

DevOps Project Kubernetes Karpenter-clusterautoscaler-VPA-HPA Monitoring
and Log Aggregation using AWS CloudWatch and Azure Monitor-
InPlacePodVerticalScaling



By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

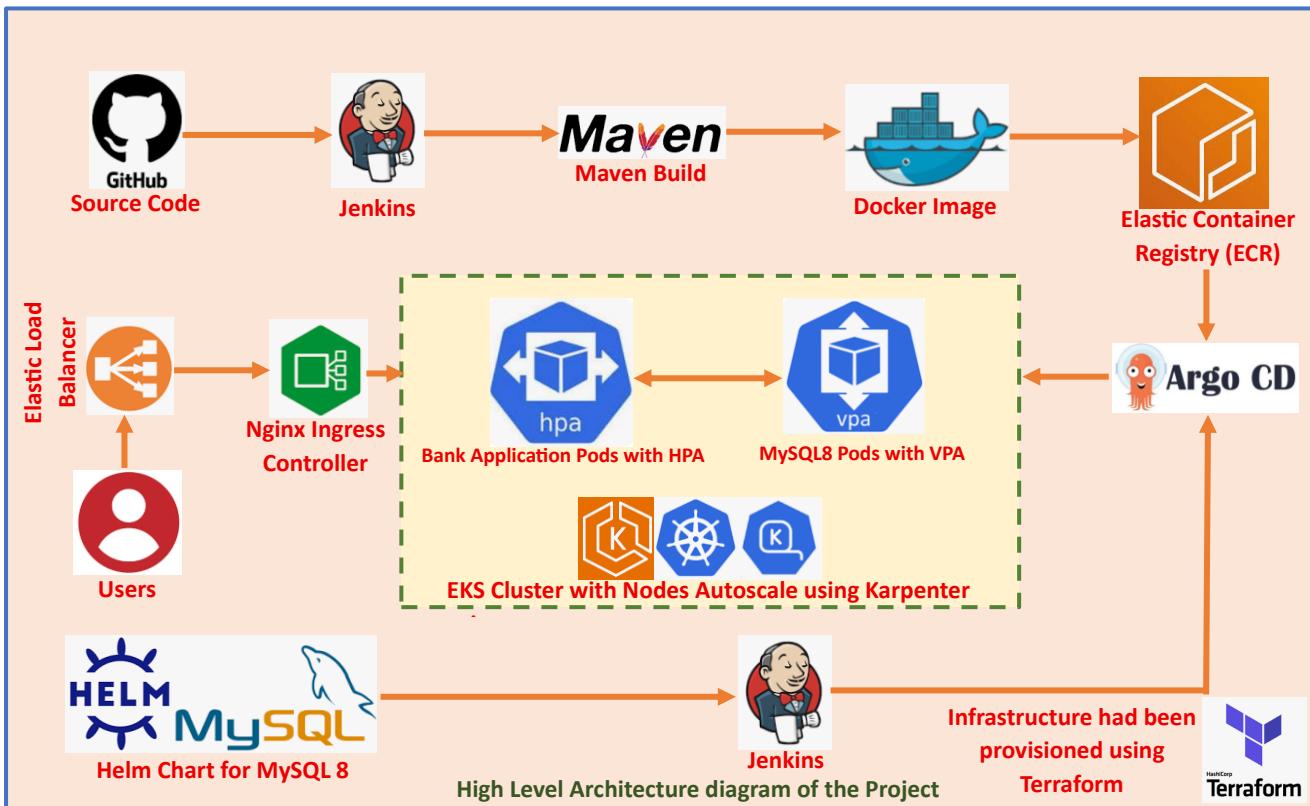
LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/>

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



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

DevOps Project Kubernetes based Karpenter-Vertical-Pod-Autoscaling (VPA), Horizontal-Pod-Autoscaling (HPA) Monitoring and Log Aggregation using CloudWatch



As shown in the architecture diagram drawn above the source code for BankApp was kept in the GitHub Repo which was deployed using the Jenkins. Maven was used as a Build Tool and Docker Image was created then pushed to ECR and then Bank Application Pods had been created using the ArgoCD. To deploy MySQL 8 Pods, I used the bitnami helm chart which was present in the GitHub Repo <https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git>. As shown in the architecture diagram drawn above MySQL 8 Pods was deployed using ArgoCD. Karpenter was used as EKS Nodes Autoscaler. To Autoscale Bank Application HPA and to autoscale MySQL 8 Pods VPA had been used. Finally, users will access the Bank Application through the Nginx Ingress Controller and hence the Kubernetes service for Bank Application. The HOST generated along with the DNS Name of the Elastic LoadBalancer was provided to Route53 to create the Record Set of A-Type with Alias. Finally, users were able to access the Two-Tier Bank Application using the generated URL.

I had provisioned the infrastructure using Terraform and the terraform script is present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git> at the path **terraform-autoscale-karpenter-hpa-vpa**.

For Terraform I used Amazon S3 Bucket to store the terraform state file and to achieve the state lock. Below Screenshot shows how I ran the terraform script and created the Resources in AWS.

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.bankapp.aws_eks_addon.csi_snapshot_controller: Still creating... [01m20s elapsed]
module.bankapp.aws_eks_addon.metrics_server: Still creating... [01m20s elapsed]
module.bankapp.aws_eks_addon.core_dns: Still creating... [01m20s elapsed]
module.bankapp.aws_eks_addon.pod_identity_agent: Creation complete after 3m36s [id=eks-demo-cluster-dev:eks-pod-identity-agent]
module.bankapp.aws_eks_addon.metrics_server: Creation complete after 1m25s [id=eks-demo-cluster-dev:metrics-server]
module.bankapp.aws_eks_addon.ebs_csi_driver: Creation complete after 1m25s [id=eks-demo-cluster-dev:aws-ebs-csi-driver]
module.bankapp.aws_eks_addon.amazon_cloudwatchObservability: Still creating... [03m40s elapsed]
module.bankapp.aws_eks_addon.guard_duty: Still creating... [03m40s elapsed]
module.bankapp.aws_eks_addon.kube_proxy: Still creating... [03m40s elapsed]
module.bankapp.aws_eks_addon.vpc_cni: Still creating... [03m30s elapsed]
module.bankapp.aws_eks_addon.csi_snapshot_controller: Still creating... [01m30s elapsed]
module.bankapp.aws_eks_addon.guard_duty: Creation complete after 3m47s [id=eks-demo-cluster-dev:aws-guardduty-agent]
module.bankapp.aws_eks_addon.kube_proxy: Creation complete after 3m47s [id=eks-demo-cluster-dev:kube-proxy]
module.bankapp.aws_eks_addon.vpc_cni: Creation complete after 3m47s [id=eks-demo-cluster-dev:vpc-cni]
module.bankapp.aws_eks_addon.amazon_cloudwatchObservability: Creation complete after 3m47s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]
module.bankapp.aws_eks_addon.csi_snapshot_controller: Still creating... [01m40s elapsed]
module.bankapp.aws_eks_addon.core_dns: Still creating... [01m40s elapsed]
module.bankapp.aws_eks_addon.core_dns: Creation complete after 1m45s [id=eks-demo-cluster-dev:coredns]
module.bankapp.aws_eks_addon.csi_snapshot_controller: Still creating... [01m50s elapsed]
module.bankapp.aws_eks_addon.csi_snapshot_controller: Creation complete after 1m55s [id=eks-demo-cluster-dev:snapshot-controller]

Apply complete! Resources: 117 added, 0 changed, 0 destroyed.

Outputs:

ecr_ec2_private_ip_alb_dns_eks = {
  "EC2_Instance_Jenkins_Master_Server_Private_IP_Address" = "10.████████"
  "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10.████████"
  "Jenkins_ALB_DNS_Name" = "jenkins-ms-████████.us-east-2.elb.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "registry_id" = [
    "02████████6",
  ]
  "repository_url" = [
    "02████████6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev",
  ]
}
```

Using this terraform script I had created an IAM Role **karpenter-eks-noderole**, which will be used to spin-up new nodes using karpenter. I had edited the configmap **aws-auth** in the namespace **kube-system** and mapped the IAM Role **karpenter-eks-noderole** as shown in the screenshot attached 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: |
    - groups:
        - system:bootstrappers
        - system:nodes
          rolearn: arn:aws:iam::02████████6:role/eks-nodegroup-role-dev
          username: system:node:{${EC2PrivateDNSName}}
    - groups:
        - system:bootstrappers
        - system:nodes
          rolearn: arn:aws:iam::02████████6:role/karpenter-eks-noderole
          username: system:node:{${EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2025-01-11T10:25:00Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1000"
  uid: d████████████████████████████████████████b
```

Then installed the Karpenter Controller using the helm as shown in the screenshot attached below.

```
[root@████████ ~]# helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.3.2" --namespace "karpenter" --create-namespaces --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks.amazonaws.com/role-arn=arn:aws:iam::02████████6: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
```

```
helm upgrade --install carpenter oci://public.ecr.aws/karpenter/karpenter --version "1.3.2" --
namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set
"serviceAccount.annotations.eks.amazonaws.com/role-
arn=arn:aws:iam::02XXXXXXXXX6:role/karpenter-controller-role" --set
controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set
controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi
```

```
[root@REDACTED ~]# kubectl get all -n carpenter
NAME                                         READY   STATUS    RESTARTS   AGE
pod/karpenter-8REDACTED-5-2REDACTED   f      1/1     Running   0          4m32s
pod/karpenter-8REDACTED-5-8REDACTED   v      1/1     Running   0          4m32s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/karpenter   ClusterIP   172.REDACTED.REDACTED   <none>        8080/TCP   4m32s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/karpenter   2/2       2           2          4m32s

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/karpenter-8REDACTED   5         2         2         4m32s
```

After installation of Karpenter Controller I applied the kubernetes manifests file for Karpenter, **karpenter.yaml** present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git> at the path **terraform-autoscale-karpenter-hpa-vpa/**. For your reference I provided the karpenter.yaml below which you can change depending on your project requirement.

```

karporter.yaml

# This example NodePool will provision general purpose instances

---

apiVersion: karporter.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: karporter.sh/capacity-type

          operator: In

          values: ["on-demand"] #[["spot"]] ### You can select on-demand or spot instance depending
on your requirement.

        - key: karporter.k8s.aws/instance-category

          operator: In

          values: ["t"] # Interested to launch t series instance      #[["c", "m", "r"]]

        - key: karporter.k8s.aws/instance-generation

          operator: Gt

          values: ["1"] # Instances launched will be greater than 1

      nodeClassRef:

        group: karporter.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

```

Then I installed ArgoCD, ArgoCD CLI and Nginx Ingress Controller as explained below. I did not explain SonarQube and Nexus in the current project if you are interested you can download and refer the PDF document present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git>.

Installation of Nginx Ingress Controller

```
kubectl create ns ingress-nginx
```

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

```
helm repo update
```

```
helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02XXXXXXXXXX6:certificate/XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-type"="elb" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-backend-protocol"="http" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-ssl-ports"="https" --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"
```

```
[root@... ~]# kubectl create ns ingress-nginx
namespace/ingress-nginx created
[root@... ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
[root@... ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. Happy Helming!
[root@... ~]# helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02XXXXXXXXXX6:certificate/XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-type"="elb" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-backend-protocol"="http" --set controller.service.annotations."service\\.beta\\.kubernetes\\.io/aws-load-balancer-ssl-ports"="https" --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"

[root@... ~]# kubectl get all -n ingress-nginx
NAME                                     READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-...          0/1     Pending   0          5m56s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
service/ingress-nginx-controller          LoadBalancer   172.17.0.11   a...us-east-2.elb.amazonaws.com   80:3183
service/ingress-nginx-controller-admission ClusterIP   172.17.0.11   <none>           443/TCP
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller  0/1     1           0          5m56s
NAME           DESIRED  CURRENT    READY   AGE
replicaset.apps/ingress-nginx-controller-...  1         1         0          5m56s
```

You can see in the screenshot attached above initially Nginx Ingress Controller Pod was in Pending State the reason behind that was Cluster did not have sufficient resource available and here Karpenter comes into the picture and Karpenter spun a new node. Initially the EKS Cluster had two node and Karpenter spun a new Node as shown in the screenshot attached below.

```
[root@... ~]# kubectl get nodes
NAME                                     STATUS   ROLES   AGE   VERSION
ip-10-... us-east-2.compute.internal   Ready   <none>  144m  v1.30.4-eks-...
ip-10-... us-east-2.compute.internal   Ready   <none>  13m   v1.30.11-eks-...
ip-10-... .us-east-2.compute.internal  Ready   <none>  144m  v1.30.4-eks-...
```

Finally, the Nginx Ingress Controller Pod was in Running State as shown in the screenshot attached below.

NAME	READY	STATUS	RESTARTS	AGE
pod/ingress-nginx-controller-[REDACTED]	1/1	Running	0	1d
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/ingress-nginx-controller	LoadBalancer	172.17.0.1	a-[REDACTED].us-east-2.elb.amazonaws.com	80:31278/TCP
service/ingress-nginx-controller-admission	ClusterIP	172.17.0.1	<none>	443/TCP
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/ingress-nginx-controller	1/1	1	1	1d
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/ingress-nginx-controller-[REDACTED]	1	1	1	1d

Installation of ArgoCD

```
kubectl create namespace argocd
```

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

You can create the password for admin user using the command as written below.

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

```
[root@REDACTED ~]# kubectl create namespace argocd
namespace/argocd created
[root@REDACTED ~]# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
[root@REDACTED ~]# kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
g[root@REDACTED ~]#
```

Then I created the ingress-rule for ArgoCD as shown in the screenshot attached below.

```
cat argocd-ingress-rule.yaml

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

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

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

```
[root@yellow ~]# kubectl get ing -n argocd
NAME           CLASS      HOSTS          ADDRESS          PORTS   AGE
minimal-ingress <none>    argocd.singhritesh85.com  a yellow.us-east-2.elb.amazonaws.com  80      8m
```

I had created the record set of A-Type with Alias in Route53 to access the ArgoCD as shown in the screenshot attached below.

The screenshot shows the AWS Route 53 console under the 'Hosted zones' section for the domain 'singhritesh85.com'. A new record set is being created for the 'argocd' alias. The 'Record type' is set to 'A – Routes traffic to an IPv4 address and some AWS resources'. The 'Route traffic to' dropdown is set to 'Alias to Application and Classic Load Balancer' with 'US East (Ohio)' selected. The target is set to 'us-east-2.elb.amazonaws.com'. The 'Evaluate target health' option is turned off. The 'Create records' button at the bottom right is highlighted.

I had updated the admin user password in ArgoCD.

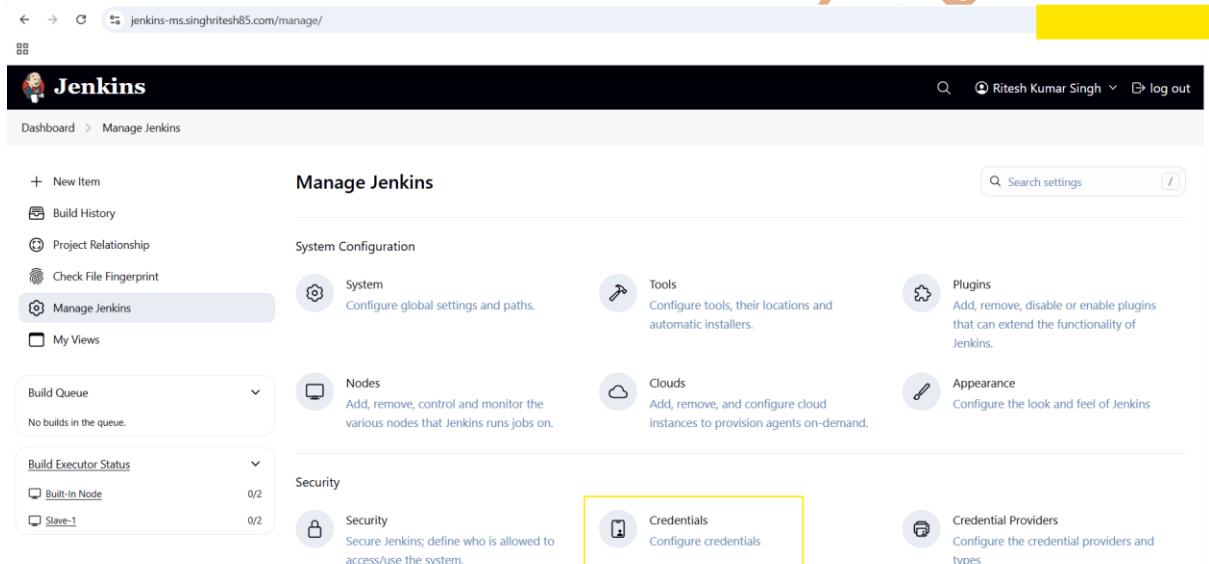
The screenshot shows the ArgoCD 'User Info' page. On the left sidebar, 'User Info' is selected. The main area shows the current user info: 'Username: admin' and 'Issuer: argocd'. A prominent yellow box highlights the 'UPDATE PASSWORD' button. A success message at the bottom right states 'Your password has been successfully updated.'.

Installation ArgoCD CLI on Jenkins Slave Node

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

Then created the Jenkins Job to deploy Pods in the EKS Cluster, the Jenkinsfile present in the GitHub Repo <https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch-eks.git> with the name of Jenkinsfile-eks-bankapp and Jenkinsfile-eks-mysql. Before running the Jenkins Job, I created three credentials of kind **secret-text** in Jenkins with the credential ID argocd_password, mysql_root_password, and mysql_database as shown in the screenshot attached below.





jenkins-ms.singhrithesh85.com/manage/credentials/store/system/domain/_/newCredentials

Jenkins

Ritesh Kumar Singh log out

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

New credentials

Kind: Secret text

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

Secret:

ID: argocd_password

Description: argocd_password

Create

jenkins-ms.singhrithesh85.com/manage/credentials/store/system/domain/_/newCredentials

Jenkins

Ritesh Kumar Singh log out

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

New credentials

Kind: Secret text

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

Secret:

ID: mysql_database

Description: mysql_database

Create

The screenshot shows the Jenkins 'New credentials' configuration page. The 'Kind' field is set to 'Secret text'. The 'Scope' field is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains 'mysql_root_password'. The 'ID' field is also 'mysql_root_password'. A description 'mysql_root_password' is provided. The 'Create' button is highlighted with a yellow box.

Ran the Jenkins Job for mysql and bankapp, the screenshot for Jenkins Job after its successful execution is as shown in the screenshot attached below. First ran the Jenkins Job for mysql and then for bankapp.

