

DevOps Project BankApp Terraform upgrade to Opentofu with secure kubernetes Cluster (EKS, AKS and GKE)



By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

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

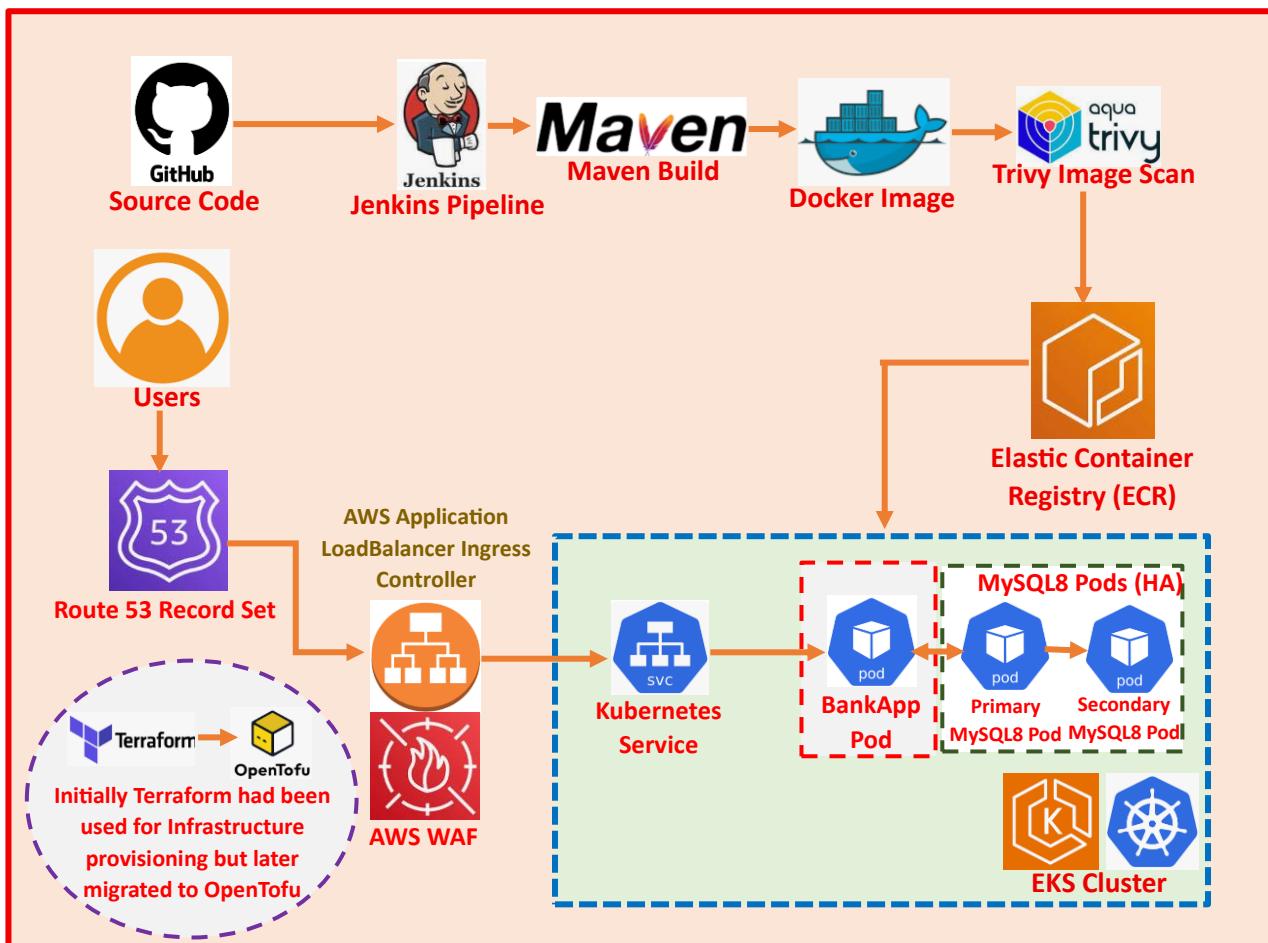
GitHub: - <https://github.com/singhritesh85>



या कुद्देन्दुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवे: सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाङ्घापहा ॥

DevOps Project BankApp Terraform upgrade to OpenTofu with Secure EKS Cluster

Module-1



Till now I used Terraform to provision the infrastructure but Hashicorp license changes due to which Terraform will introduce restrictions for commercial use, due to this reason many organisations are migrating from Terraform to OpenTofu. However, I would like to mention here that DevOps Team who used it as a tool to provision the infrastructure will not be affected but those who provides it as a Service will be affected. You already aware that Terraform state file was kept locally or in the S3 Bucket / Azure Storage Account Container / GCP Bucket. The problem was its state file was not encrypted, so anyone who had the access of Terraform Server (when state file was kept locally) or Bucket or Storage Account Container can easily see its state file. But if you use OpenTofu, you will not face such an issue. OpenTofu is an Open Source, community-driven Infrastructure as Code (IaC) tool, it is a fork of Hashicorp Terraform. I will migrate from Terraform to OpenTofu at the end of Module-1.

This DevOps project deals will creation of infrastructure using Terraform, GitHub was used to keep the source Code and Jenkins had been used as CI/CD Tool. The build tool used in this project was Maven and Docker Image was pushed to the Elastic Container Registry (ECR) which was finally deployed to the Elastic Kubernetes Services (EKS). The Bank Application Pods was accessed using Application LoadBalancer Ingress Controller and hence using the kubernetes service as shown in the screenshot attached above. I applied WAF Policy (to Limit the Rate) to the Bank Application URL and

to prevent from SQL Injection. As this Bank Application was open to entire world so rate limit and SQL Injection was applied through WAF Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot (by the Hackers) so that its speed becomes slow or this website hangs to refrain from such a situation WAF Policy Rate Limit was applied. Finally, I migrated from Terraform to OpenTofu. For EKS Cluster I used cilium in chain mode with VPC CNI the reason to implement this was secure (encrypted with WireGuard) Node-to-Node Communication. I implemented Istio for encrypted Pod-to-Pod Communication and implemented **OPA** (Open Policy Agent) and **Falco** to Secure the EKS Cluster. I had used Aqua Security Trivy image scan for vulnerability scanning during CI/CD Pipeline as shown in the end-to-end architecture diagram of the project drawn above. I used Guard Duty to add extra layer of security to the EKS Cluster. In EKS Cluster with version 1.28 and more envelope encryption is by default implemented, if you do not use AWS Customer Managed Key then using AWS Managed Key envelope encryption gets implemented. The AWS Customer Managed Key enables you to rotate it as per the standard followed by your organisation. Usually after 90 days the AWS Customer Managed Key should be rotated but you can rotate it as per your organisation need. If you use Amazon Managed Key then it will be rotated after 1 year (365 days) by the AWS. In this project I used AWS Customer Managed Key for envelope encryption and to encrypt the all the EBS Volumes used in this project.

In EKS however it is possible to encrypt all the volumes either by Amazon Managed Key or AWS Customer Managed Key but envelope encryption provides extra layer of security by encrypting the etcd.

Terraform script to create the infrastructure is present at the path **terraform-route53-hosted-zone-with-acm-certificate** (this terraform script is handy for you to create the Route53 hosted-zone and AWS Certificate Manager SSL Certificate) in the GitHub Repo

<https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>. Commands as written below had been used to create infrastructure using Terraform.

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

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

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

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

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

```

module.route53_hosted_zone.aws_acm_certificate.acm_cert: Creating...
module.route53_hosted_zone.aws_acm_certificate.acm_cert: Creation complete after 5s [id=arn:aws:acm:us-east-2:02...6:certificate/]
[...]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Creation complete after 31s [...]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Creating...
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Creation complete after 31s [...]
inghrithesh85.com._CNAME]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Creating...
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m40s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m50s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m00s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m10s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m20s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Creation complete after 1m23s [id=... UTC]

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

Outputs:

route53_hosted_zone_details = {
  "certificate_arn" = "arn:aws:acm:us-east-2:02...6:certificate/"
  "hosted_zone_id" = "..."
  "hosted_zone_name_servers" = tolist([
    ...
    ...
    ...
  ])
}

```

Now the Route53 Hosted Zone and AWS Certificate Manager SSL Certificate became ready. Then I created the other AWS Resources using the terraform script present at the path **terraform-eks-cluster-waf-with-velero-backup-with-cmk** in the GitHub Repo

<https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>.

```

module.eks_cluster.aws_eks_addon.core_dns: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.core_dns: Creation complete after 14s [id=eks-demo-cluster-dev:coredns]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Creation complete after 34s [id=eks-demo-cluster-dev:snapshot-controller]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Creation complete after 35s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Creation complete after 44s [id=eks-demo-cluster-dev:aws-ebs-csi-driver]
module.eks_cluster.aws_eks_addon.metrics_server: Creation complete after 44s [id=eks-demo-cluster-dev:metrics-server]

Apply complete! Resources: 104 added, 0 changed, 0 destroyed.

Outputs:

ecr_ec2_private_ip_eks_details = {
  "alb_dns_name" = "jenkins-ms-...-us-east-2.elb.amazonaws.com"
  "alb_ingress_controller_role_arn" = "arn:aws:iam::02...6:role/aws-alb-ingress-controller-role"
  "aws_wafv2_web_acl_arn" = "arn:aws:wafv2:us-east-2:0...6:regional/webacl/aws-wafv2-rate-limit/"
  "eks_cluster_endpoint" = "https://...-us-east-2.eks.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "jenkins_master_instance_id" = "i-..."
  "jenkins_master_private_ip" = "10.0.0.1"
  "jenkins_slave_instance_id" = "i-..."
  "jenkins_slave_private_ip" = "10.0.0.2"
  "karpeneter_node_spun_iam_role_arn" = "arn:aws:iam::02...6:role/karpeneter-eks-noderole"
  "registry_id" = "02...6"
  "repository_url" = "02...6.dkr.ecr.us-east-2.amazonaws.com/bankapp"
  "velero_backup_iam_role_arn" = "arn:aws:iam::02...6:role/eks-velero-backup-role"
  "velero_backup_s3_bucket_name" = "dexter-velero-backup"
}

```

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

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

```

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::02XXXXXXX6:role/eks-nodegroup-role-dev
      groups:
        - system:bootstrappers
        - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
    - rolearn: arn:aws:iam::02XXXXXXX6:role/karpenter-eks-noderole
      groups:
        - system:bootstrappers
        - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2025-01-10T10:25:00Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1000000000000000000"
  uid: 00000000000000000000000000000000
~
~
```

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

```

helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.2" --
namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set
"serviceAccount.annotations.eks.amazonaws.com/role-
arn=arn:aws:iam::02XXXXXXX6:role/karpenter-controller-role" --set
controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set
controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi
```

```
[root@XXXXXXXXXX ~]# helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.2" --namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks.amazonaws.com/role-arn=arn:aws:iam::02XXXXXXX6:role/karpenter-controller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi
```

Then ran the kubernetes Manifests file **karpenter.yaml** present at the path **terraform-eks-cluster-waf-with-velero-backup-with-cmk** in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>. For your reference **karpenter.yaml** I had also provided below.

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

karpenter.yaml

```

# This example NodePool will provision general purpose instances

---

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

```

```

kind: EC2NodeClass
name: default

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

```

[root@ REDACTED ~]# kubectl get pods -n karpenter	READY	STATUS	RESTARTS	AGE
NAME				
karpenter- REDACTED	1/1	Running	0	5m16s
karpenter- REDACTED	1/1	Running	0	5m16s

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

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

```
helm repo update
```

```
helm install velero vmware-tanzu/velero --namespace velero --create-namespace --set
configuration.backupStorageLocation[0].provider=aws --set
configuration.backupStorageLocation[0].bucket=dexter-velero-backup --set
configuration.volumeSnapshotLocation[0].config.region=us-east-2 --set
configuration.volumeSnapshotLocation[0].provider=aws --set serviceAccount.create=true --set
serviceAccount.name=velero-server --set
serviceAccount.server.annotations."eks\.amazonaws\.com/role-
arn"="arn:aws:iam::02XXXXXXXXXX6:role/eks-velero-backup-role" --set
initContainers[0].name=velero-plugin-for-aws --set initContainers[0].image="velero/velero-
plugin-for-aws:v1.12.2" --set initContainers[0].volumeMounts[0].mountPath="/target" --set
initContainers[0].volumeMounts[0].name="plugins" --version=10.1.0 --set
credentials.useSecret=false
```

```
[root@ ~]# helm repo add vmware-tanzu https://vmware-tanzu.github.io/helm-charts/
"vmware-tanzu" has been added to your repositories
[root@ ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "vmware-tanzu" chart repository
Update Complete. Happy Helm-ing!
[root@ ~]# helm install velero vmware-tanzu/velero --namespace velero --create-namespace --set configuration.backupStorageLocation[0].provider=aws --set configuration.backupStorageLocation[0].bucket=dexter-velero-backup --set configuration.volumeSnapshotLocation[0].config.region=us-east-2 --set configuration.volumeSnapshotLocation[0].provider=aws --set serviceAccount.create=true --set serviceAccount.name=velero-server --set serviceAccount.server.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::02XXXXXXXXXX6:role/eks-velero-backup-role" --set initContainers[0].name=velero-plugin-for-aws --set initContainers[0].volumeMounts[0].mountPath="/target" --set initContainers[0].volumeMounts[0].name="plugins" --version=10.1.0 --set credentials.useSecret=false
NAME: velero
LAST DEPLOYED: Fri Sep 12 09:36:32 2025
NAMESPACE: velero
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Check that the velero is up and running:

  kubectl get deployment/velero -n velero

Check that the secret has been created:

  kubectl get secret/velero -n velero

Once velero server is up and running you need the client before you can use it
1. wget https://github.com/vmware-tanzu/velero/releases/download/v1.16.2/velero-v1.16.2-darwin-amd64.tar.gz
2. tar -xvf velero-v1.16.2-darwin-amd64.tar.gz -C velero-client

More info on the official site: https://velero.io/docs
```

NAME	READY	STATUS	RESTARTS	AGE
velero-	1/1	Running	0	97s

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

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

```
helm repo update
```

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set
clusterName=eks-demo-cluster-dev --set serviceAccount.create=true --set region=us-east-2 --set
vpcId=vpc-0XXXXXXXXXXXXXX2 --set serviceAccount.name=aws-load-balancer-controller --set
serviceAccount.annotations."eks\.amazonaws\.com/role-
arn"="arn:aws:iam::02XXXXXXXXXX6:role/aws-alb-ingress-controller-role" --set enableWafv2=true
```

```
[root@yellow ~]# helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
[root@yellow ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "vmware-tanzu" chart repository
...Successfully got an update from the "eks" chart repository
Update Complete. Happy Helming!
[root@yellow ~]# helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=eks-demo-cluster-dev --set serviceAccount.create=true --set region=us-east-2 --set vpcId=vpc-XXXXXXXXXX --set serviceAccount.name=aws-load-balancer-controller --set serviceAccount.annotations."eks.amazonaws.com/role-arn"="arn:aws:iam::01XXXXXXXXXX:role/aws-alb-ingress-controller-role" --set enableWafv2=true
NAME: aws-load-balancer-controller
LAST DEPLOYED: Fri Sep 12 09:42:17 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed

[root@yellow ~]# kubectl get pods -n kube-system|grep -i "aws-load-balancer-controller"
aws-load-balancer-controller-XXXXXXXXXX 1/1 Running 0 2m
aws-load-balancer-controller-XXXXXXXXXX 1/1 Running 0 2m
```

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

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

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

Following steps I used to install the ArgoCD.

kubectl create namespace argocd

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

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

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

```
[root@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
w[redacted]@yellow ~]#
```

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

```
[root@192.168.1.10 ~]# cat argocd-ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-ingress
  namespace: argocd
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arm: arn:aws:acm:us-east-2:123456789012:certificate/12345678901234567890123456789012
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80, "HTTPS": 443}]'
spec:
  ingressClassName: alb
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server
            port:
              number: 443

[root@192.168.1.10 ~]# kubectl apply -f argocd-ingress-rule.yaml
ingress.networking.k8s.io/argocd-ingress created
[root@192.168.1.10 ~]# kubectl get ing -A --watch
NAMESPACE   NAME           CLASS      HOSTS           ADDRESS
argocd       argocd-ingress   alb        argocd.singhritesh85.com   k8s-argocd-argocd-in-12345678901234567890123456789012.us-east-2.elb.amazonaws.com   PORTS   AGE
                                                               80      9s
```

Route 53 > Hosted zones > singhritesh85.com > Create record

Quick create record

Record name .singhritesh85.com

Record type A – Routes traffic to an IPv4 address and some AWS resources

Alias

Route traffic to Alias to Application and Classic Load Balancer

US East (Ohio)

Dualstack.k8s-argocd-argocd-in-12345678901234567890123456789012.us-east-2.elb.amazonaws.com

Routing policy Simple routing

Evaluate target health No

Add another record

Create records

Let's get stuff deployed!



Username _____

Password _____

SIGN IN

argo

```

cat argocd-ingress-rule.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-ingress
  namespace: argocd
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:02XXXXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
spec:
  ingressClassName: alb
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server
            port:
              number: 443

```

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

The image displays two screenshots of the Argo CD web interface.

Top Screenshot (Applications Page):

- URL: argocd.singhritesh85.com/applications
- Left sidebar: Shows the Argo logo, version v3.1.5+cfeed49, and navigation links for Applications, Settings, User Info (highlighted with a yellow box), and Documentation.
- Header: Applications, + NEW APP, SYNC APPS, REFRESH APPS, Search applications...
- Content: A large circular icon with three stacked cards, indicating "No applications available to you just yet". Below it is the text "Create new application to start managing resources in your cluster" and a "CREATE APPLICATION" button.
- Right sidebar: APPLICATIONS TILES, Log out.

Bottom Screenshot (User Info Page):

- URL: argocd.singhritesh85.com/user-info?changePassword=false
- Left sidebar: Same as the top screenshot.
- Header: User Info, UPDATE PASSWORD (button highlighted with a yellow box), USER INFO, Log out.
- Content: Displays user information: Username: admin and Issuer: argocd.

The top screenshot shows the 'Update account password' dialog with fields for Current Password, New Password, and Confirm New Password. The 'SAVE NEW PASSWORD' button is highlighted. The bottom screenshot shows the 'User Info' page with a success message: 'Your password has been successfully updated.'

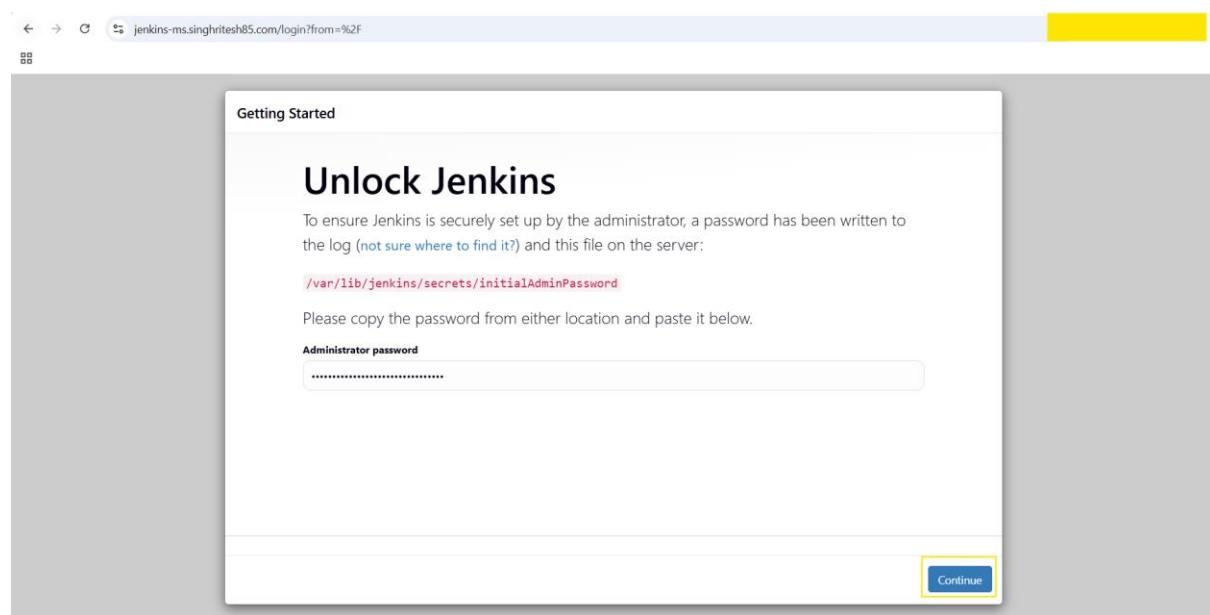
To create the URL for accessing Jenkins I did the entry for Jenkins LoadBalancer DNS Name in Route53 to create the record set of A-Type as shown in the screenshot attached below.

The screenshot shows the 'Quick create record' section of the AWS Route 53 console. It includes fields for Record name (jenkins-ms), Record type (A - Routes traffic to an IPv4 address and some AWS resources), Route traffic to (Alias to Application and Classic Load Balancer, US East (Ohio)), and Routing policy (Simple routing). A success message at the bottom right says 'Create records'.

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



[root@jenkins-master ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
2022singh85



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

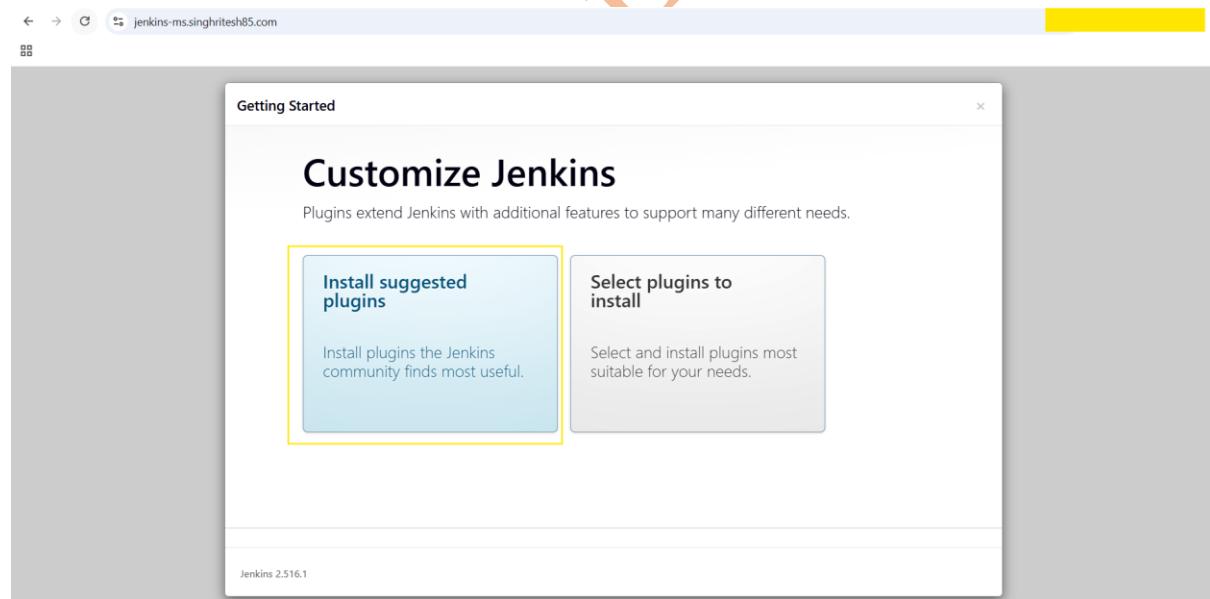
Please copy the password from either location and paste it below.

Administrator password

.....

Continue

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



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

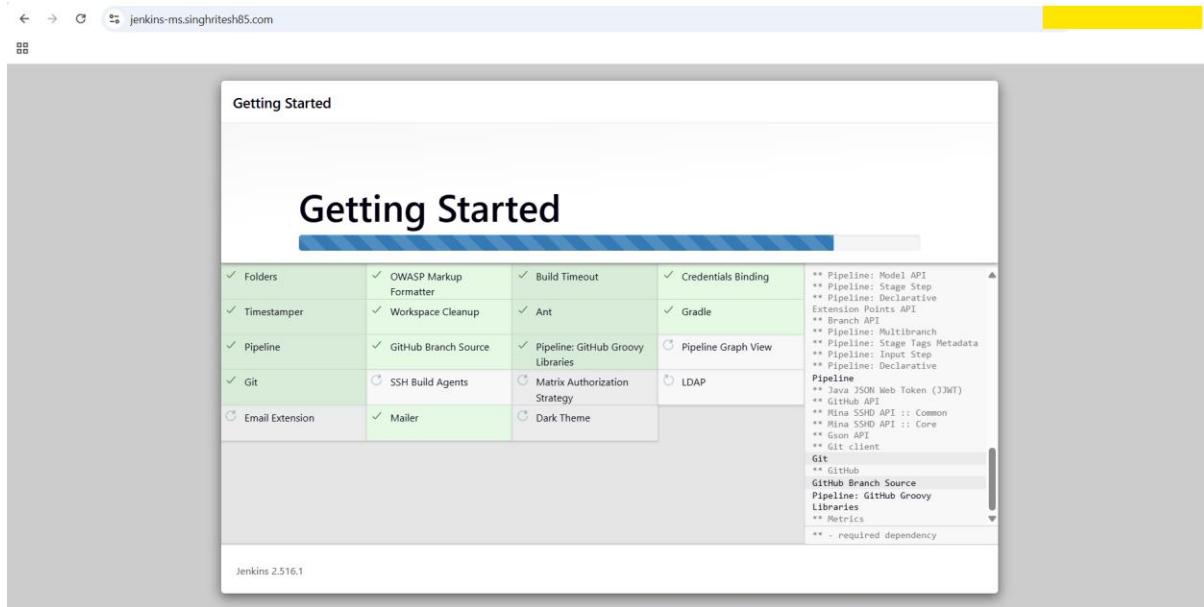
Install suggested plugins

Install plugins the Jenkins community finds most useful.

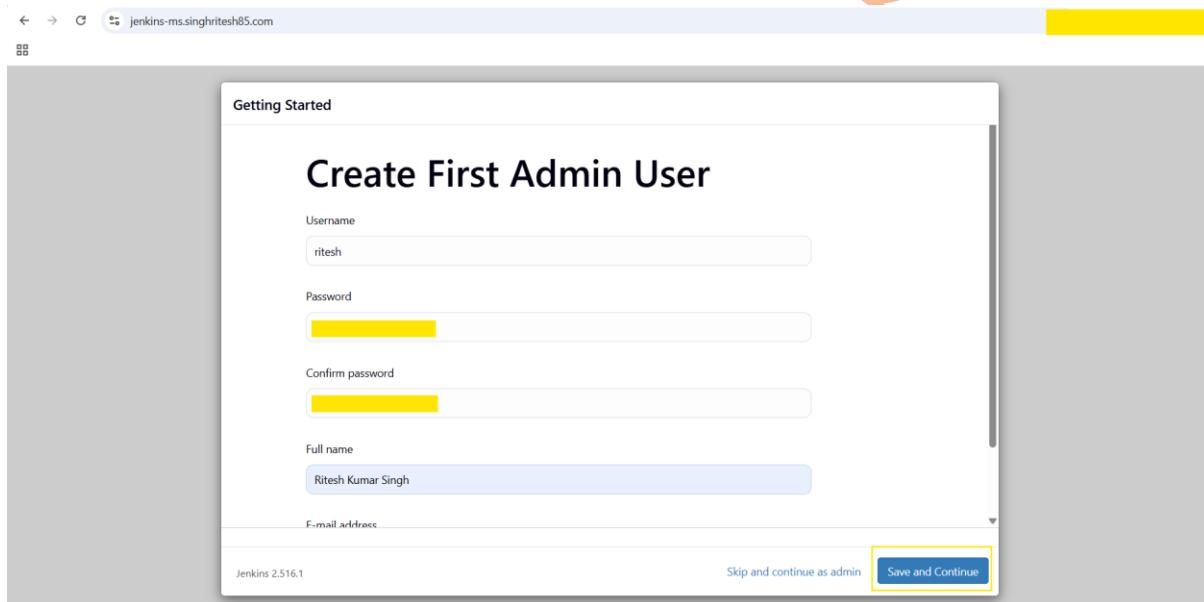
Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.516.1



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



The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.516.1 Not now Save and Finish

Jenkins 2.516.1 Start using Jenkins

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

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

- Set up an agent
- Configure a cloud
- Learn more about distributed builds ?

REST API Jenkins 2.516.2

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

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted)

Username with password

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

Username ?
jenkins

Treat username as secret ?

Password ?
[REDACTED]

ID ?
jenkins-cred

Description ?
jenkins-cred

Create

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted)

Username with password

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

Username ?
[REDACTED]

Treat username as secret ?

Password ?
[REDACTED]

ID ?
github-cred

Description ?
github-cred

Create

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
jenkins-cred	jenkins/***** (jenkins-cred)	Username with password	jenkins-cred
github-cred	singhritesh85/***** (github-cred)	Username with password	github-cred
ARGOCD_PASSWORD	ARGOCD_PASSWORD	Secret text	ARGOCD_PASSWORD

Icon: S M L

REST API Jenkins 2.516.2

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

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

- Set up an agent
- Configure a cloud
- Learn more about distributed builds

REST API Jenkins 2.516.1

Manage Jenkins

System Configuration

- System**: Configure global settings and paths.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Tools**: Configure tools, their locations and automatic installers.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance**: Configure the look and feel of Jenkins.

Security

- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials.
- Credential Providers**: Configure the credential providers and types.
- Users**: Create/delete/modify users that can log in to this Jenkins.

Nodes

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	16.57 GiB	512.00 MiB	▲ 1.86 GiB	0ms
	Data obtained	0.34 sec	0.34 sec	0.34 sec	0.34 sec	0.33 sec	0.33 sec

Icon: S M L

+ New Node Configure Monitors

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 / Manage Jenkins / Nodes

Name ?
Slave-1

Description ?
This is a Slave Node.

Plain text Preview

Number of executors ?
2

Remote root directory ?
/home/jenkins

Labels ?
Slave-1

Save

Jenkins / Manage Jenkins / Nodes

Usage ?
Use this node as much as possible

Launch method ?
Launch agents via SSH

Host ?
10.0.2.15

Credentials ?
***** (jenkins-cred)

+ Add

Host Key Verification Strategy ?
Non verifying Verification Strategy

Advanced ▾

Save

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

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

The screenshot shows the Jenkins 'Nodes' page. It lists two nodes: 'Built-In Node' and 'Slave-1'. Both nodes are of type 'Linux (amd64)' and are marked as 'In sync'. The 'Free Disk Space' and 'Free Swap Space' columns show values like '16.53 GiB' and '512.00 MiB'. The 'Free Temp Space' column shows values like '1.86 GiB' and '1.87 GiB', both with a warning icon. The 'Response Time' column shows '0ms' and '22ms' respectively. A legend at the bottom indicates that the warning icon represents '1.86 GiB' and '1.87 GiB'. The page also includes links for 'REST API' and 'Jenkins 2.516.2'.

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

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

Create the storage class named as **dexter**, this storage class will provision the EBS volume (and encrypt the EBS volume using AWS Customer Managed Key) dynamically using the EBS CSI driver deployed as an Add Ons in the EKS Cluster.

