger token: mytoken

Orphaned Item Strategy

Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed

Save **Apply**

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

Jenkins

Dashboard >

New Item **Add description**

Build History **All** **+**

S	W	Name	Last Success	Last Failure	Last Duration
♂	☀	bankapp	3 min 32 sec log	N/A	0.44 sec D
♂	☀	mysql	13 min log	N/A	0.4 sec D

Build Queue No builds in the queue. **Build Executor Status** **Icon:** S M L **Build-in Node** 0/2 **Slave-1** 0/2

REST API Jenkins 2.504.1

The image contains two side-by-side screenshots of the Jenkins web interface, both titled "Status".

Top Screenshot (bankapp Pipeline):

- Page Title:** jenkins-ms.singhrithesh85.com/job/bankapp/
- Header:** Jenkins, Ritesh Kumar Singh, log out
- Breadcrumbs:** Dashboard > bankapp >
- Section:** bankapp
- Description:** 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
- Actions:** 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.
- Build Queue:** (dropdown menu)

Bottom Screenshot (mysql Pipeline):

- Page Title:** jenkins-ms.singhrithesh85.com/job/mysql/
- Header:** Jenkins, Ritesh Kumar Singh, log out
- Breadcrumbs:** Dashboard > mysql >
- Section:** mysql
- Description:** Jenkins multibranch pipeline to create MySQL Pods in dev and stage branch.
- Branches:** (2)

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁️	dev	22 min #6	41 min #3	6.8 sec
✓	☀️	stage	20 min #7	44 min #2	8.5 sec
- Actions:** 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.
- Build Queue:** (dropdown menu)

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

```
cat ingress-rule-dev.yaml

---

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

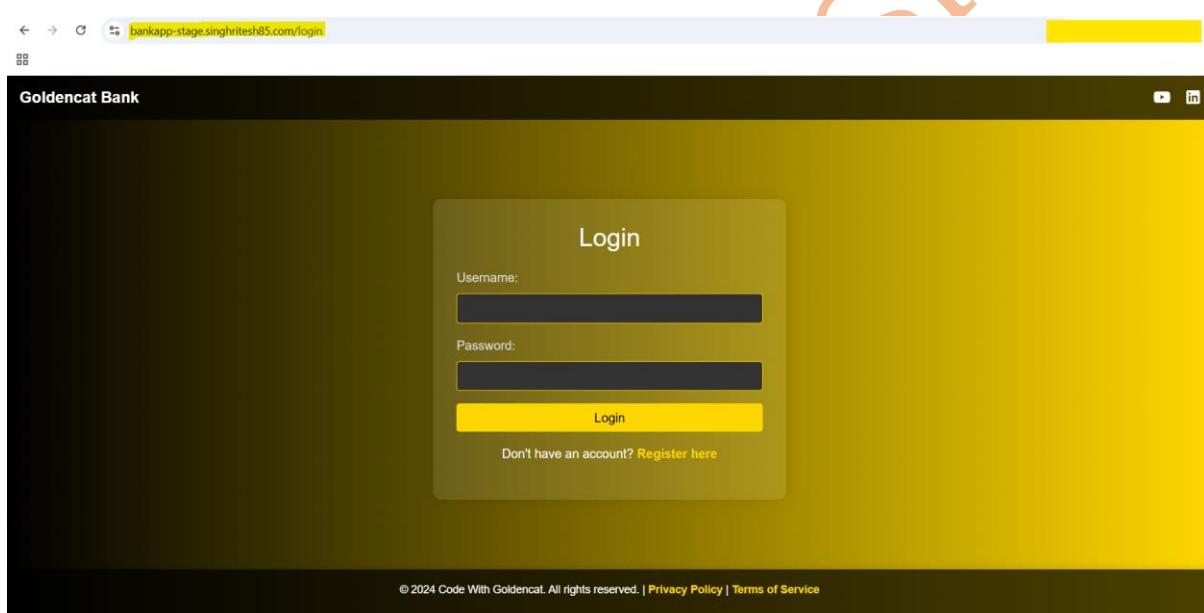
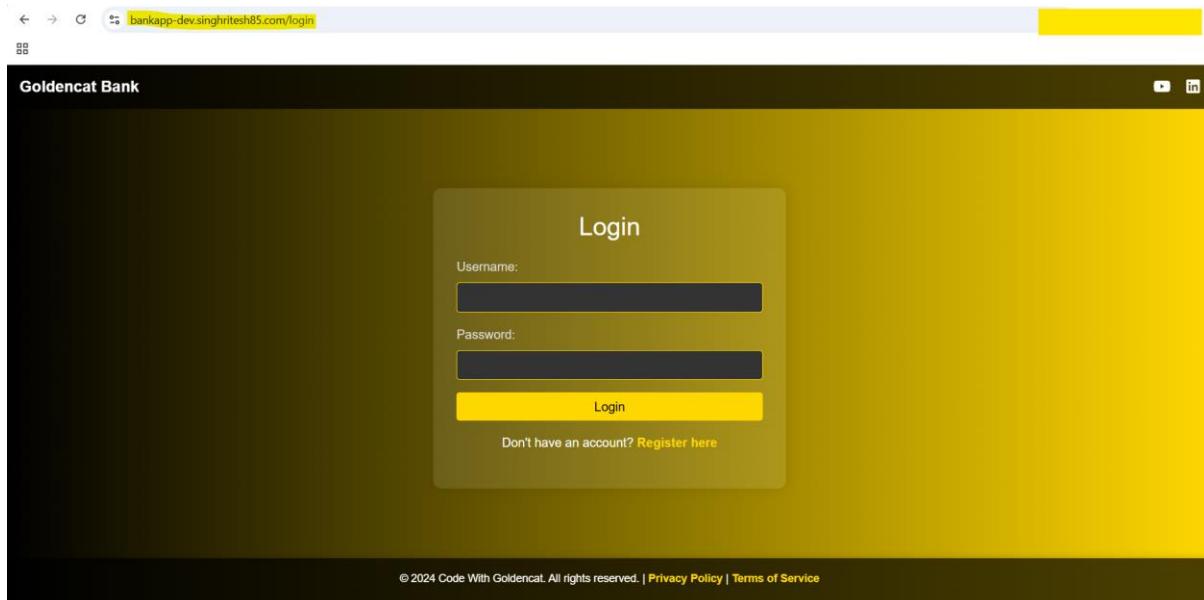
```
cat ingress-rule-stage.yaml
```

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

---

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

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



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

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

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

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

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