```
[jenkins@[REDACTED] ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          [REDACTED]
mysql-1   1/1     Running   0          [REDACTED]
mysql-2   1/1     Running   0          [REDACTED]
```

Karpenter was doing its job and spun a new node for EKS to schedule the mysql pods as shown in the screenshot attached below.

NAME	STATUS	ROLES	AGE	VERSION
ip-[REDACTED].us-east-2.compute.internal	Ready	<none>	[REDACTED]	v1.30.4-eks-[REDACTED]
ip-[REDACTED].us-east-2.compute.internal	Ready	<none>	[REDACTED]	v1.30.11-eks-[REDACTED]
ip-[REDACTED].us-east-2.compute.internal	Ready	<none>	[REDACTED]	v1.30.4-eks-[REDACTED]
ip-[REDACTED].us-east-2.compute.internal	Ready	<none>	[REDACTED]	v1.30.11-eks-[REDACTED]

I want to mention here that I wanted to perform Real User Monitoring (RUM) using CloudWatch. So, I got the Script to be inserted into the source code as shown in the screenshot attached below.

```

1 import { AwsRum, AwsRumConfig } from 'aws-rum-web';
2
3 + try {
4   const config: AwsRumConfig = {
5     sessionSampleRate: 1 ,
6     identityPoolId: "████████████████████████████████",
7     endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
8     telemetries: ["http","errors","performance"] ,
9     allowCookies: true ,
10    enableXRay: false ,
11    signing: true // If you have a public resource policy and wish to send unsigned requests please set this to false
12  };
13
14  const APPLICATION_ID: string = "████████████████";
15  const APPLICATION_VERSION: string = '1.0.0';
16  const APPLICATION_REGION: string = 'us-east-2';
17
18 + const awsRum: AwsRum = new AwsRum(
19   APPLICATION_ID,
20   APPLICATION_VERSION,
21   APPLICATION_REGION,
22   config
23 );

```

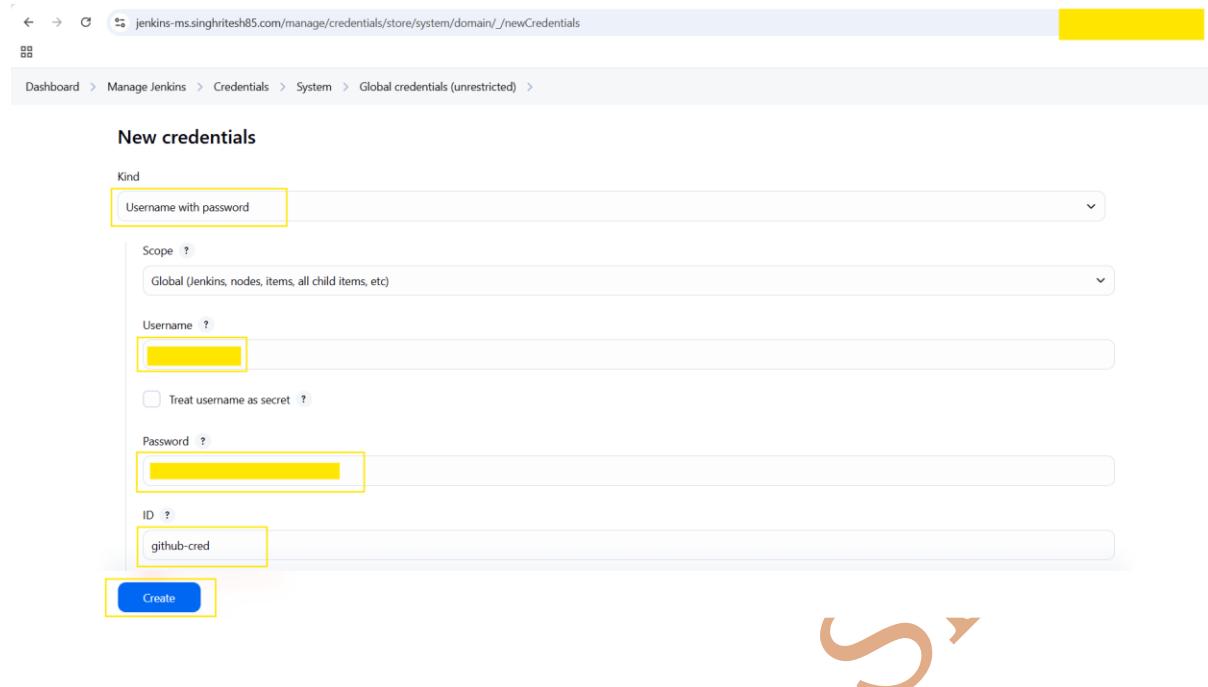
I copied this script and added in the file dashboard.html, login.html, register.html and transactions.html under the line as shown in the screenshot attached below.

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <script>
5     (function(n,i,v,r,s,c,x,z){x=window.AwsRumClient={q:[],n:n,i:i,v:v,r:r,c:c};window[n]=function(c,p){x.q.push({c:c,p:p});};z=document.createElement('script'));
6     .src='████████████████',
7     .type='text/javascript',
8     .version='1.0.0',
9     .region='us-east-2',
10    .src='https://client.rum.us-east-1.amazonaws.com/1.19.0/cwr.js',
11    .config={
12      sessionSampleRate: 1 ,
13      identityPoolId: "████████████████████████████████",
14      endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
15      telemetries: ["http","errors","performance"] ,
16      allowCookies: true ,
17      enableXRay: false ,
18      signing: true // If you have a public resource policy and wish to send unsigned requests please set this to false
19    }
20  );

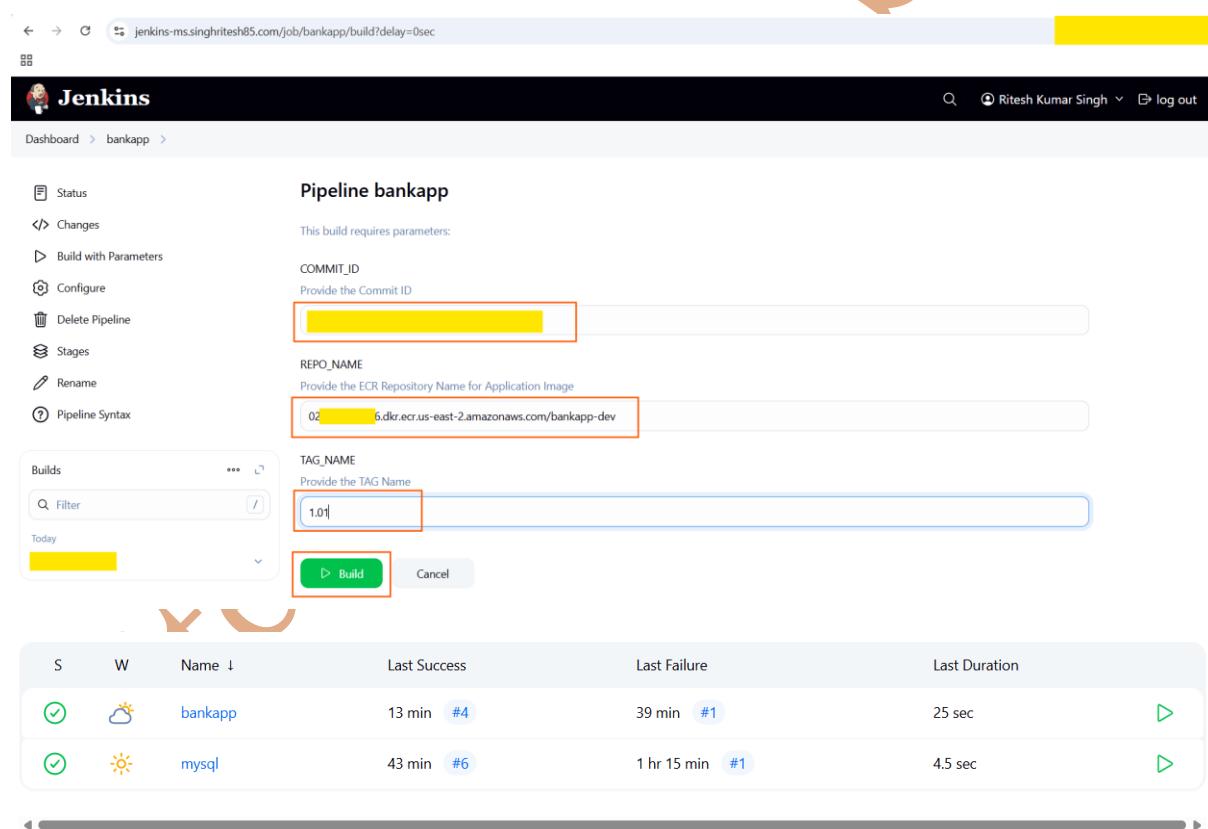
```

In the similar way I added in more three files login.html, register.html and transactions.html. Then I ran the Jenkins Job for bankapp and for that I provided string parameters as shown in the screenshot attached below. Before running this Jenkins Job, I created the github credentials in Jenkins credentials as shown in the screenshot attached below.



The screenshot shows the Jenkins 'New credentials' configuration page. A 'Username with password' credential is being created with the following details:

- Kind:** Username with password
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Username:** [REDACTED]
- Treat username as secret:** [unchecked]
- Password:** [REDACTED]
- ID:** github-cred
- Create:** [button]



The screenshot shows the Jenkins Pipeline bankapp build configuration page. The pipeline requires parameters:

- COMMIT_ID:** Provide the Commit ID [REDACTED]
- REPO_NAME:** Provide the ECR Repository Name for Application Image [REDACTED]
- TAG_NAME:** Provide the TAG Name [REDACTED]

A build button is highlighted with a red box.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	cloud	bankapp	13 min #4	39 min #1	25 sec
✓	sun	mysql	43 min #6	1 hr 15 min #1	4.5 sec

Then I created the ingress rule for bankapp and Record Set of A-Type with Alias in Route53 as shown in the screenshot attached below.

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

```
[jenkins@REDACTED ~]$ kubectl get ing -n bankapp --watch
NAME      CLASS   HOSTS          ADDRESS
bankapp-ingress  nginx  bankapp.singhritesh85.com  a[REDACTED].us-east-2.elb.amazonaws.com  80    2m35s
```

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




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

Quick create record

Record name **.singhritesh85.com**

Record type

Alias

Route traffic to

US East (Ohio)

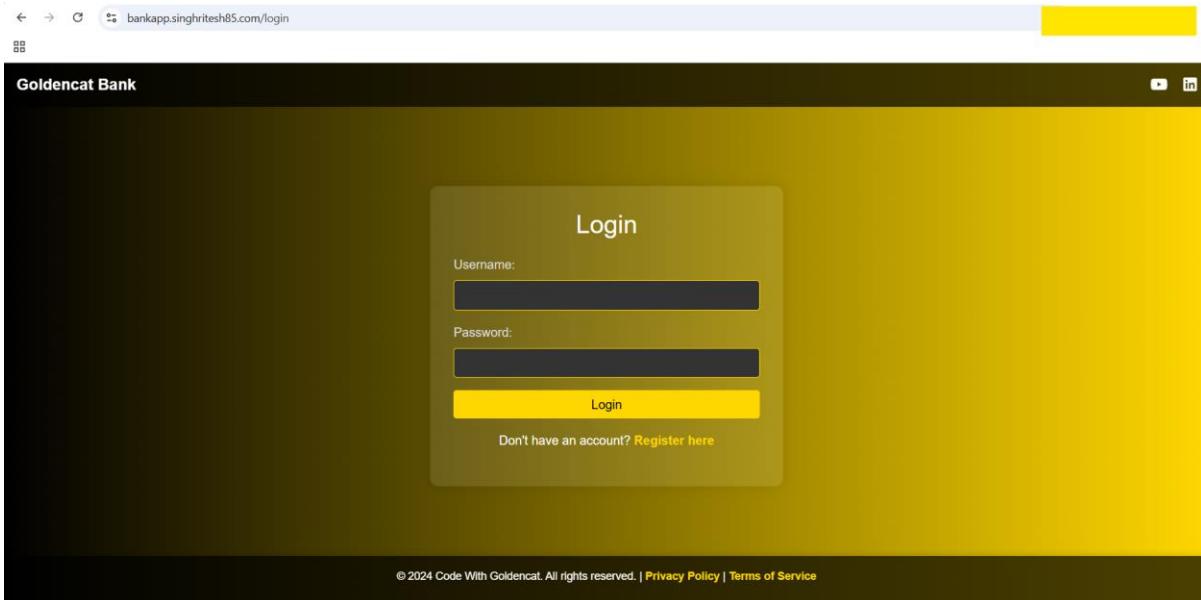
Alias hosted zone ID

Routing policy

Evaluate target health No

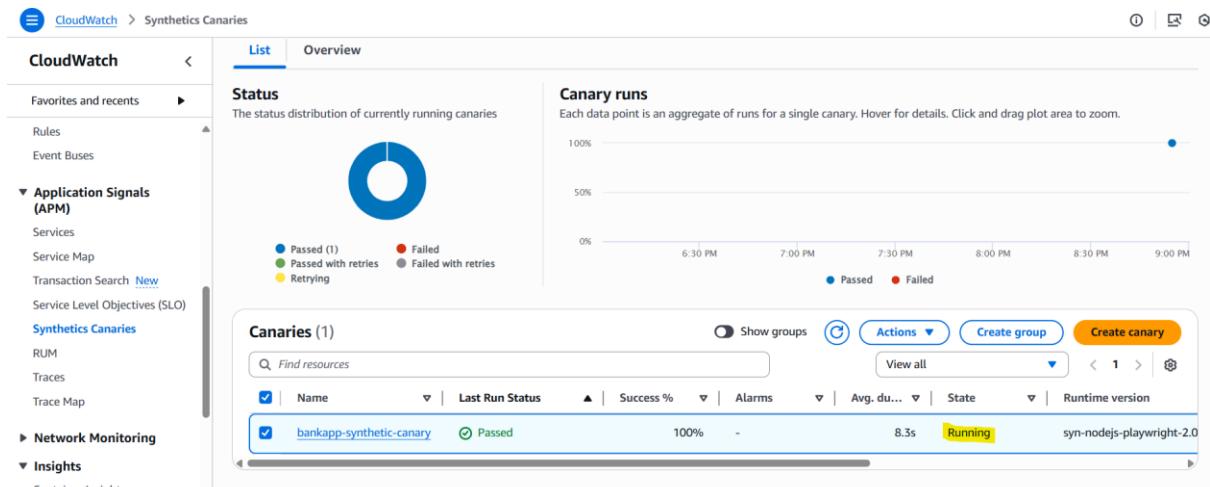
Add another record **Create records**

Finally, I was able to access the Application as shown in the screenshot attached below. After I found I was able to access the Bank Application then I confirmed the SNS Topic Subscription for the Group Email ID otherwise unnecessary Emails will be sent to the Group Email ID.

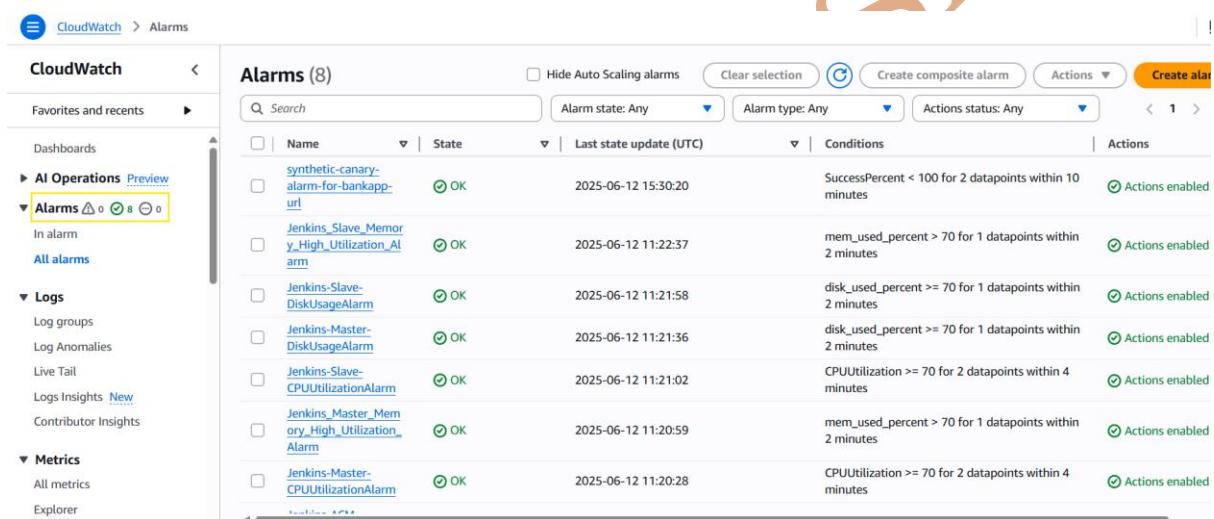


The screenshot for ArgoCD after the successful deployment of pods is as shown in the screenshot attached below.