```
[root@yellow ~]# cat storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: dexter
provisioner: ebs.csi.aws.com    #ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: arn:aws:kms:us-east-2:0[REDACTED]:key/[REDACTED]
[root@yellow ~]# kubectl apply -f storage-class.yaml
storageclass.storage.k8s.io/dexter created
[root@yellow ~]# kubectl get sc
NAME      PROVISIONER          RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
dexter    ebs.csi.aws.com     Delete          WaitForFirstConsumer  false                  2023-07-11T11:45:45Z
gp2       kubernetes.io/aws-ebs Delete          WaitForFirstConsumer  false                  2023-07-11T11:45:45Z
```

The storage class named as **dexter** will be used during the deployment of MySQL8 Pods.

```
cat storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: dexter
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
kmsKeyId: arn:aws:kms:us-east-2:02XXXXXXXXXX6:key/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

Installation of Istio in Amazon EKS Cluster

To install Istio Service mesh using helm chart use the command as written below.

```
helm repo add istio https://istio-release.storage.googleapis.com/charts
helm repo update
kubectl create ns istio-system
helm install istio-base istio/base -n istio-system
helm install istiod istio/istiod -n istio-system
helm install istio-ingressgateway istio/gateway --set service.type=ClusterIP -n istio-system
helm install istio-egressgateway istio/gateway --set service.type=ClusterIP -n istio-system
cat server.crt intermediate1.crt intermediate2.crt root.crt > tls.crt
kubectl create secret generic cacerts -n istio-system --from-file=server.crt --from-file=tls.key --from-file=root.crt --from-file=tls.crt
[root@yellow ~]# helm repo add istio https://istio-release.storage.googleapis.com/charts
"istio" has been added to your repositories

[root@yellow ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "vmware-tanzu" chart repository
...Successfully got an update from the "eks" chart repository
...Successfully got an update from the "istio" chart repository
Update Complete. Happy Helming!
```

```
[root@yellow ~]# kubectl create ns istio-system
namespace/istio-system created
[root@yellow ~]# helm install istio-base istio/base -n istio-system
```

```
[root@yellow ~]# helm install istiod istio/istiod -n istio-system
[root@yellow ~]# helm install istio-ingressgateway istio/gateway --set service.type=ClusterIP -n istio-system
[root@yellow ~]# helm install istio-egressgateway istio/gateway --set service.type=ClusterIP -n istio-system
[root@yellow ~]# cat server.crt intermediate1.crt intermediate2.crt root.crt > tls.crt
[root@yellow ~]# kubectl create secret generic cacerts -n istio-system --from-file=server.crt --from-file=tls.key --from-file=root.crt --from-file=tls.crt
secret/cacerts created
```

I had installed Kiali, Prometheus and Grafana as shown in the screenshot attached below.

kubectl apply -f <https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/kiali.yaml>
 kubectl apply -f <https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/prometheus.yaml>
 kubectl apply -f <https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/grafana.yaml>

```
[root@yellow ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/kiali.yaml
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
service/kiali created
deployment.apps/kiali created
[root@yellow ~]# kubectl get pods -n istio-system --watch
NAME           READY   STATUS    RESTARTS   AGE
istio-egressgateway-[REDACTED]   1/1     Running   0          21m
istio-ingressgateway-[REDACTED]   1/1     Running   0          26m
istiod-[REDACTED]                 1/1     Running   0          27m
kiali-[REDACTED]                  1/1     Running   0          48s

[root@yellow ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/prometheus.yaml
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
service/prometheus created
deployment.apps/prometheus created
[root@yellow ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/grafana.yaml
serviceaccount/grafana created
configmap/grafana created
service/grafana created
deployment.apps/grafana created
configmap/istio-grafana-dashboards created
configmap/istio-services-grafana-dashboards created
```

NAME	READY	STATUS	RESTARTS	AGE
grafana-[REDACTED]	1/1	Running	0	s
istio-egressgateway-[REDACTED]	1/1	Running	0	m
istio-ingressgateway-[REDACTED]	1/1	Running	0	m
istiod-[REDACTED]	1/1	Running	0	m
kiali-[REDACTED]	1/1	Running	0	m
prometheus-[REDACTED]	2/2	Running	0	s

```
[root@yellow ~]# kubectl get virtualservice -n istio-system
NAME           GATEWAYS   HOSTS   AGE
telemetry-kiali-virtual-service  ["istio-telemetry-gateway"]  ["kiali.singhrites85.com"]  30m
```

To enable peer-to-peer authentication (mTLS) in Istio Service mesh I used the STRICT mode and the manifests file as shown below. If you use mode as PERMISSIVE mode then it accepts both http (plain text) and https (Mutual TLS) based communication. I created a kubernetes secrets named as cacerts as shown in the screenshot attached above. I will use this kubernetes secret for the creation of gateway.

I created the record set in Route53 for kiali corresponding to DNS Name of the LoadBalancer for the kiali service as shown in the screenshot attached below.

```
[root@XXXXXXXXXX ~]# cat kiali-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kiali-ingress
  namespace: istio-system # Or the namespace where Kiali is deployed
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip # Or 'instance'
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]' # If using HTTPS
    ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:XXXXXXXXXX:certificate/XXXXXXXXXXXXXX
spec:
  rules:
  - host: kiali.singhritesh85.com # Replace with your desired hostname
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kiali # Name of the Kiali service created by the operator
            port:
              number: 20001 # Kiali's default port
[root@XXXXXXXXXX ~]# kubectl get ing -n istio-system
NAME           CLASS      HOSTS             ADDRESS
kiali-ingress   <none>    kiali.singhritesh85.com   k8s-istiosys-kialiing-XXXXXXXXXX.us-east-2.elb.amazonaws.com   PORTS   AGE
                                                               80      48m
```

cat kiali-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kiali-ingress
  namespace: istio-system
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]'
    ingress.kubernetes.io/backend-protocol: HTTPS
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:XXXXXXXXXX:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXXXX
spec:
  ingressClassName: alb
  rules:
  - host: kiali.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kiali
            port:
              number: 20001
```

Records (7)		DNSSEC signing		Hosted zone tags (1)	
Records (7) <small>Info</small>					
		<small>Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.</small>			
<input type="checkbox"/>	Record name	Type	Routing p...	Alias	Value/Route traffic
	▼	▼	▲	▼	▼
<input type="checkbox"/>	singhritesh85.com	NS	Simple	-	No
<input type="checkbox"/>	singhritesh85.com	SOA	Simple	-	No
<input type="checkbox"/>	[REDACTED]	CNAME	Simple	-	No
<input type="checkbox"/>	[REDACTED]	CNAME	Simple	-	No
<input type="checkbox"/>	argocd.singhritesh85.com	A	Simple	-	Yes
<input type="checkbox"/>	jenkins-ms.singhritesh85.com	A	Simple	-	Yes
<input type="checkbox"/>	kiali.singhritesh85.com	A	Simple	-	Yes

```
[root@[REDACTED] ~]# cat peerauthentication.yaml
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata:
  name: "default"
  namespace: istio-system
spec:
  mtls:
    mode: STRICT
```

cat peerauthentication.yaml

```
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata:
  name: "default"
  namespace: istio-system
spec:
  mtls:
    mode: STRICT
```

```
[root@[REDACTED] ~]# kubectl get virtualservice -n istio-system
NAME                                GATEWAYS          HOSTS          AGE
telemetry-kiali-virtual-service     ["istio-telemetry-gateway"]  ["kiali.singhritesh85.com"]  45m
[root@[REDACTED] ~]# kubectl get destinationrule -n istio-system
NAME      HOST          AGE
kiali    kiali.istio-system.svc.cluster.local  45m
```

```
cat gtw-vsc-destrule-kiali.yaml
---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-telemetry-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
    - port:
        number: 80
        name: http
        protocol: HTTP
      hosts:
        - "kiali.singhrithesh85.com"
      tls:
        httpsRedirect: true
    - port:
        number: 443
        name: https
        protocol: HTTPS
      hosts:
        - "kiali.singhrithesh85.com"
      tls:
        mode: SIMPLE
        credentialName: cacerts
  ---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
```

```
name: telemetry-kiali-virtual-service
namespace: istio-system
spec:
  hosts:
    - "kiali.singhritesh85.com"
  gateways:
    - istio-telemetry-gateway
  http:
    - route:
        - destination:
            host: kiali.istio-system.svc.cluster.local
        port:
          number: 20001
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: kiali
  namespace: istio-system
spec:
  host: kiali.istio-system.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE
```

Kiali dashboard is as shown in the screenshot attached below which confirms mesh-wide mTLS had been enabled.

The top screenshot shows the Kiali Overview page. It lists four namespaces: default, istio-system, karptenter, and velero. Each namespace has a summary card. The default namespace has 1 label (Istio config) and 0 applications. The istio-system namespace has 1 label (Istio config) and 6 applications. The karptenter namespace has 2 labels (Istio config) and 1 application. The velero namespace has 2 labels (Istio config) and 1 application.

The bottom screenshot shows the Kiali Mesh page. It displays a network diagram with nodes: kiali (blue hexagon), Prometheus (red hexagon), Grafana (orange hexagon), and istiod (green hexagon). Below the diagram, there are buttons for 'Display', 'Find...', 'Hide...', 'Help', 'Reset', 'Last 1m', 'Every 1m', and a refresh icon. On the right side, there is a detailed view for the istiod pod. It shows the following information:

- istiod**: Version 1.27.1, Mesh-wide mTLS is enabled, configured by default.
- Outbound policy**: ALLOW_ANY.
- Proxy sync time**: N/A.
- Min TLS version**: TLSV1_2.

Metrics for Memory and CPU are shown as line graphs. The Configuration section includes download and commit buttons.

Installation of Cilium in Chain Mode with VPC CNI in Amazon EKS Cluster

To enable Node-to-Node encryption with wireguard install cilium in chain mode with VPC-CNI follow the steps as written below.

helm repo add cilium <https://helm.cilium.io/>

helm repo update

```
helm upgrade --install cilium cilium/cilium --version 1.18.1 --namespace kube-system --set
cni.chainingMode=aws-cni --set kubeProxyReplacement=false --set cni.exclusive=false --set
enableIPv4Masquerade=false --set endpointRoutes.enabled=true --set encryption.enabled=true --
set encryption.nodeEncryption=true --set encryption.type=wireguard --set l7Proxy=false
```

kubectl get pods -n kube-system --watch| grep -i "cilium"

kubectl -n kube-system exec -it ds/cilium -- cilium status | grep Encryption

```
[root@[REDACTED] ~]# helm repo add cilium https://helm.cilium.io/
"cilium" 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 "vmware-tanzu" chart repository
...Successfully got an update from the "eks" chart repository
...Successfully got an update from the "cilium" chart repository
...Successfully got an update from the "istio" chart repository
Update Complete. ☺Happy Helming!☺
```

```
[root@[REDACTED] ~]# helm upgrade --install cilium cilium/cilium --version 1.18.1 --namespace kube-system --set cni.chainingMode=aws-cni --set kubeProxyReplication=false --set cni.exclusive=false --set enableIPv4Masquerade=false --set endpointRoutes.enabled=true --set encryption.enabled=true --set encryption.nodeEncryption=true --set encryption.type=wireguard --set l7Proxy=false
```

[root@[REDACTED] ~]# kubectl get pods -n kube-system --watch grep -i "cilium"				
cilium-[REDACTED]	1/1	Running	0	113s
cilium-[REDACTED]	1/1	Running	0	113s
cilium-[REDACTED]	1/1	Running	0	113s
cilium-operator-[REDACTED]	1/1	Running	0	113s
cilium-operator-[REDACTED]	1/1	Running	0	113s
cilium-[REDACTED]	1/1	Running	0	87s
cilium-[REDACTED]	1/1	Running	0	63s

To get confirmation regarding Node-to-Node encryption run the below command.

kubectl -n kube-system exec -it ds/cilium -- cilium status | grep Encryption

```
[root@jenkins-slave ~]# kubectl -n kube-system exec -it ds/cilium -- cilium status | grep Encryption
Encryption:           Wireguard  [NodeEncryption: Enabled, cilium_wg0 (Pubkey: [REDACTED], Port: 51871, Peers: 4)]
```

As shown in the screenshot attached above Node-to-Node Encryption in EKS had been Enabled.

Installation of Open Policy Agent (OPA) Gatekeeper in Amazon EKS Cluster

To install OPA using Helm follow the steps as written below.

helm repo add gatekeeper https://open-policy-agent.github.io/gatekeeper/charts

helm repo update

helm install gatekeeper gatekeeper/gatekeeper --namespace gatekeeper-system --create-namespace

kubectl get pods -n gatekeeper-system --watch

```
[root@[REDACTED] ~]# helm repo add gatekeeper https://open-policy-agent.github.io/gatekeeper/charts
"gatekeeper" 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 "gatekeeper" chart repository
...Successfully got an update from the "vmware-tanzu" chart repository
...Successfully got an update from the "eks" chart repository
...Successfully got an update from the "cilium" chart repository
...Successfully got an update from the "istio" chart repository
Update Complete. ☺Happy Helming!☺
```

```
[root@[REDACTED] ~]# helm install gatekeeper gatekeeper/gatekeeper --namespace gatekeeper-system --create-namespace
```

NAME	READY	STATUS	RESTARTS	AGE
gatekeeper-audit-[REDACTED]	1/1	Running	0	90s
gatekeeper-controller-manager-[REDACTED]	1/1	Running	0	90s
gatekeeper-controller-manager-[REDACTED]	1/1	Running	0	90s
gatekeeper-controller-manager-[REDACTED]	1/1	Running	0	90s

You can check the logs for OPA using the command as shown in the screenshot attached below.

kubectl logs -l control-plane=audit-controller -n gatekeeper-system

```
[root@[REDACTED] ~]# kubectl logs -l control-plane=audit-controller -n gatekeeper-system
{"level":"info","ts":1758008534.146585,"msg":"Starting workers","controller":"syncset-controller","worker count":1}
{"level":"info","ts":1758008593.6449754,"logger":"controller","msg":"auditing constraints and violations","process":"audit","audit_id":"2025-09-16T07:43:13Z","event_type":"audit_started"}
{"level":"info","ts":1758008593.674333,"logger":"controller","msg":"no constraint is found with apiversion","process":"audit","audit_id":"2025-09-16T07:43:13Z","constraint_apiversion":"constraints.gatekeeper.sh/vbeta1"}
{"level":"info","ts":1758008593.6743734,"logger":"controller","msg":"auditing is complete","process":"audit","audit_id":"2025-09-16T07:43:13Z","event_type":"audit_finished","duration":"29.429417ms"}
{"level":"info","ts":1758008653.6453779,"logger":"controller","msg":"auditing constraints and violations","process":"audit","audit_id":"2025-09-16T07:44:13Z","event_type":"audit_started"}
{"level":"info","ts":1758008653.6742582,"logger":"controller","msg":"no constraint is found with apiversion","process":"audit","audit_id":"2025-09-16T07:44:13Z","constraint_apiversion":"constraints.gatekeeper.sh/vbeta1"}
{"level":"info","ts":1758008653.6745365,"logger":"controller","msg":"auditing is complete","process":"audit","audit_id":"2025-09-16T07:44:13Z","event_type":"audit_finished","duration":"29.16974ms"}
 {"level":"info","ts":1758008713.6448324,"logger":"controller","msg":"auditing constraints and violations","process":"audit","audit_id":"2025-09-16T07:45:13Z","event_type":"audit_started"}
 {"level":"info","ts":1758008713.6736693,"logger":"controller","msg":"no constraint is found with apiversion","process":"audit","audit_id":"2025-09-16T07:45:13Z","constraint_apiversion":"constraints.gatekeeper.sh/vbeta1"}
 {"level":"info","ts":1758008713.6737137,"logger":"controller","msg":"auditing is complete","process":"audit","audit_id":"2025-09-16T07:45:13Z","event_type":"audit_finished","duration":"28.910995ms"}
```

Here, I created OPA Policy (Constraint Template and Constraint) which will disallow the creation of pod with container had the capability of host machine (Privileged = true) and elevated privileges (allowPrivilegeEscalation = true) in the namespace bankapp and mysql, and no one can create the LoadBalancer Service in the namespace bankapp and mysql. If any one wants to create the service, then create the ClusterIP service and expose the pods using Ingress Rule through the created ClusterIP service. It will also not allow anyone to use the Image with latest tag and do not allow to create the Resources Pod, Deployment, Statefulset or Daemonset in the namespace mysql and bankapp if require resource requests and limits per Pod is not mentioned.

```
[root@[REDACTED] ~]# kubectl apply -f constrainttemplate.yaml
constrainttemplate.templates.gatekeeper.sh/k8sdisallowprivileged created
constrainttemplate.templates.gatekeeper.sh/loadbalancerconstraint created
constrainttemplate.templates.gatekeeper.sh/k8scontainerresourcerequirements created
constrainttemplate.templates.gatekeeper.sh/k8sdisallowedtags created
[root@[REDACTED] ~]# kubectl apply -f constraint.yaml
k8sdisallowprivileged.constraints.gatekeeper.sh/psp-privileged-container created
loadbalancerconstraint.constraints.gatekeeper.sh/loadbalancerconstraint created
k8scontainerresourcerequirements.constraints.gatekeeper.sh/pod-resource-requirements created
k8sdisallowedtags.constraints.gatekeeper.sh/container-image-must-not-have-latest-tag created
```

You can list the constraint template and constraint currently deployed and used in the cluster using the command **kubectl get constrainttemplate** and **kubectl get constraints** respectively.

```
cat constrainttemplate.yaml
```

```
---
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8sdisallowprivileged
spec:
  crd:
    spec:
      names:
        kind: K8sDisallowPrivileged
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8sdisallowprivileged

        violation[{"msg": msg}] {
          container := input.review.object.spec.containers[_]
          container.securityContext.privileged == true
          msg := sprintf("Privileged container %v is not allowed", [container.name])
        }

        violation[{"msg": msg}] {
          initContainer := input.review.object.spec.initContainers[_]
          initContainer.securityContext.privileged == true
          msg := sprintf("Privileged init container %v is not allowed", [initContainer.name])
        }

        violation[{"msg": msg}] {
          container := input.review.object.spec.containers[_]
          container.securityContext.allowPrivilegeEscalation == true
        }
```

```

msg := sprintf("allowPrivilegeEscalation container %v is not allowed", [container.name])
}

violation[{"msg": msg}] {
    initContainer := input.review.object.spec.initContainers[_]
    initContainer.securityContext.allowPrivilegeEscalation == true
    msg := sprintf("allowPrivilegeEscalation init container %v is not allowed", [initContainer.name])
}
---

apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: loadbalancerconstraint
spec:
  crd:
    spec:
      names:
        kind: LoadBalancerConstraint
targets:
  - target: admission.k8s.gatekeeper.sh
rego: |
  package insomniacoder.constraint
  violation[{"msg": msg}] {
    input.review.kind.kind = "Service"
    input.review.operation = "CREATE"
    input.review.object.spec.type = "LoadBalancer"
    msg := "Service type LoadBalancer are restricted"
  }
---

apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8scontainerresourcerequirements

```

```

spec:
crd:
  spec:
    names:
      kind: K8sContainerResourceRequirements
    validation:
      openAPIV3Schema:
        type: object
        properties:
          spec:
            type: object
targets:
  - target: admission.k8s.gatekeeper.sh
    rego: |
      package k8scontainerresourcerequirements

      violation[{"msg": msg}] {
        container := input.review.object.spec.containers[_]
        not container.resources.requests.cpu
        msg := sprintf("Container '%v' must have CPU requests defined", [container.name])
      }

      violation[{"msg": msg}] {
        container := input.review.object.spec.containers[_]
        not container.resources.requests.memory
        msg := sprintf("Container '%v' must have memory requests defined", [container.name])
      }

      violation[{"msg": msg}] {
        container := input.review.object.spec.containers[_]
        not container.resources.limits.cpu
        msg := sprintf("Container '%v' must have CPU limits defined", [container.name])
      }

```

```

}

violation[{"msg": msg}] {
    container := input.review.object.spec.containers[_]
    not container.resources.limits.memory
    msg := sprintf("Container '%v' must have memory limits defined", [container.name])
}
---  

apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8sdisallowedtags
spec:
  crd:
    spec:
      names:
        kind: K8sDisallowedTags
      validation:
        openAPIV3Schema:
          type: object
          properties:
            tags:
              type: array
              description: Disallowed container image tags.
            items:
              type: string
      targets:
        - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8sdisallowedtags
        violation[{"msg": msg}] {
            container := input_containers[_]

```

```
tags := [tag_with_prefix | tag := input.parameters.tags[_]; tag_with_prefix := concat(":", ["",
tag])]

strings.any_suffix_match(container.image, tags)

msg := sprintf("container <%v> uses a disallowed tag <%v>; disallowed tags are %v",
[container.name, container.image, input.parameters.tags])

}

violation[{"msg": msg}] {

    container := input_containers[_]

    not contains(container.image, ":")

    msg := sprintf("container <%v> didn't specify an image tag <%v>", [container.name,
container.image])

}

input_containers[c] {

    c := input.review.object.spec.containers[_]

}

input_containers[c] {

    c := input.review.object.spec.initContainers[_]

}
```

Ritesh Kumar Singh

```
cat constraint.yaml

apiVersion: constraints.gatekeeper.sh/v1beta1

kind: K8sDisallowPrivileged

metadata:

  name: psp-privileged-container

spec:

  match:

    kinds:

      - apiGroups: [""]

        kinds: ["Pod"]

    namespaces:

      - "bankapp"

      - "mysql"

---

apiVersion: constraints.gatekeeper.sh/v1beta1

kind: LoadBalancerConstraint

metadata:

  name: loadbalancerconstraint

spec:

  match:

    kinds:

      - apiGroups: [""]

        kinds: ["Service"]

    namespaces:

      - "bankapp"

      - "mysql"

---

apiVersion: constraints.gatekeeper.sh/v1beta1

kind: K8sContainerResourceRequirements

metadata:

  name: pod-resource-requirements
```

```
spec:  
  match:  
    kinds:  
      - apiGroups: [""]  
        kinds: ["Pod"]  
      - apiGroups: ["apps"]  
        kinds: ["Deployment", "StatefulSet", "DaemonSet"]  
  namespaces:  
    - "bankapp"  
    - "mysql"  
---  
apiVersion: constraints.gatekeeper.sh/v1beta1  
kind: K8sDisallowedTags  
metadata:  
  name: container-image-must-not-have-latest-tag  
spec:  
  match:  
    kinds:  
      - apiGroups: [""]  
        kinds: ["Pod"]  
      - apiGroups: ["apps"]  
        kinds: ["Deployment", "StatefulSet", "DaemonSet"]  
  namespaces:  
    - "bankapp"  
    - "mysql"  
parameters:  
  tags: ["latest"]
```

Installation of Falco in Amazon EKS Cluster

To install Falco in Amazon EKS Cluster, follow the steps as written below.

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
```

```
helm repo update
```

```
helm install falco falcosecurity/falco --namespace falco --create-namespace --set
falcoseidekick.enabled=true --set falcoseidekick.webui.enabled=true --set
falcoseidekick.webui.redis.storageEnabled=false
```

```
kubectl get pods -n falco --watch
```

```
[root@XXXXXXXXXX ~]# helm install falco falcosecurity/falco --namespace falco --create-namespace
[XXXXXXXXXX ~]# helm repo update
[XXXXXXXXXX ~]# helm install falco falcosecurity/falco --namespace falco --create-namespace --set falcoseidekick.enabled=true --set falcoseidekick.webui.enabled=false
Error: INSTALLATION FAILED: expected at most two arguments, unexpected arguments: --set, falcoseidekick.webui.redis.storageEnabled=false
[XXXXXXXXXX ~]# helm install falco falcosecurity/falco --namespace falco --create-namespace --set falcoseidekick.enabled=true --set falcoseidekick.webui.enabled=false
NAME: falco
LAST DEPLOYED: XXXXXXXXXX 2025
NAMESPACE: falco
STATUS: deployed
REVISION: 1
NOTES:
Falco agents are spinning up on each node in your cluster. After a few
seconds, they are going to start monitoring your containers looking for
security issues.

No further action should be required.

NOTICE:
It seems you are loading the following plugins [container], please make sure to install them by specifying the correct reference to falcoctl.config.artifact.i
nstall.refs: [falco-rules:4 ghcr.io/falcosecurity/plugins/plugin/container:0.3.6]Ignore this notice if the value of falcoctl.config.artifact.install.refs is c
orrect already.
```

I had applied a custom falco rule, if anyone will try to access the /etc directory to write in a file then it will be notified through Falco as shown in the screenshot attached below.

```
[root@XXXXXXXXXX ~]# cat dexter-falco-rule.yaml
customRules:
  custom-rules.yaml: |-
    - rule: Write below etc
      desc: An attempt to write to /etc directory
      condition: >
        (evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)
        and fd.name.startswith '/etc'
      output: "File below /etc opened for writing | file=%fd.name pcmdline=%proc.pcmdline gparent=%proc.parent pname=%proc.name[2] gparent=%proc.parent pname=%proc.name[3] ggpname=%proc.parent pname=%proc.parent ggpname=%proc.parent pname=%proc.parent"
      [4] evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.parent pname=%proc.parent cmdline=%proc.cmdline terminal=%proc.terminal"
      priority: WARNING
      tags: [filesystem, mitre_persistence]
```

```
helm upgrade --install falco falcosecurity/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcoseidekick.enabled=true --set falcoseidekick.webui.enabled=true --set falcoseidekick.webui.redis.storageEnabled=false
```

```
[root@XXXXXXXXXX ~]# helm upgrade --install falco falcosecurity/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcoseidekick.enabled=true --set falcoseidekick.webui.enabled=true --set falcoseidekick.webui.redis.storageEnabled=false
```

```
cat dexter-falco-rule.yaml
```

```
customRules:
```

```
  custom-rules.yaml: |-
```

- rule: Write below etc

desc: An attempt to write to /etc directory

condition: >

(evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)

and fd.name startswith /etc

output: "File below /etc opened for writing | file=%fd.name pc cmdline=%proc.pc cmdline gparent=%proc.ename[2] ggparent=%proc.ename[3] gggparent=%proc.ename[4] evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname command=%proc.cmdline terminal=%proc.tty"

priority: WARNING

tags: [filesystem, mitre_persistence]

Created Ingress Rule to access Falco ui as shown in the screenshot attached below.

```
[root@REDACTED ~]# cat falco-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:REDACTED:certificate/REDACTED
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
spec:
  ingressClassName: alb
  rules:
    - host: falco.singhrites85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: falco-falcosidekick-ui
                port:
                  number: 2802
```



```
[root@REDACTED ~]# kubectl apply -f falco-ingress-rule.yaml
ingress.networking.k8s.io/falco-ui-ingress created
[root@REDACTED ~]# kubectl get ing -A
NAMESPACE     NAME           CLASS      HOSTS          ADDRESS
argocd        argocd-ingress alb        argocd.singhrites85.com   k8s-argocd-argocd-in-REDACTED.us-east-2.elb.amazonaws.com  80   2d
falco         falco-ui-ingress alb        falco.singhrites85.com   k8s-falco-falcoui-REDACTED.us-east-2.elb.amazonaws.com  80   2d
istio-system  kiali-ingress  alb        kiali.singhrites85.com   k8s-istiosys-kialiing-REDACTED.us-east-2.elb.amazonaws.com  80   2d
```

```
cat falco-ingress-rule.yaml
```

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:02XXXXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
spec:
  ingressClassName: alb
  rules:
    - host: falco.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: falco-falcosidekick-ui
              port:
                number: 2802
```

I had created the entry for falco HOST resulted from ingress rule as shown in the screenshot attached above in Route53 to create the Record Set of A-Type as shown in the screenshot attached below.

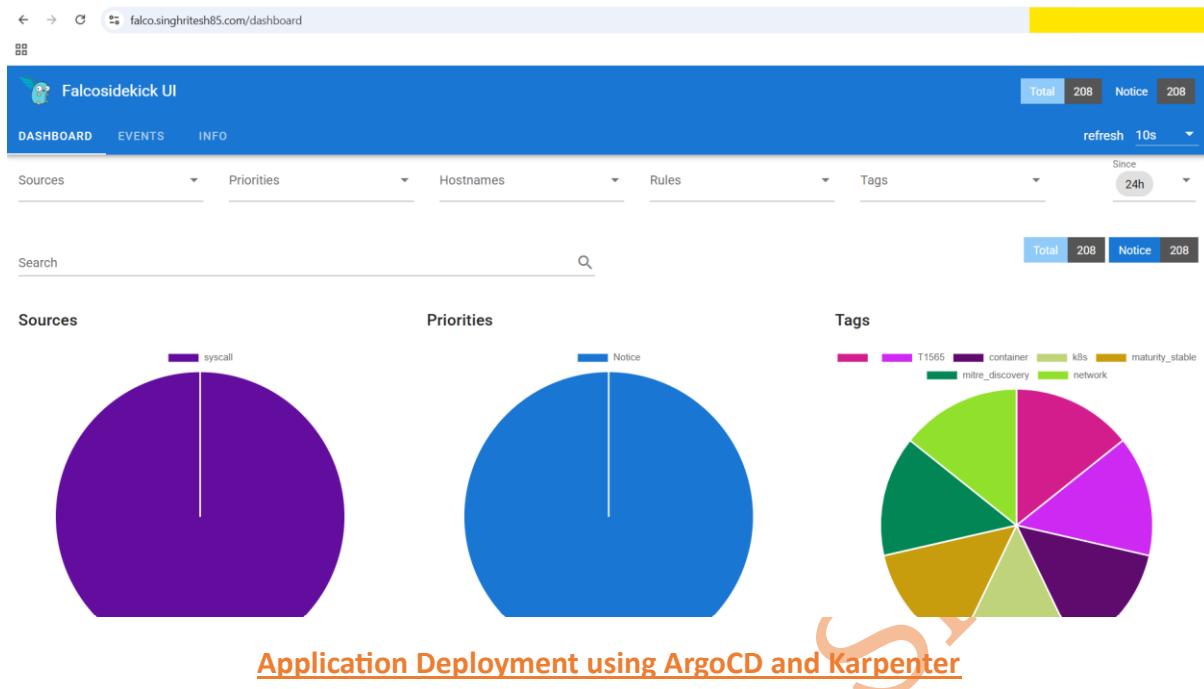
The screenshot shows the 'Create record' page in the AWS Route 53 console. A new A record is being created for the domain 'singhritesh85.com'. The 'Record name' field contains 'falco'. The 'Record type' is set to 'A – Routes traffic to an IPv4 address and some AWS resources'. The 'Route traffic to' section is set to 'Alias' and points to an Application Load Balancer named 'dualstack.k8s-falco-falcouii-...'. The 'Routing policy' is set to 'Simple routing'. The 'Evaluate target health' option is turned off ('No'). At the bottom right, there is a large orange button labeled 'Create records' with a hand cursor icon pointing at it.

The screenshot shows the Falcosidekick UI login page. It features a blue header bar with the text 'Falcosidekick UI'. Below the header is a navigation bar with links for 'DASHBOARD', 'EVENTS', and 'INFO'. On the right side of the header, there is a 'refresh 10s' button. The main area contains a login form with fields for 'Login' and 'Password', and a 'LOGIN' button.

2023 - Falco Authors

Use the username/password as admin/admin and logged-in into the Falco ui as shown in the screenshot attached below. It is also possible to check falco logs using the command

`kubectl logs -l "app.kubernetes.io/instance=falco" -n falco`



Application Deployment using ArgoCD and Karpenter

To deploy Application using ArgoCD and Karpenter through Jenkins I followed the steps as written below.

First deploy MySQL 8 Pods, for that I created the Jenkins Job as shown in the screenshot attached below.

```
[root@yellow ~]# kubectl get pvc -n mysql
NAME                      STATUS   VOLUME
mysql-persistent-storage-mysql-0  Bound   pvc-
mysql-persistent-storage-mysql-1  Bound   pvc-
mysql-persistent-storage-mysql-2  Bound   pvc-
[root@yellow ~]# kubectl get pods -n mysql
NAME     READY   STATUS    RESTARTS   AGE
mysql-0  1/1     Running   0          9m24s
mysql-1  1/1     Running   0          2m46s
mysql-2  1/1     Running   0          2m19s
```

The Jenkinsfile to deploy MySQL 8 Pods is provided for your reference blow.

Jenkinsfile-MySQL 8

```

pipeline{
    agent{
        node{
            label "Slave-1"
            customWorkspace "/home/jenkins/mysql"
        }
    }
    environment{
        JAVA_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
        PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
        ARGOCD_PASSWORD=credentials('ARGOCD_PASSWORD')
    }
    stages{
        stage("MySQL-Deployment"){
            steps{
                //MySQL
                sh 'argocd login argocd.singhritesh85.com --username admin --password $ARGOCD_PASSWORD --skip-test-tls --grpc-web'
                sh 'argocd app create mysql --project default --repo https://github.com/singhritesh85/kustomize-bankapp.git --path ./overlay/mysql --dest-namespace mysql --sync-option CreateNamespace=true --dest-server https://kubernetes.default.svc --upsert'
                sh 'argocd app sync mysql'
            }
        }
    }
}

```

Then I created the Jenkins Job to deploy bankapp pods as shown in the screenshot attached below.

```
[root@REDACTED ~]# kubectl get pods -n bankapp --watch
NAME           READY   STATUS    RESTARTS   AGE
bankapp-6fcb47f9dd-km4k2   1/1     Running   0          2m47s
```

Jenkinsfile-Bankapp

```

pipeline{
    agent{
        node{
            label "Slave-1"
            customWorkspace "/home/jenkins/bankapp"
        }
    }
    environment{
        JAVA_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
        PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
        ARGOCD_PASSWORD=credentials('ARGOCD_PASSWORD')
    }
    stages{
        stage("Clone-Code"){
            steps{
                cleanWs()
                checkout scmGit(branches: [[name: 'main']], extensions: [], userRemoteConfigs: [[credentialsId: 'github-cred', url: 'https://github.com/singhrithesh85/Bank-App-Kustomize.git']])
            }
        }
        stage("Build"){
            steps{
                sh 'mvn clean install'
            }
        }
        stage("Docker Build"){
            steps{
                sh 'docker system prune -f --all'
                sh 'docker build -t 027330342406.dkr.ecr.us-east-2.amazonaws.com/bankapp:1.0.1 -f Dockerfile-Project-1 .'
                sh 'trivy image --exit-code 0 --severity MEDIUM,HIGH 027330342406.dkr.ecr.us-east-2.amazonaws.com/bankapp:1.0.1'
            }
        }
    }
}

```

```

sh 'trivy image --exit-code 1 --severity CRITICAL 027330342406.dkr.ecr.us-east-
2.amazonaws.com/bankapp:1.0.1'

sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --
password-stdin 027330342406.dkr.ecr.us-east-2.amazonaws.com'

sh 'docker push 027330342406.dkr.ecr.us-east-2.amazonaws.com/bankapp:1.0.1'

}

}

stage("Deployment"){

steps{

sh 'argocd login argocd.singhritesh85.com --username admin --password
$ARGOCD_PASSWORD --skip-test-tls --grpc-web'

sh 'argocd app create bankapp --project default --repo
https://github.com/singhritesh85/kustomize-bankapp.git --path ./overlay/bankapp --revision main --dest-namespace bankapp --sync-option CreateNamespace=true --dest-server
https://kubernetes.default.svc --upsert'

sh 'argocd app sync bankapp'

}

}

}

}

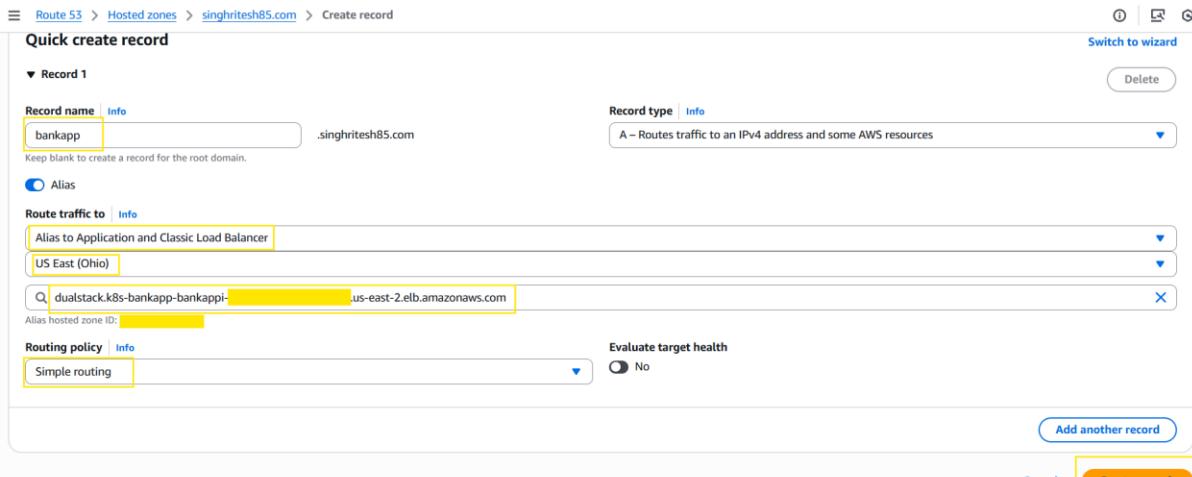
```

I created the Ingress Rule for BankApp and did the entry in Route53 to create the A-Type Record Set with HOST and DNS Name of the LoadBalancer as shown in the screenshot attached below.

```

[root@REDACTED ~]# kubectl get svc -n bankapp
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP  172.REDACTED.79   <none>        80/TCP   23m
[REDACTED]# cat ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:REDACTED:certificate/REDACTED
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/wafv2-acl-arn: arn:aws:wafv2:us-east-2:REDACTED:regional/webacl/aws-wafv2-rate-limit/
spec:
  ingressClassName: alb
  rules:
  - host: bankapp.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80
[REDACTED]# kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/bankapp-ingress created
[REDACTED]# kubectl get ing -n bankapp
NAME           CLASS      HOSTS          ADDRESS
bankapp-ingress   alb      bankapp.singhritesh85.com   k8s-bankapp-bankappi:REDACTED.us-east-2.elb.amazonaws.com   PORTS   AGE
[REDACTED]# kubectl get svc -n bankapp
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bankapp-service   ClusterIP  172.REDACTED.79   <none>        80/TCP   9s

```



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

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-2:0XXXXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb-ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/wafv2-acl-arn: arn:aws:wafv2:us-east-2:0XXXXXXXXXX6:regional/webacl/aws-wafv2-rate-limit/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
spec:
  ingressClassName: alb
  rules:
  - host: bankapp.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
    backend:
      service:
        name: bankapp-service
      port:
        number: 80
```

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

The image contains three screenshots of a web application for "Goldencat Bank".

- Login Screen:** The URL is bankapp.singhritesh85.com/login. It features a "Login" button at the top right of a central box. Below it are fields for "Username" and "Password", both with placeholder text. A "Login" button is at the bottom right of the box. Below the box is a link to "Register here".
- Registration Screen:** The URL is bankapp.singhritesh85.com/register. It features a "Register" button at the top right of a central box. Below it are fields for "Username" and "Password", both with placeholder text. A "Register" button is at the bottom right of the box. Below the box is a link to "Login here".
- Dashboard Screen:** The URL is bankapp.singhritesh85.com/dashboard. It shows a welcome message "Welcome, ritesh" and a current balance of "\$0.00". Below this is a box titled "Account Details" containing "Account Number: 1" and "Account Type: Savings". At the bottom are three buttons: "Deposit", "Withdraw", and "Transfer Money". The top navigation bar includes links for "Dashboard", "Transactions", and "Logout".

Migrate from Terraform to Opentofu

To Migrate from Terraform to Opentofu make sure about below mentioned points.

1. Take the backup of state file
2. Take the backup of existing Terraform Script.

For backing up terraform state file I created a new s3 bucket and copied the existing state file to the new bucket

```
[root@yellow ~]# aws s3 mb s3://dexter-backup-statefile --region=us-east-2
make_bucket: dexter-backup-statefile

[redacted]
[redacted] ~]# aws s3 ls --region=us-east-2
2025- [redacted] dexter-backup-statefile
2025- [redacted] dexter-velero-backup
2025- [redacted] dolo-dempo
2025- [redacted] s3bucketcapturealblogjenkins

[root@yellow ~]# aws s3 sync s3://dolo-dempo/ s3://dexter-backup-statefile/
copy: s3://dolo-dempo/state/route53/terraform.tfstate to s3://dexter-backup-statefile/state/route53/terraform.tfstate
copy: s3://dolo-dempo/state/dev/terraform.tfstate to s3://dexter-backup-statefile/state/dev/terraform.tfstate
```

For terraform script I have new set of script present in the directory **opentofu-eks-cluster-waf-with-velero-backup-with-cmk** and **opentofu-route53-hosted-zone-with-acm-certificate** as shown in the screenshot attached below.

```
[root@yellow ~]# ls
opentofu-eks-cluster-waf-with-velero-backup-with-cmk  terraform-eks-cluster-waf-with-velero-backup-with-cmk
opentofu-route53-hosted-zone-with-acm-certificate  terraform-route53-hosted-zone-with-acm-certificate
```

The only difference between the Terraform and Opentofu codebase is backend.tf file because in Opentofu I am going to encrypt the state file.

Below is the backend.tf for Terraform.

```
[root@yellow ~]# cat terraform-route53-hosted-zone-with-acm-certificate/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/route53/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ## dynamodb_table = "terraform-state"
  }
}

[redacted]
[redacted] ~]# cat terraform-eks-cluster-waf-with-velero-backup-with-cmk/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/dev/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ## dynamodb_table = "terraform-state"
  }
}
```

Below is the backend.tf for Opentofu before migration.

```
[root@... ~]# cat opentofu-route53-hosted-zone-with-acm-certificate/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/route53/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }

  plan {
    method = method.aes_gcm.passphrase
    fallback {
      method = method.unencrypted.read_unencrypted_state
    }
  }
}
}
```



```
[root@... ~]# cat opentofu-eks-cluster-waf-with-velero-backup-with-cmk/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/dev/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }

  plan {
    method = method.aes_gcm.passphrase
    fallback {
      method = method.unencrypted.read_unencrypted_state
    }
  }
}
..
```

Below is the backend.tf for Opentofu after migration.

```
[root@opentofu ~]# cat opentofu-route53-hosted-zone-with-acm-certificate/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/route53/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}

[root@opentofu 3 ~]# cat opentofu-eks-cluster-waf-with-velero-backup-with-cmk/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/dev/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

I changed in backend.tf because the state file of terraform is not encrypted and you already created the resources using terraform which resources might be utilized in some project so you cannot delete the existing resources created using terraform and the create the new one and terraform state file was unencrypted.

You can follow below steps to install Opentofu.

```
cd /opt/
```

```
curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
```

```
chmod +x install-opentofu.sh
```

```
./install-opentofu.sh --install-method rpm
```

```
rm -f install-opentofu.sh
```

```
[root@XXXXXXXXXX ~]# cd /opt/
[root@XXXXXXXXXX opt]# curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
[root@XXXXXXXXXX opt]# chmod +x install-opentofu.sh
[root@XXXXXXXXXX opt]# ./install-opentofu.sh --install-method rpm
```

```
[root@XXXXXXXXXX opt]# rm -f install-opentofu.sh
```

```
[root@XXXXXXXXXX ~]# tofu --version
OpenTofu v1.10.6
on linux_amd64
```

Below commands you should run for migration.

```
tofu init -reconfigure
```

```
tofu validate
```

```
tofu plan
```

```
tofu apply -auto-approve
```

```
[root@XXXXXXXXXX ~]# cd opentofu-route53-hosted-zone-with-acm-certificate/main/
[root@XXXXXXXXXX main]# tofu init -reconfigure
```

```
[root@XXXXXXXXXX main]# tofu validate
Success! The configuration is valid.
```

```
[root@XXXXXXXXXX main]# tofu plan
```

```
No changes. Your infrastructure matches the configuration.
```

```
OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

```
[root@XXXXXXXXXX main]# tofu apply -auto-approve
```

```
OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

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

```
Outputs:
```

```
route53_hosted_zone_details = {
  "certificate_arn" = "arn:aws:acm:us-east-2:02XXXXXXXXXX:certificate/XXXXXXXXXX"
  "hosted_zone_id" = "XXXXXXXXXX"
  "hosted_zone_name_servers" = tolist([
    "",
    "XXXXXXXXXX",
    "XXXXXXXXXX",
  ])
}
```

Migration had been done, below is the backend.tf after Migration.

```
[root@yellow ~]# cat opentofu-route53-hosted-zone-with-acm-certificate/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/route53/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    #####dynamodb_table = "terraform-state"
  }
  encryption {
    #   method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      #   fallback {
      #     method = method.unencrypted.read_unencrypted_state
      #   }
    }

    plan {
      method = method.aes_gcm.passphrase
      #   fallback {
      #     method = method.unencrypted.read_unencrypted_state
      #   }
    }
  }
}

[root@yellow ~]# cd opentofu-eks-cluster-waf-with-velero-backup-with-cmk/main/
[root@yellow main]# tofu init -reconfigure
```

```
[root@yellow main]# tofu validate
Success! The configuration is valid.
```

```
[root@yellow main]# tofu plan
```

Plan: 0 to add, 5 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so OpenTofu can't guarantee to take exactly these actions if you run "tofu apply" now.

```
[root@yellow main]# tofu apply -auto-approve
```

```

Plan: 0 to add, 5 to change, 0 to destroy.
module.eks_cluster.aws_route_table.private_route_table_3: Modifying... [id=████████]
module.eks_cluster.aws_route_table.private_route_table_2: Modifying... [id=████████]
module.eks_cluster.aws_route_table.private_route_table_1: Modifying... [id=████████]
module.eks_cluster.aws_route_table.private_route_table_2: Modifications complete after 0s [id=████████]
module.eks_cluster.aws_route_table.private_route_table_3: Modifications complete after 0s [id=████████]
module.eks_cluster.aws_route_table.private_route_table_1: Modifications complete after 0s [id=████████]
module.eks_cluster.aws_eks_node_group.eksnode: Modifying... [id=eks-demo-cluster-dev:eks-nodegroup-dev]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Modifying... [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still modifying... [id=eks-demo-cluster-dev:amazon-cloudwatch-observability, 10s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Modifications complete after 14s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]

Apply complete! Resources: 0 added, 5 changed, 0 destroyed.

Outputs:

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

```

Migration had been done, below is the backend.tf after Migration.