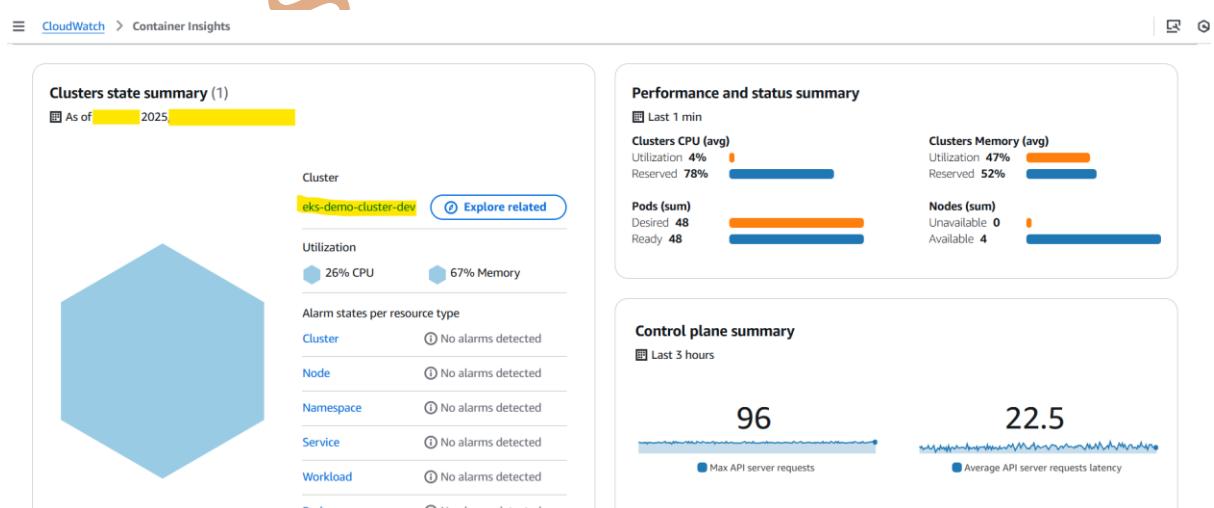
I started the synthetic canary after I found I was able to access the Application through the URL.

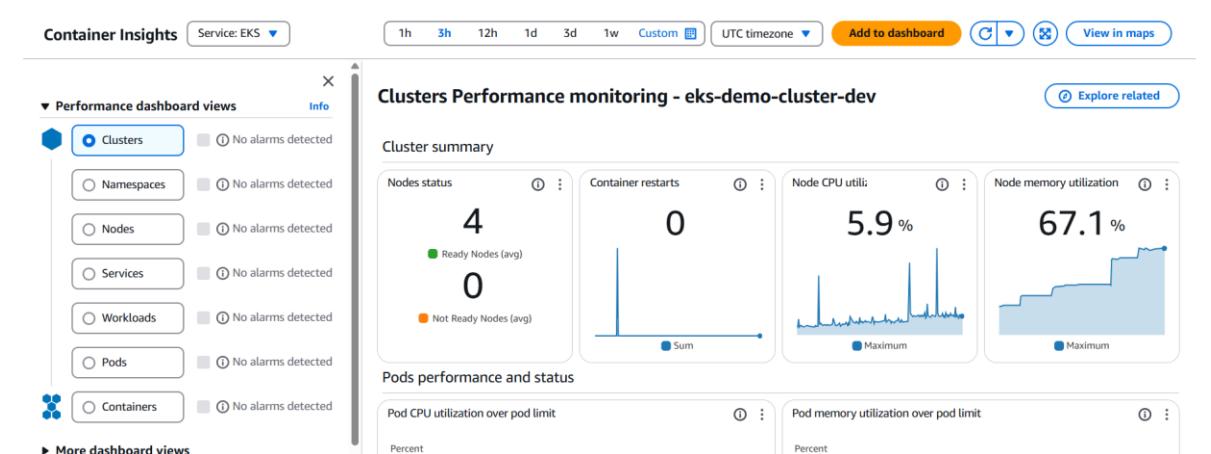
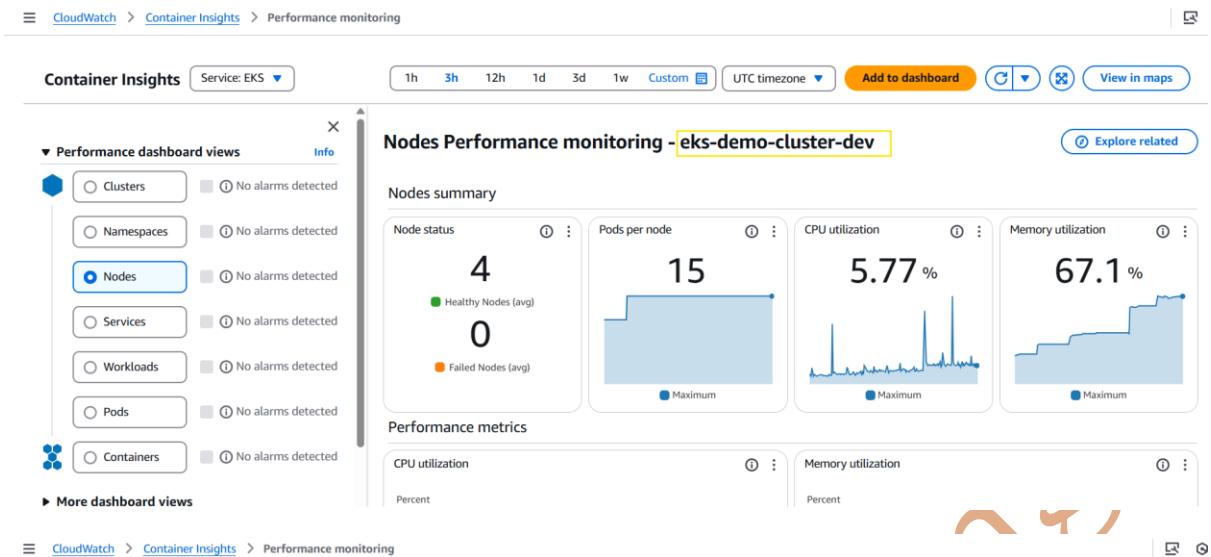


The CloudWatch Alarm was generated using Terraform as shown in the screenshot attached below.

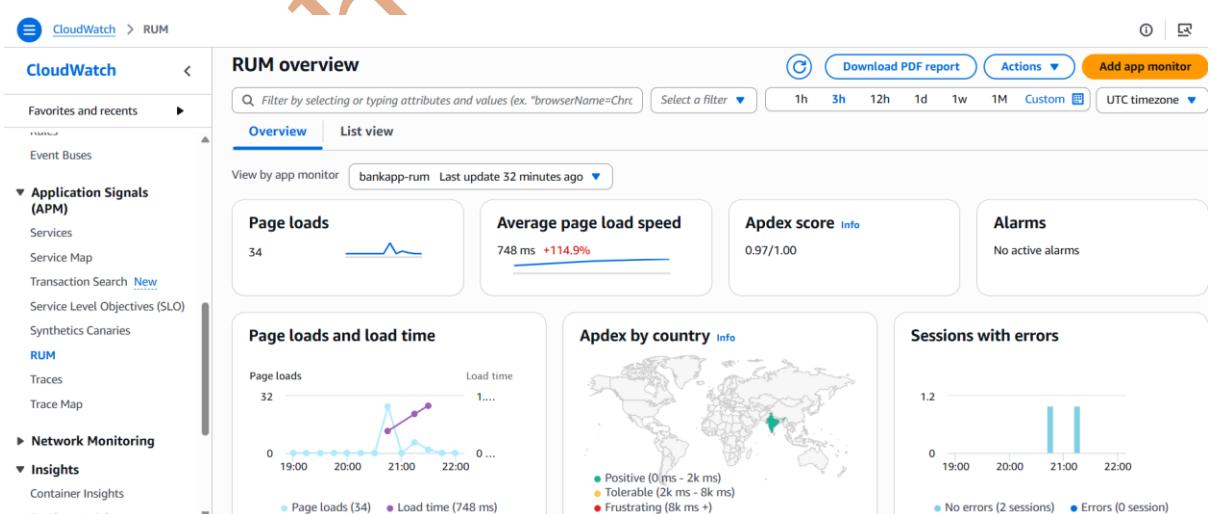


I enabled container Insight as shown in the screenshot attached below.



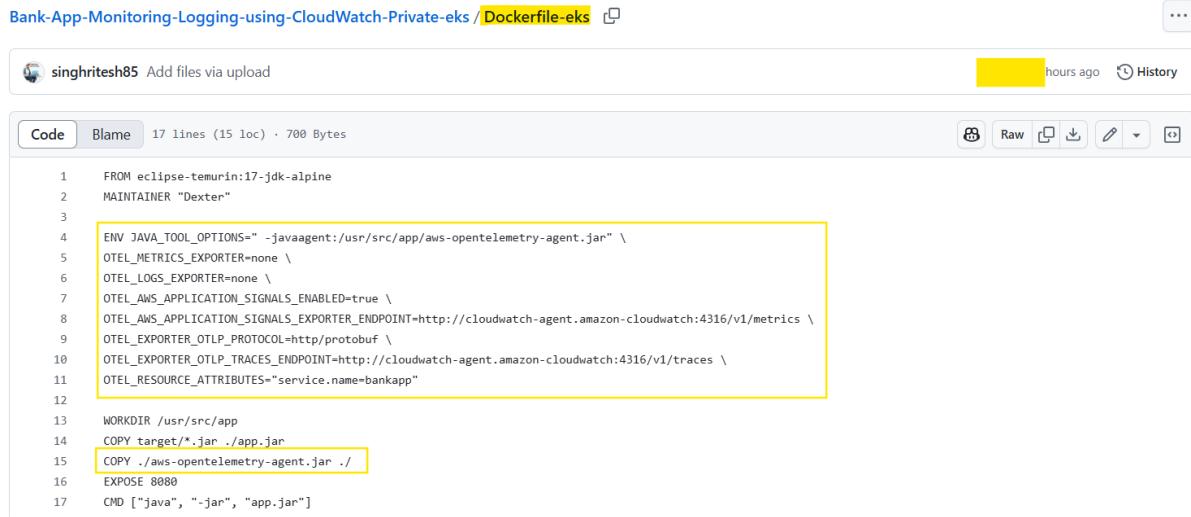


I am performing Real User Monitoring (RUM) using CloudWatch RUM as shown in the screenshot attached below.



I perform Application Performance Monitoring (APM) using CloudWatch for Bank Application in EKS Cluster for that I installed the Amazon CloudWatch Observability and Amazon EKS Pod Identity Agent Add-Ons and created EKS Pod Identity Association for bankapp and mysql using Terraform, further this the Dockerfile present in the GitHub Repo <https://github.com/singhritesh85/Bank-App>

[Monitoring-Logging-using-CloudWatch-eks.git](#) and added annotation in the Helm-Chart <https://github.com/singhritesh85/helm-repo-for-ArgoCD-cloudwatch-apm.git> is as shown in the screenshot attached below.



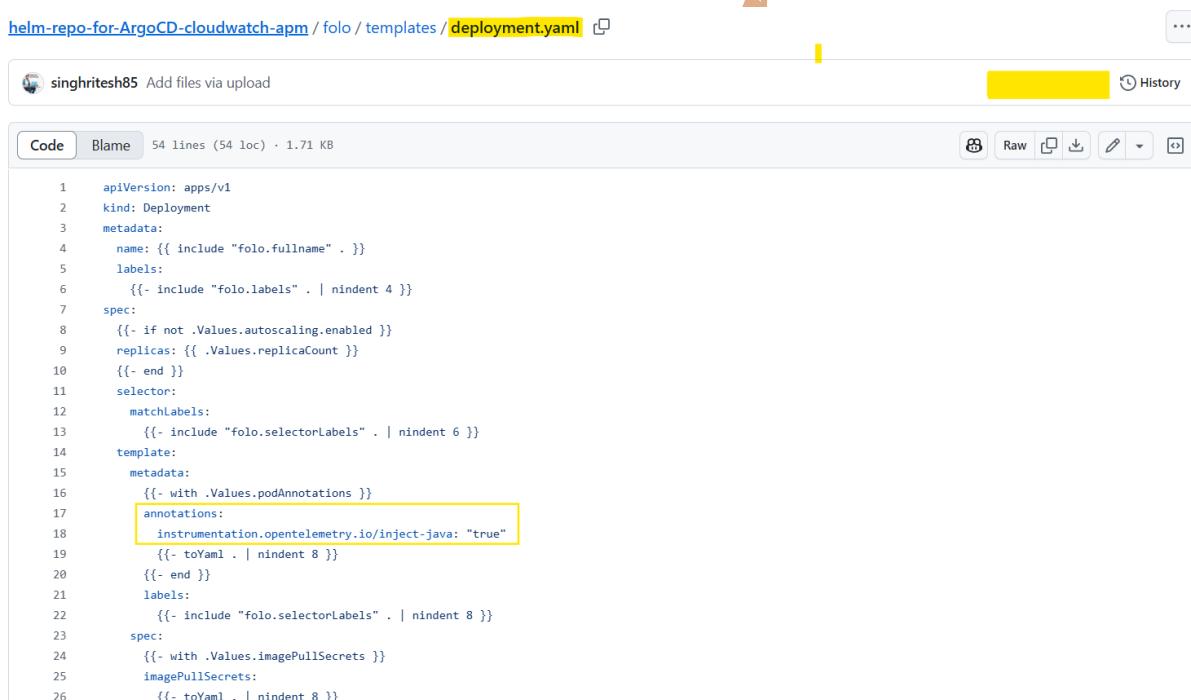
```

FROM eclipse-temurin:17-jdk-alpine
MAINTAINER "Dexter"

ENV JAVA_TOOL_OPTIONS="-javaagent:/usr/src/app/aws-opentelemetry-agent.jar" \
    OTEL_METRICS_EXPORTER=none \
    OTEL_LOGS_EXPORTER=none \
    OTEL_AWS_APPLICATION_SIGNALS_ENABLED=true \
    OTEL_AWS_APPLICATION_SIGNALS_EXPORTER_ENDPOINT=http://cloudwatch-agent.amazon-cloudwatch:4316/v1/metrics \
    OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf \
    OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://cloudwatch-agent.amazon-cloudwatch:4316/v1/traces \
    OTEL_RESOURCE_ATTRIBUTES="service.name=bankapp"

WORKDIR /usr/src/app
COPY target/*.jar ./app.jar
COPY ./aws-opentelemetry-agent.jar ./
EXPOSE 8080
CMD ["java", "-jar", "app.jar"]

```



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "folo.fullname" . }}
  labels:
{{- include "folo.labels" . | nindent 4 }}
spec:
{{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
{{- end }}
  selector:
    matchLabels:
{{- include "folo.selectorLabels" . | nindent 6 }}
template:
  metadata:
{{- with .Values.podAnnotations }}
    annotations:
{{- instrumentation.opentelemetry.io/inject-java: "true" | toYaml . | nindent 8 }}
{{- end }}
    labels:
{{- include "folo.selectorLabels" . | nindent 8 }}
  spec:
{{- with .Values.imagePullSecrets }}
    imagePullSecrets:
{{- toYaml . | nindent 8 }}

```

The Services, Service Map, Traces and Trace Map is as shown in the screenshot attached below.

CloudWatch Services

CloudWatch

- Favorites and recent
- Event Buses
- Application Signals (APM)**
 - Services
 - Service Map
 - Transaction Search [New](#)
 - Service Level Objectives (SLO)
 - Synthetics Canaries
 - RUM
 - Traces
 - Trace Map
- Network Monitoring**
- Insights**
 - Container Insights

CloudWatch Service Map

Service Map

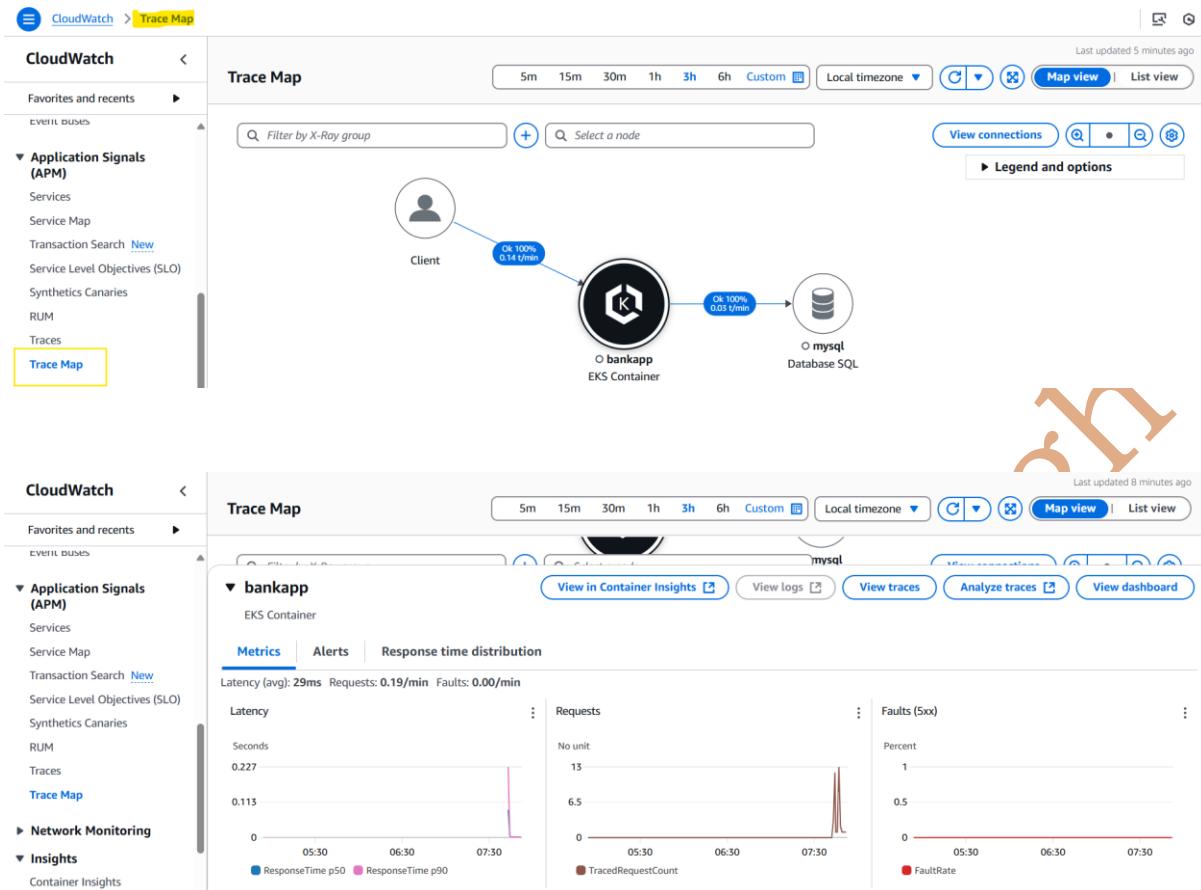
Last updated now

30m 1h 3h 12h Custom UTC timezone

CloudWatch Traces

CloudWatch

- Favorites and recent
- Event Buses
- Application Signals (APM)**
 - Services
 - Service Map
 - Transaction Search [New](#)
 - Service Level Objectives (SLO)
 - Synthetics Canaries
 - RUM
 - Traces**
 - Trace Map
- Network Monitoring**
- Insights**
 - Container Insights
 - Database Insights



Horizontal Pod Autoscaler (HPA) and Vertical Pod Autoscaler (VPA)

I had already installed the Metrics Server in EKS as Add-On using Terraform. First, we need to install the CRD for Vertical Pod Autoscaler (VPA) using the steps as mentioned below.