```

[root@████████ 3 ~]# cat opentofu-eks-cluster-waf-with-velero-backup-with-cmk/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/dev/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ## dynamodb_table = "terraform-state"
  }
  encryption {
    #   method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      #   fallback {
      #     method = method.unencrypted.read_unencrypted_state
      #   }
    }

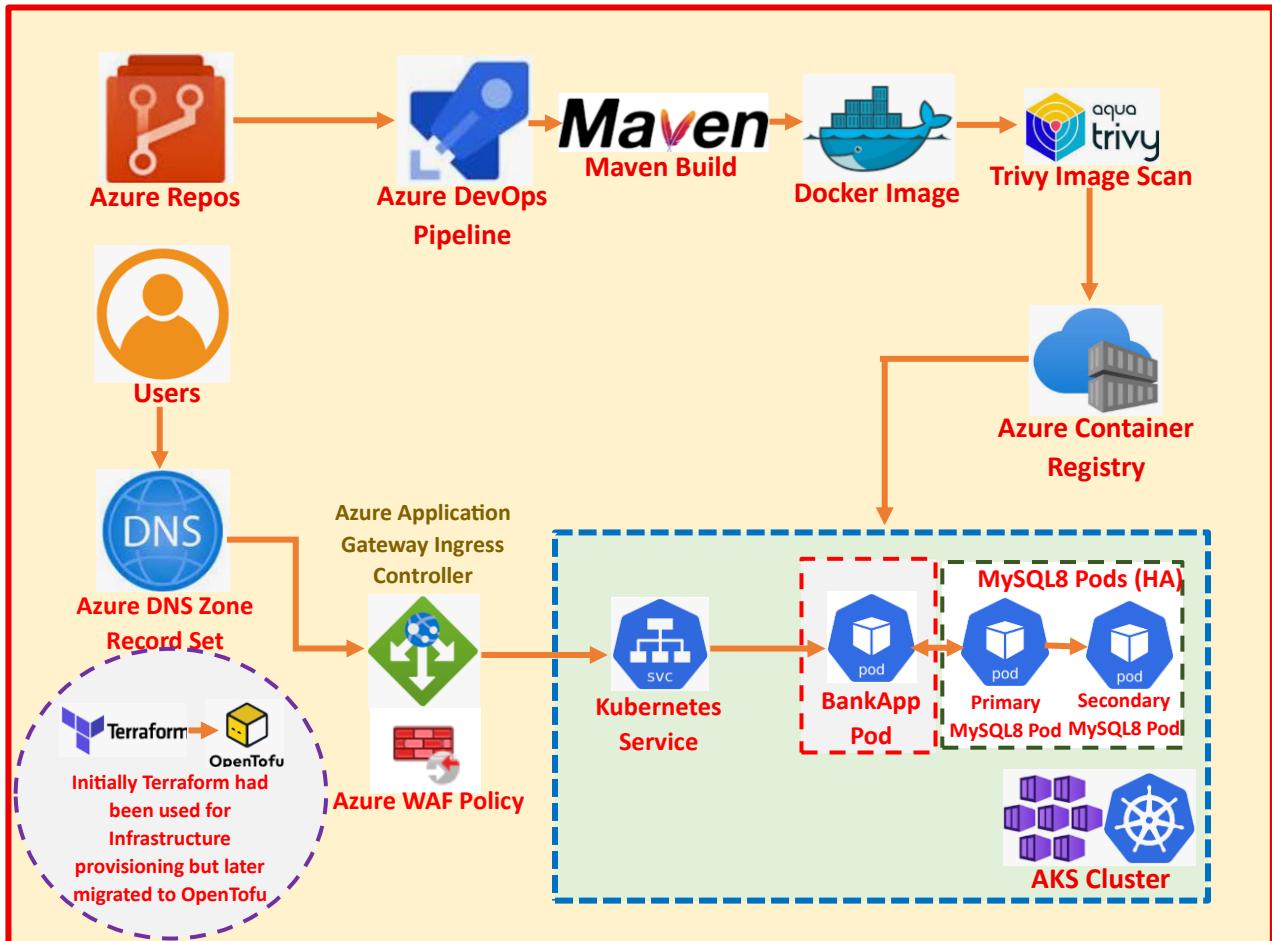
    plan {
      method = method.aes_gcm.passphrase
      #   fallback {
      #     method = method.unencrypted.read_unencrypted_state
      #   }
    }
  }
}

```

Now check state file in s3 bucket you will find it is encrypted. However, if you run **tofu show**, you can see the state file but in S3 Bucket it is encrypted.

DevOps Project BankApp Terraform upgrade to OpenTofu with Secure AKS Cluster

Module-2



Till now I used Terraform to provision the infrastructure but Hashicorp license changes due to which Terraform will introduce restrictions for commercial use, due to this reason many organisations are migrating from Terraform to OpenTofu. However, I would like to mention here that DevOps Team who used it as a tool to provision the infrastructure will not be affected but those who provides it as a Service will be affected. You already aware that Terraform state file was kept locally or in the S3 Bucket / Azure Storage Account Container / GCP Bucket. The problem was its state file was not encrypted, so anyone who had the access of Terraform Server (when state file was kept locally) or Bucket or Storage Account Container can easily see its state file. But if you use OpenTofu, you will not face such an issue. OpenTofu is an Open Source, community-driven Infrastructure as Code (IaC) tool, it is a fork of Hashicorp Terraform. I will migrate from Terraform to OpenTofu at the end of Module-2.

This DevOps project deals will creation of infrastructure using Terraform, GitHub was used to keep the source Code and Azure DevOps had been used as CI/CD Tool. The build tool used in this project was Maven and Docker Image was pushed to the Azure Container Registry (ACR) which was finally deployed to the Azure Kubernetes Services (AKS). The Bank Application Pods was accessed using Application Gateway Ingress Controller (AGIC) and hence using the kubernetes service as shown in the screenshot attached above. I applied WAF Policy (to Limit the Rate) to the Bank Application URL and to prevent from SQL Injection. As this Bank Application was open to entire world so rate limit

and SQL Injection was applied through WAF Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot (by the Hackers) so that its speed becomes slow or this website hangs to refrain from such a situation WAF Policy Rate Limit was applied. Finally, I migrated from Terraform to OpenTofu. For AKS Cluster Node-to-Node encryption (at rest) is enabled by setting **host_encryption_enabled = true** using Terraform and Node-to-Node encryption (in transit) is enabled by running the command **kubectl patch felixconfiguration default --type='merge' -p '{"spec":{"wireguardEnabled":true}}'** after creation of the AKS cluster as shown in the screenshot attached below. In AKS cluster nodes kernel has WireGuard installed already, so there is no manual installation required.

I implemented Istio for encrypted Pod-to-Pod Communication and implemented **Kyverno** (Open-Source Policy Engine) and **Falco** to Secure the AKS Cluster. Kyverno was developed by Nirmata and donated to CNCF. It comes under CNCF Incubating Project and used by Organisation like Adidas, Wayfair, US Department of Defence, Spotify, Bloomberg, Robinhood and Razorpay. Like OPA Gatekeeper, Kyverno is a **dynamic admission controller**. I had used Aqua Security Trivy image scan for vulnerability scanning during CI/CD Pipeline as shown in the end-to-end architecture diagram of the project drawn above. In AKS Cluster to enable envelope encryption I used **key_management_service** in Terraform Script. The Azure Key-Vault Key enables you to rotate it as per the standard followed by your organisation. Usually after 90 days the Azure Key-Vault Key should be rotated but you can rotate it as per your organisation need. If you use Azure Platform Managed Key then it will be rotated by Azure, however Azure did not announce it publicly that after specific number of days it will be rotated by Azure. In this project I used Azure Key-Vault Key for envelope encryption and to encrypt the all the disks used in this project.

In AKS however it is possible to encrypt all the disks either by Azure Platform Managed Key or Azure Key-Vault Key but envelope encryption provides extra layer of security by encrypting the etcd.

Terraform script to create the infrastructure is present at the path **terraform-azure-dns-zone** (this terraform script is handy for you to create the Azure DNS Zone) in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>. Commands as written below had been used to create infrastructure using Terraform.

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

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

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

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

After successful execution of the command **terraform apply -auto-approve** make sure you do the entry for Nameservers of the Azure DNS Zone in your domain name service provider's Nameserver.

```

+ soa_record (known after apply)
}

# module.dns.azurerm_resource_group.dns_rg will be created
+ resource "azurerm_resource_group" "dns_rg" {
  + id      = (known after apply)
  + location = "eastus"
  + name    = "dns-resource-group"
}

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

Changes to Outputs:
+ azure_dns_zone_name_and_nameserver = {
  + dns_zone_name = "singhritesh85.com"
  + name_servers = (known after apply)
}

module.dns.azurerm_resource_group.dns_rg: Creating...
module.dns.azurerm_resource_group.dns_rg: Still creating... [00m10s elapsed]
module.dns.azurerm_resource_group.dns_rg: Creation complete after 11s [id=/subscriptions/[REDACTED]/resourceGroups/dns-resource-group]
module.dns.azurerm_dns_zone.dns_zone: Creating...
module.dns.azurerm_dns_zone.dns_zone: Creation complete after 5s [id=/subscriptions/[REDACTED]/resourceGroups/dns-resource-group/providers/Microsoft.Network/dnsZones/singhritesh85.com]

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

Outputs:

azure_dns_zone_name_and_nameserver = {
  "dns_zone_name" = "singhritesh85.com"
  "name_servers" = toset([
    "192.168.1.1",
    "192.168.1.2",
    "192.168.1.3",
    "192.168.1.4"
  ])
}

```

Finally, the Azure DNS Zone had been created. Then I created the other Azure Resources using the terraform script present at the path **terraform-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault** in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>.

```

bankapp-rg/providers/Microsoft.RecoveryServices/vaults/bankapp-recovery-vault/backupFabrics/Azure/protectionContainers/iaasvmcontainer;iaasvmcontainerv2;bankapp-rg;devopsagent-vm/protectedItems/VM;iaasvmcontainerv2;bankapp-rg;devopsagent-vm]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [03m30s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [03m40s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [03m50s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Still creating... [04m00s elapsed]
module.aks.azurerm_kubernetes_cluster_extension.aks_cluster_extension: Creation complete after 4m2s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster/providers/Microsoft.KubernetesConfiguration/extensions/bankapp-extension]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Creating...
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Creating...
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Still creating... [00m10s elapsed]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Still creating... [00m10s elapsed]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Still creating... [00m20s elapsed]
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Still creating... [00m20s elapsed]
module.aks.azurerm_role_assignment.backup_vault_data_contributor_on_storage: Creation complete after 28s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Storage/storageAccounts/bankappbackupsa2025/providers/Microsoft.Authorization/roleAssignments/[REDACTED]]
module.aks.azurerm_role_assignment.backup_extension_and_storage_account_permission: Creation complete after 29s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Storage/storageAccounts/bankappbackupsa2025/providers/Microsoft.Authorization/roleAssignments/[REDACTED]]
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Creating...
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Still creating... [00m10s elapsed]
module.aks.azurerm_data_protection_backup_instance_kubernetes_cluster.aks_backup_instance: Creation complete after 17s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.DataProtection/backupVaults/bankapp-backup-vault/backupInstances/bankapp-backup-instance]

Apply complete! Resources: 62 added, 0 changed, 0 destroyed.

Outputs:

acr_azurevm_private_ip_and_aks_details_waf_policy_name_and_application_gateway_name = {
  "acr_login_server" = "bankappcontainer24registry.azurecr.io"
  "aks_id" = "/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster"
  "aks_name" = "bankapp-cluster"
  "application_gateway_name" = "app-gtw-ingress-controller"
  "azurevm_devopsagent_privateip" = "10.0.0.10"
  "web_application_firewall_policy_name" = "bankapp-wafpolicy"
}

```

[root@[REDACTED] ~]# kubectl patch felixconfiguration default --type='merge' -p '{"spec":{"wireguardEnabled":true}}'

[root@[REDACTED] ~]# kubectl get felixconfiguration default -o yaml

After enabling wireguard encryption in AKS Cluster I checked felixconfiguration resource to see if WireGuard is enabled or not and found **spec.wireguardEnabled: true** as shown below.

```

[root@[REDACTED] ~]# kubectl get felixconfiguration default -o yaml

spec:
  wireguardEnabled: true

```

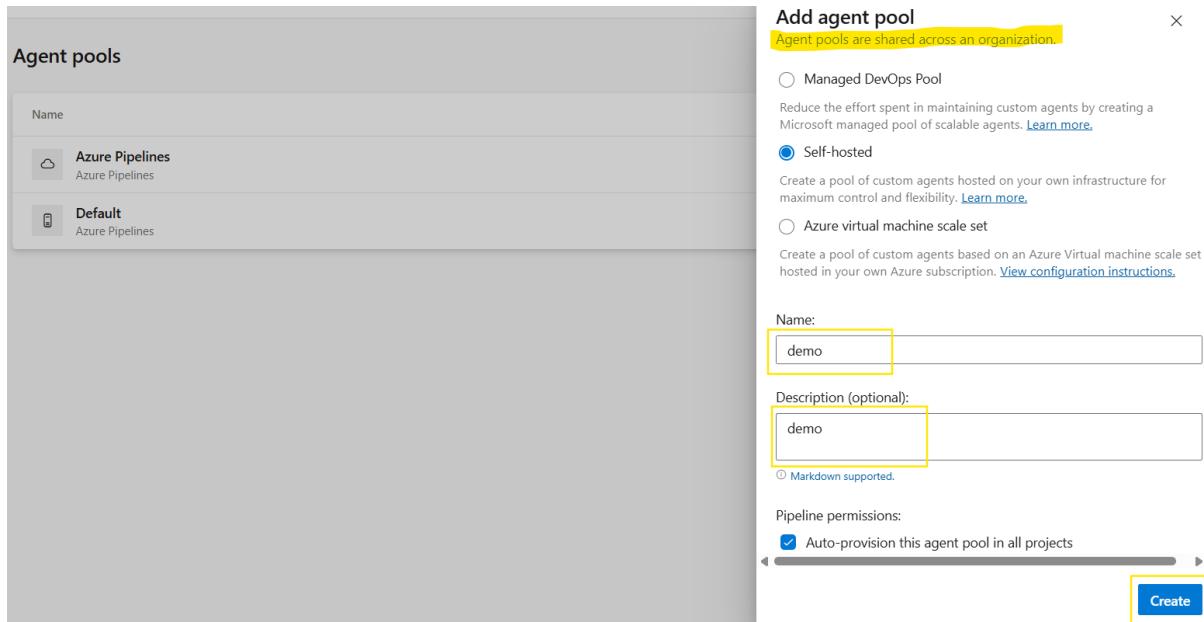
In the installed AKS Cluster OPA Gatekeeper is already installed in the namespace **gatekeeper-system** and I explained it in Module-1, so in Module-2 I will install Kyverno Policy Agent using Helm and achieve the same goal which I explained in Module-1 with OPA Gatekeeper. If you want to go with OPA Gatekeeper then create the OPA Policy (Constraint Template and Constraint) and directly apply them to use here.

Installation of Azure DevOps Agent

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

The screenshot shows two main sections of the Azure DevOps UI:

- Organization settings:** This section is visible at the top. It includes a search bar, a 'New organization' button, a 'Projects' tab (selected), and a 'my-demo-project' card. The 'Organization settings' link is highlighted with a yellow box.
- Agent pools:** This section is visible below. It has a sidebar with navigation links like Organization Settings, Security, Boards, Pipelines, Repos, Artifacts, and Storage. The 'Agent pools' link under Pipelines is highlighted with a yellow box. The main area shows an 'Overview' form with fields for Name, Privacy URL, Description, Time zone, Geography, and Region. A note at the bottom says "Changes made will affect all projects and members of the organization". A 'Save' button is present. At the top right of this section is a 'Security' button and an 'Add pool' button, both highlighted with yellow boxes.



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

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

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

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

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

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

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

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

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

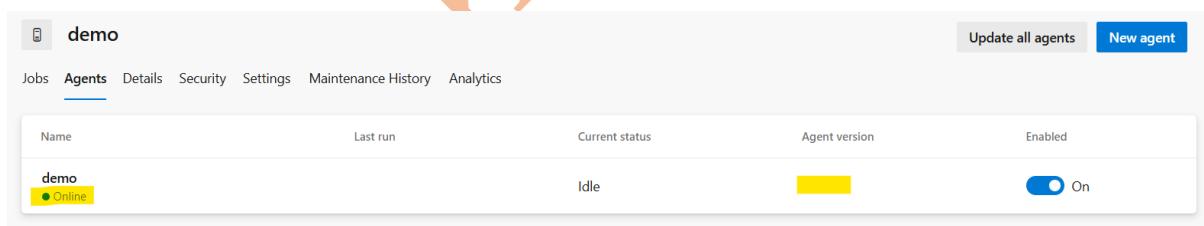
>> Register Agent:
Enter agent pool (press enter for default) > demo
Enter agent name (press enter for devopsagent-vm) > demo
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) > 2025
2025 Settings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

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

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

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

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



Name	Last run	Current status	Agent version	Enabled
demo ● Online		Idle	[REDACTED]	<input checked="" type="checkbox"/> On

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

The screenshot shows the Azure DevOps interface for a project named 'M'. The left sidebar lists various project management sections like Overview, Summary, Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The 'Project settings' tab is selected. In the main area, there's a 'About this project' section with a 'Help others to get on board!' placeholder and a 'Add Project Description' button. To the right is a cartoon illustration of a person running on clouds. Below this is a 'Service connections' section with a 'New service connection' button. The sidebar also includes sections for Boards, Pipelines, and XAML build services, with 'Service connections' highlighted.

This dialog box is titled 'New GitHub service connection'. It has several fields: 'Authentication method' (radio buttons for 'Grant authorization' and 'Personal Access Token', with 'Grant authorization' selected); 'OAuth Configuration' (dropdown set to 'AzurePipelines'); 'Service connection details' (text input field for 'Service Connection Name' containing 'GitHub-Account'); and optional fields for 'Service Management Reference (optional)' and 'Description (optional)'. A red arrow points from the text 'Authorize your GitHub Account' in the main service connections list to the 'Authorize' button in this dialog.

New GitHub service connection

Authentication

OAuth Configuration

AzurePipelines

GitHub-Account Change

Service connection details

Service Connection Name

GitHub-Account

Service Management Reference (optional)

Description (optional)

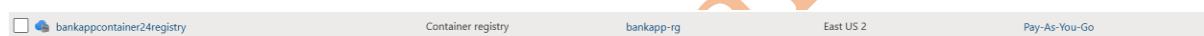
Security

Grant access permission to all pipelines

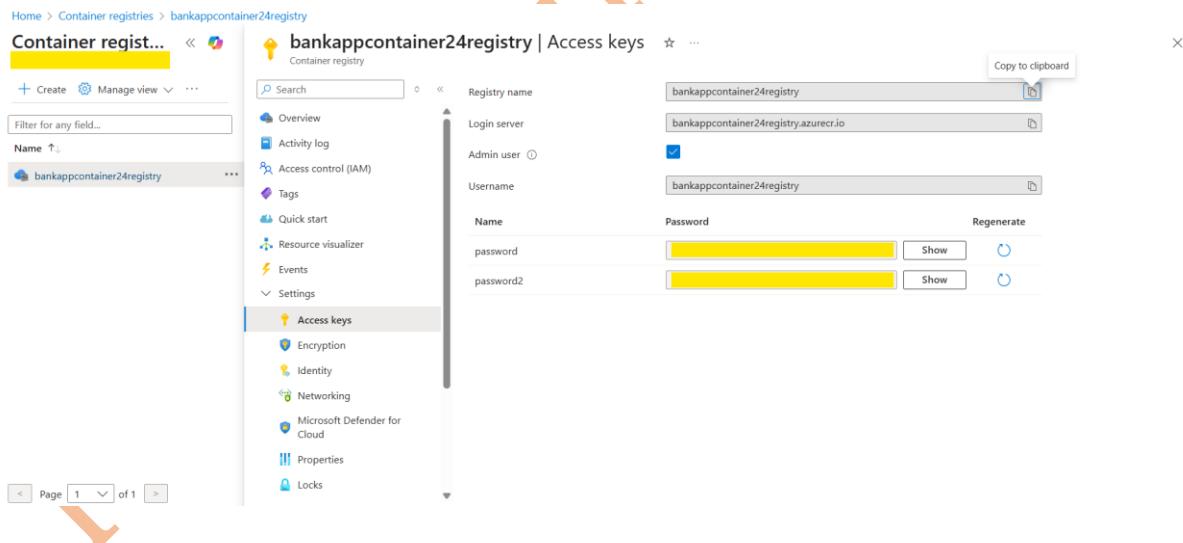
Learn more Troubleshoot

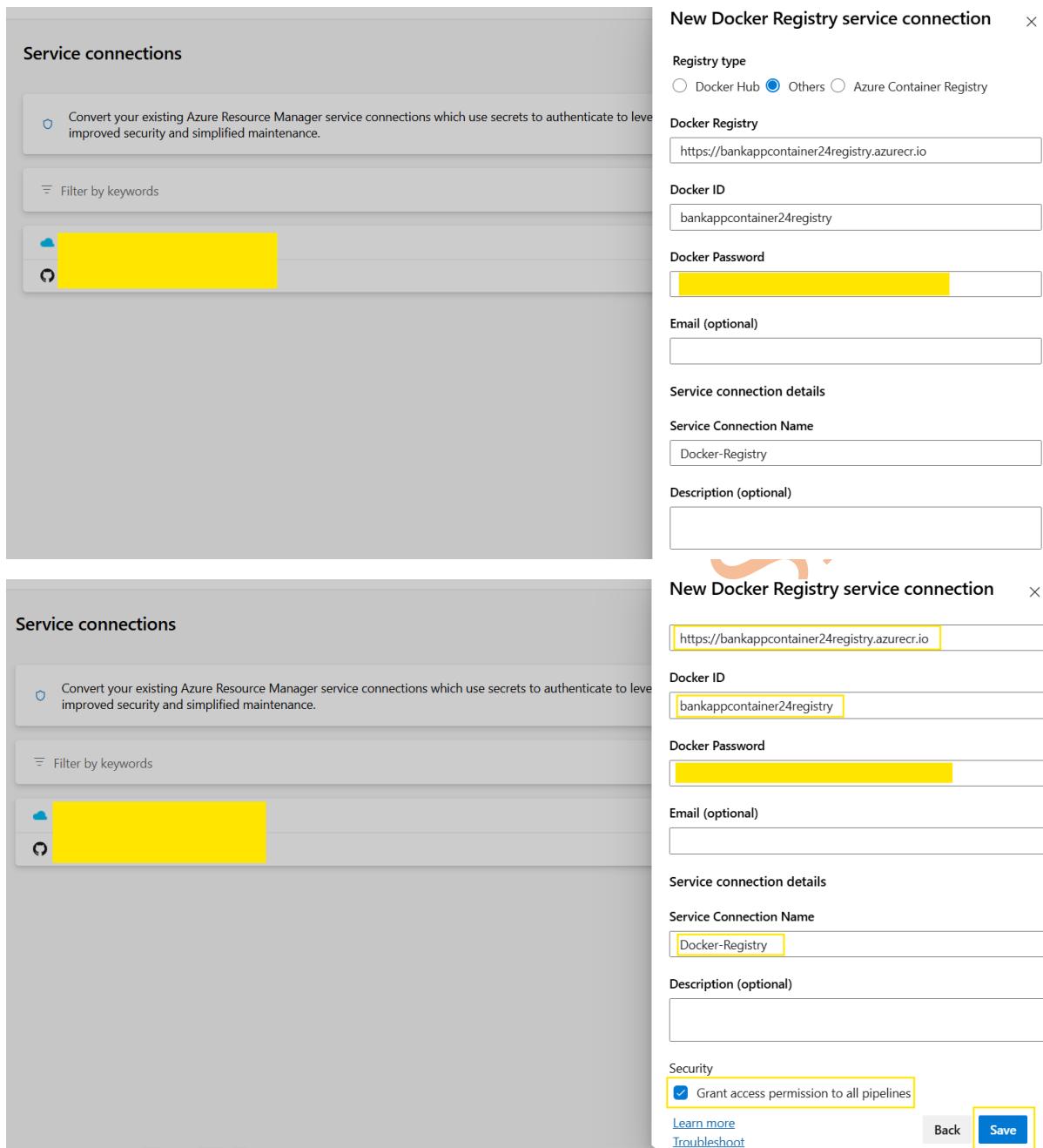
Back Save

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



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





New Docker Registry service connection

Registry type

- Docker Hub
- Others
- Azure Container Registry

Docker Registry

Docker ID

Docker Password

Email (optional)

Service connection details

Service Connection Name

Description (optional)

New Docker Registry service connection

Docker ID

Docker Password

Email (optional)

Service connection details

Service Connection Name

Description (optional)

Security

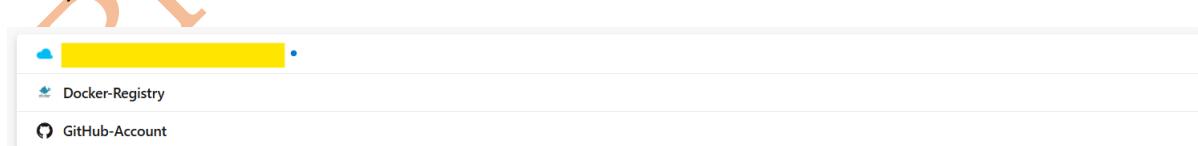
Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)

[Back](#) [Save](#)

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



Docker-Registry

GitHub-Account

Installation of Kyverno Policy Engine

To install Kyverno Policy Engine follow the steps as written below.

```
helm repo add kyverno https://kyverno.github.io/kyverno/
helm repo update
helm install kyverno kyverno/kyverno --set admissionController.replicas=3 --set enablePolicyException=true --set backgroundController.replicas=2 --set cleanupController.replicas=2 --set reportsController.replicas=2 --namespace kyverno --create-namespace
kubectl get pods -n kyverno
```

```
[root@yellow ~]# helm repo add kyverno https://kyverno.github.io/kyverno/
"kyverno" has been added to your repositories
[root@yellow ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "kyverno" chart repository
Update Complete. ⚡Happy Helming!⚡
[root@yellow ~]# helm install kyverno kyverno/kyverno --set admissionController.replicas=3 --set enablePolicyException=true --set backgroundController.replicas=2 --set cleanupController.replicas=2 --set reportsController.replicas=2 --namespace kyverno --create-namespace
```

NAME	READY	STATUS	RESTARTS	AGE
kyverno-admission-controller- yellow	1/1	Running	0	
kyverno-admission-controller- yellow	1/1	Running	0	
kyverno-admission-controller- yellow	1/1	Running	0	
kyverno-background-controller- yellow	1/1	Running	0	
kyverno-background-controller- yellow	1/1	Running	0	
kyverno-cleanup-controller- yellow	1/1	Running	0	
kyverno-cleanup-controller- yellow	1/1	Running	0	
kyverno-reports-controller- yellow	1/1	Running	0	
kyverno-reports-controller- yellow	1/1	Running	0	

I had applied the Kyverno policy as shown in the screenshot attached below.

```
[root@yellow ~]# kubectl apply -f kyverno-policy.yaml
clusterpolicy.kyverno.io/no-loadbalancer-service created
clusterpolicy.kyverno.io/disallow-latest-tag created
clusterpolicy.kyverno.io/disallow-privileged-and-escalation created
clusterpolicy.kyverno.io/require-requests-limits created
[root@yellow ~]# kubectl get clusterpolicy
NAME          ADMISSION   BACKGROUND   READY   AGE      MESSAGE
disallow-latest-tag  true        true        True    6m56s  Ready
disallow-privileged-and-escalation  true        true        True    6m55s  Ready
no-loadbalancer-service   true        true        True    6m56s  Ready
require-requests-limits   true        true        True    6m55s  Ready
```

This policy restricts use of the Service type LoadBalancer

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: no-loadbalancer-service
  annotations:
    policies.kyverno.io/title: Disallow Service Type LoadBalancer
    policies.kyverno.io/category: Sample
    policies.kyverno.io/severity: medium
    policies.kyverno.io/subject: Service
    policies.kyverno.io/minversion: 1.6.0
    policies.kyverno.io/description: >-
```

Especially in cloud provider environments, a Service having type LoadBalancer will cause the provider to respond by creating a load balancer somewhere in the customer account. This adds cost and complexity to a deployment. Without restricting this ability, users may easily overrun established budgets and security practices set by the organization. This policy restricts use of the Service type LoadBalancer.

```
spec:
  validationFailureAction: Enforce  ### Block the resource creation if the policy is violated
  background: true
  rules:
    - name: no-LoadBalancer
      match:
        any:
          - resources:
              kinds:
                - Service
              namespaces:
                - bankapp
                - mysql
```

```

validate:
  message: "Service of type LoadBalancer is not allowed."
pattern:
spec:
  type: "!LoadBalancer"
---

### This policy validates that the image specifies a tag and that it is not called latest ####
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-latest-tag
  annotations:
    policies.kyverno.io/title: Disallow Latest Tag
    policies.kyverno.io/category: Best Practices
    policies.kyverno.io/minversion: 1.6.0
    policies.kyverno.io/severity: medium
    policies.kyverno.io/subject: Pod
    policies.kyverno.io/description: >-
      The ':latest' tag is mutable and can lead to unexpected errors if the
      image changes. A best practice is to use an immutable tag that maps to
      a specific version of an application Pod. This policy validates that the image
      specifies a tag and that it is not called 'latest'.
spec:
  validationFailureAction: Enforce  ### Block the resource creation if the policy is violated
  background: true
rules:
  - name: require-image-tag
    match:
      any:
        - resources:
          kinds:

```

```
- Pod
namespaces:
- bankapp
- mysql

validate:
message: "An image tag is required."
foreach:
- list: "request.object.spec.containers"
pattern:
image: "*:*"
- list: "request.object.spec.initContainers"
pattern:
image: "*:*"
- list: "request.object.spec.ephemeralContainers"
pattern:
image: "*:*"
- name: validate-image-tag
match:
any:
- resources:
kinds:
- Pod
namespaces:
- bankapp
- mysql

validate:
message: "Using a mutable image tag e.g. 'latest' is not allowed."
foreach:
- list: "request.object.spec.containers"
pattern:
image: "!*:latest"
```

```

- list: "request.object.spec.initContainers"
  pattern:
    image: "!*:latest"

- list: "request.object.spec.ephemeralContainers"
  pattern:
    image: "!*:latest"

---
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-privileged-and-escalation
  annotations:
    policies.kyverno.io/title: Disallow Privileged and Privilege Escalation
    policies.kyverno.io/category: Pod Security Standards (Restricted)
    policies.kyverno.io/severity: medium
    policies.kyverno.io/subject: Pod
    policies.kyverno.io/description: >-
      Privileged mode disables most security mechanisms. Privilege escalation,
      such as via set-user-ID or set-group-ID file mode, should not be allowed.
      This policy ensures Pods do not request privileged mode and that
      allowPrivilegeEscalation is set to false.

spec:
  validationFailureAction: Enforce
  background: true
  rules:
    - name: disallow-privileged-containers
      match:
        any:
          - resources:
              kinds:
                - Pod

```

```

namespaces:
- bankapp
- mysql

validate:
message: "Privileged containers are disallowed."

pattern:
spec:
=(initContainers):
- =(securityContext):
  =(Privileged): "false"
containers:
- =(securityContext):
  =(Privileged): "false"

- name: disallow-privilege-escalation

match:
any:
- resources:
  kinds:
  - Pod

namespaces:
- bankapp
- mysql

validate:
message: "Privilege escalation is disallowed."

pattern:
spec:
=(initContainers):
- =(securityContext):
  =(allowPrivilegeEscalation): "false"
containers:
- =(securityContext):
  =(allowPrivilegeEscalation): "false"

```

This policy validates that all containers have something specified for memory and CPU requests and memory limits

apiVersion: kyverno.io/v1

kind: ClusterPolicy

metadata:

name: require-requests-limits

annotations:

policies.kyverno.io/title: Require Limits and Requests

policies.kyverno.io/category: Best Practices, EKS Best Practices

policies.kyverno.io/severity: medium

policies.kyverno.io/subject: Pod

policies.kyverno.io/minversion: 1.6.0

policies.kyverno.io/description: >-

As application workloads share cluster resources, it is important to limit resources requested and consumed by each Pod. It is recommended to require resource requests and limits per Pod, especially for memory and CPU. If a Namespace level request or limit is specified, defaults will automatically be applied to each Pod based on the LimitRange configuration.