```
git clone https://github.com/kubernetes/autoscaler.git
cd autoscaler/vertical-pod-autoscaler/hack/
./vpa-up.sh
```

```
[root@... ~]# git clone https://github.com/kubernetes/autoscaler.git
Cloning into 'autoscaler'...
remote: Enumerating objects: 224382, done.
remote: Counting objects: 100% (1862/1862), done.
remote: Compressing objects: 100% (1238/1238), done.
remote: Total 224382 (delta 1205), reused 625 (delta 624), pack-reused 222520 (from 2)
Receiving objects: 100% (224382/224382), 248.20 MiB | 22.86 MiB/s, done.
Resolving deltas: 100% (145393/145393), done.
Updating files: 100% (8051/8051), done.
[root@... ~]# cd autoscaler/vertical-pod-autoscaler/hack/
[root@... hack]# ./vpa-up.sh
```

Make sure you run the above three commands from Amazon Linux 2023 Machine. As the requirement to run the above three commands from a Machine where OpenSSL is installed with 1.1.1 or later version installed.

```
[root@yellow ~]# kubectl get crd |grep -i "VerticalPodAutoscaler"
verticalpodautoscalercheckpoints.autoscaling.k8s.io      2025
verticalpodautoscalers.autoscaling.k8s.io                 2025
```

```
[root@yellow hack]# kubectl get pods -n kube-system|grep -i "vpa"
vpa-admission-controller- 1/1     Running   0
vpa-recommender-          1/1     Running   0
vpa-updater-              1/1     Running   0
```

At this point I checked the cpu and memory of container mysql present in mysql-0 pod in the namespace mysql which as shown in the screenshot attached below.

```
[jenkins@yellow ~]$ kubectl describe pod mysql-0 -n mysql
Containers:
  mysql:
    Container ID:  containerd://
    Image:         docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:      docker.io/bitnami/mysql@sha256:1
    Port:          3306/TCP
    Host Port:    0/TCP
    SeccompProfile: RuntimeDefault
    State:         Running
      Started:    2025
    Ready:        True
    Restart Count: 0
    Limits:
      cpu:  650m
      memory:  650Mi
    Requests:
      cpu:  650m
      memory:  650Mi
```

Now, I created the Jenkins Job with the name of hpa-vpa to deploy the Horizontal Pod Autoscaling (HPA) and Vertical Pod Autoscaling (VPA) using the Kubernetes Manifests file which is present in the GitHub Repo <https://github.com/singhritesh85/kubernetes-manifests.git> at the path **hpa-vpa**. The Jenkinsfile with the name of **Jenkinsfile-hpa-vpa** is present in the GitHub Repo <https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch-eks.git>.

Kubernetes Manifests file hpa-vpa.yaml

```
---  
apiVersion: autoscaling/v2  
kind: HorizontalPodAutoscaler  
metadata:  
  name: bankapp-hpa  
  namespace: bankapp  
spec:  
  scaleTargetRef:  
    apiVersion: apps/v1  
    kind: Deployment  
    name: bankapp-folo  
  minReplicas: 1  
  maxReplicas: 3  
  metrics:  
    - type: Resource  
      resource:  
        name: cpu  
      target:  
        type: Utilization  
        averageValue: 1 #####averageUtilization: 50 ##### For demonstration I used a targated value of 1  
---  
apiVersion: "autoscaling.k8s.io/v1"  
kind: VerticalPodAutoscaler  
metadata:  
  name: mysql-vpa #####mysql-primary-vpa  
  namespace: mysql  
spec:  
  targetRef:  
    apiVersion: "apps/v1"
```

```
kind: StatefulSet
name: mysql    ###mysql-primary
updatePolicy:
  updateMode: Auto
resourcePolicy:
  containerPolicies:
    - containerName: 'mysql'  ###
      minAllowed:
        cpu: 600m
        memory: 650Mi
      maxAllowed:
        cpu: 850m    #maximum vpa will be allocating this many cpus even if demand is higher.
        memory: 850Mi
  controlledResources: ["cpu", "memory"]
```

Ritesh Kumar Singh

Jenkinsfile-hpa-vpa

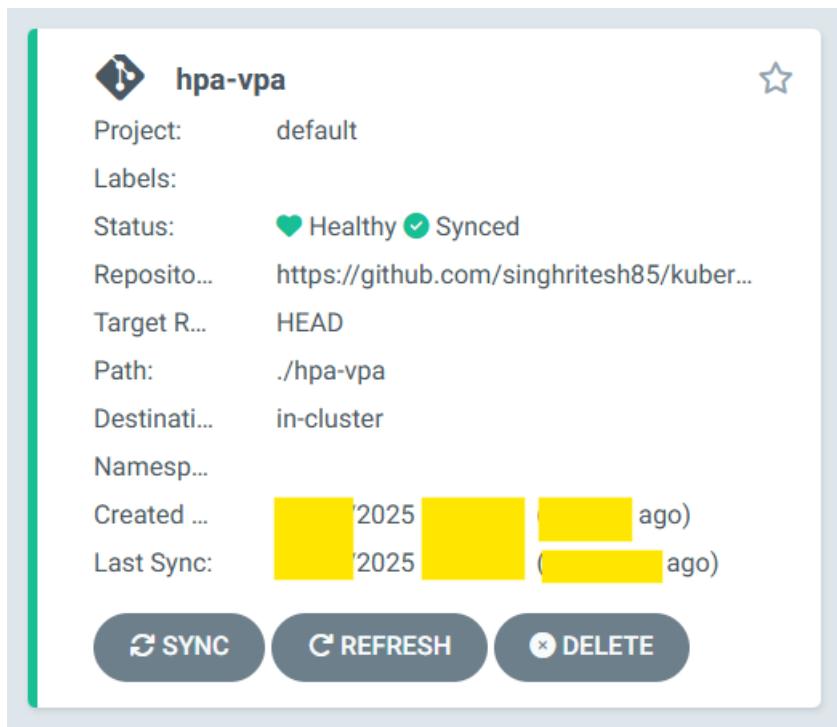
```

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 hpa-vpa --project default --repo https://github.com/singhritesh85/kubernetes-manifests.git --path ./hpa-vpa --dest-server https://kubernetes.default.svc --upsert'
                sh 'argocd app sync hpa-vpa'
            }
        }
    }
}

```

The screenshot for Jenkins Job after running it successfully is as shown in the screenshot attached below.

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	☀️	bankapp	1 hr 14 min #5	3 hr 11 min #1	24 sec	▶
✓	☁️	hpa-vpa	6.3 sec #7	1 hr 36 min #5	3.5 sec	▶
✓	☁️	mysql	1 hr 18 min #9	1 hr 20 min #8	7.4 sec	▶



```
[root@[REDACTED] ~]# kubectl get hpa -n bankapp
NAME      REFERENCE          TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
bankapp-hpa   Deployment/bankapp-folo   cpu: 2m/1    1          3           1          [REDACTED]
[root@[REDACTED] ~]# kubectl get vpa -n mysql
NAME      MODE      CPU      MEM      PROVIDED      AGE
mysql-vpa  Auto     600m    650Mi   True        [REDACTED]
```

Finally, BankApp and MySQL8 Pods were scaled horizontally and vertically respectively as shown in the screenshot attached below. You can see in case bankapp pods scaled to desired number of replicas while in case of MySQL8 Pods got created one-by-one as shown in the screenshot attached below for bankapp and mysql. In case of mysql8 pods (as a part of mysql statefulset) as the pods got created one by one so it should not affect the users however it is suggestable to use Pod Disruption Budget (PDB) when you use the VPA in Auto mode.

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

I accessed the mysql pods from bankapp pod using kubernetes service (which also acted as the LoadBalancer) and not with the headless service. So, when one pod goes down during the VPA (Vertical Pod Autoscale) then other pods of MySQL could be accessible from the bankapp pod. If needed you can implement primary secondary (HA) for MySQL Pods.

```
[root@ip-172-31-0-77 ~]# kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          9m22s
mysql-1   0/1     Running   0          19s
mysql-2   1/1     Running   0          8m7s
mysql-1   0/1     Running   0          30s
mysql-1   1/1     Running   0          30s
mysql-2   1/1     Running   0          8m46s
mysql-2   1/1     Terminating 0          8m46s
mysql-2   1/1     Terminating 0          8m46s
mysql-2   0/1     Terminating 0          8m47s
mysql-2   0/1     Terminating 0          8m48s
mysql-2   0/1     Terminating 0          8m48s
mysql-2   0/1     Pending    0          0s
mysql-2   0/1     Pending    0          0s
mysql-2   0/1     Pending    0          2m42s
mysql-2   0/1     Pending    0          2m44s
mysql-2   0/1     Pending    0          3m16s
mysql-2   0/1     Pending    0          3m24s
mysql-2   0/1     Pending    0          3m34s
mysql-2   0/1     Init:0/1  0          3m34s
mysql-2   0/1     Init:0/1  0          3m49s
mysql-2   0/1     PodInitializing 0          3m50s
mysql-2   0/1     Running    0          3m51s
mysql-0   1/1     Running   0          14m
mysql-0   1/1     Terminating 0          14m
mysql-0   1/1     Terminating 0          14m
mysql-0   0/1     Pending    0          0s
mysql-0   0/1     Pending    0          0s
mysql-0   0/1     Init:0/1  0          0s
mysql-2   0/1     Running   0          4m9s
mysql-2   1/1     Running   0          4m9s
mysql-0   0/1     Init:0/1  0          25s
mysql-0   0/1     PodInitializing 0          26s
mysql-0   0/1     Running   0          27s
mysql-0   0/1     Running   0          44s
mysql-0   1/1     Running   0          44s
```

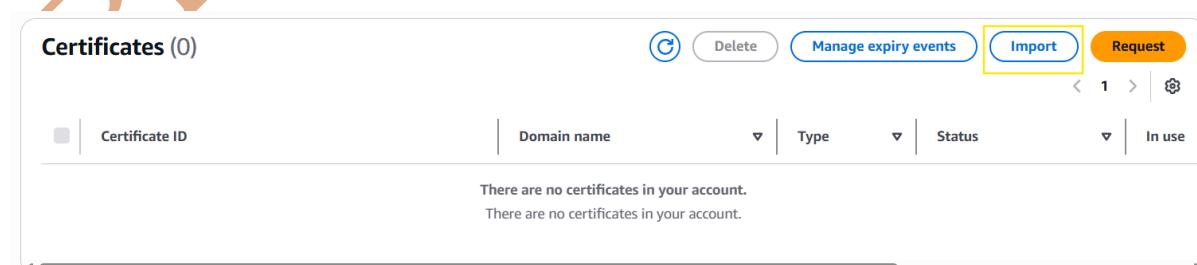
```
[root@yellow ~]# kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          yellow
mysql-1   1/1     Running   0          yellow
mysql-2   1/1     Running   0          yellow
```

```
Containers:
  mysql:
    Container ID: containerd://[REDACTED]
    Image: docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID: docker.io/bitnami/mysql@sha256:[REDACTED]
    Port: 3306/TCP
    Host Port: 0/TCP
    SeccompProfile: RuntimeDefault
    State:
      Started: 2025-06-14T10:25:41Z
      Ready: True
    Restart Count: 0
    Limits:
      cpu: 800m
      memory: 850Mi
    Requests:
      cpu: 800m
      memory: 850Mi
```

I checked the logs of vpa-admission-controller pod present in the namespace kube-system as found the entry as shown below which matched with the mysql container cpu and memory as shown in the screenshot attached above.

```
[root@[REDACTED] ~]# kubectl logs -f vpa-admission-controller-[REDACTED] -n kube-system
I0615 [REDACTED] 1 handler.go:95] "Processing vpa" vpa={"kind":"VerticalPodAutoscaler","apiVersion":"autoscaling.k8s.io/v1","metadata":{"name":"mysql-vpa","namespace":"mysql","creationTimestamp":null,"annotations":{"kubectl.kubernetes.io/last-applied-configuration":{"apiVersion":"autoscaling.k8s.io/v1","kind":"VerticalPodAutoscaler"},"metadata":{"annotations":{},"name":"mysql-vpa","namespace":"mysql"}, "spec":{"resourcePolicy":[{"containerName":"mysql","controlledResources":["cpu","memory"], "maxAllowed":{"cpu":1024m,"memory":1024Mi}, "minAllowed":{"cpu":800m,"memory":850Mi}}, {"targetRef":{"apiVersion":"apps/v1","kind":"StatefulSet","name":"mysql"}, "updatePolicy":{"updateMode":"Auto","resourcePolicy":{"containerPolicies":[{"containerName":"mysql","minAllowed":{"cpu":800m,"memory":850Mi}}, {"maxAllowed":{"cpu":1024m,"memory":1Gi}], "controlledResources":["cpu","memory"]}}}, "status":{}}}
I0615 [REDACTED] 1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-1"
I0615 [REDACTED] 1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/memory","value":850Mi}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":850Mi}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":mysql}]
I0615 [REDACTED] 1 handler.go:81] "Admitting pod" pod="mysql/mysql-2"
I0615 [REDACTED] 1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-2"
I0615 [REDACTED] 1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/memory","value":850Mi}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":850Mi}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":mysql}]
I0615 [REDACTED] 1 handler.go:81] "Admitting pod" pod="mysql/mysql-0"
I0615 [REDACTED] 1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-0"
I0615 [REDACTED] 1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/memory","value":850Mi}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":800m}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":850Mi}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":mysql}]
```

Finally, Pods had been scaled as shown in the screenshots attached above. In this project (Module-1), I had used the SSL Certificate Provided by AWS Certificate Manager which is DNS Verified. The DNS Verified SSL Certificate will be Auto-Renewed. You can also use the SSL Certificate provided by the SSL Certificate Provider with AWS Certificate Manager, in that case you need to import it as shown in the screenshot attached below.



Certificate body

```
-----END CERTIFICATE-----
```

Certificate private key

```
-----END PRIVATE KEY-----
```

Certificate chain - optional | [Info](#)
Paste the PEM-encoded certificate chain below.

Tags [Info](#)
No tags associated with the resource.
[Add new tag](#)
You can add up to 50 tags.

Certificate status

Identifier [REDACTED]

ARN arn:aws:acm:us-east-2:02[REDACTED]6:certificate/[REDACTED]

Type Imported

Status Issued

Domains [REDACTED]

Domain
bankapp.singhritesh85.com

You can renew an imported SSL Certificate in AWS Certificate Manager, for this to happen firstly get the new certificate from your Certificate issuer then manually reimport it into ACM as shown in the screenshot attached below. Finally provide the certificate body and the private key of the new certificate which you obtained from the certificate issuer and it will be renewed. The advantage of renewing an imported SSL Certificate in AWS Certificate manager is its ARN (Amazon Resource Name) will not change. So, wherever you were using it earlier you can use now without any service disruption.

Certificate status

Identifier [REDACTED]

ARN arn:aws:acm:us-east-2:02[REDACTED]6:certificate/[REDACTED]

Type Imported

Status Issued

Domains [REDACTED]

Domain
bankapp.singhritesh85.com

Reimport **Delete**

Usage of EKS-1.33 to support the feature-gate **InPlacePodVerticalScaling**. The feature-gate InPlacePodVerticalScaling will dynamically adjust the CPU and memory of the running pod without needing to recreate it. The InPlacePodVerticalScaling feature-gate is supported by EKS 1.33. In the demonstration written above I used EKS 1.30 either you can migration EKS from 1.30 -> 1.31 -> 1.32 -> 1.33 or you can directly use EKS 1.33 from the initial stage instead EKS 1.30 Cluster. If you are migrating EKS from 1.30 then you cannot migrate directly from 1.30 to 1.33 you need to migrate it to 1.31 first then 1.32 then to 1.33. The Terraform script I used to create the EKS Cluster is available in GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git> at the path **terraform-autoscale-karpenter-hpa-vpa**, you can do below changes to create the EKS Cluster 1.33 using this Terraform script.

```

ami_type = ["AL2023_x86_64_STANDARD", "AL2_x86_64", "CUSTOM"]
release_version = ["1.23.16-20230217", "1.24.10-20230217", "1.25.6-20230217", "1.26.12-20240110", "1.27.9-20240110", "1.28.5-20240110", "1.29.8-20240110", "1.30.4-20240110"]
kubernetes_version = ["1.23", "1.24", "1.25", "1.26", "1.27", "1.28", "1.29", "1.30", "1.31", "1.32", "1.33"]
capacity_type = ["ON_DEMAND", "SPOT"]
env = ["dev", "stage", "prod"]
ebs_csi_name = "aws-ebs-csi-driver"
ebs_csi_version = ["v1.39.0-eksbuild.1", "v1.28.0-eksbuild.1", "v1.27.0-eksbuild.1", "v1.26.1-eksbuild.1", "v1.25.0-eksbuild.1"] ##### v1.21.0-eksbuild.1
addon_version_garduddiy = ["v1.18.0-eksbuild.2", "v1.18.0-eksbuild.1", "v1.8.1-eksbuild.2", "v1.4.1-eksbuild.2", "v1.3.1-eksbuild.1", "v1.2.0-eksbuild.3"]
addon_version_kubeproxy = ["v1.30.9-eksbuild.3", "v1.28.15-eksbuild.9", "v1.27.18-eksbuild.2", "v1.27.8-eksbuild.4", "v1.27.8-eksbuild.1", "v1.27.6-eksbuild.2"]
addon_version_vpc_cni = ["v1.19.5-eksbuild.1", "v1.19.2-eksbuild.5", "v1.16.3-eksbuild.2", "v1.16.0-eksbuild.1", "v1.15.5-eksbuild.1", "v1.15.1-eksbuild.1"]
addon_version_coredns = ["v1.11.4-eksbuild.2", "v1.10.1-eksbuild.18", "v1.10.1-eksbuild.7", "v1.10.1-eksbuild.6", "v1.10.1-eksbuild.5", "v1.10.1-eksbuild.4"]
csi_snapshot_controller_version = ["v8.2.0-eksbuild.1", "v8.1.0-eksbuild.2", "v8.1.0-eksbuild.1", "v7.0.1-eksbuild.1", "v6.3.2-eksbuild.1"]
addon_version_observability = ["v4.1.0-eksbuild.1", "v4.0.0-eksbuild.1", "v3.7.0-eksbuild.1", "v3.5.0-eksbuild.1"]
addon_version_podidentityagent = ["v1.3.7-eksbuild.2", "v1.3.5-eksbuild.2", "v1.3.4-eksbuild.1", "v1.3.0-eksbuild.1", "v1.2.0-eksbuild.1"]
addon_version_metrics_server = ["v0.7.2-eksbuild.3", "v0.7.2-eksbuild.2", "v0.7.2-eksbuild.1"]

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

resource "aws_eks_node_group" "eksnodes" {
  cluster_name  = "${var.eks_cluster}-${var.env}"
  node_group_name = "${var.node_group_name}-${var.env}"
  node_role_arn = aws_iam_role.eksnoderole.arn
  subnet_ids   = aws_subnet.private_subnet.*.id  ##### element(aws_subnet.private_subnet[*].id, count.index) #var.subnet_ids

#  subnet_ids = ["subnet-", "subnet-", "subnet-"] ##### Private Subnet List for Private EKS NodeGroup
#  instance_types = [ var.instance_types[1] ]
#  disk_size     = var.disk_size
#  ami_type       = var.ami_type[0]
#  capacity_type = var.capacity_type[0]
#  release_version = var.release_version[10]
}

```

You can check the feature-gates enabled using the command **kubectl get --raw /metrics | grep kubernetes_feature_enabled**.

```
[root@ ~]# kubectl get --raw /metrics | grep kubernetes_feature_enabled | grep InPlacePodVerticalScaling
kubernetes_feature_enabled{name="InPlacePodVerticalScaling",stage="BETA"} 1
kubernetes_feature_enabled{name="InPlacePodVerticalScalingAllocatedStatus",stage="DEPRECATED"} 0
kubernetes_feature_enabled{name="InPlacePodVerticalScalingExclusiveCPUs",stage="ALPHA"} 0
```

As I informed earlier **InPlacePodVerticalScaling** allows dynamically adjust the CPU and memory resources of a running pod without restart it. Now the question arises how to change the CPU and memory of a running pod without restarting the pod. To achieve this either you run the kubectl patch command with the pod to edit the CPU and memory or kubectl edit command to edit CPU and memory of a running pod with **--subresource=resize** (the prerequisites is kubectl version should be greater than or equal to 1.32) as shown in the screenshot attached below.

```
[root@ ~]# kubectl edit pod mysql-0 -n mysql --subresource=resize
```

Initially the CPU and memory was 650m and 650Mi, I changed it to 1.5 and 1.5Gi (for limits) and 1 and 1Gi for requests as shown in the screenshot attached below.

```

resizePolicy:
- resourceName: cpu
  restartPolicy: NotRequired
- resourceName: memory
  restartPolicy: NotRequired
resources:
limits:
  cpu: 650m
  memory: 650Mi
requests:
  cpu: 650m
  memory: 650Mi

```

I did the changes as shown in the screenshot attached below.

```

resizePolicy:
- resourceName: cpu
  restartPolicy: NotRequired
- resourceName: memory
  restartPolicy: NotRequired
resources:
limits:
  cpu: 1.5
  memory: 1.5Gi
requests:
  cpu: 1
  memory: 1Gi

```

```
[root@[REDACTED] ~]# kubectl edit pod mysql-0 -n mysql --subresource=resize
pod/mysql-0 edited
```

After updating CPU and memory, I checked the CPU and memory as shown in the screenshot attached below and found changes had been reflected in the pod.

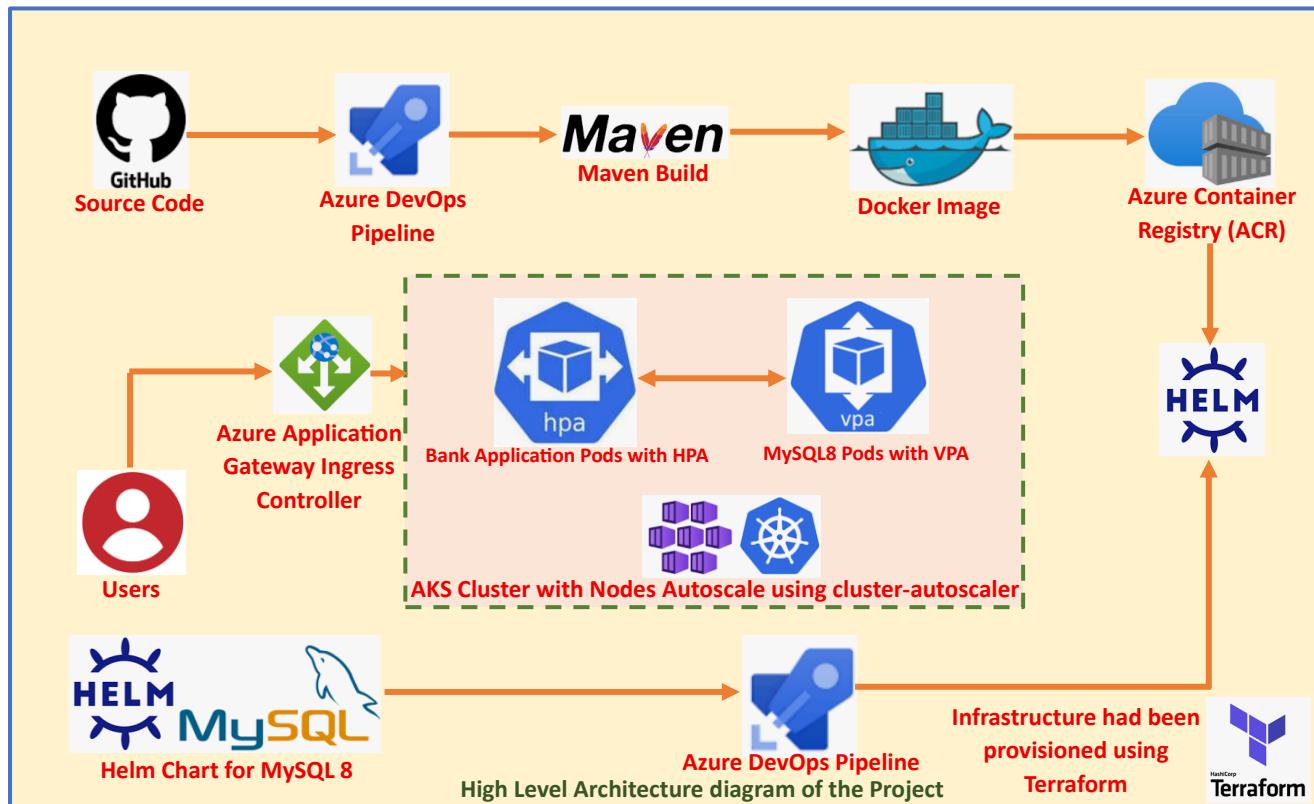
```

Ready:          True
Restart Count: 0
Limits:
  cpu:    1500m
  memory: 1536Mi
Requests:
  cpu:    1
  memory: 1Gi

```

```
[root@[REDACTED] ~]# kubectl get pods -n mysql
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          20m
mysql-1   1/1     Running   0          19m
mysql-2   1/1     Running   0          18m
```

**DevOps Project Kubernetes based Cluster-Autoscaler-Vertical-Pod-Autoscaling (VPA),
Horizontal-Pod-Autoscaling (HPA) Monitoring and Log Aggregation using Azure Monitor**



As shown in the architecture diagram drawn above the source code for BankApp was kept in the GitHub Repo which was deployed using the Azure DevOps Pipeline. Maven was used as a Build Tool and Docker Image was created then pushed to ACR and then Bank Application Pods had been created using the Helm. To deploy MySQL 8 Pods, I used the bitnami helm chart which was present in the GitHub Repo <https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git>. As shown in the architecture diagram drawn above MySQL 8 Pods was deployed using Helm. AKS Cluster Autoscaler was used as AKS Nodes Autoscaler. To Autoseale Bank Application HPA and to autoscale MySQL 8 Pods VPA had been used. Finally, users will access the Bank Application through the Application Gateway Ingress Controller and hence the Kubernetes service for Bank Application. The HOST generated along with the Public IP Address was provided to Azure DNS Zone to create the Record Set of A-Type. Finally, users were able to access the Two-Tier Bank Application using the generated URL.

I installed Azure CLI and used **az login** to authenticate and authorize the Azure Account with Terraform.

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

Copied and pasted the generated URL for the page and entered the generated code (after running the command **az login**) in web browser page to sign-in into Azure Account.

I had provisioned the infrastructure using Terraform and the terraform script is present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git> at the path **terraform-cluster-autoscale-hpa-vpa**.

For Terraform I used Azure Storage Account Container to store the terraform state file and to achieve the state lock. Below Screenshot shows how I ran the terraform script and created the Resources in Azure.

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

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

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

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

```
module.aks.azurerm_monitor_metric_alert_rule3: Still creating... [01m40s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m40s elapsed]
module.aks.azurerm_monitor_metric_alert_rule1: Still creating... [01m40s elapsed]
module.aks.azurerm_monitor_metric_alert_rule2: Still creating... [01m50s elapsed]
module.aks.azurerm_monitor_metric_alert_rule3: Still creating... [01m50s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m50s elapsed]
module.aks.azurerm_monitor_metric_alert_rule1: Still creating... [01m50s elapsed]
module.aks.azurerm_monitor_metric_alert_rule2: Still creating... [02m00s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert_rule1: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert_rule2: Creation complete after 2m2s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Insights/metricAlerts/bankapp-alert-rule2]
module.aks.azurerm_monitor_metric_alert_rule3: Creation complete after 2m2s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Insights/metricAlerts/bankapp-alert-rule3]
module.aks.azurerm_monitor_metric_alert_rule1: Creation complete after 2m2s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.Insights/metricAlerts/bankapp-alert-rule1]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Creation complete after 2m7s [id=/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster/agentPools/userpool]
module.aks.null_resource.kubectl: Creating...
module.aks.null_resource.kubectl: Provisioning with 'local-exec'...
module.aks.null_resource.kubectl (local-exec): Executing: ["./bin/bash" "-c" "az account set --subscription $(az account show --query id|tr -d '\"') && az aks get-credentials --resource-group bankapp-rg --name bankapp-cluster --overwrite-existing && chmod 600 ~/.kube/config"]
module.aks.null_resource.kubectl (local-exec): WARNING: Merged "bankapp-cluster" as current context in /root/.kube/config
module.aks.null_resource.kubectl: Creation complete after 3s [REDACTED]

Apply complete! Resources: 51 added, 0 changed, 0 destroyed.

Outputs:

acr_azurevm_private_ip_and_aks_details = {
  "acr_login_server" = "bankappcontainer24registry.azurecr.io"
  "aks_id" = "/subscriptions/[REDACTED]/resourceGroups/bankapp-rg/providers/Microsoft.ContainerService/managedClusters/bankapp-cluster"
  "aks_name" = "bankapp-cluster"
  "azurevm_devopsagent_privateip" = "10.[REDACTED]"
}
```

Initially I kept the Synthetic Monitoring and Alert for Synthetic Monitoring in disabled mode after creating the Pods for MySQL8 and bankapp I will change it to enable mode otherwise email will be sent to the Group Email ID.

The Aks Cluster and the Terraform Server existed in different VNet if you want to access the AKS Cluster using the kubectl command from the Terraform Server then first you need to create the Virtual Network Link and VNet Peering between Terraform Server VNet and AKS Cluster VNet as shown in the screenshot attached below.

Home > Resource groups > bankapp-rg > [REDACTED].privatelink.eastus2.azurek8s.io | Virtual Network Links >

Add Virtual Network Link ...

[REDACTED].privatelink.eastus2.azurek8s.io

Link name *

dexter

Virtual network details

i Only virtual networks with Resource Manager deployment model are supported for linking with Private DNS zones. Virtual networks with Classic deployment model are not supported.

I know the resource ID of virtual network i

Subscription *

Pay-As-You-Go

Virtual Network *

Terraform-Server-vnet (ritesh)

Configuration

Enable auto registration i

Enable fallback to internet i

Create

Cancel

i Give feedback

Home > Resource groups > bankapp-rg > [REDACTED].privatelink.eastus2.azurek8s.io | Virtual Network Links

Private DNS zone

Link Name	Link Status	Virtual Network	Auto-Registration	Fallback to Internet
bankapp-cluster-dns-link	Completed	bankapp-vnet	Disabled	Disabled
dexter	Completed	Terraform-Server-vnet	Disabled	Disabled

Ritesh

Home > Network foundation | Virtual networks > Terraform-Server-vnet | Peerings >

Add peering

Terraform-Server-vnet

Virtual network peering enables you to seamlessly connect two or more virtual networks in Azure. This will allow resources in either virtual network to directly connect and communicate with resources in the peered virtual network.

Remote virtual network summary

Peering link name *	<input type="text" value="peer-2"/>
Virtual network deployment model ⓘ	<input checked="" type="radio"/> Resource manager <input type="radio"/> Classic
I know my resource ID ⓘ	<input type="checkbox"/>
Subscription *	<input type="text" value="Pay-As-You-Go"/>
Virtual network *	<input type="text" value="bankapp-vnet (bankapp-rg)"/>

Remote virtual network peering settings

Allow 'bankapp-vnet' to access 'Terraform-Server-vnet'

Add

Cancel

Home > Network foundation | Virtual networks > Terraform-Server-vnet | Peerings >

Add peering

Terraform-Server-vnet

Server

Local virtual network summary

Peering link name *	<input type="text" value="peer-1"/>
---------------------	-------------------------------------

Local virtual network peering settings

Allow 'Terraform-Server-vnet' to access 'bankapp-vnet'

Allow 'Terraform-Server-vnet' to receive forwarded traffic from 'bankapp-vnet'

Allow gateway or route server in 'Terraform-Server-vnet' to forward traffic to 'bankapp-vnet'

Enable 'Terraform-Server-vnet' to use 'bankapp-vnet's' remote gateway or route server

Add

Cancel

The screenshot shows two separate Azure Virtual Network Peering configurations:

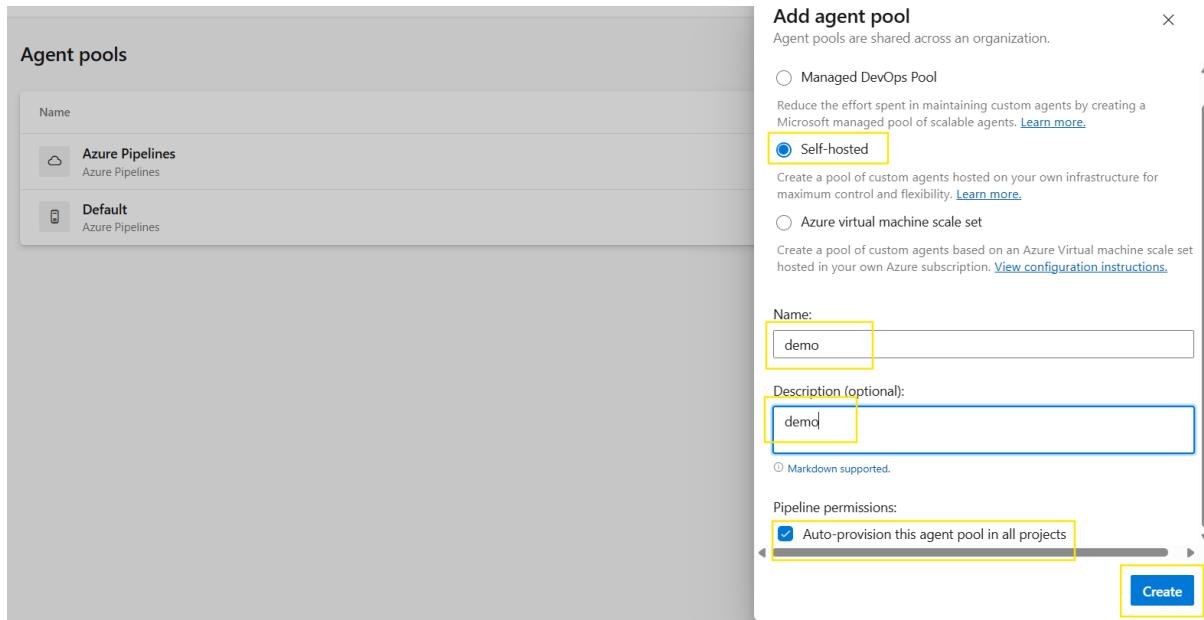
- Terraform-Server-vnet | Peering:** Shows one peering entry named "peer-1". Status: Fully Synchronized, Peering state: Connected, Remote virtual network name: bankapp-vnet, Virtual network: Terraform-Server-vnet, Cross-tenant: No.
- bankapp-vnet | Peering:** Shows one peering entry named "peer-2". Status: Fully Synchronized, Peering state: Connected, Remote virtual network name: Terraform-Server-vnet, Virtual network: bankapp-vnet, Cross-tenant: No.