This policy validates that all containers have something specified for memory and CPU requests and memory limits.

spec:

validationFailureAction: Audit

background: true

rules:

- name: validate-resources

match:

any:

- resources:

kinds:

- Pod

```
- Deployment
- StatefulSet
- DaemonSet
namespaces:
- bankapp
- mysql
validate:
message: "CPU and memory resource requests and memory limits are required for containers."
pattern:
spec:
containers:
- resources:
requests:
    memory: "?*"
    cpu: "?*"
limits:
    memory: "?*"
=(initContainers):
- resources:
requests:
    memory: "?*"
    cpu: "?*"
limits:
    memory: "?*"
=(ephemeralContainers):
- resources:
requests:
    memory: "?*"
    cpu: "?*"
limits:
    memory: "?*"
```

Installation of Falco in AKS Cluster

Falco is a tool that captures and analyses logs within a runtime environment to detect suspicious and malicious activity. I had installed here Falco with the custom rule that if anyone will try to access the /etc directory to write in a file then its logs will be captured by the Falco.

cat dexter-falco-rule.yaml

customRules:

 custom-rules.yaml: |-

 - rule: Write below etc

 desc: An attempt to write to /etc directory

 condition: >

 (evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)

 and fd.name startswith /etc

 output: "File below /etc opened for writing | file=%fd.name pcmdline=%proc.pcmdline gparent=%proc.pname[2] ggparent=%proc.pname[3] gggparent=%proc.pname[4] evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname command=%proc.cmdline terminal=%proc.tty"

 priority: WARNING

 tags: [filesystem, mitre_persistence]

helm repo add falco https://falcosecurity.github.io/charts

helm repo update

helm upgrade --install falco falco https://falcosecurity.github.io/charts/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcosidekick.enabled=true --set falcosidekick.webui.enabled=true --set falcosidekick.webui.redis.storageEnabled=false

kubectl get pods -n falco --watch

```
[root@XXXXXXXXXX ~]# cat dexter-falco-rule.yaml
customRules:
  custom-rules.yaml: |-"
    - rule: Write below etc
      desc: An attempt to write to /etc directory
      condition: >
        (evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)
        and fd.name startswith /etc
      output: "File below /etc opened for writing | file=%fd.name pcmdline=%proc.pcmdline gparent=%proc.pname[2] ggparent=%proc.pname[3] gggparent=%proc.pname[4] evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname command=%proc.cmdline terminal=%proc.tty"
      priority: WARNING
      tags: [filesystem, mitre_persistence]
[root@XXXXXXXXXX ~]# helm repo add falco https://falcosecurity.github.io/charts
"falco" 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 "kyverno" chart repository
...Successfully got an update from the "falco" chart repository
Update Complete. Happy Helm-ing!
[root@XXXXXXXXXX ~]# helm upgrade --install falco falco https://falcosecurity.github.io/charts/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcosidekick.enabled=true --set falcosidekick.webui.enabled=true --set falcosidekick.webui.redis.storageEnabled=false
```

NAME	READY	STATUS	RESTARTS	AGE
falco-falcosidekick-[REDACTED]	1/1	Running	0	92s
falco-falcosidekick-[REDACTED]	1/1	Running	0	92s
falco-falcosidekick-[REDACTED]	1/1	Running	0	92s
falco-falcosidekick-[REDACTED]	1/1	Running	0	92s
falco-falcosidekick-[REDACTED]	1/1	Running	0	92s
falco-[REDACTED]	2/2	Running	0	92s
falco-[REDACTED]	2/2	Running	0	92s

I had created Ingress Rule for Falco as shown in the screenshot attached below.

```
[root@REDACTED ~]# cat falco-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-secret
  rules:
  - host: falco.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: falco-falcosidekick-ui
            port:
              number: 2802
[root@REDACTED ~]# kubectl apply -f falco-ingress-rule.yaml
ingress.networking.k8s.io/falco-ui-ingress created
[root@REDACTED ~]# kubectl get ing -n falco
NAME           CLASS          HOSTS          ADDRESS        PORTS   AGE
falco-ui-ingress  azure-application-gateway  falco.singhritesh85.com  68.[REDACTED]  80, 443  9s
```

Created the kubernetes secrets as shown in the screenshot attached below.

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

I did the entry for Ingress HOST with External IP Address in Azure DNS Zone to create the Record Set of A-Type as shown in the screenshot attached below.

Add record set

singhritesh85.com

Name: falco

Type: A - IPv4 Address records

Alias record set: No

TTL: 1

TTL unit: Hours

IP address: 68.183.131.131

0.0.0.0

Add Cancel Give feedback

Finally, I was able to access Falco UI as shown in the screenshot attached below. You can login with username and password admin and admin respectively.

falco.singhritesh85.com/login

Falcosidekick UI

DASHBOARD EVENTS INFO refresh 10s

Login

Password

LOGIN

It is also possible to check falco logs using the command

`kubectl logs -l "app.kubernetes.io/instance=falco" -n falco`

```
cat falco-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
    - secretName: ingress-secret
  rules:
    - host: falco.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: falco-falcosidekick-ui
                port:
                  number: 2802
```

Installation of Istio in AKS Cluster using Helm

To install Istio Service mesh using helm chart use the command as written below.

```
helm repo add istio https://istio-release.storage.googleapis.com/charts
```

```
helm repo update
```

```
kubectl create ns istio-system
```

```
helm install istio-base istio/base -n istio-system
```

```
helm install istiod istio/istiod -n istio-system
```

```
[root@XXXXXXXXXX ~]# helm repo add istio https://istio-release.storage.googleapis.com/charts
"istio" 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 "falcosecurity" chart repository
...Successfully got an update from the "istio" chart repository
...Successfully got an update from the "kyverno" chart repository
Update Complete. Happy Helming!
[root@XXXXXXXXXX ~]# kubectl create ns istio-system
namespace/istio-system created

[root@XXXXXXXXXX ~]# helm install istio-base istio/base -n istio-system
[root@XXXXXXXXXX ~]# helm install istiod istio/istiod -n istio-system
```

Then I had installed Kiali, Prometheus and Grafana as shown in the screenshot attached below.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/kiali.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/prometheus.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/grafana.yaml
```

```
[root@XXXXXXXXXX ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/kiali.yaml
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
service/kiali created
deployment.apps/kiali created
[root@XXXXXXXXXX ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/prometheus.yaml
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
service/prometheus created
deployment.apps/prometheus created
[root@XXXXXXXXXX ~]# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/grafana.yaml
serviceaccount/grafana created
configmap/grafana created
service/grafana created
deployment.apps/grafana created
configmap/istio-grafana-dashboards created
configmap/istio-services-grafana-dashboards created
```

```
[root@XXXXXXXXXX ~]# kubectl get pods -n istio-system --watch
NAME                  READY   STATUS    RESTARTS   AGE
grafana-cdb9db549-5652h   1/1     Running   0          57s
istiod-589d764-xbx7c      1/1     Running   0          5m7s
kiali-56f54f58f9-57k1h     1/1     Running   0          77s
prometheus-7bf56b6bc-xv66q   2/2     Running   0          66s
```

To establish mesh-wide mTLS I created the peer authentication as shown in the screenshot attached below.

```
cat peerauthentication.yaml  
apiVersion: security.istio.io/v1  
kind: PeerAuthentication  
metadata:  
  name: "default"  
  namespace: istio-system  
spec:  
  mtls:  
    mode: STRICT
```

```
[root@XXXXXXXXXX ~]# cat peerauthentication.yaml  
apiVersion: security.istio.io/v1  
kind: PeerAuthentication  
metadata:  
  name: "default"  
  namespace: istio-system  
spec:  
  mtls:  
    mode: STRICT  
[root@XXXXXXXXXX ~]# kubectl apply -f peerauthentication.yaml  
peerauthentication.security.istio.io/default created
```

```
[root@XXXXXXXXXX ~]# kubectl get peerauthentication -n istio-system  
NAME      MODE   AGE  
default   STRICT  38s
```

```
cat kiali-ingress-rule.yaml

# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n istio-system

---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kiali-ingress
  namespace: istio-system
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
    - secretName: ingress-secret
  rules:
    - host: kiali.singhritesh85.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: kiali
                port:
                  number: 20001
```

I created the Ingress Rule to access Kiali as shown in the screenshot attached below.

kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n istio-system command had been used to create the kubernetes tls secrets as shown in the screenshot attached below.

```
[root@yellow ~]# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n istio-system
secret/ingress-secret created
[root@yellow ~]# cat kiali-ingress-rule.yaml
# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n istio-system
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kiali-ingress
  namespace: istio-system
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-secret
  rules:
  - host: kiali.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kiali
            port:
              number: 20001
[root@yellow ~]# kubectl apply -f kiali-ingress-rule.yaml
ingress.networking.k8s.io/kiali-ingress created
[root@yellow ~]# kubectl get ing -n istio-system
NAME           CLASS      HOSTS             ADDRESS          PORTS   AGE
kiali-ingress   azure-appg...  kiali.singhritesh85.com  68.***.***.***  80, 443  8s
```

I did the entry for HOST resulted from Kiali Ingress Rule and External IP in Azure DNS Zone to create the Record Set of A-Type as shown in the screenshot attached below.

Name	Type
@	NS
falco	A

Add record set

singhritesh85.com

Name

Type

Alias record set

TTL *

TTL unit

IP address

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

The screenshot shows two views of the Kiali interface. The top view is the 'Mesh' section, displaying a traffic graph for Sep 21, 03:51:19 PM ... 03:52:19 PM. It shows a central node labeled 'kiali' connected to three other nodes: 'Prometheus' (red), 'Grafana' (orange), and 'Istiod' (blue). Below the graph are metrics for 'Memory' and 'CPU' usage over time. The bottom view is the 'Overview' section, showing a grid of namespaces: calico-system, dataprotection-microsoft, default, falco, gatekeeper-system, and istio-system. Each namespace entry includes information about Istio config, mTLS status, and application count.

Creation of Azure DevOps Pipeline

Before creation of Azure DevOps Pipeline create the storage class named as **dexter** as shown in screenshot attached below. Then created namespaces mysql, bankapp and Kubernetes Secrets for Azure Container Registry as mentioned below.

```
kubectl create ns bankapp
```

```
kubectl create ns mysql
```

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

```
[root@yellow ~]# kubectl create ns bankapp
namespace/bankapp created
[root@yellow ~]# kubectl create ns mysql
namespace/mysql created
[root@yellow ~]# kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappcontainer24registry.azurecr.io --docker-username=bankappcontainer24registry --docker-password=[REDACTED] -n bankapp
secret/bankapp-auth created
[root@yellow ~]# kubectl get secrets -n bankapp
NAME          TYPE           DATA   AGE
bankapp-auth  kubernetes.io/dockerconfigjson  1      10s
```

cat storage-class.yaml

kind: StorageClass

apiVersion: storage.k8s.io/v1

metadata:

name: dexter

provisioner: disk.csi.azure.com

parameters:

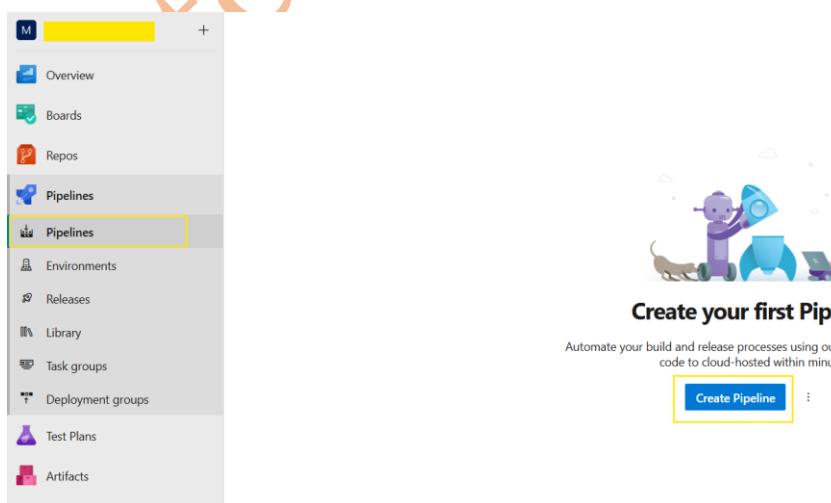
skuname: StandardSSD_LRS

kind: managed

diskEncryptionSetID: "/subscriptions/5XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX/resourceGroups/bankapp-rg/providers/Microsoft.Compute/diskEncryptionSets/bankapp-aks-des" ### Resource ID of Disk Encryption Set

```
[root@yellow ~]# cat storage-class.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: dexter
  provisioner: disk.csi.azure.com
parameters:
  skuname: StandardSSD_LRS
  kind: managed
  diskEncryptionSetID: "/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Compute/diskEncryptionSets/bankapp-aks-des" ### Resource ID of Disk Encryption Set
[root@yellow ~]# kubectl apply -f storage-class.yaml
storageclass.storage.k8s.io/dexter created
```

Then I created the Azure DevOps CI/CD pipeline as shown in the screenshot attached below. First, I created the CI/CD Pipeline to deploy MySQL Pods and after successfully running the MySQL 8 Pods I created the CI/CD Pipeline to deploy bankapp pods.



New pipeline

Where is your code?

- Azure Repos Git YAML
Free private Git repositories, pull requests, and code search
- Bitbucket Cloud YAML
Hosted by Atlassian
- Github YAML**
Home to the world's largest community of developers
- Github Enterprise Server YAML
The self-hosted version of Github Enterprise
- Other Git
Any generic Git repository
- Subversion
Centralized version control by Apache

Use the [classic editor](#) to create a pipeline without YAML.

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

my-demo-project / my-demo-project / Pipelines

✓ Connect **Select** Configure Review

New pipeline

Select a repository

Filter by keywords

My repositories

- singhritesh85/kustomize-bankapp** Yesterday
- singhritesh85/Bank-App-Kustomize Tuesday
- singhritesh85/DevOps-Project-BankApp-WAF-Backup-LoadTest Aug 28
- singhritesh85/terraform-google-cloud-platform Aug 26
- singhritesh85/Bank-App-GKE Aug 25
- singhritesh85/DevOps-Project-BankApp-CI_CD-using-GitLab-GCP-and-AWS Aug 21
- singhritesh85/terraform-script Aug 15
- singhritesh85/Bank-Ann-AKS-WAF**

my-demo-project / my-demo-project / Pipelines

✓ Connect ✓ Select **Configure** Review

New pipeline

Configure your pipeline

Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.

Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.

Show more

Configure your pipeline

Existing Azure Pipelines YAML file

/azure-pipelines.yml

Branch: stage

Path: /azure-pipelines.yml

Continue

```
[demo@devopsagent-vm ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          2m35s
mysql-1   1/1     Running   0          119s
mysql-2   1/1     Running   0          77s
^C[demo@devopsagent-vm ~]$
[demo@devopsagent-vm ~]$ kubectl get pvc -n mysql --watch
NAME           STATUS   VOLUME
mysql-persistent-storage-mysql-0   Bound   pvc-
mysql-persistent-storage-mysql-1   Bound   pvc-
mysql-persistent-storage-mysql-2   Bound   pvc-
```

CAPACITY	ACCESS MODES	STORAGECLASS	VOLUME ATTRIBUTES CLASS	AGE
1Gi	RWO	dexter	<unset>	2m46s
1Gi	RWO	dexter	<unset>	2m10s
1Gi	RWO	dexter	<unset>	88s

As shown in the screenshot attached above PVC and MySQL 8 Pods had been created successfully. Then I created the CI/CD Pipeline to deploy BankApp Pods as shown in the screenshot attached below.

Pipelines

Recent All Runs

MySQL

9m ago

New pipeline

The screenshots illustrate the process of creating a new pipeline in Azure DevOps:

- Select a repository:** Shows a list of repositories from the user 'singhritesh85'. The repository 'singhritesh85/kustomize-bankapp' is selected.
- Configure your pipeline:** Shows the configuration options. The 'Existing Azure Pipelines YAML file' option is selected, and a modal dialog is open for selecting an existing YAML file.
- Select an existing YAML file:** A modal dialog shows the 'Branch' dropdown set to 'stage' and the 'Path' input field containing '/azure-pipelines-1.yml'. The 'Continue' button is highlighted.

Added the two variables GITACCOUNT_USER and GITACCOUNT_PAT as shown in the screenshot attached below.

Azure DevOps / my-demo-project / Pipelines

✓ Connect ✓ Select ✓ Configure Review

New pipeline

Review your pipeline YAML

singhritesh85/kustomize-bankapp / azure-pipelines-1.yml

```

1 trigger:
2   - none
3
4 pr: none
5 pool:
6   - name: demo
7   - demands:
8     - agent.name -equals demo
9
10 stages:
11   - stage: "CloneandBuild"
12     displayName: CloneandBuild
13     jobs:
14       - job: "CloneandBuild"
15         displayName: CloneandBuild
16         steps:
17           - checkout: self
18           - script:
19             - echo "Cloning the external GitHub repository"
20             - git clone -b main https://$(GITACCOUNT_USER):$(GITACCOUNT_PAT)@github.com/singhritesh85/Bank-App-Kustomize.git
21             - task: Maven@4
22               ...

```

Variables Run Show assistant

Azure DevOps / my-demo-project / Pipelines

✓ Connect ✓ Select ✓ Configure Review

New pipeline

Review your pipeline YAML

singhritesh85/kustomize-bankapp / azure-pipelines-1.yml

```

1 trigger:
2   - none
3
4 pr: none
5 pool:
6   - name: demo
7   - demands:
8     - agent.name -equals demo
9
10 stages:
11   - stage: "CloneandBuild"
12     displayName: CloneandBuild
13     jobs:
14       - job: "CloneandBuild"
15         displayName: CloneandBuild
16         steps:
17           - checkout: self
18           - script:
19             - echo "Cloning the external GitHub repository"
20             - git clone -b main https://$(GITACCOUNT_USER):$(GITACCOUNT_PAT)@github.com/singhritesh85/Bank-App-Kustomize.git
21             - task: Maven@4
22               ...

```

New variable

Name: GITACCOUNT_USER

Value:

Keep this value secret

Let users override this value when running this pipeline

To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example: \${GITACCOUNT_USER}

To use a variable in a script, use environment variable syntax. Replace `_` and space with `_`, capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:

- Batch script: \$(GITACCOUNT_USER)
- Powershell script: \$env:GITACCOUNT_USER
- Bash script: \$(GITACCOUNT_USER)

To use a secret variable in a script, you must explicitly map it as an environment variable.

Learn about variables Cancel OK

Azure DevOps / my-demo-project / Pipelines

✓ Connect ✓ Select ✓ Configure Review

New pipeline

Review your pipeline YAML

singhritesh85/kustomize-bankapp / azure-pipelines-1.yml

```

1 trigger:
2   - none
3
4 pr: none
5 pool:
6   - name: demo
7   - demands:
8     - agent.name -equals demo
9
10 stages:
11   - stage: "CloneandBuild"
12     displayName: CloneandBuild
13     jobs:
14       - job: "CloneandBuild"
15         displayName: CloneandBuild
16         steps:
17           - checkout: self
18           - script:
19             - echo "Cloning the external GitHub repository"
20             - git clone -b main https://$(GITACCOUNT_USER):$(GITACCOUNT_PAT)@github.com/singhritesh85/Bank-App-Kustomize.git
21             - task: Maven@4
22               ...

```

Variables

Search variables +

GITACCOUNT_USER

Learn about variables Cancel Save

New pipeline

Review your pipeline YAML

```

1 trigger:
2   - none
3
4 pr: none
5 pool:
6   - name: demo
7   - demands:
8     - agent.name -equals demo
9
10 stages:
11   - stage: "CloneandBuild"
12     displayName: CloneandBuild
13     jobs:
14       - job: "CloneandBuild"
15         displayName: CloneandBuild
16         steps:
17           - checkout: self-
18           - script: |
19             echo "Cloning the external GitHub repository"
20             git clone -b main https://$(GITACCOUNT_USER):$(GITACCOUNT_PAT)@github.com/singhritesh85/kustomize-bankapp.git
21           - task: Maven@4

```

Variables

Name	Value
GITACCOUNT_PAT	[REDACTED]
GITACCOUNT_USER	[REDACTED]

After Running the pipeline, I found Pods had been created successfully as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl get pods -n bankapp --watch
NAME          READY   STATUS    RESTARTS   AGE
bankapp-[REDACTED]   1/1     Running   0          11s

[demo@devopsagent-vm ~]$ kubectl get svc -n bankapp
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
bankapp-service   ClusterIP   10.[REDACTED]      <none>        80/TCP      38s
```

Created the Ingress Rule as shown in the screenshot attached below.

```
cat ingress-rule.yaml

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

---

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

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

[demo@devopsagent-vm ~]$ kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n bankapp
secret/ingress-secret created
[demo@devopsagent-vm ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/bankapp-ingress created
[demo@devopsagent-vm ~]$ kubectl get ing -n bankapp
NAME           CLASS      HOSTS          ADDRESS        PORTS   AGE
bankapp-ingress  azure-application-gateway  bankapp.singhritesh85.com  68.192.168.129  80, 443  8s
[demo@devopsagent-vm ~]$ kubectl get secrets -n bankapp
NAME          TYPE       DATA   AGE
bankapp-auth  kubernetes.io/dockerconfigjson  1      75m
ingress-secret kubernetes.io/tls            2      29s
```

Home > singhritesh85.com

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

Name

Type

TTL

Value

Alias record set

TTL

TTL unit

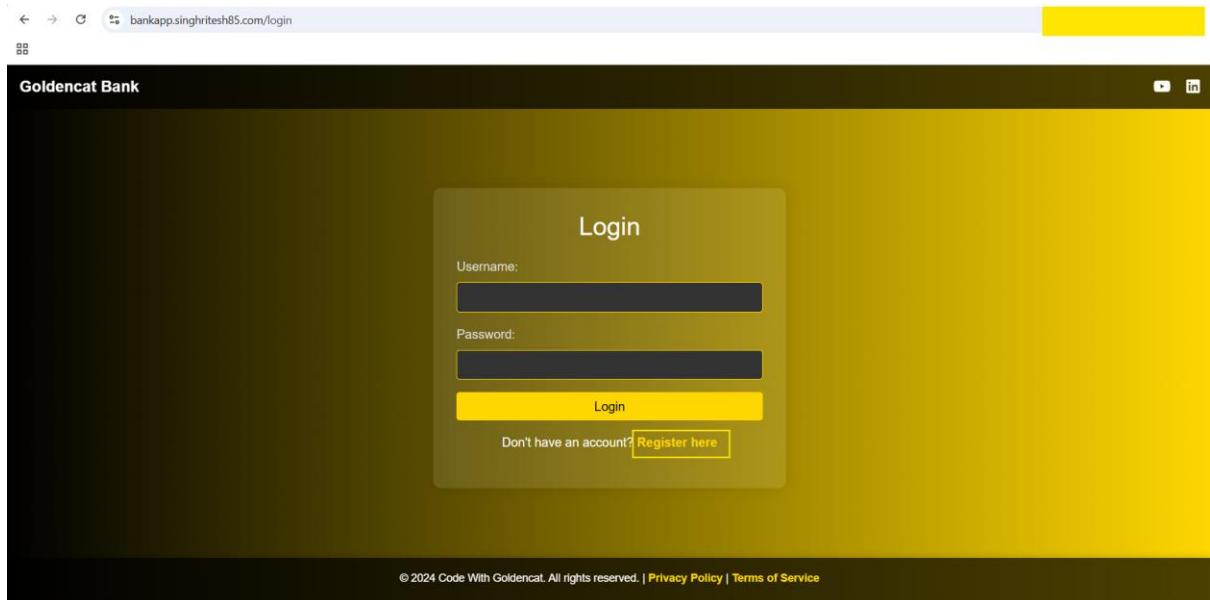
IP address

Add

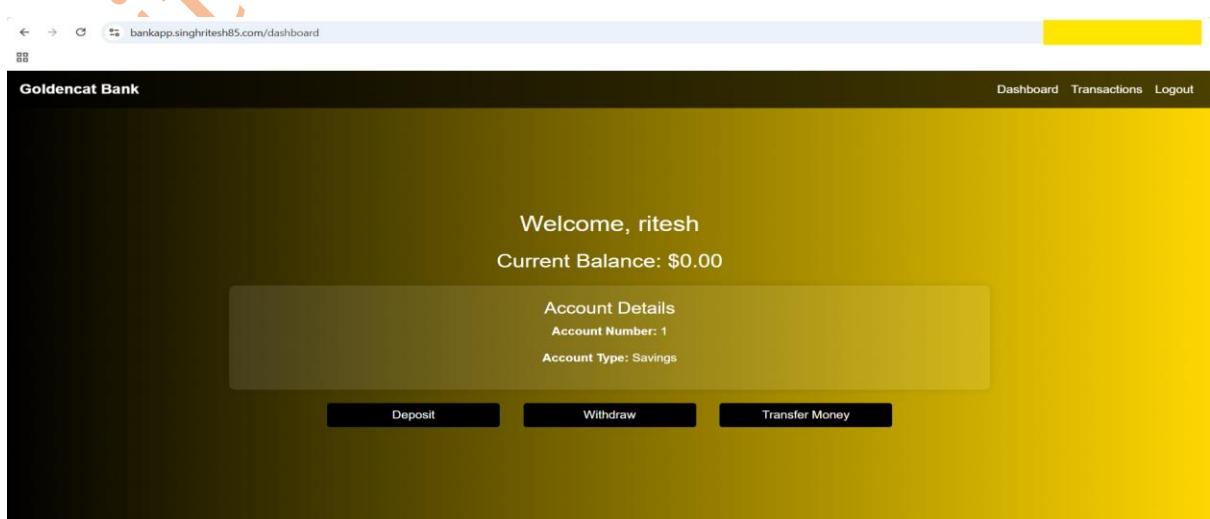
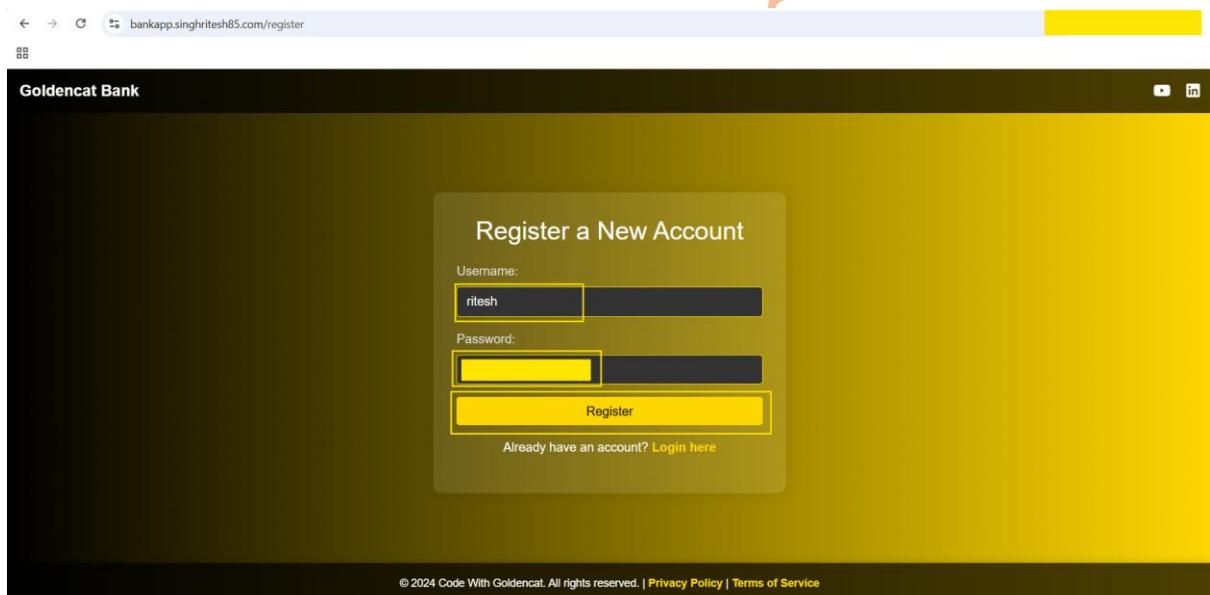
Cancel

Give feedback

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



I registered with a user and logged-in with the same user and found that I was able to login with the newly created user successfully as shown in the screenshot attached below.