```
[root@Terraform-Server ~]# kubectl get nodes
NAME                               STATUS   ROLES      AGE     VERSION
aks-agentpool-[REDACTED]-vmss000000   Ready    <none>    94m    v1.33.0
aks-userpool-[REDACTED]-vmss000000   Ready    <none>    87m    v1.33.0
```

Installation of Self-hosted Azure DevOps Agent

To install Azure DevOps Self-host agent, I followed the steps as mentioned below.

Open the Azure DevOps URL and then go to the **Organization Settings > Agent Pools > Add Pools > Add Agent Pool** and the Add the new Agent Pool as shown in the screenshot attached below.



In Agent Pool the select the newly created Agent and Add the New Agent using the below mentioned steps.

Then run the commands on your Azure DevOps Self-hosted Agent which was created using the Terraform as shown in the screenshot attached below.

```
[ritesh@devopsagent-vm ~]$ sudo -i
[root@devopsagent-vm ~]# cd /opt/
[root@devopsagent-vm opt]# mkdir myagent && cd myagent
[root@devopsagent-vm myagent]# wget https://download.agent.dev.azure.com/agent/4.255.0/vsts-agent-linux-x64-4.255.0.tar.gz
--2025 [100%] - https://download.agent.dev.azure.com/agent/4.255.0/vsts-agent-linux-x64-4.255.0.tar.gz
Resolving download.agent.dev.azure.com (download.agent.dev.azure.com)... [REDACTED] [REDACTED] [REDACTED] ...
Connecting to download.agent.dev.azure.com (download.agent.dev.azure.com) |[REDACTED]|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147552820 (141M) [application/octet-stream]
Saving to: 'vsts-agent-linux-x64-4.255.0.tar.gz'

vsts-agent-linux-x64-4.255.0.tar.gz    100%[=====] 140.72M   274MB/s   in 0.5s
2025 [REDACTED] (274 MB/s) - 'vsts-agent-linux-x64-4.255.0.tar.gz' saved [147552820/147552820]

[root@devopsagent-vm myagent]# tar -xvf vsts-agent-linux-x64-4.255.0.tar.gz

[root@devopsagent-vm myagent]# chown -R demo:demo /opt/myagent/
[root@devopsagent-vm myagent]# ./bin/installdependencies.sh
```

```
[root@devopsagent-vm myagent]# su - demo
[demo@devopsagent-vm ~]$ cd /opt/myagent/
[demo@devopsagent-vm myagent]$ ./config.sh

agent v4.255.0          (commit [REDACTED])

>> End User License Agreements:
Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building 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 > https://dev.azure.com/[REDACTED]
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

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

Enter work folder (press enter for _work) >
2025-[REDACTED]: Settings Saved.
[demo@devopsagent-vm myagent]$ sudo ./svc.sh install
Creating launch agent in /etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service
Run as user: demo
Run as uid: 1001
gid: 1001
/sbin/sestatus
SELinux status:           enabled
Created symlink /etc/systemd/system/multi-user.target.wants/vsts.agent.[REDACTED].demo.demo.service → /etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service.
[demo@devopsagent-vm myagent]$ sudo ./svc.sh start

/etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service
● vsts.agent.[REDACTED].demo.demo.service - Azure Pipelines Agent ([REDACTED].demo.demo)
   Loaded: loaded (/etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2025 [REDACTED] UTC; 13ms ago
     Main PID: [REDACTED] (runsvc.sh)
       Tasks: 2 (limit: [REDACTED])
      Memory: 896.0K
        CGroup: /system.slice/vsts.agent.[REDACTED].demo.demo.service
                  └─[REDACTED] /bin/bash /opt/myagent/runsvc.sh
                      ├─[REDACTED] ./node20_1/bin/node ./bin/AgentService.js

Jun 22 06:43:29 devopsagent-vm systemd[1]: Started Azure Pipelines Agent ([REDACTED].demo.demo).
Jun 22 06:43:29 devopsagent-vm runsvc.sh[14089]: .path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/sonar-scanner/bin:/opt/apache-maven/bin:/local/bin
Jun 22 06:43:29 devopsagent-vm runsvc.sh[14092]: v20.18.2
Hint: Some lines were ellipsized, use -l to show in full.
[demo@devopsagent-vm myagent]$ sudo ./svc.sh status

/etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service
● vsts.agent.singhriteshkumar2510554.demon.demo.service - Azure Pipelines Agent ([REDACTED].demo.demo)
   Loaded: loaded (/etc/systemd/system/vsts.agent.[REDACTED].demo.demo.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2025 [REDACTED] UTC; 4s ago
     Main PID: [REDACTED] (runsvc.sh)
       Tasks: 25 (limit: [REDACTED])
      Memory: 60.8M
        CGroup: /system.slice/vsts.agent.[REDACTED].demo.demo.service
                  ├─[REDACTED] /bin/bash /opt/myagent/runsvc.sh

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

Now the Self-hosted Azure DevOps Agent is in Online mode as shown in the screenshot attached below.

The screenshot shows the 'Agents' tab selected in the Azure DevOps interface. A single agent named 'demo' is listed, showing it is currently 'Online'. The 'Enabled' switch is turned on. Other tabs like 'Jobs', 'Details', 'Security', 'Settings', 'Maintenance History', and 'Analytics' are also visible.

Name	Last run	Current status	Agent version	Enabled
demo ● Online		Idle	4.255.0	<input checked="" type="checkbox"/> On

Then I went to Project in the **Azure DevOps Organization > Project Settings > Service connections** and created a new service connection for Azure Container Registry and GitHub Account as shown in the screenshot attached below.

New service connection

Choose a service or connection type

Docker Registry

Docker Registry

[Learn more](#)

[Next](#)

New Docker Registry service connection

Registry type

Docker Hub Others Azure Container Registry

Docker Registry

https://bankappcontainer24registry.azurecr.io

Docker ID

bankappcontainer24registry

Docker Password

.....

Email (optional)

Service connection details

Service Connection Name

Docker-Registry

Description (optional)

Description (optional)

Security

- Grant access permission to all pipelines

[Learn more](#)
[Troubleshoot](#)
[Back](#)
[Save](#)

x

Service connections

Convert your existing Azure Resource Manager service connections which use secrets to authenticate to leverage improved security and simplified maintenance.

Filter by keywords

Service Connection
Docker-Registry

New GitHub service connection

Authentication method

Grant authorization

Personal Access Token

OAuth Configuration

AzurePipelines

GitHub Change

Service connection details

Service Connection Name

Description (optional)

Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)

Back Save

Then I had created the Azure DevOps Pipeline first for mysql and then for bankapp and executed in them in the same order.

- [my-demo-project](#)
- [Overview](#)
- [Boards](#)
- [Repos](#)
- [Pipelines](#)
- [Pipelines](#)
- [Environments](#)
- [Releases](#)
- [Library](#)
- [Task groups](#)
- [Deployment groups](#)
- [Test Plans](#)
- [Artifacts](#)
- [Project settings](#)

Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

[Create Pipeline](#)

Connect Select Configure Review

New pipeline

Where is your code?



Azure Repos Git YAML

Free private Git repositories, pull requests, and code search



Bitbucket Cloud YAML

Hosted by Atlassian



GitHub YAML

Home to the world's largest community of developers



GitHub Enterprise Server YAML

The self-hosted version of GitHub Enterprise



Other Git

Any generic Git repository



Subversion

Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.



✓ Connect

Select

Configure

Review

New pipeline

Select a repository

Filter by keywords

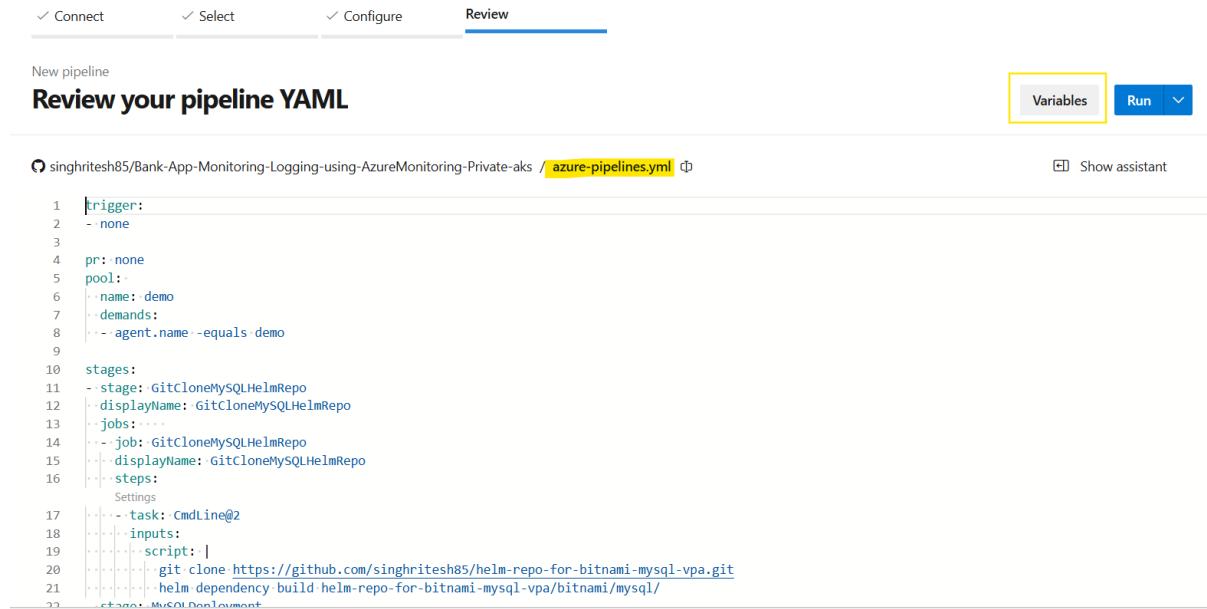
My repositories ▼



singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks private

3m ago





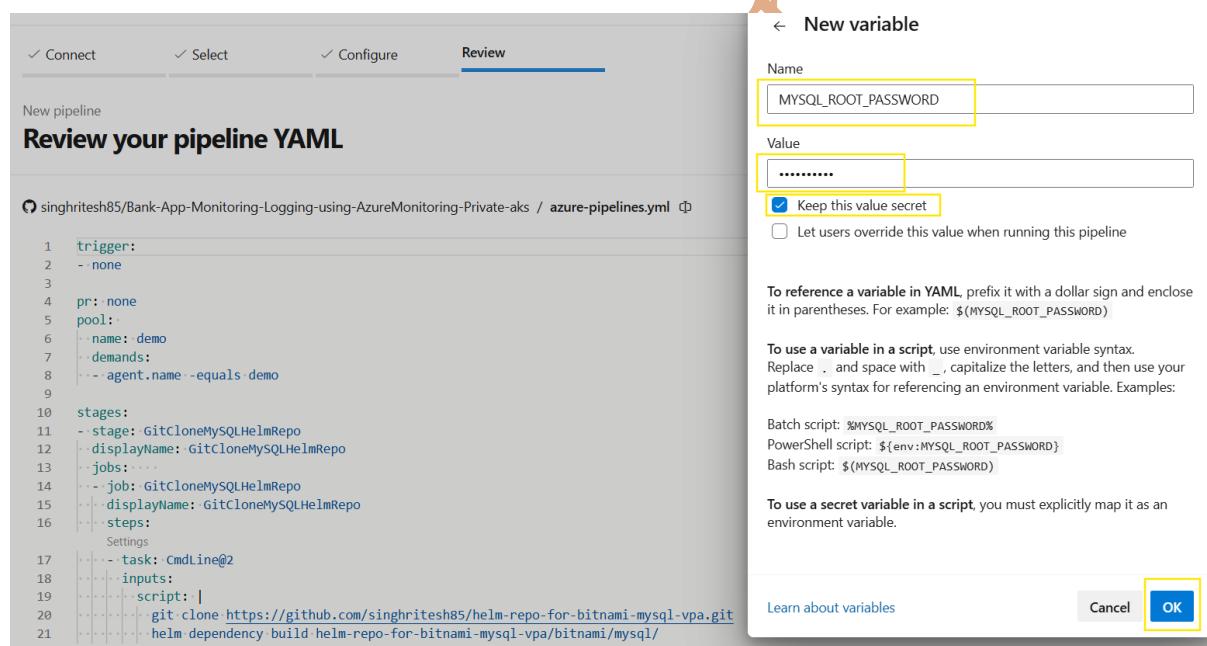
The screenshot shows the 'Review' step of a new pipeline named 'Review your pipeline YAML'. The pipeline configuration file is 'azure-pipelines.yml'. The code is as follows:

```

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: GitCloneMySQLHelmRepo
12     displayName: GitCloneMySQLHelmRepo
13     jobs:
14       - job: GitCloneMySQLHelmRepo
15         displayName: GitCloneMySQLHelmRepo
16         steps:
17           - task: CmdLine@2
18             inputs:
19               script: |
20                 git clone https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git
21                 helm dependency build helm-repo-for-bitnami-mysql-vpa/bitnami/mysql/
22             stage: MySQLDevelopment

```

Before running this pipeline, I created two variables `MYSQL_ROOT_PASSWORD` and `MYSQL_DATABASE` as shown in the screenshot attached below.



The screenshot shows the 'New variable' dialog in the 'Review' step of the pipeline. A variable named `MYSQL_ROOT_PASSWORD` is being created with a value of '*****'. The 'Keep this value secret' checkbox is checked. The dialog also includes instructions for referencing variables in YAML and examples for different scripts.

Name	<code>MYSQL_ROOT_PASSWORD</code>
Value	*****
<input checked="" type="checkbox"/> Keep this value secret	
<input type="checkbox"/> 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: <code>\$(MYSQL_ROOT_PASSWORD)</code>	
To use a variable in a script, use environment variable syntax. Replace <code>.</code> and space with <code>_</code> , capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:	
Batch script: <code>\$MYSQL_ROOT_PASSWORD%</code> PowerShell script: <code>\$(env:MYSQL_ROOT_PASSWORD)</code> Bash script: <code>\$(MYSQL_ROOT_PASSWORD)</code>	
To use a secret variable in a script, you must explicitly map it as an environment variable.	
Learn about variables	
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

The screenshot shows the Azure Pipelines interface for reviewing a pipeline YAML file. The top navigation bar includes 'Connect', 'Select', 'Configure', 'Review' (which is selected), and 'Variables'. The pipeline name is 'Review your pipeline YAML'.

Variables panel:

- Search bar: 'Search variables' with a '+' button.
- Variable: 'MYSQL_ROOT_PASSWORD' with value '*****'.
- Buttons: 'Learn about variables', 'Cancel', and 'Save'.

New variable dialog:

- Name: 'MYSQL_DATABASE' (highlighted with a yellow box).
- Value: '*****' (highlighted with a yellow box).
- Checkboxes:
 - Keep this value secret (highlighted with a yellow box).
 - Let users override this value when running this pipeline.
- Text: 'To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example: \$(MYSQL_DATABASE)'.
- Text: '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:'.
- Text: 'Batch script: %MYSQL_DATABASE%'
'PowerShell script: \${env:MYSQL_DATABASE}'
'Bash script: \${MYSQL_DATABASE}'
- Text: 'To use a secret variable in a script, you must explicitly map it as an environment variable.'
- Buttons: 'Learn about variables', 'Cancel', and 'OK' (highlighted with a yellow box).

pipeline panel:

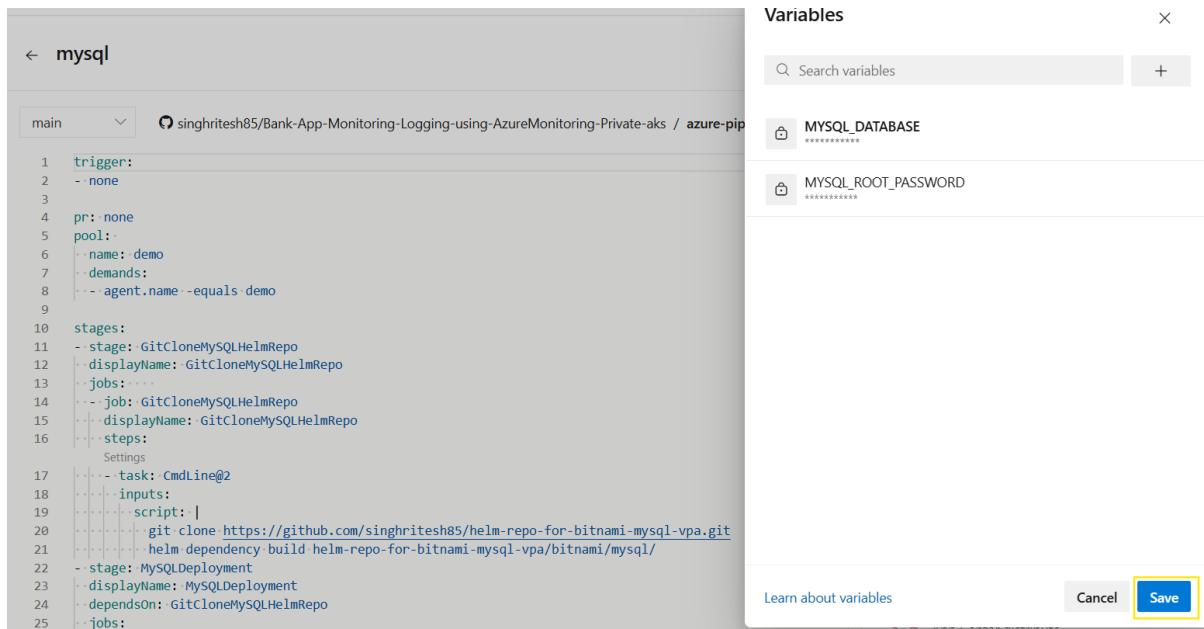
```

trigger:
- none

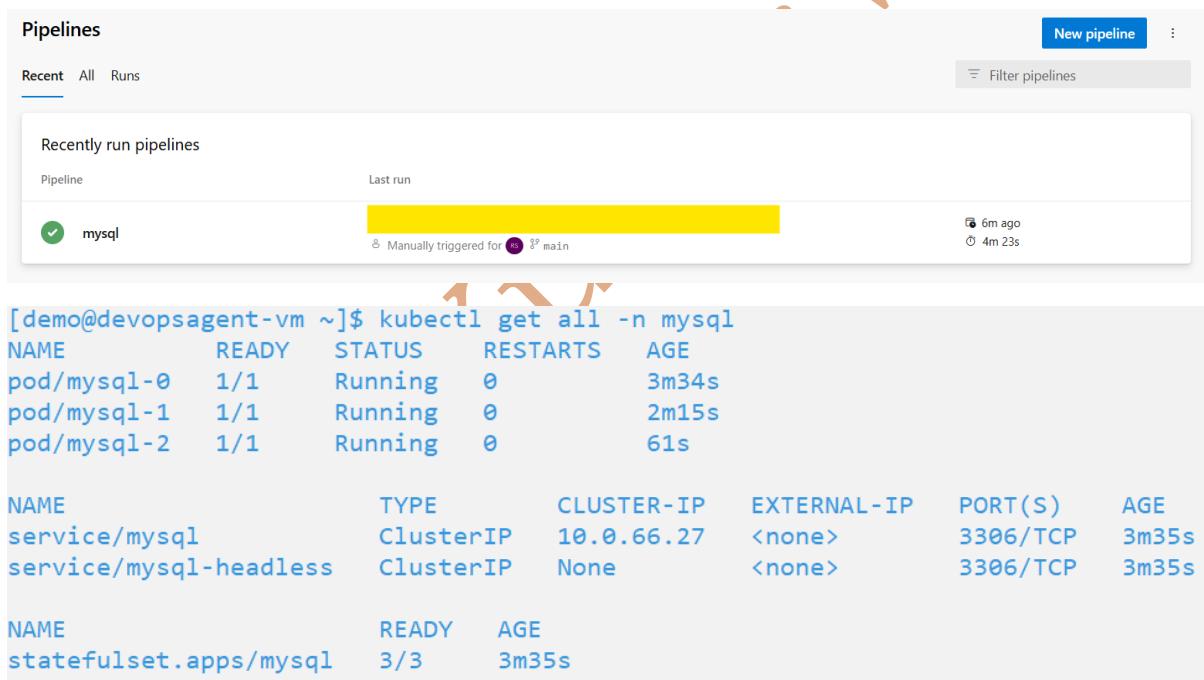
pr: none
pool:
- name: demo
- demands:
  - agent.name -equals demo

stages:
- stage: GitCloneMySQLHelmRepo
  displayName: GitCloneMySQLHelmRepo
  jobs:
  - job: GitCloneMySQLHelmRepo
    displayName: GitCloneMySQLHelmRepo
    steps:
    - task: CmdLine@2
      inputs:
        script: |
          git clone https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git
          helm dependency build helm-repo-for-bitnami-mysql-vpa/bitnami/mysql/
  - stage: MySQLDeployment
    displayName: MySQLDeployment
    dependsOn: GitCloneMySQLHelmRepo
    jobs:

```



Then I ran this Azure DevOps Pipeline, its screenshot after its successful execution is as shown below.



Before creating and running the Azure DevOps Pipeline I create kubernetes secrets for ACR (Azure Container Registry) as shown in the screenshot attached below.

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

```
[root@DESKTOP-1D9FVHJ ~]# kubectl create ns bankapp
namespace/bankapp created
[root@DESKTOP-1D9FVHJ ~]# kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappcontainer24registry.azurecr.io --docker-username=bankappcontainer24registry --docker-password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -n bankapp
secret/bankapp-auth created
```

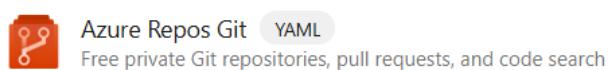
Then I created and ran the Azure DevOps Pipeline for BankApp as shown in the screenshot attached below.

The screenshot shows the Azure DevOps Pipelines interface. At the top, there's a header with 'Pipelines', 'New pipeline' (button), and a 'Filter pipelines' search bar. Below the header, there are tabs for 'Recent', 'All', and 'Runs'. Under 'Recently run pipelines', there's a card for a pipeline named 'mysql'. The card includes a green checkmark icon, the pipeline name, a yellow progress bar indicating the last run, a note that it was 'Manually triggered for main', and two timestamped entries: '7m ago' and '4m 23s'.

Connect **Select** **Configure** **Review**

New pipeline

Where is your code?



Use the classic editor to create a pipeline without YAML.

✓ Connect **Select** **Configure** **Review**

New pipeline

Select a repository

The screenshot shows the 'Select a repository' step of the pipeline creation wizard. It features a search bar labeled 'Filter by keywords' and a dropdown menu labeled 'My repositories'. A single repository card is visible, belonging to a user named 'singhrites85' with the repository name 'Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks'. The repository is marked as 'private' and was updated '20m ago'.

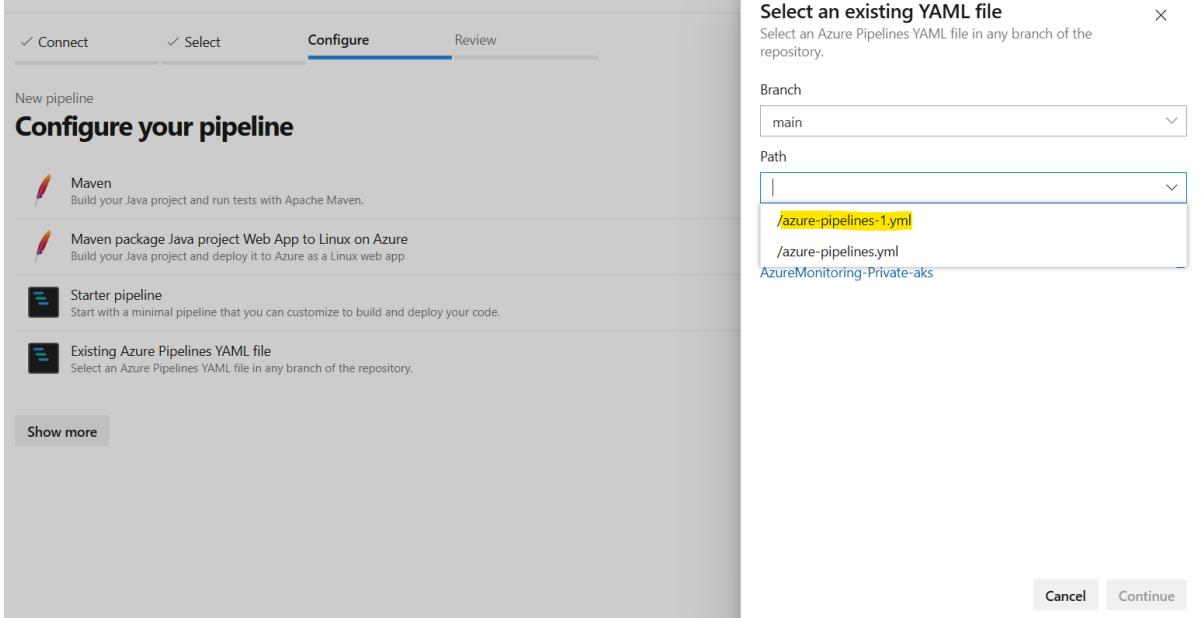
✓ Connect ✓ Select **Configure** Review

New pipeline

Configure your pipeline

-  **Maven**
Build your Java project and run tests with Apache Maven.
-  **Maven package Java project Web App to Linux on Azure**
Build your Java project and deploy it to Azure as a Linux web app
-  **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



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

Branch: main

Path:

- /azure-pipelines-1.yml
- /azure-pipelines.yml
- AzureMonitoring-Private-aks

Cancel **Continue**

The screenshot shows the 'Configure' step of a new pipeline creation process. On the left, a sidebar lists options like Maven, Maven package Java project Web App to Linux on Azure, Starter pipeline, and Existing Azure Pipelines YAML file. The 'Existing Azure Pipelines YAML file' option is selected. A modal dialog titled 'Select an existing YAML file' appears, prompting the user to choose a file from a repository. The 'Branch' dropdown is set to 'main' and the 'Path' dropdown is set to '/azure-pipelines-1.yml'. The path is highlighted with a yellow box. The 'Continue' button at the bottom right of the modal is also highlighted.

Configure your pipeline

- Maven
- Maven package Java project Web App to Linux on Azure
- Starter pipeline
- Existing Azure Pipelines YAML file

Show more

Select an existing YAML file

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

Branch: main

Path: /azure-pipelines-1.yml

singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks

Cancel Continue

The screenshot shows the 'Review' step where the YAML configuration is displayed. The code defines a pipeline with a trigger for the 'main' branch, no pull requests, and a pool named 'demo'. It includes variables for image pull secrets and stages for building the application. Below the code, a preview section shows two recent runs: 'bankapp' (7m ago) and 'mysql' (11m ago), both triggered manually. The 'Run' button is highlighted with a yellow box.

New pipeline

Review your pipeline YAML

```

1 trigger:
2 - main
3
4 pr: none
5 pool:
6   name: demo
7   demands:
8     - agent.name-equals demo
9
10 variables:
11   - imagePullSecret: 'bankapp-auth'
12
13 stages:
14   - stage: "Build"
15     displayName: Build
16     jobs:
17       - job: "Build"
18         displayName: Build
19         steps:
20           - task: Maven@4
21             inputs:

```

Variables Run

Recent All Runs

Pipelines

New pipeline

Recently run pipelines

Pipeline	Last run
bankapp	7m ago Manually triggered for main
mysql	11m ago Manually triggered for main

Filter pipelines

```
[demo@devopsagent-vm ~]$ kubectl get all -n bankapp
NAME                                         READY   STATUS    RESTARTS   AGE
pod/bankapp-folo-[REDACTED]                   1/1     Running   0          2m48s
service/bankapp-folo   ClusterIP  10.43.128.100 <none>        80/TCP   2m48s
deployment.apps/bankapp-folo      1/1     UP-TO-DATE 1           1          2m48s
replicaset.apps/bankapp-folo-[REDACTED] 1         DESIRED   CURRENT   READY   AGE
[REDACTED]                                     1         1         1         1         2m49s
```

The source code to deploy the Bankapp is present in the GitHub Repo <https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-aks.git>. To access the bankapp application I created the kubernetes secretees and Ingress Rule as shown in the screenshot attached below.

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

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

```

cat ingress-rule-azure.yaml

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

---

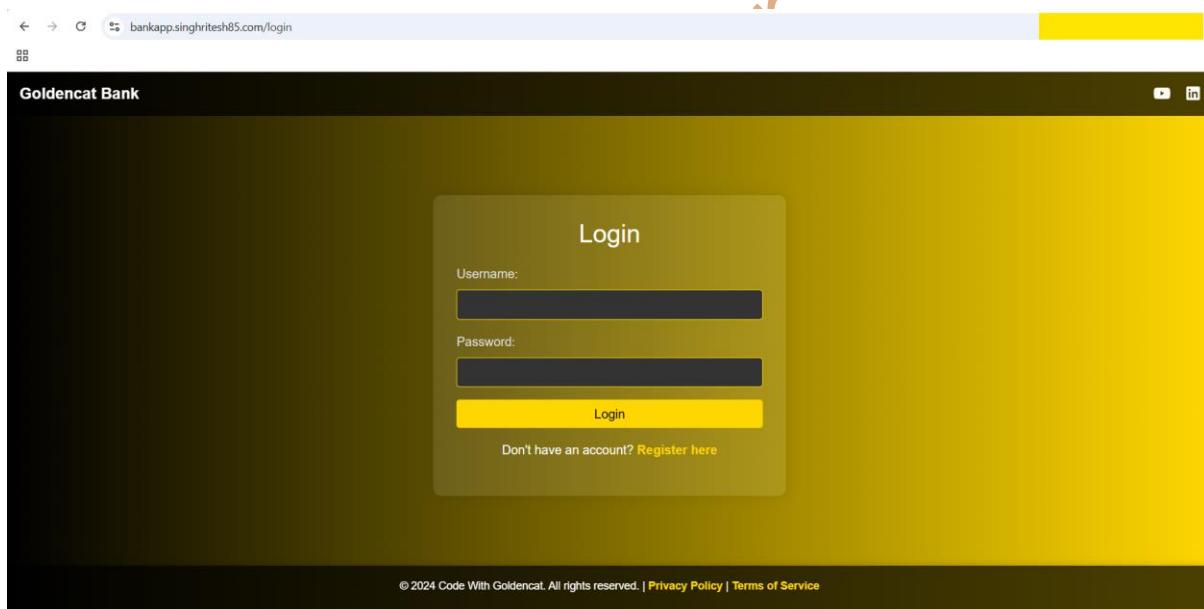
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-tls
  rules:
  - host: bankapp.singhritesh85.com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

```

I create the URL to access the bank application by providing the HOST and Public IP Address in Azure DNS Zone to create the record set of A-Type as shown in the screenshot attached below.

The screenshot shows the Azure DNS Management portal for the domain `singhritesh85.com`. On the left, the 'Recordsets' section is selected. A new record set is being created with the name `bankapp`, type `A - IPv4 Address records`, and TTL of 1 hour. The IP address is currently set to `0.0.0.0`.

At last, I was able to access the application using the URL as shown in the screenshot attached below.



I had registered a new user and logged-in with the same user and checked in the MySQL8 database the same user created as shown in the screenshot attached below.

The image displays three screenshots of a mobile banking application interface for Goldencat Bank, showing the registration, login, and dashboard pages.

Registration Page:

- The URL is bankapp.singhritesh85.com/register.
- The title is "Goldencat Bank".
- The main heading is "Register a New Account".
- Fields include "Username" (ritesh) and "Password".
- A "Register" button is highlighted with a red border.
- Text at the bottom says "Already have an account? [Login here](#)".
- Footer text: "© 2024 Code With Goldencat. All rights reserved. | [Privacy Policy](#) | [Terms of Service](#)".

Login Page:

- The URL is bankapp.singhritesh85.com/login.
- The title is "Goldencat Bank".
- The main heading is "Login".
- Fields include "Username" (ritesh) and "Password".
- A "Login" button is highlighted with a red border.
- Text at the bottom says "Don't have an account? [Register here](#)".
- Footer text: "© 2024 Code With Goldencat. All rights reserved. | [Privacy Policy](#) | [Terms of Service](#)".