Migration from Terraform to OpenTofu

To Migrate from Terraform to Opentofu make sure about below mentioned points.

1. Take the backup of state file
 2. Take the backup of existing Terraform Script.

For backing up terraform state file I created a new Azure Storage Account and container then copied the existing state file to the new bucket

```
[root@Terraform-Server ~]# az storage account create --name dexter202509 --resource-group bankapp-rg --location "East US 2" --sku Standard_LRS
[{"created": true}
[{"root@Terraform-Server ~]# az storage container create --name terraform-state --account-name dexter202509 --account-key
[{"root@Terraform-Server ~]# az storage account list --query "[].name" --output tsv
terraformstate08022024
bankappbackupsa2025
dexter202509
[{"root@Terraform-Server ~]# az storage container generate-sas --name terraform-state --account-name terraformstate08022024 --expiry 2025-09-21T23:00Z --permissions acdlrw --account-key
[{"root@Terraform-Server ~]# az storage container generate-sas --name terraform-state --account-name dexter202509 --expiry 2025-09-21T23:00Z --permissions acdlrw --account-key
[{"root@Terraform-Server ~]#
```

Installed azcopy as shown in the screenshot attached below.

```
[root@Terraform-Server ~]# wget https://aka.ms/downloadazcopy-v10-linux
[root@Terraform-Server ~]# tar -xvf downloadazcopy-v10-linux --strip-components=1
[root@Terraform-Server ~]# sudo mv azcopy /usr/local/bin/
[root@Terraform-Server ~]# chmod +x /usr/local/bin/azcopy

[root@Terraform-Server ~]# azcopy copy 'https://terraformstate08022024.blob.core.windows.net/terraform-state?se=2025-09-01T00:00:00Z' 'https://dexter202509.blob.core.windows.net/terraform-state/?se=2025-09-01T00:00:00Z' --recursive
INFO: Scanning...
INFO: Any empty folders will not be processed, because source and/or destination doesn't have full folder support
Job [REDACTED] has started
Log file is located at: /root/.azcopy/[REDACTED].log

100.0 %, 2 Done, 0 Failed, 0 Pending, 0 Skipped, 2 Total, 2-sec Throughput (Mb/s): 0.8656

Job [REDACTED] summary
Elapsed Time (Minutes): 0.0334
Number of File Transfers: 2
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 2
Number of File Transfers Completed: 2
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 0
Number of Folder Transfers Skipped: 0
Number of Symbolic Links Skipped: 0
Number of Hardlinks Converted: 0
Number of Special Files Skipped: 0
Total Number of Bytes Transferred: 216839
Final Job Status: Completed
```

For terraform script I have new set of script present in the directory **opentofu-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault** and **opentofu-azure-dns-zone** as shown in the screenshot attached below.

The only difference between the Terraform and Opentofu codebase is backend.tf file because in Opentofu I am going to encrypt the state file.

Below is the backend.tf for Terraform.

```
[root@yellow ~]# cat terraform-azure-dns-zone/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name       = "terraform-state"
    key                 = "dns-zone/terraform.tfstate"
    subscription_id     = "5[REDACTED]"      ##### Provide Subscription ID of your Azure Account.
  }
}

[root@yellow ~]# cat terraform-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name       = "terraform-state"
    key                 = "state/terraform.tfstate"
    subscription_id     = "5[REDACTED]"      ##### Provide Subscription ID of your Azure Account.
  }
}
```

Below is the backend.tf for Opentofu before migration.

```
[root@yellow ~]# cat opentofu-azure-dns-zone/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name       = "terraform-state"
    key                 = "dns-zone/terraform.tfstate"
    subscription_id     = "5[REDACTED]"      ##### Provide Subscription ID of your Azure Account.
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ##### To read unencrypted state file
    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
    plan {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }
}
```

```
[root@XXXXXXXXXX ~]# cat opentofu-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name  = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name        = "terraform-state"
    key                  = "state/terraform.tfstate"
    subscription_id      = "5XXXXXXXXXX"      ##### Provide Subscription ID of your Azure Account.
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ##### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }

    plan {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }
}
```

Below is the backend.tf for Opentofu after migration.



```
[root@XXXXXXXXXX ~]# cat opentofu-azure-dns-zone/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name  = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name        = "terraform-state"
    key                  = "dns-zone/terraform.tfstate"
    subscription_id      = "5XXXXXXXXXX"      ##### Provide Subscription ID of your Azure Account.
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ##### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
#      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
#      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

```
[root@yellow ~]# cat opentofu-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name  = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name        = "terraform-state"
    key                  = "state/terraform.tfstate"
    subscription_id      = "5XXXXXXXXXXXX"      ### Provide Subscription ID of your Azure Account.
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file
    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }
  plan {
    method = method.aes_gcm.passphrase
    fallback {
      method = method.unencrypted.read_unencrypted_state
    }
  }
}
}
```

I changed in backend.tf because the state file of terraform is not encrypted and you already created the resources using terraform which resources might be utilized in some project so you cannot delete the existing resources created using terraform and the create the new one and terraform state file was unencrypted.

You can follow below steps to install Opentofu.

cd /opt/

curl --proto '=https' --tlsv1.2 -fsSL <https://get.opentofu.org/install-opentofu.sh> -o install-opentofu.sh

chmod +x install-opentofu.sh

./install-opentofu.sh --install-method rpm

rm -f install-opentofu.sh

```
[root@yellow ~]# cd /opt/
[root@yellow opt]# curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
[root@yellow opt]# chmod +x install-opentofu.sh
[root@yellow opt]# ./install-opentofu.sh --install-method rpm
```

```
[root@yellow opt]# rm -f install-opentofu.sh
```

```
[root@yellow ~]# tofu --version
```

OpenTofu v1.10.6

on linux_amd64

Below commands you should run for migration.

tofu init -reconfigure

tofu validate

tofu plan

tofu apply -auto-approve

```
[root@XXXXXXXXXX ~]# cd opentofu-azure-dns-zone/main/
[root@XXXXXXXXXX main]# tofu init -reconfigure
```

```
[root@XXXXXXXXXX main]# tofu validate
Success! The configuration is valid.
```

```
[root@XXXXXXXXXX main]# tofu plan
```

```
No changes. Your infrastructure matches the configuration.
```

```
OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

```
[root@XXXXXXXXXX main]# tofu apply -auto-approve
```

```
No changes. Your infrastructure matches the configuration.
```

```
OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

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

```
Outputs:
```

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

Migration had been done, below is the backend.tf after Migration.

```
[root@XXXXXXXXXX ~]# cat opentofu-azure-dns-zone/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name  = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name       = "terraform-state"
    key                 = "dns-zone/terraform.tfstate"
    subscription_id     = "5XXXXXXXXXX"    ### Provide Subscription ID of your Azure Account.
  }
  encryption {
    #   method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      # fallback {
      #   method = method.unencrypted.read_unencrypted_state
      # }
    }
    plan {
      method = method.aes_gcm.passphrase
      # fallback {
      #   method = method.unencrypted.read_unencrypted_state
      # }
    }
  }
}
```

```
[root@XXXXXXXXXX ~]# cd opentofu-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault/main/
[root@XXXXXXXXXX main]# tofu init -reconfigure
```

```
[root@[REDACTED] main]# tofu validate
Success! The configuration is valid.
```

```
[root@[REDACTED] main]# tofu plan
```

Plan: 0 to add, 4 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so OpenTofu can't guarantee to take exactly these actions if you run "tofu apply" now.

```
[root@[REDACTED] main]# tofu apply -auto-approve
```

```
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 20s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 30s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 40s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 50s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 1m10s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 1m10s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 1m20s elapsed]
module.aks.azurerm_application_gateway.appgtw: Still modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller, 1m30s elapsed]
module.aks.azurerm_application_gateway.appgtw: Modifications complete after 1m42s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Network/applicationGateways/app-gtw-ingress-controller]
module.aks.azurerm_kubernetes_cluster.aks_cluster: Modifying... [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster]
module.aks.azurerm_kubernetes_cluster.aks_cluster: Modifications complete after 8s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster]

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

Outputs:

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

Migration had been done, below is the backend.tf after Migration.

```
[root@[REDACTED] ~]# cat opentofu-aks-cluster-waf-with-backup-envelope-encryption-azure-keyvault/main/backend.tf
terraform {
  backend "azurerm" {
    resource_group_name = "ritesh"
    storage_account_name = "terraformstate08022024"
    container_name       = "terraform-state"
    key                 = "state/terraform.tfstate"
    subscription_id      = "5[REDACTED]"      ### Provide Subscription ID of your Azure Account.
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

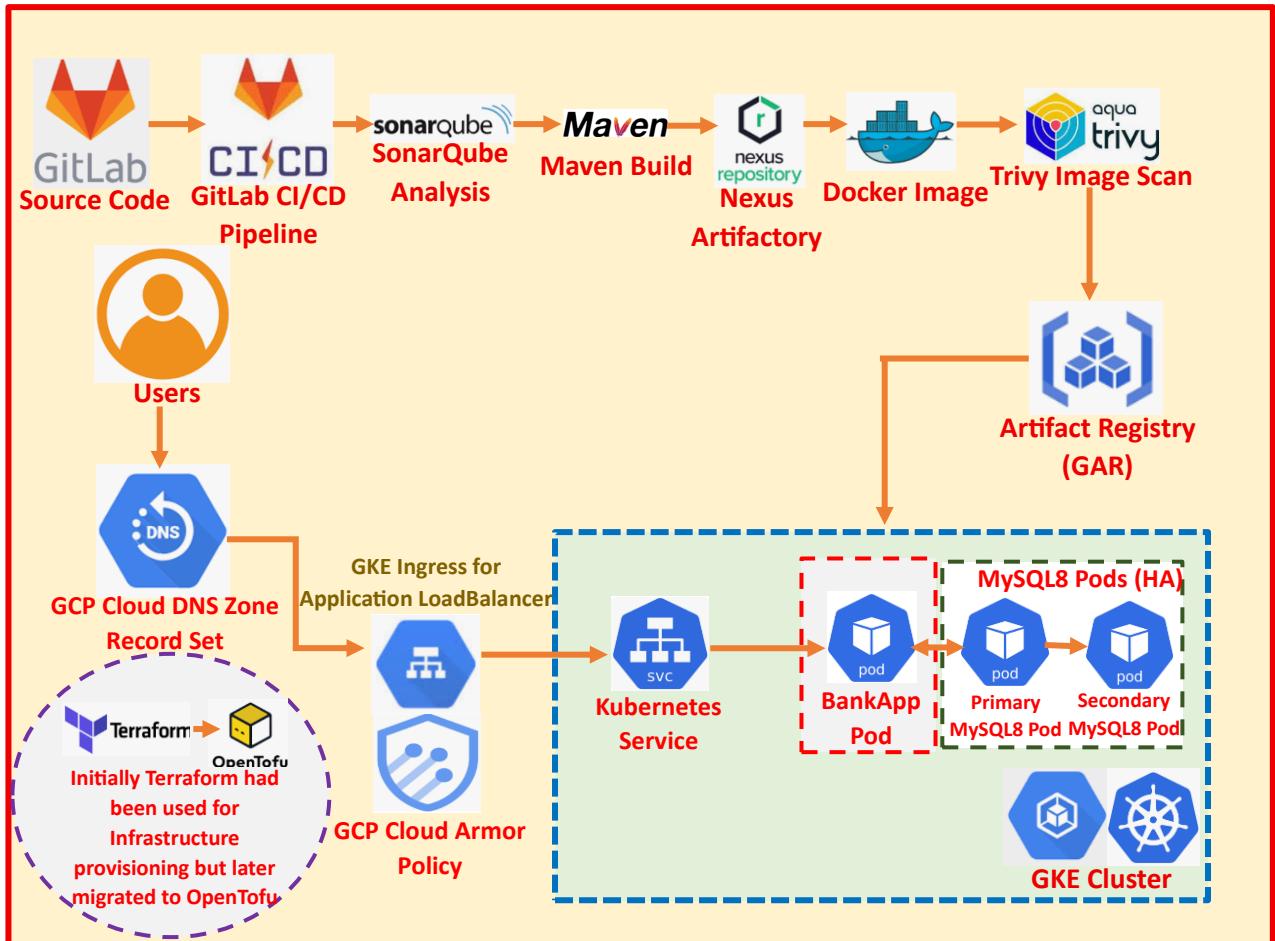
    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

Now check state file in Azure Storage Account Container you will find it is encrypted. However, if you run **tofu show**, you can see the state file but in Azure Storage Account Container it is encrypted.

DevOps Project BankApp Terraform upgrade to OpenTofu with Secure GKE Standard Cluster

Module-3



Till now I used Terraform to provision the infrastructure but Hashicorp license changes due to which Terraform will introduce restrictions for commercial use, due to this reason many organisations are migrating from Terraform to OpenTofu. However, I would like to mention here that DevOps Team who used it as a tool to provision the infrastructure will not be affected but those who provides it as a Service will be affected. You already aware that Terraform state file was kept locally or in the S3 Bucket / Azure Storage Account Container / GCP Bucket. The problem was its state file was not encrypted, so anyone who had the access of Terraform Server (when state file was kept locally) or Bucket or Storage Account Container can easily see its state file. But if you use OpenTofu, you will not face such an issue. OpenTofu is an Open Source, community-driven Infrastructure as Code (IaC) tool, it is a fork of Hashicorp Terraform. I will migrate from Terraform to OpenTofu at the end of Module-3.

This DevOps project deals will creation of infrastructure using Terraform, GitLab was used to keep the source Code and GitLab CI/CD had been used as CI/CD Tool. For Code Analysis SonarQube had been used, the build tool used in this project was Maven, Sonatype Nexus3 was used to keep the Artifacts and Docker Image was pushed to the GCP Artifact Registry (GAR) which was finally deployed to the

Google Kubernetes Engine (GKE) Standard Cluster. The Bank Application Pods was accessed using GKE Ingress for Application Load Balancer and hence using the kubernetes service as shown in the screenshot attached above. I applied GCP Cloud Armor Policy (to Limit the Rate) to the Bank Application URL and to prevent from SQL Injection. As this Bank Application was open to entire world so rate limit and SQL Injection was applied through GCP Cloud Armor Policy so that non-of-the IP Address can access the Bank Application more than 40 times in one minute. There may be the possibility that the Bank Application URL accessed using the Bot (by the Hackers) so that its speed becomes slow or this website hangs to refrain from such a situation GCP Cloud Armor Policy Rate Limit was applied. Finally, I migrated from Terraform to OpenTofu. For GKE Standard Cluster Node-to-Node encryption (in transit) was enabled by setting `in_transit_encryption_config = "IN_TRANSIT_ENCRYPTION_INTER_NODE_TRANSPARENT"` using Terraform.

I implemented Istio for encrypted Pod-to-Pod Communication and implemented **Open Policy Agent Gateway** (Open-Source Policy Engine) and **Falco** to Secure the GKE Standard Cluster. It comes under CNCF Graduated Project and used by Organisation like Allstate, Bloomberg, BNY, CapitalOne, Cisco, Intuit, and Marsh McLennan. The OPA Gatekeeper is a **dynamic admission controller**. I had used Aqua Security Trivy image scan for vulnerability scanning during CI/CD Pipeline as shown in the end-to-end architecture diagram of the project drawn above. In GKE Standard Cluster to enable envelope encryption I used **database_encryption** in Terraform Script. The GCP Customer Managed Encryption Key (CMEK) enables you to rotate it as per the standard followed by your organisation. Usually after 90 days the GCP CMEK Key should be rotated but you can rotate it as per your organisation need. If you use Google Managed Encryption Keys (GMEK) then it will be rotated by Google, however there is no specific number of days after which Google rotates it. In this project I used GCP CMEK for envelope encryption and to encrypt the all the disks used in this project.

In GKE Standard Cluster, however it is possible to encrypt all the disks either by Google Managed Encryption Key (GMEK) or Customer Managed Encryption Key (CMEK) but envelope encryption provides extra layer of security by encrypting the etcd.

For Authentication and Authorization of AWS Account, I installed **aws cli** on GCP VM instance and created an IAM user in AWS account and used its access key and secret key as shown in the screenshot attached below.

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

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

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

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

# User specific aliases and functions

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

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

PATH="$PATH:/usr/local/bin"
[root@terraform-server ~]# echo $PATH
/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/root/bin
[root@terraform-server ~]# aws --version
aws-cli/2.30.7 Python/3.13.7 Linux/4.18.0-553.72.1.el8_10.x86_64 exe/x86_64.rocky.8

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

Terraform script to create the infrastructure is present at the path **terraform-gcp-cloud-dns-aws-acm-certificate** (this terraform script is handy for you to create the GCP Cloud DNS Zone and AWS Certificate Manager SSL Certificate) in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>. Commands as written below had been used to create infrastructure using Terraform.

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

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

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

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

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

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

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

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

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

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

```
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-kas: (pid 31152) 279s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-workhorse: (pid 31913) 20s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: logrotate: (pid 30724) 313s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: nginx: (pid 31923) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: node-exporter: (pid 31934) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgres-exporter: (pid 31981) 14s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgresql: (pid 30904) 290s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: prometheus: (pid 31958) 17s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: puma: (pid 31344) 148s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis: (pid 30770) 307s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis-exporter: (pid 31950) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: sidekiq: (pid 31845) 36s
module.gitlab.null_resource.gitlab_server (remote-exec): run: alertmanager: (pid 31973) 17s; run: log: (pid 31701) 100s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitaly: (pid 31939) 20s; run: log: (pid 30838) 300s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-exporter: (pid 31948) 20s; run: log: (pid 31588) 118s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-kas: (pid 31152) 281s; run: log: (pid 31162) 280s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-workhorse: (pid 31913) 22s; run: log: (pid 31443) 136s
module.gitlab.null_resource.gitlab_server (remote-exec): run: logrotate: (pid 30724) 315s; run: log: (pid 30732) 314s
module.gitlab.null_resource.gitlab_server (remote-exec): run: nginx: (pid 31923) 21s; run: log: (pid 31496) 128s
module.gitlab.null_resource.gitlab_server (remote-exec): run: node-exporter: (pid 31934) 21s; run: log: (pid 31553) 122s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgres-exporter: (pid 31981) 16s; run: log: (pid 31738) 94s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgresql: (pid 30904) 292s; run: log: (pid 30928) 289s
module.gitlab.null_resource.gitlab_server (remote-exec): run: prometheus: (pid 31958) 19s; run: log: (pid 31663) 104s
module.gitlab.null_resource.gitlab_server (remote-exec): run: puma: (pid 31344) 150s; run: log: (pid 31357) 147s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis: (pid 30770) 309s; run: log: (pid 30779) 308s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis-exporter: (pid 31950) 20s; run: log: (pid 31619) 112s
module.gitlab.null_resource.gitlab_server (remote-exec): run: sidekiq: (pid 31845) 38s; run: log: (pid 31386) 141s
module.gitlab.null_resource.gitlab_server: Creation complete after 12m38s [id=XXXXXXXXXXXXXX]
```

```
Outputs:  
  
ec2_private_ip_alb_dns = {  
    "EC2_Instance_GitLab_Server_Private_IP_Address" = "10.10.10.10"  
    "GitLab_ALB_DNS_Name" = "gitlab-10.10.10.10.us-east-2.elb.amazonaws.com"  
}  
  
Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-141a81111/
```

Then I created the other GCP Resources using the terraform script present at the path **terraform-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek** in the GitHub Repo

<https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>. As a part of disaster recovery, I created the Lifecycle Policy to create the EBS backed AMI periodically 11:00 AM and 11:00 PM UTC (twice daily) for SonarQube and Nexus Server, snapshot schedule for gitlab-runner and GKE backup plan to take the backup daily twice a day (11:00 AM UTC and 11:00 PM UTC).

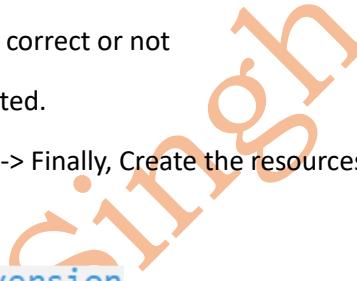
terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

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

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

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

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



```
[root@yellow ~]# gcloud --version
Google Cloud SDK 526.0.0
alpha 2025.06.06
beta 2025.06.06
bq 2.1.18
bundled-python3-unix 3.12.9
core 2025.06.06
gcloud-crc32c 1.0.0
gke-gcloud-auth-plugin 0.5.10
gsutil 5.34
kubectl 1.33.4

[root@yellow ~]# gcloud auth login
You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

You can run:
$ gcloud config set account `ACCOUNT`
to switch accounts if necessary.

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

Do you want to continue (Y/n)? Y
Go to the following link in your browser, and complete the sign-in prompts:
[REDACTED]

Once finished, enter the verification code provided in your browser: [REDACTED]
You are now logged in as [REDACTED].
Your current project is [REDACTED]. You can change this setting by running:
$ gcloud config set project PROJECT_ID
[root@yellow ~]# gcloud container clusters get-credentials bankapp-gke-cluster --region us-central1 --project [REDACTED]
Fetching cluster endpoint and auth data.
kubeconfig entry generated for bankapp-gke-cluster.



| Status                              | Name                | Location    | Number of nodes | Total vCPUs | Total memory | Notifications | Labels               |
|-------------------------------------|---------------------|-------------|-----------------|-------------|--------------|---------------|----------------------|
| <input checked="" type="checkbox"/> | bankapp-gke-cluster | us-central1 | 2               | 4           | 8 GB         |               | goog-terra... : true |



[root@yellow ~]# kubectl get nodes
NAME STATUS ROLES AGE VERSION
gke-bankapp-gke-clus-bankapp-linux-no- Ready <none> 7m51s v1.33.4-gke.1172000
gke-bankapp-gke-clus-bankapp-linux-no- Ready <none> 7m56s v1.33.4-gke.1172000
```

Installation of OPA Gateway Policy Engine

I had installed OPA Gateway as shown in the screenshot attached below.

```
helm repo add gatekeeper https://open-policy-agent.github.io/gatekeeper/charts
helm repo update
helm install gatekeeper/gatekeeper --name-template=gatekeeper --namespace gatekeeper-system --create-namespace
kubectl get pods -n gatekeeper-system
```

```
[root@XXXXXXXXXX ~]# helm repo add gatekeeper https://open-policy-agent.github.io/gatekeeper/charts
"gatekeeper" 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 "gatekeeper" chart repository
Update Complete. ⚡Happy Helm-ing!⚡
[root@XXXXXXXXXX ~]# helm install gatekeeper/gatekeeper --name-template=gatekeeper --namespace gatekeeper-system --create-namespace
NAME: gatekeeper
LAST DEPLOYED: YYYY-MM-DD HH:MM 2025
NAMESPACE: gatekeeper-system
STATUS: deployed
REVISION: 1
TEST SUITE: None

[root@XXXXXXXXXX ~]# kubectl get pods -n gatekeeper-system
NAME                                         READY   STATUS    RESTARTS   AGE
gatekeeper-audit-XXXXXXXXXX                   1/1     Running   0          89s
gatekeeper-controller-manager-XXXXXXXXXX       1/1     Running   0          88s
gatekeeper-controller-manager-XXXXXXXXXX       1/1     Running   0          89s
gatekeeper-controller-manager-XXXXXXXXXX       1/1     Running   0          88s
```

You can check the logs for OPA using the command as shown in the screenshot attached below.

kubectl logs -l control-plane=audit-controller -n gatekeeper-system

```
[root@XXXXXXXXXX ~]# kubectl logs -l control-plane=audit-controller -n gatekeeper-system
{"level": "info", "ts": "1758704970.2340465", "logger": "controller", "msg": "auditing is complete", "process": "audit", "audit_id": "2025-09-24T09:09:30Z", "event_type": "audit_finished", "duration": "29.946014ms"}
{"level": "info", "ts": "1758705030.2034974", "logger": "controller", "msg": "auditing constraints and violations", "process": "audit", "audit_id": "2025-09-24T09:10:30Z", "event_type": "audit_started"}
{"level": "info", "ts": "1758705030.2696376", "logger": "controller", "msg": "no constraint is found with apiversion", "process": "audit", "audit_id": "2025-09-24T09:10:30Z", "constraints_apiversion": "constraints.gatekeeper.sh/v1beta1"}
{"level": "info", "ts": "1758705030.2696836", "logger": "controller", "msg": "auditing is complete", "process": "audit", "audit_id": "2025-09-24T09:10:30Z", "event_type": "audit_finished", "duration": "66.219891ms"}
 {"level": "info", "ts": "1758705090.204344", "logger": "controller", "msg": "auditing constraints and violations", "process": "audit", "audit_id": "2025-09-24T09:11:30Z", "event_type": "audit_started"}
 {"level": "info", "ts": "1758705090.2350106", "logger": "controller", "msg": "no constraint is found with apiversion", "process": "audit", "audit_id": "2025-09-24T09:11:30Z", "constraints_apiversion": "constraints.gatekeeper.sh/v1beta1"}
 {"level": "info", "ts": "1758705090.235044", "logger": "controller", "msg": "auditing is complete", "process": "audit", "audit_id": "2025-09-24T09:11:30Z", "event_type": "audit_finished", "duration": "30.736843ms"}
 {"level": "info", "ts": "1758705150.2046235", "logger": "controller", "msg": "auditing constraints and violations", "process": "audit", "audit_id": "2025-09-24T09:12:30Z", "event_type": "audit_started"}
 {"level": "info", "ts": "1758705150.2046235", "logger": "controller", "msg": "no constraint is found with apiversion", "process": "audit", "audit_id": "2025-09-24T09:12:30Z", "constraints_apiversion": "constraints.gatekeeper.sh/v1beta1"}
 {"level": "info", "ts": "1758705150.2507737", "logger": "controller", "msg": "auditing is complete", "process": "audit", "audit_id": "2025-09-24T09:12:30Z", "event_type": "audit_finished", "duration": "46.185006ms"}
```

Here, I created OPA Policy (Constraint Template and Constraint) which will disallow the creation of pod with container had the capability of host machine (Privileged = true) and elevated privileges (allowPrivilegeEscalation = true) in the namespace bankapp and mysql, and no one can create the LoadBalancer Service in the namespace bankapp and mysql. If anyone wants to create the service, then create the ClusterIP service and expose the pods using Ingress Rule through the created ClusterIP service. It will also not allow anyone to use the Image with latest tag and do not allow to create the Resources Pod, Deployment, Statefulset or Daemonset in the namespace mysql and bankapp if require resource requests and limits per Pod is not mentioned.

```
[root@yellow ~]# kubectl apply -f constrainttemplate.yaml
constrainttemplate.templates.gatekeeper.sh/k8sdisallowprivileged created
constrainttemplate.templates.gatekeeper.sh/loadbalancerconstraint created
constrainttemplate.templates.gatekeeper.sh/k8scontainerresourcerequirements created
constrainttemplate.templates.gatekeeper.sh/k8sdisallowedtags created
[root@yellow ~]# kubectl apply -f constraint.yaml
k8sdisallowprivileged.constraints.gatekeeper.sh/psp-privileged-container created
loadbalancerconstraint.constraints.gatekeeper.sh/loadbalancerconstraint created
k8scontainerresourcerequirements.constraints.gatekeeper.sh/pod-resource-requirements created
k8sdisallowedtags.constraints.gatekeeper.sh/container-image-must-not-have-latest-tag created
```

You can list the constraint template and constraint currently deployed and used in the cluster using the command **kubectl get constrainttemplate** and **kubectl get constraints** respectively.

```
[root@yellow ~]# kubectl get constrainttemplate
NAME          AGE
k8scontainerresourcerequirements  2m6s
k8sdisallowedtags      2m5s
k8sdisallowprivileged    2m6s
loadbalancerconstraint    2m6s
[root@yellow ~]# kubectl get constraint
NAME                                     ENFORCEMENT-ACTION  TOTAL-VIOLATIONS
k8scontainerresourcerequirements.constraints.gatekeeper.sh/pod-resource-requirements  deny      0
NAME                                     ENFORCEMENT-ACTION  TOTAL-VIOLATIONS
k8sdisallowedtags.constraints.gatekeeper.sh/container-image-must-not-have-latest-tag  deny      0
NAME                                     ENFORCEMENT-ACTION  TOTAL-VIOLATIONS
k8sdisallowprivileged.constraints.gatekeeper.sh/psp-privileged-container    deny      0
NAME                                     ENFORCEMENT-ACTION  TOTAL-VIOLATIONS
loadbalancerconstraint.constraints.gatekeeper.sh/loadbalancerconstraint      deny      0
```

The constrainttemplate.yaml and constraint.yaml files used are same as that was used and provided in **Module-1**.

Installation of Falco in GKE Standard Cluster

Falco is a tool that captures and analyses logs within a runtime environment to detect suspicious and malicious activity. I had installed here Falco with the custom rule that if anyone will try to access the /etc directory to write in a file then its logs will be captured by the Falco.

```
cat dexter-falco-rule.yaml
customRules:
  custom-rules.yaml: |-
    - rule: Write below etc
      desc: An attempt to write to /etc directory
      condition: >
        (evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)
        and fd.name startswith /etc
      output: "File below /etc opened for writing | file=%fd.name pc cmdline=%proc.pc cmdline
gparent=%proc.pname[2] gparent=%proc.pname[3] gparent=%proc.pname[4] evt_type=%evt.type
user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name
proc_exepath=%proc.exepath parent=%proc.pname command=%proc.cmdline terminal=%proc.tty"
      priority: WARNING
      tags: [filesystem, mitre_persistence]
```

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
```

```
helm repo update
```

```
helm upgrade --install falco falcosecurity/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcosidekick.enabled=true --set falcosidekick.webui.enabled=true --set falcosidekick.webui.redis.storageEnabled=false
```

```
kubectl get pods -n falco --watch
```

```
[root@yellow ~]# helm repo add falcosecurity https://falcosecurity.github.io/charts
"falcosecurity" has been added to your repositories
[root@yellow ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "gatekeeper" chart repository
...Successfully got an update from the "falcosecurity" chart repository
Update Complete. Happy Helming!
```

```
[root@yellow ~]# cat dexter-falco-rule.yaml
customRules:
  custom-rules.yaml: |-
    - rule: Write below etc
      desc: An attempt to write to /etc directory
      condition: >
        (evt.type in (openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)
        and fd.name startsWith /etc
      output: "File below /etc opened for writing | file=%fd.name pcmdline=%proc.cmdline gparent=%proc.parent pname[%2] gparent=%proc.parent name[%3] ggpname=%proc.parent name[%4] evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.parent command=%proc.cmdline terminal=%proc.tty"
      priority: WARNING
      tags: [filesystem, mitre_persistence]

[root@yellow ~]# helm upgrade --install falco falcosecurity/falco -f dexter-falco-rule.yaml --namespace falco --create-namespace --set falcosidekick.enabled=true --set falcosidekick.webui.enabled=true --set falcosidekick.webui.redis.storageEnabled=false
```

I had created Ingress Rule for Falco as shown in the screenshot attached below.

```
[root@yellow ~]# kubectl create secret tls ingress-secret --key server.key --cert fyellowa.crt -n falco
secret/ingress-secret created
[root@yellow ~]# cat falco-ingress-rule.yaml
# kubectl create secret tls ingress-secret --key server.key --cert fyellowa.crt -n falco
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    kubernetes.io/ingress.class: "gce" ### Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
spec:
  ingressClassName: "gce"
  tls:
  - hosts:
    - falco.singhritesh85.com
      secretName: ingress-tls
  rules:
  - host: "falco.singhritesh85.com"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: falco-falcosidekick-ui
            port:
              number: 2802
[root@yellow ~]# kubectl apply -f falco-ingress-rule.yaml
ingress.networking.k8s.io/falco-ui-ingress created

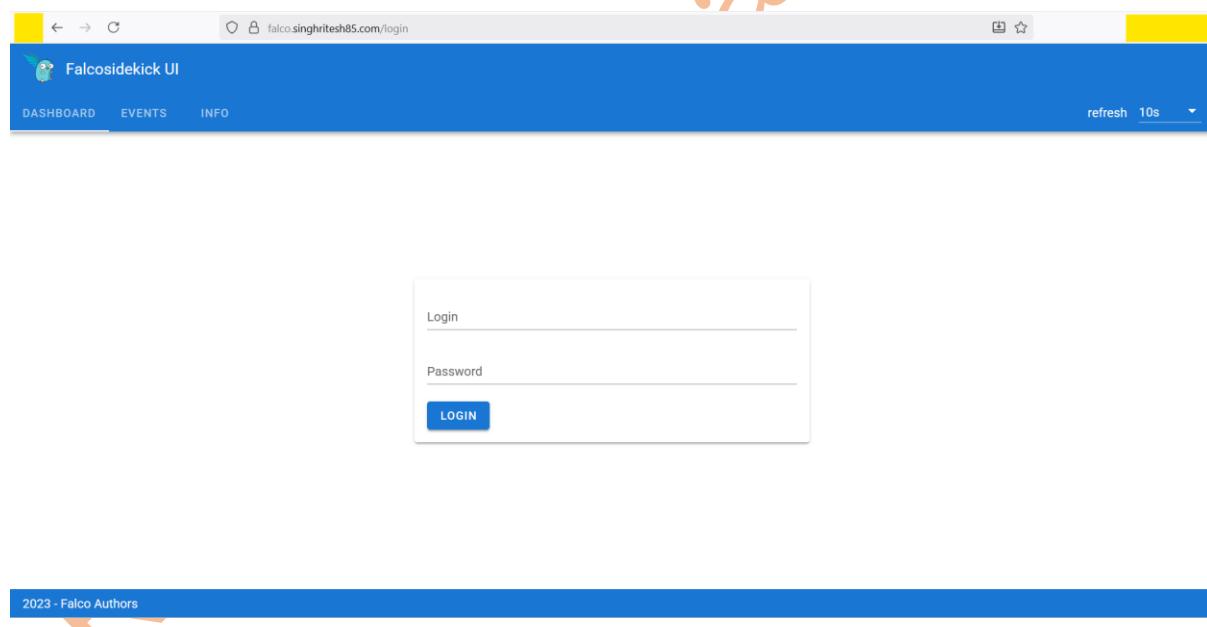
[root@yellow ~]# kubectl get ing -A
NAMESPACE     NAME           CLASS      HOSTS           ADDRESS          PORTS      AGE
falco        falco-ui-ingress   gce       falco.singhritesh85.com  34.yellow  80, 443   2m58s
```

I did the entry for Ingress HOSTS with External IP Address in GCP Cloud DNS Zone to create the Record Set of A-Type as shown in the screenshot attached below.

Network Services / Zones / Zone: bankapp-public-zone / Create record set

<ul style="list-style-type: none"> + Load balancing + Cloud DNS + Cloud CDN + Cloud NAT + Cloud Service Mesh (Traffic Manager) + Service Directory + Cloud Domains + Private Service Connect + SSL policies + Service Extensions <hr/> + Marketplace	< Create record set DNS name: falco .singhrites85.com. ? Resource record type: A ? TTL *: 5 ? TTL unit: minutes ? IPv4 Address ? IPv4 Address 1 *: 34. ? Select + Add item Create Cancel
---	--

Finally, I was able to access Falco UI as shown in the screenshot attached below. You can login with username and password admin and admin respectively.



It is also possible to check falco logs using the command

```
kubectl logs -l "app.kubernetes.io/instance=falco" -n falco
```

```
cat falco-ingress-rule.yaml

# kubectl create secret tls ingress-tls --key server.key --cert XXXXXXXXXXXXXXXXX.crt -n falco

---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: falco-ui-ingress
  namespace: falco
  annotations:
    kubernetes.io/ingress.class: "gce" ##### Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
spec:
  ingressClassName: "gce"
  tls:
    - hosts:
        - falco.singhritesh85.com
      secretName: ingress-tls
  rules:
    - host: "falco.singhritesh85.com"
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: falco-falcosidekick-ui
              port:
                number: 2802
```

Installation of Istio in GKE Standard Cluster using Helm

To install Istio Service mesh using helm chart use the command as written below.

```
helm repo add istio https://istio-release.storage.googleapis.com/charts
```

```
helm repo update
```

```
kubectl create ns istio-system
```

```
helm install istio-base istio/base -n istio-system
```

```
helm install istiod istio/istiod -n istio-system
```

```
[root@yellow ~]# helm repo add istio https://istio-release.storage.googleapis.com/charts
"istio" has been added to your repositories
[root@yellow ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "falcosecurity" chart repository
...Successfully got an update from the "istio" chart repository
...Successfully got an update from the "kyverno" chart repository
Update Complete. Happy Helming!
[root@yellow ~]# kubectl create ns istio-system
namespace/istio-system created

[root@yellow ~]# helm install istio-base istio/base -n istio-system
[root@yellow ~]# helm install istiod istio/istiod -n istio-system
```

To establish mesh-wide mTLS I created the peer authentication as shown in the screenshot attached below.

```
cat peerauthentication.yaml
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata:
  name: "default"
  namespace: istio-system
spec:
  mtls:
    mode: STRICT
```

```
[root@XXXXXXXXXX ~]# cat peerauthentication.yaml
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata:
  name: "default"
  namespace: istio-system
spec:
  mtls:
    mode: STRICT
[root@XXXXXXXXXX ~]# kubectl apply -f peerauthentication.yaml
peerauthentication.security.istio.io/default created

[root@XXXXXXXXXX ~]# kubectl get peerauthentication -n istio-system
NAME      MODE     AGE
default   STRICT   38s
```

At this stage GKE Standard Cluster is Ready, Now I will do the entry for GitLab LoadBalancer DNS Name, Sonatype Nexus LoadBalancer DNS Name and SonarQube LoadBalancer DNS Name in the GCP Cloud DNS Zone to create the three Record Set of CNAME Type as shown in the screenshot attached below.

Create record set



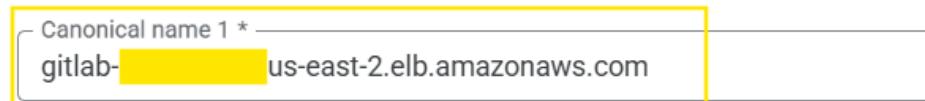
DNS name
gitlab .singhritesh85.com. 

Resource record type
CNAME 

TTL *
5 

TTL unit
minutes 

Canonical name



Canonical name 1 *
gitlab-[REDACTED]us-east-2.elb.amazonaws.com 

Example: server-1.example.com.

 + Add item

 Create

Cancel

DNS name ?

Resource record type ▼ ?

TTL * ?

TTL unit ▼ ?

Canonical name ?

Canonical name 1 * ?

Example: server-1.example.com.

+ Add item

Save Cancel

DNS name ?

Resource record type ▼ ?

TTL * ?

TTL unit ▼ ?

Canonical name ?

Canonical name 1 * ?

Example: server-1.example.com.

+ Add item

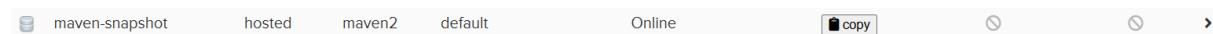
Save Cancel

After creating CNAME Type Record Set in GCP Cloud DNS Zone I was able to access GitLab, Nexus Artifactory and SonarQube as shown in the screenshot attached below.

The image contains three vertically stacked screenshots of web interfaces:

- Top Screenshot:** A GitLab Enterprise Edition login page. The URL in the address bar is `gitlab.singhritesh85.com/users/sign_in`. The page features a logo, fields for 'Username or primary email' and 'Password', a 'Remember me' checkbox, a 'Sign In' button, and a link to 'Forgot your password?'. Below the sign-in form is a link to 'Register now'.
- Middle Screenshot:** A Sonatype Nexus Repository OSS 3.68.1-02 interface. The URL is `nexus.singhritesh85.com/#browse/welcome`. It shows a 'Welcome' banner with links for 'Release Notes', 'Documentation', and 'Community'. A central 'Notification Center' box displays a 'Sign In' dialog over a message about a 'Nexus Repository Cloud' promotional offer. A notification bubble indicates 'You have 4 notifications to review'. The top navigation bar includes links for 'Explore', 'Help', 'About GitLab', and 'GitLab community forum'.
- Bottom Screenshot:** A SonarQube login page. The URL is `sonarqube.singhritesh85.com/sessions/new?return_to=%2F`. It features a 'Log in to SonarQube' header, fields for 'Login' and 'Password', and 'Log in' and 'Cancel' buttons. At the bottom, it includes a footer note: 'SonarQube™ technology is powered by SonarSource SA' and links for 'LGPL v3', 'Community', 'Documentation', and 'Plugins'.

I had created a Repository in Sonatype Nexus with the name **maven-snapshot** as shown in the screenshot attached below.



For first time login-into the GitLab UI you need root password which was present at the path **/etc/gitlab/initial_root_password** on the GitLab Server (EC2 Instance) as shown in the screenshot attached below.

```
[root@gitlab-server ~]# cat /etc/gitlab/initial_root_password
# WARNING: This value is valid only in the following conditions
#   1. If provided manually (either via `GITLAB_ROOT_PASSWORD` environment variable or via `gitlab_rails['initial_root_password']` setting in `gitlab.r
b`, it was provided before database was seeded for the first time (usually, the first reconfigure run).
#   2. Password hasn't been changed manually, either via UI or via command line.
#
#       If the password shown here doesn't work, you must reset the admin password following https://docs.gitlab.com/ee/security/reset_user_password.html#r
eset-your-root-password.

Password: [REDACTED]

# NOTE: This file will be automatically deleted in the first reconfigure run after 24 hours.
```

After login for the first time, I changed the admin password.

Configure GitLab Runner as shown in the screenshot attached below.

```
[root@bankapp-gitlab-runner ~]# gitlab-runner register
Runtime platform                                arch=amd64 os=linux pid=[REDACTED] revision=[REDACTED] version=18.4.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.singhritesh85.com
Enter the registration token:
[REDACTED]
Verifying runner... is valid correlation_id=[REDACTED] runner=[REDACTED]
Enter a name for the runner. This is stored only in the local config.toml file:
[bankapp-gitlab-runner]: gitlab-runner-1
Enter an executor: custom, shell, ssh, virtualbox, docker-windows, docker+machine, kubernetes, docker-autoscaler, parallels, docker, instance: shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
```

Now you can see Runner had been configured as shown in the screenshot attached below.

All	Instance	Group	Project
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Search or filter results...			
Online	Offline	Stale	
1	0	0	
<input type="checkbox"/>	Status	Runner configuration	Owner
<input type="checkbox"/>	Online Idle	#1 [REDACTED] [REDACTED] Instance 0 0 Last contact: 1 minute ago 10 0 Created by Administrator 7 minutes ago gitlab-runner-1	Administrator

Then open the file **/etc/gitlab-runner/config.toml** and configured clone_url = "https://gitlab.singhritesh85.com" as shown in the screenshot attached below.

```
[root@bankapp-gitlab-runner ~]# vim /etc/gitlab-runner/config.toml
```

```

concurrent = 1
check_interval = 0
connection_max_age = "15m0s"
shutdown_timeout = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "gitlab-runner-1"
  url = "https://gitlab.singhritesh85.com"
  clone_url = "https://gitlab.singhritesh85.com"
  id = 1
  token = "XXXXXXXXXXXXXX"
  token_obtained_at = 2025
  token_expires_at =
  executor = "shell"
[runners.cache]
  MaxUploadedArchiveSize = 0
  [runners.cache.s3]
  [runners.cache.gcs]
  [runners.cache.azure]

```

I created two Repository in GitLab named as **kustomize-bankapp** and then pushed the source code and Kustomize Manifests Yaml files to the GitLab Repository as shown in the screenshot attached below.

```

[root@XXXXXX ~]# cd demo/
[root@XXXXXX demo]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /root/demo/.git/
[root@XXXXXX demo]# git config --local user.name "Administrator"
[root@XXXXXX demo]# git config --local user.email "XXXXXXXXXXXXXX@gmail.com"
[root@XXXXXX demo]# git remote add origin https://XXXXXXXXXXXXXX@gitlab.singhritesh85.com/dexter/kustomize-bankapp.git
[root@XXXXXX ~]# git clone --branch dev https://github.com/singhritesh85/kustomize-bankapp.git
Cloning into 'kustomize-bankapp'...
remote: Enumerating objects: 199, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 199 (delta 2), reused 0 (delta 0), pack-reused 193 (from 1)
Receiving objects: 100% (199/199), 44.83 KiB | 882.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.
[root@XXXXXX ~]# cp -r kustomize-bankapp/* demo/
[root@XXXXXX ~]# cd demo/
[root@XXXXXX demo]# git checkout -b dev
Switched to a new branch 'dev'
[root@XXXXXX demo]# git add .
[root@XXXXXX demo]# git commit -m "demo"
[dev (root-commit) 0 8] demo
 10 files changed, 149 insertions(+)
 create mode 1 4 README.md

```

```

[root@XXXXXX demo]# git push origin dev
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (18/18), 2.58 KiB | 880.00 KiB/s, done.
Total 18 (delta 1), reused 0 (delta 0), pack-reused 0
To https://gitlab.singhritesh85.com/dexter/kustomize-bankapp.git

```

dev kustomize-bankapp

kustomize-bankapp

Delete .gitlab-ci.yml
Administrator authored just now

+ Find file Code ...

bcaa2a56 History

Name	Last commit	Last update
base	demo	4 hours ago
overlay	demo	4 hours ago
README.md	demo	4 hours ago

```
[root@] # mkdir bolo
[root@] # cd bolo/
[root@] # git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /root/bolo/.git/
[root@] # git config --local user.name "Administrator"
[root@] # git config --local user.email "[REDACTED]@gmail.com"
[root@] # git remote add origin https://[REDACTED]@gitlab.singhritesh85.com/dexter/bank-app-kustomize.git
[root@] # cd
[root@] # git clone --branch main https://github.com/singhritesh85/Bank-App-Kustomize.git
Cloning into 'Bank-App-Kustomize'...
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 51 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (51/51), 21.62 KiB | 3.09 MiB/s, done.
Resolving deltas: 100% (5/5), done.
[root@] # cp -r Bank-App-Kustomize/* bolo/
[root@] # cd bolo/
[root@] # git checkout -b dev
Switched to a new branch 'dev'
[root@] # git add .
[root@] # git commit -m "dexter"
```

[root@] bolo]# git push origin dev

Enumerating objects: 42, done.

Counting objects: 100% (42/42), done.

Delta compression using up to 2 threads

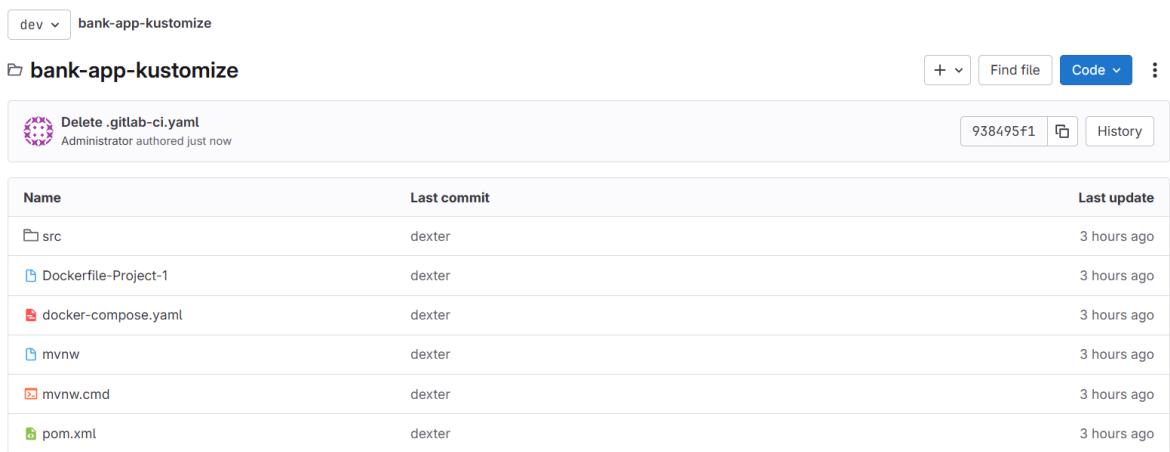
Compressing objects: 100% (29/29), done.

Writing objects: 100% (42/42), 18.13 KiB | 2.01 MiB/s, done.

Total 42 (delta 2), reused 0 (delta 0), pack-reused 0

To https://gitlab.singhritesh85.com/dexter/bank-app-kustomize.git

* [new branch] dev -> dev



The screenshot shows a GitLab repository interface. At the top, there's a dropdown menu set to 'dev' and a search bar containing 'bank-app-kustomize'. Below the search bar, a message indicates a file named '.gitlab-ci.yaml' has been deleted by an administrator. To the right, there are buttons for '+', 'Find file', 'Code', and a three-dot menu. A commit hash '938495f1' and a 'History' button are also visible. The main area displays a table of files with columns for Name, Last commit, and Last update. All files listed were last updated 3 hours ago by 'dexter'.

Name	Last commit	Last update
src	dexter	3 hours ago
Dockerfile-Project-1	dexter	3 hours ago
docker-compose.yaml	dexter	3 hours ago
mvnw	dexter	3 hours ago
mvnw.cmd	dexter	3 hours ago
pom.xml	dexter	3 hours ago

Then I created the storage class as shown in the screenshot attached below.



The terminal session shows the creation of a StorageClass named 'dexter'. It starts with the command 'cat storage-class.yaml' which displays the YAML configuration. This is followed by 'kubectl apply -f storage-class.yaml' which creates the StorageClass. Finally, 'kubectl get sc' is run to list the StorageClasses, where 'dexter' is shown with its details: PROVISIONER, RECLAIMPOLICY, VOLUMEBINDINGMODE, ALLOWVOLUMEEXPANSION, and AGE.

```
[root@XXXXXXXXXX ~]# cat storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: dexter
provisioner: pd.csi.storage.gke.io
parameters:
  type: pd-standard
  disk-encryption-kms-key: projects/XXXX-XXXXXXX-XXXXXX/locations/us-central1/keyRings/dexter-gke-key-ring/cryptoKeys/bankapp-gke-disk-encryption-key
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
[root@XXXXXXXXXX ~]# kubectl apply -f storage-class.yaml
storageclass.storage.k8s.io/dexter created
[root@XXXXXXXXXX ~]# kubectl get sc
NAME      PROVISIONER      RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
dexter   pd.csi.storage.gke.io   Delete        WaitForFirstConsumer   true               7s
```

cat storage-class.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: dexter
provisioner: pd.csi.storage.gke.io
parameters:
  type: pd-standard
  disk-encryption-kms-key: projects/XXXX-XXXXXXX-XXXXXX/locations/us-central1/keyRings/dexter-gke-key-ring/cryptoKeys/bankapp-gke-disk-encryption-key
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
```

After Creation of Storage Class, I had created the two namespaces **mysql** and **bankapp** then I had created the first GitLab CI/CD Pipeline to deploy MySQL 8 Pods as shown in the screenshot attached below.

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

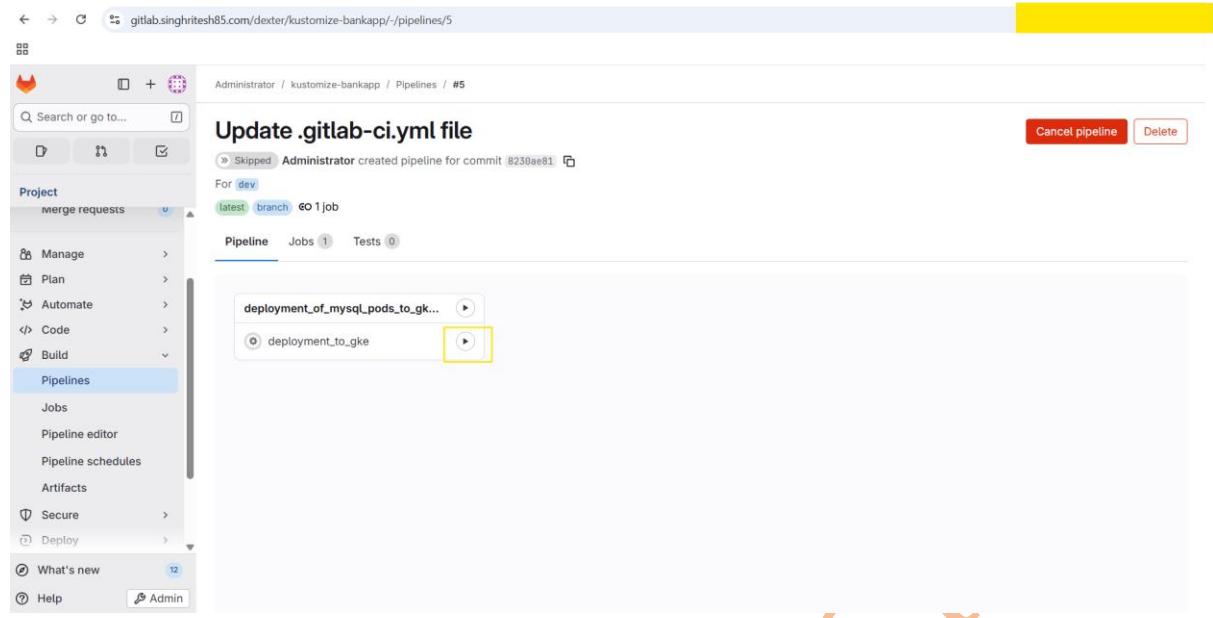
Go to the GitLab Repository kustomize-bankapp > Build > Pipeline editor > Configure Pipeline and provide the details to create .gitlab-ci.yml. For your reference this file present in GitHub Repo <https://github.com/singhrithesh85/kustomize-bankapp.git> (dev branch).

```

1 default:
2   tags:
3     - gitlab-runner-1
4
5   stages:
6     - deployment_of_mysql_pods_to_gke_cluster
7
8   deployment_to_gke:
9     stage: deployment_of_mysql_pods_to_gke_cluster
10    script:
11      - kubectl apply -k overlay/mysql
12      - kubectl get pods -n mysql
13    when: manual
14
15   only:
16     - dev

```

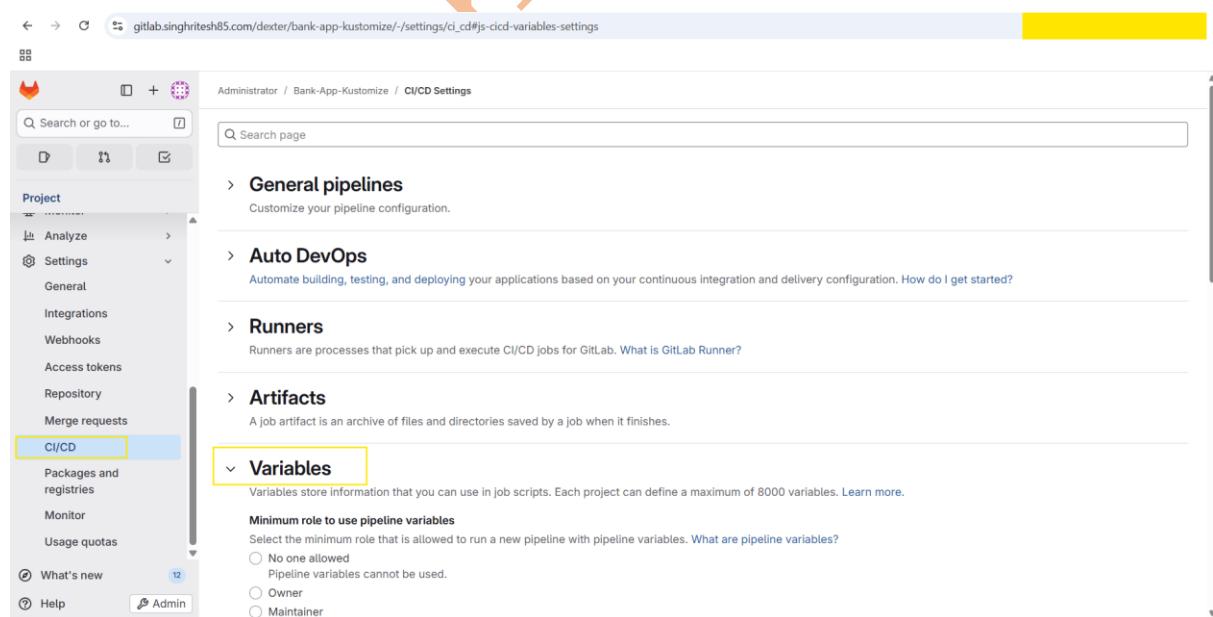
Then Go to the Pipelines and Run the Pipeline as shown in the screenshot attached below.



```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          77s
mysql-1   1/1     Running   0          63s
mysql-2   1/1     Running   0          48s
```

After successful deployment of all the MySQL 8 Pods, I created the GitLab CI/CD Pipeline to deploy Bank Application Pods as shown below, but before that I create three variables named as SONARQUBE_TOKEN, GITLAB_USER, GITLAB_PASSWORD and HELM_REPO_FOR_GKE_ACCESS_TOKEN as shown in the screenshot attached below.

Go to GitLab Repo bank-app-kustomize > Settings > CICD > Variables as show below.



The screenshot shows two nearly identical configurations for a CI/CD variable named 'SONARQUBE_TOKEN' on the 'CI/CD Variables' settings page.

Common Configuration:

- Key:** SONARQUBE_TOKEN
- Value:**
- Environment:** All (default)
- Flags:**
 - Masked**: Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.
 - Masked and hidden**: Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.
 - Protect variable**: Export variable to pipelines running on protected branches and tags only.
 - Expand variable reference**: \$ will be treated as the start of a reference to another variable.
- Description (optional):** The description of the variable's value or usage.
- Key:** SONARQUBE_TOKEN
- Value:**

Differences between the two configurations:

- Configuration 1 (Top):** Has a note: "Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help prevent this by hiding the value from logs and pipelines. It's important to note that while masked variables are not guaranteed to be completely secure, they provide an additional layer of protection for sensitive data."
- Configuration 2 (Bottom):** Has a note: "You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?"

The screenshot shows two nearly identical screenshots of the GitLab CI/CD Settings page for a project named "Bank-App-Kustomize". Both screenshots are from the same URL: gitlab.singhritesh85.com/dexter/bank-app-kustomize/-/settings/ci_cd#js-cicd-variables-settings.

Project Variables:

- SONARQUBE_TOKEN** (Protected, Masked, Expanded): Value is masked as '.....'.
- NEXUS_USER** (Protected, Expanded): Value is masked as '.....'.

Flags:

- Visible**: Can be seen in job logs.
- Masked**: Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.
- Masked and hidden**: Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

Project variables description:

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help protect sensitive values, but is not a guaranteed method to prevent malicious users from accessing variables. How can I make my variables safe?

Pipeline trigger tokens:

Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a user.

Deploy freezes:

Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the deployment freeze added here. Learn more. Specify deploy freezes using cron syntax.

The screenshot shows two screenshots of the GitLab CI/CD Settings page. The top screenshot displays 'CI/CD Variables' with three entries: NEXUS_PASSWORD, NEXUS_USER, and SONARQUBE_TOKEN, all set to masked values. The bottom screenshot shows the same variables listed under 'CI/CD Variables' and includes sections for 'Pipeline trigger tokens' and 'Deploy freezes'.

Key	Value	Environments	Actions
NEXUS_PASSWORD	All (default)	
NEXUS_USER	All (default)	
SONARQUBE_TOKEN	All (default)	

Then Authenticate GAR (GCP Artifact Registry) as shown in the screenshot attached below.

```
[gitlab-runner@bankapp-gitlab-runner ~]$ gcloud auth configure-docker us-central1-docker.pkg.dev
Adding credentials for: us-central1-docker.pkg.dev
After update, the following will be written to your Docker config file located at [/home/gitlab-runner/.docker/config.json]:
{
  "credHelpers": {
    "us-central1-docker.pkg.dev": "gcloud"
  }
}

Do you want to continue (Y/n)? Y
Docker configuration file updated.
```

Go to the GitLab Repository **bank-app-kustomize** > Build > Pipeline editor > Configure Pipeline and provide the details to create `.gitlab-ci.yml`. For your reference this file present in GitHub Repo [https://github.com/singhritesh85/Bank-App-Kustomize.git \(main branch\)](https://github.com/singhritesh85/Bank-App-Kustomize.git (main branch)).

The screenshot shows two views of the GitLab Pipeline editor for a project named 'Bank-App-Kustomize'.

Top View: Shows a summary message: "Optimize your workflow with CI/CD Pipelines". It includes a button to "Configure pipeline".

Bottom View: Shows a pipeline named "running" for commit "28c1bfac". The status is "Pipeline syntax is correct. Learn more". Below is the pipeline configuration code:

```

1 default:
2   tags:
3     - gitlab-runner-1
4
5 stages:
6   - build_and_sonarqube_analysis
7   - nexus_artifact_upload
8   - docker_image_build
9   - deployment_to_gke_cluster
10
11 build_job:
12   stage: build_and_sonarqube_analysis
13   before_script:
14     - export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.16.0.8-2.el8.x86_64"
15     - export PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
16   script:
17     - echo "Building artifact..."
18     - mvn clean install sonar:sonar -Dsonar.qualitygate.wait=true -Dsonar.qualitygate.timeout=600 -Dsonar.projectKey=bankapp -Dsonar.projectName=Bank-App-Kustomize
19   artifacts:
20     paths:

```

Below the code, a terminal window shows the command: `[gitlab-runner@[REDACTED] ~]$ kubectl get pods -n bankapp`. The output is:

NAME	READY	STATUS	RESTARTS	AGE
bankapp-[REDACTED]	1/1	Running	0	10m

Created Kubernetes Secret ingress-tls and Ingress Rule as shown in the screenshot attached below.

kubectl create secret tls ingress-tls --key server.key --cert 1XXXXXXXXXXXXXX.crt --namespace bankapp

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl create secret tls ingress-tls --key server.key --cert [REDACTED].crt --namespace bankapp
secret/ingress-tls created
```

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

```
cat backendconfig-bankapp.yaml
```

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

```
kubectl get backendconfig -n bankapp
```



```
[gitlab-runner@[REDACTED] ~]$ kubectl edit svc bankapp-service -n bankapp
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    cloud.google.com/neg: '{"ingress":true}'
    cloud.google.com/neg-status: '{"network_endpoint_groups":{"80":"[REDACTED"]}}, "zones":["us-central1-a", "us-central1-c"]'
    cloud.google.com/backend-config: '{"default": "bankapp-gke-cloud-armor-policy"}'
    kubernetes.io/last-applied-configuration: |
      {"apiVersion": "v1", "kind": "Service", "metadata": {"annotations": {}, "name": "bankapp-service", "namespace": "bankapp"}, "spec": {"ports": [{"port": 80, "targetPort": 8080}], "selector": {"app": "bankapp"}, "type": "ClusterIP"}}
    creationTimestamp: "2025-01-11T10:25:15Z"
  name: bankapp-service
  namespace: bankapp
  resourceVersion: "[REDACTED]"
  uid: [REDACTED]
spec:
  clusterIP: [REDACTED]
  clusterIPs:
  - [REDACTED]
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: bankapp
    sessionAffinity: None
    type: ClusterIP
  status:
    loadBalancer: {}
```

```
[gitlab-runner@bankapp-gitlab-runner ~]$ cat ingress-rule.yaml
# kubectl create secret tls ingress-tls --key server.key --cert 1XXXXXXXXXXXXXX.crt --namespace bankapp
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dexter-ingress
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: "gce" # Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
#    cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'
spec:
  ingressClassName: "gce"
  tls:
  - hosts:
    - bankapp.singhritesh85.com
    secretName: ingress-tls
  rules:
  - host: "bankapp.singhritesh85.com"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: bankapp-service
            port:
              number: 80
```

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

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get ing -n bankapp --watch
NAME      CLASS   HOSTS           ADDRESS      PORTS      AGE
dexter-ingress   gce   bankapp.singhritesh85.com  34.123.123.123  80, 443  5m47s
```

Ritesh Kumar Singh

```

cat ingress-rule.yaml

# kubectl create secret tls ingress-tls --key server.key --cert 1XXXXXXXXXXXXXX.crt --namespace
bankapp

---

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

```

I had done the entry of HOSTS and EXTERNAL IP Address in the GCP Cloud DNS Zone to create the Record Set of A-Type as shown in the screenshot attached below.

[Create record set](#)

DNS name [?](#)

Resource record type [?](#)

TTL * [?](#)

TTL unit [?](#)

IPv4 Address [?](#)

IPv4 Address 1 * [Select](#)

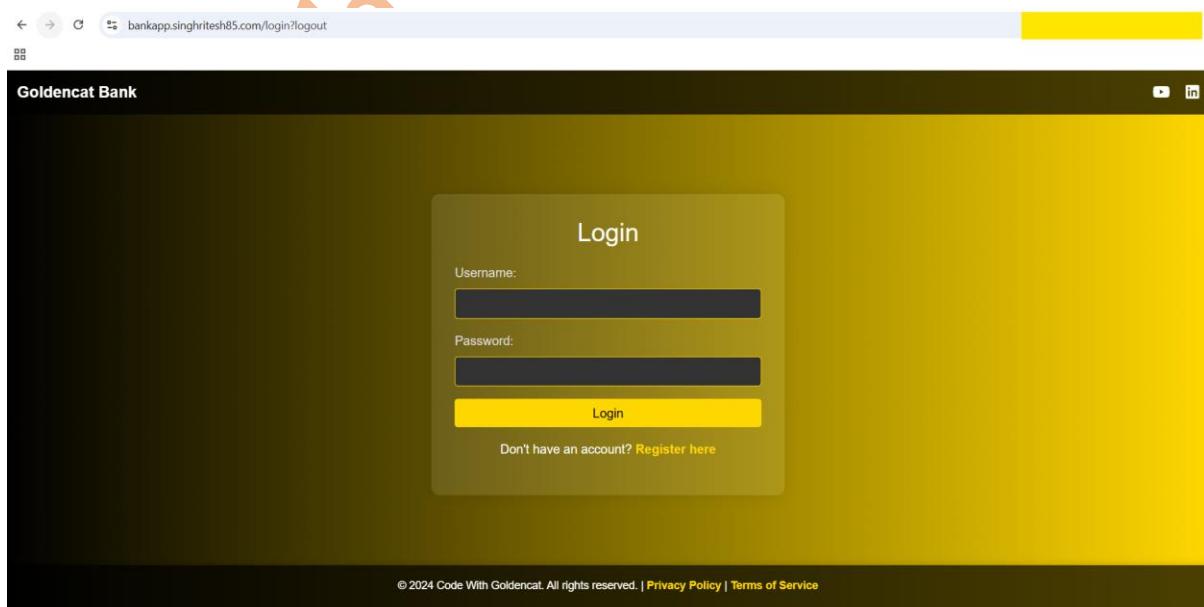
Example: 192.0.2.91

[+ Add item](#)

[Create](#)

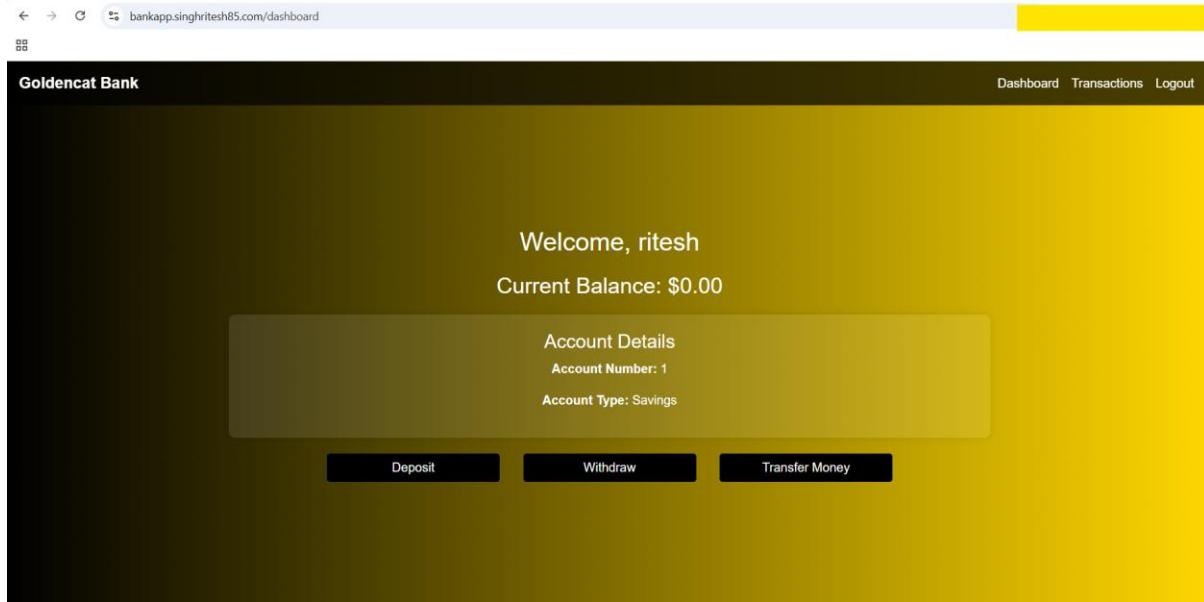
[Cancel](#)

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



I registered with a user and I was able to login successfully with same registered user as shown in the screenshot attached below.

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



Migration from Terraform to Opentofu

As mentioned at the starting of the project that lastly, I will migrate from Terraform to Opentofu. So, in this section I am explaining how to migrate from Terraform to Opentofu.

To Migrate from Terraform to Opentofu make sure about below mentioned points.

1. Take the backup of state file
2. Take the backup of existing Terraform Script.

For backing up terraform state file I created a new GCP bucket and copied the existing state file to the new bucket

```
[root@[REDACTED] ~]# gsutil mb -l US-CENTRAL1 gs://dexter-therema
Creating gs://dexter-therema/...
[root@terraform-server ~]# gcloud storage ls
gs://dexter-therema/
gs://dolo-dempo/
[root@terraform-server ~]# gcloud storage cp --recursive gs://dolo-dempo/* gs://dexter-therema/
Copying gs://dolo-dempo/state/cloud-dns/default.tfstate to gs://dexter-therema/state/cloud-dns/default.tfstate
Copying gs://dolo-dempo/state/gke-standard/terraform.tfstate/default.tfstate to gs://dexter-therema/state/gke-standard/terraform.tfstate/default.tfstate
Completed files 2/2 | 222.1kiB/222.1kiB
Average throughput: 14.9MiB/s
[root@[REDACTED] ~]# gcloud storage ls gs://dexter-therema/
gs://dexter-therema/state/
[root@[REDACTED] ~]# gcloud storage ls gs://dexter-therema/state/
gs://dexter-therema/state/cloud-dns/
gs://dexter-therema/state/gke-standard/
```

For backing up terraform state file (present is s3 Bucket, for creation of GitLab Server) I created a new s3 bucket and copied the existing state file to the new bucket

```
[root@[REDACTED] ~]# aws s3 mb s3://dexter-backup-statefile --region=us-east-2
make_bucket: dexter-backup-statefile
```

```
[root@REDACTED ~]# aws s3 ls --region=us-east-2
2025- REDACTED dexter-backup-statefile
2025- REDACTED dexter-velero-backup
2025- REDACTED dolo-dempo
2025- REDACTED s3bucketcapturealblogjenkins
```

```
[root@REDACTED ~]# aws s3 sync s3://dolo-dempo/ s3://dexter-backup-statefile/
```

For terraform script I have new set of script present in the directory **opentofu-gcp-cloud-dns-aws-acm-certificate**, **opentofu-gitlab-server** and **opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek** as shown in the screenshot attached below.

```
[root@terraform-server ~]# ls
opentofu-gcp-cloud-dns-aws-acm-certificate
opentofu-gitlab-server
opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek
```

```
terraform-gcp-cloud-dns-aws-acm-certificate
terraform-gitlab-server
terraform-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek
```

The only difference between the Terraform and Opentofu codebase is backend.tf file because in Opentofu I am going to encrypt the state file.

Below is the backend.tf for Terraform.

```
[root@terraform-server ~]# cat terraform-gcp-cloud-dns-aws-acm-certificate/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix  = "state/cloud-dns"
  }
}
[root@terraform-server ~]# cat terraform-gitlab-server/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/gitlab/terraform.tfstate"
    region     = "us-east-2"
    encrypt    = true
    use_lockfile = true    ## dynamodb_table = "terraform-state"
  }
}
[root@terraform-server ~]# cat terraform-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix  = "state/gke-standard"
  }
}
```

Below is the backend.tf for Opentofu before migration.

```
[root@terraform-server ~]# cat opentofu-gcp-cloud-dns-aws-acm-certificate/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix  = "state/cloud-dns"
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }

    plan {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }
}
```

Ritesh Kumar Singh

```
[root@terraform-server ~]# cat opentofu-gitlab-server/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/gitlab/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    #####dynamodb_table = "terraform-state"
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }

  plan {
    method = method.aes_gcm.passphrase
    fallback {
      method = method.unencrypted.read_unencrypted_state
    }
  }
}
}
```

```
[root@terraform-server ~]# cat opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix   = "state/gke-standard"
  }
  encryption {
    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      fallback {
        method = method.unencrypted.read_unencrypted_state
      }
    }
  }

  plan {
    method = method.aes_gcm.passphrase
    fallback {
      method = method.unencrypted.read_unencrypted_state
    }
  }
}
}
```

Below is the backend.tf for Opentofu after migration.

```
[root@terraform-server ~]# cat opentofu-gcp-cloud-dns-aws-acm-certificate/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix  = "state/cloud-dns"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

```
[root@terraform-server ~]# cat opentofu-gitlab-server/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/gitlab/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

```
[root@terraform-server ~]# cat opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix   = "state/gke-standard"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

I changed in backend.tf because the state file of terraform is not encrypted and you already created the resources using terraform which resources might be utilized in some project so you cannot delete the existing resources created using terraform and create the new one and terraform state file was unencrypted.

You can follow below steps to install Opentofu.

```
cd /opt/
```

```
curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
```

```
chmod +x install-opentofu.sh
```

```
./install-opentofu.sh --install-method rpm
```

```
rm -f install-opentofu.sh
```

```
[root@yellow ~]# cd /opt/
[root@yellow opt]# curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
[root@yellow opt]# chmod +x install-opentofu.sh
[root@yellow opt]# ./install-opentofu.sh --install-method rpm
```

```
[root@yellow opt]# rm -f install-opentofu.sh
```

```
[root@yellow ~]# tofu --version
```

```
OpenTofu v1.10.6
```

```
on linux_amd64
```

Below commands you should run for migration.

```
tofu init -reconfigure
```

```
tofu validate
```

```
tofu plan
```

```
tofu apply -auto-approve
```

```
[root@terraform-server ~]# cd opentofu-gcp-cloud-dns-aws-acm-certificate/main/
[root@terraform-server main]# tofu init -reconfigure
```

```
[root@terraform-server main]# tofu validate
Success! The configuration is valid.
```

```
[root@[REDACTED] main]# tofu plan
```

```
No changes. Your infrastructure matches the configuration.

OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

```
[root@[REDACTED] main]# tofu apply -auto-approve
```

```
No changes. Your infrastructure matches the configuration.

OpenTofu has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

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

Outputs:

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

Migration had been done, below is the backend.tf after Migration.

```
[root@terraform-server ~]# cat opentofu-gcp-cloud-dns-aws-acm-certificate/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-dempo"
    prefix   = "state/cloud-dns"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

```
[root@terraform-server ~]# cd opentofu-gitlab-server/main/
[root@terraform-server main]# tofu init -reconfigure
```

```
[root@[REDACTED] main]# tofu validate
Success! The configuration is valid.
```

```
[root@[REDACTED] main]# tofu plan
```

Plan: 0 to add, 3 to change, 0 to destroy.

Note: You didn't use the `-out` option to save this plan, so OpenTofu can't guarantee to take exactly these actions if you run "tofu apply" now.

```
[root@[REDACTED] main]# tofu apply -auto-approve
```

```
Plan: 0 to add, 3 to change, 0 to destroy.
module.gitlab.aws_route_table.private_route_table_3: Modifying... [id=rtb-[REDACTED]]
module.gitlab.aws_route_table.private_route_table_1: Modifying... [id=rtb-[REDACTED]]
module.gitlab.aws_route_table.private_route_table_2: Modifying... [id=rtb-[REDACTED]]
module.gitlab.aws_route_table.private_route_table_3: Modifications complete after 0s [id=rtb-[REDACTED]]
module.gitlab.aws_route_table.private_route_table_2: Modifications complete after 0s [id=rtb-[REDACTED]]
module.gitlab.aws_route_table.private_route_table_1: Modifications complete after 0s [id=rtb-[REDACTED]]
```

Apply complete! Resources: 0 added, 3 changed, 0 destroyed.

Outputs:

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

Migration had been done, below is the backend.tf after Migration.

```
[root@terraform-server ~]# cat opentofu-gitlab-server/main/backend.tf
terraform {
  backend "s3" {
    bucket      = "dolo-dempo"
    key         = "state/gitlab/terraform.tfstate"
    region      = "us-east-2"
    encrypt     = true
    use_lockfile = true    ###dynamodb_table = "terraform-state"
  }
  encryption {
    #   method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
      #      fallback {
      #        method = method.unencrypted.read_unencrypted_state
      #      }
    }

    plan {
      method = method.aes_gcm.passphrase
      #      fallback {
      #        method = method.unencrypted.read_unencrypted_state
      #      }
    }
  }
}
```

```
[root@terraform-server ~]# cd opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek/main/
[root@terraform-server main]# tofu init -reconfigure
```

```
[root@REDACTED main]# tofu validate
Success! The configuration is valid.
```

```
[root@REDACTED main]# tofu plan
```

```
Plan: 0 to add, 4 to change, 0 to destroy.
```

```
Note: You didn't use the -out option to save this plan, so OpenTofu can't guarantee to take exactly these actions if you run "tofu apply" now.
```

```
[root@REDACTED main]# tofu apply -auto-approve
```

Migration had been done, below is the backend.tf after Migration.

```
[root@terraform-server ~]# cat opentofu-gke-cluster-cloudarmor-with-backup-envelope-encryption-kms-cmek/main/backend.tf
terraform {
  backend "gcs" {
    bucket  = "dolo-demo"
    prefix   = "state/gke-standard"
  }
  encryption {
#    method "unencrypted" "read_unencrypted_state" {} ### To read unencrypted state file

    key_provider "pbkdf2" "statefile_encryption_password" {
      passphrase = var.encryption_passphrase
    }

    method "aes_gcm" "passphrase" {
      keys = key_provider.pbkdf2.statefile_encryption_password
    }

    state {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }

    plan {
      method = method.aes_gcm.passphrase
#      fallback {
#        method = method.unencrypted.read_unencrypted_state
#      }
    }
  }
}
```

Now check state file in GCP Bucket (For GCP Cloud DNS Zone and GKE Creation) and S3 Bucket (For GitLab Server Creation) you will find it is encrypted. However, if you run **tofu show**, you can see the state file but in GCP Bucket and S3 Bucket it is encrypted.

Source Code: <https://github.com/singhritesh85/Bank-App-Kustomize.git>

GitHub Repo: <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>

Kustomize Manifests Repo: <https://github.com/singhritesh85/kustomize-bankapp.git>

dev branch: - GKE

stage branch: - AKS

main branch: - EKS

Terraform Script: <https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git>

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

<https://open-policy-agent.github.io/gatekeeper/website/>

<https://kyverno.io/docs/>

<https://falco.org/docs/>

<https://aws.amazon.com/blogs/containers/transparent-encryption-of-node-to-node-traffic-on-amazon-eks-using-wireguard-and-cilium/>

Ritesh Kumar Singh