Dashboard Page:

- The URL is bankapp.singhritesh85.com/dashboard.
- The title is "Goldencat Bank".
- The top navigation bar includes "Dashboard", "Transactions", and "Logout".
- Welcome message: "Welcome, ritesh".
- Current Balance: "\$0.00".
- Account Details box shows:
 - Account Number: 1
 - Account Type: Savings
- Action buttons: "Deposit", "Withdraw", and "Transfer Money".

```
[demo@devopsagent-vm ~]$ kubectl exec -it mysql-2 -n mysql -- mysql -h localhost -u root --password
Defaulted container "mysql" out of: mysql, preserve-logs-symlinks (init)
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 291
Server version: 8.4.0 Source distribution

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

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

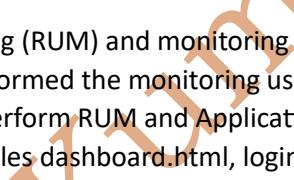
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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

Database changed
mysql> show tables;
+-----+
| Tables_in_bankappdb |
+-----+
| account           |
| transaction       |
+-----+
2 rows in set (0.01 sec)

mysql> select * from account;
+----+-----+-----+-----+
| id | balance | password            | username |
+----+-----+-----+-----+
| 1  | 80.00   | [REDACTED]          | ritesh   |
+----+-----+-----+-----+
```

I performed Real User Monitoring (RUM) and monitoring using Application Insight for the deployed Bank Application. I had also performed the monitoring using container insight for the container workload deployed to AKS. To perform RUM and Application Insight I added below shown JavaScript (Web) SDK Loader Script in the files dashboard.html, login.html, register.html and transactions.html.



[Bank-App-Monitoring-Logging-using-AzureMonitoring-aks /src/main/resources/templates/dashboard.html](#)

singhrites85 Add files via upload 60ee1c9 · 31 minutes ago History

Code Blame 215 lines (192 loc) · 13.2 KB

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <script type="text/javascript">
5      !(function (cfg){function e(){cfg.onInit&&cfg.onInit(n)}var x,w,D,t,E,n,C=window,O=document,b=C.location,q="script",I="ingestionendpoint",L="disableExceptionTrac
6      src: "https://js.monitor.azure.com/scripts/b/ai.3.gbl.min.js",
7      // name: "appInsights", // Global SDK Instance name defaults to "appinsights" when not supplied
8      // ld: 0, // Defines the load delay (in ms) before attempting to load the sdk. -1 = block page load and add to head. (default) = 0ms load after timeout,
9      // useXhr: 1, // Use XHR instead of fetch to report failures (if available),
10     // dle: true, // Prevent the SDK from reporting load failure log
11     // crossOrigin: "anonymous", // When supplied this will add the provided value as the cross origin attribute on the script tag
12     // onInit: null, // Once the application insights instance has loaded and initialized this callback function will be called with 1 argument -- the sdk instance (
13     // sri: false, // Custom optional value to specify whether fetching the snippet from integrity file and do integrity check
14     cfg: { // Application Insights Configuration
15         connectionString: "InstrumentationKey=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX;IngestionEndpoint=https://eastus2-3.in.applicationinsights.azure.com;LiveEndpoint=[REDACTED]
16     }};
17     </script>
18     <title>Dashboard - Goldencat Bank</title>
19     <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
20     <style>
```

In the similar way I had added the same JavaScript (Web) SDK Loader Script in the files login.html, register.html and transactions.html.

I had added the Env variable and application insight agent jar file in the Dockerfile as shown in the screenshot attached below.

Bank-App-Monitoring-Logging-using-AzureMonitoring-aks / Dockerfile-Project-1 [...](#)

 singhritesh85 Add files via upload 60ee1c9 · 34 minutes ago [History](#)

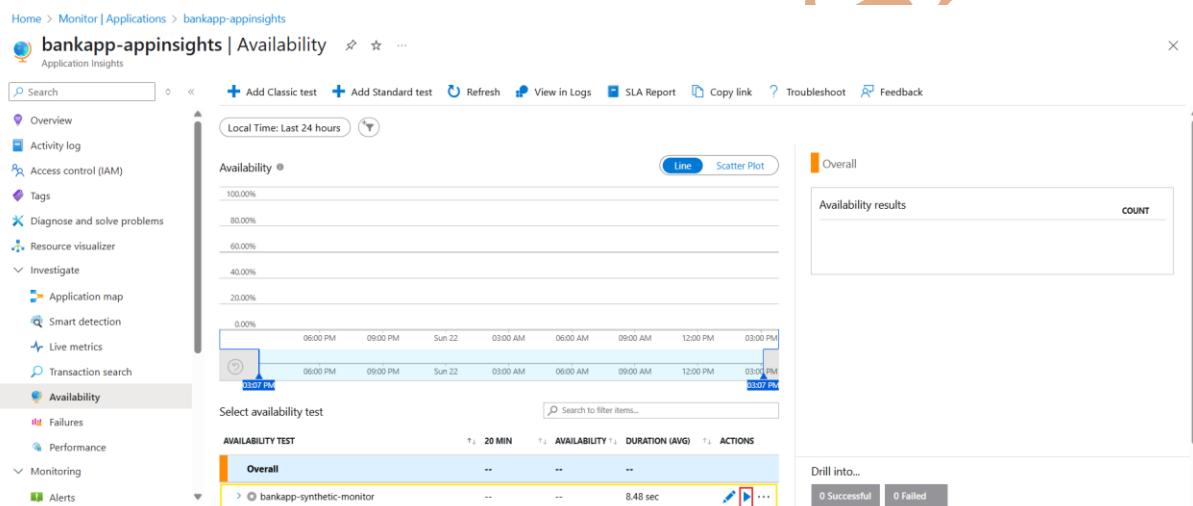
[Code](#) [Blame](#) 10 lines (8 loc) · 546 Bytes

```

1  FROM eclipse-temurin:17-jdk-alpine
2  MAINTAINER "Dexter"
3
4  ENV APPLICATIONINSIGHTS_CONNECTION_STRING=InstrumentationKey=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX;IngestionEndpoint=https://eastus2-3.in.applicationinsights.azure.com
5
6  WORKDIR /usr/src/app
7  COPY target/*.jar ./app.jar
8  COPY ./applicationinsights-agent-3.7.1.jar ./
9  EXPOSE 8080
10 CMD ["java", "-javaagent:/usr/src/app/applicationinsights-agent-3.7.1.jar", "-jar", "app.jar"]

```

Then I enabled the Synthetic Monitoring and its Alert which I created because of execution of Terraform Script. To enable it go to Azure Monitor > Insights (Applications) > Select the Application Insight which you created using the Terraform Script > Availability then enables it as shown in the screenshot attached below.



The Application is enabled now as shown in the screenshot attached below.



Then I enabled the Alert for Application Insight as shown in the screenshot attached below.

Home > Monitor | Applications > bankapp-appinsights | Alerts >

Alert rules ...

Subscription : Pay-As-You-Go Target resource type : all Target scope : bankapp-appinsights Signal type : all Severity : all Status : Enabled No grouping

Name ↑↓	Condition	Severity ↑↓	Target scope	Target resource type	Signal type
<input type="checkbox"/> Failure Anomalies - bankapp-ap...	Failure Anomalies detected	3 - Informational	bankapp-appinsights	Application Insights	Smart detector

Status
2 selected
Apply Cancel

Home > Monitor | Applications > bankapp-appinsights | Alerts >

Alert rules ...

Subscription : Pay-As-You-Go Target resource type : all Target scope : bankapp-appinsights Signal type : all Severity : all Status : Enabled No grouping

Name ↑↓	Condition	Severity ↑↓	Target scope	Target resource type	Signal type
<input type="checkbox"/> Failure Anomalies - bankapp-ap...	Failure Anomalies detected	3 - Informational	bankapp-appinsights	Application Insights	Smart detector

Status
2 selected
Select all
Enabled
Disabled

Home > Monitor | Applications > bankapp-appinsights | Alerts >

Alert rules ...

Subscription : Pay-As-You-Go Target resource type : all Target scope : bankapp-appinsights Signal type : all Severity : all Status : all No grouping

Name ↑↓	Condition	Severity ↑↓	Target scope	Target resource type	Signal type ↑↓	Status ↑↓
<input type="checkbox"/> alert-for-bankapp-url-and-ssl-ex...	availabilityResults/availabilityPer...	3 - Informational	bankapp-appinsights	Application Insights	Metrics	<input checked="" type="checkbox"/> Enable
<input type="checkbox"/> Failure Anomalies - bankapp-ap...	Failure Anomalies detected	3 - Informational	bankapp-appinsights	Application Insights	Smart detector	<input type="checkbox"/> Disable

Enable
Disable
Delete

Home > Monitor | Applications > bankapp-appinsights | Alerts >

Alert rules ...

Subscription : Pay-As-You-Go Target resource type : all Target scope : bankapp-appinsights Signal type : all Severity : all Status : all No grouping

Name ↑↓	Condition	Severity ↑↓	Target scope	Target resource type	Signal type ↑↓	Status ↑↓
<input type="checkbox"/> alert-for-bankapp-url-and-ssl-ex...	availabilityResults/availabilityPer...	3 - Informational	bankapp-appinsights	Application Insights	Metrics	<input checked="" type="checkbox"/> Enabled
<input type="checkbox"/> Failure Anomalies - bankapp-ap...	Failure Anomalies detected	3 - Informational	bankapp-appinsights	Application Insights	Smart detector	<input checked="" type="checkbox"/> Enabled

You can see the Application Map, Live Metrics and Transaction Search in Application Insight as shown in the screenshot attached below.

Home > Application Insights > bankapp-appinsights

Application Insights

bankapp-appinsights | Application map

bankapp-appinsights | Live metrics

bankapp-appinsights | Transaction search

You can filter out the Standard Availability Test (Synthetic Monitoring) in Azure Monitor Application Insight as shown in the screenshot attached below.

Home > Application Insights > bankapp-appinsights

Application Insights

bankapp-appinsights | Availability

Local Time: Last 24 hours

Availability

Overall

Availability results

COUNT	
Successful	49
Failed	41

Drill into... 49 Successful 41 Failed

Overall

Availability results

COUNT	
Successful	102
Failed	80

Drill into... 102 Successful 80 Failed

Overall

Availability results

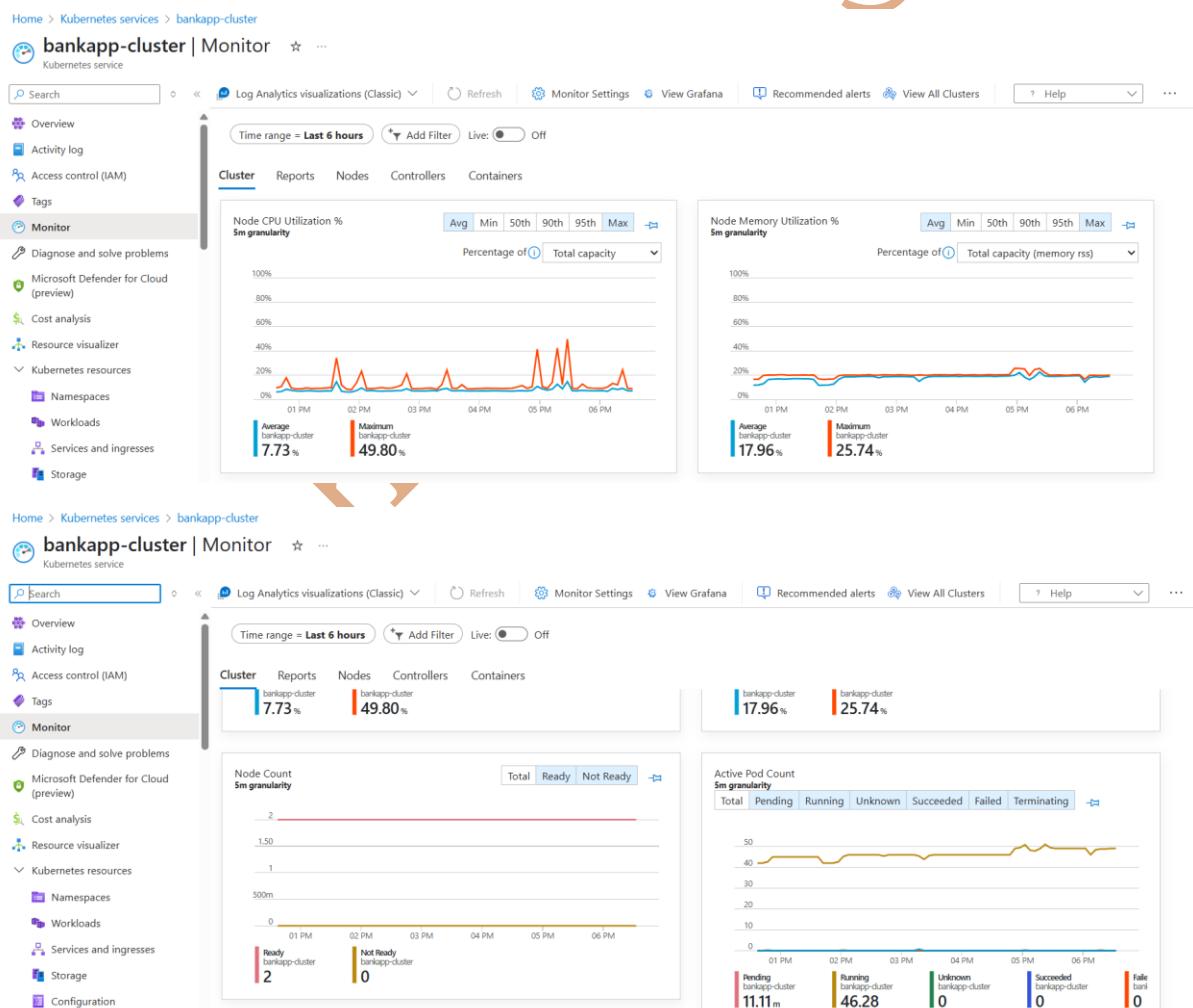
COUNT	
Successful	102
Failed	80

Drill into... 102 Successful 80 Failed

AVAILABILITY TEST	20 MIN	AVAILABILITY	DURATION (AVG)	ACTIONS
✓ bankapp-sy...	100.00%	55.80%	3.38 sec	
✓ East Asia	100.00%	55.17%	3.83 sec	
✓ East US	100.00%	60.00%	2.31 sec	
✓ North C...	100.00%	56.25%	4.30 sec	
✓ South C...	100.00%	53.33%	1.85 sec	

The Azure alert **alert-for-bankapp-url-and-ssl-expiration-synthetic-monitor** also monitors the SSL expiration and it will trigger an email to the Group Email ID configured in the Azure Action Group 30 days before expiration.

Monitoring containerized workload deployed to AKS using container insight is as shown in the screenshot attached below.



I had deployed the Vertical Pod Autoscaler using the commands as written below in the AKS Cluster.

```
git clone https://github.com/kubernetes/autoscaler.git
cd autoscaler/vertical-pod-autoscaler/hack/
./vpa-up.sh
```

The requirement to run the above commands is OpenSSL must be 1.1.1 or later version.

```
[root@XXXXXXXXXX ~]# openssl version
OpenSSL 1.1.1k FIPS 25 Mar 2021
[root@Terraform-Server ~]# git clone https://github.com/kubernetes/autoscaler.git
Cloning into 'autoscaler'...
remote: Enumerating objects: 224484, done.
remote: Counting objects: 100% (1900/1900), done.
remote: Compressing objects: 100% (1245/1245), done.
remote: Total 224484 (delta 1245), reused 655 (delta 655), pack-reused 222584 (from 4)
Receiving objects: 100% (224484/224484), 248.11 MiB | 30.78 MiB/s, done.
Resolving deltas: 100% (145429/145429), done.
Updating files: 100% (8051/8051), done.
[root@XXXXXXXXXX ~]# cd autoscaler/
.git/                      LICENSE          addon-resizer/
.github/                   OWNERS           balancer/
.gitignore                 OWNERS_ALIASES   builder/
.pre-commit-config.yaml    README.md       charts/
CONTRIBUTING.md            SECURITY_CONTACTS cluster-autoscaler/
[root@XXXXXXXXXX ~]# cd autoscaler/vertical-pod-autoscaler/hack/
[root@XXXXXXXXXX hack]# ./vpa-up.sh

[root@XXXXXXXXXX hack]# kubectl get pods -n kube-system | grep vpa
vpa-admission-controller-XXXXXXXXXX        1/1   Running   0          4s
vpa-recommender-XXXXXXXXXX        1/1   Running   0          5s
vpa-updater-XXXXXXXXXX        1/1   Running   0          5s
```

I had implemented the HPA and VPA in AKS cluster using the Kubernetes manifests file present in GitHub Repo <https://github.com/singhritesh85/kubernetes-manifests.git> in the file **hpa.yaml** and **vpa.yaml**. The kubernetes manifests file was deployed to the AKS Cluster using Azure DevOps Pipeline as shown in the screenshot attached below.

✓ Connect **Select** Configure Review

New pipeline

Select a repository

Filter by keywords My repositories ▾ ×

-  singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks private
2h ago
-  singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-aks
2h ago
-  singhritesh85/helm-repo-for-bitnami-mysql-vpa
Thursday
-  singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-A...
Monday
-  singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch-eks
Monday
-  singhritesh85/kubernetes-manifests
Jun 15
-  singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch-Private-eks private
Jun 14

✓ 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



Review your pipeline YAML

```

trigger:
- none
pr: none
pool:
- name: demo
  demands:
    - agent.name -equals demo
stages:
- stage: HPA_VPA_in_AKS
  displayName: HPA_VPA_in_AKS
  jobs:
    - deployment: HPA_VPA_in_AKS
      displayName: HPA_VPA_in_AKS
      environment: "dev"
      workspace:
        clean: all
        strategy:
          runOnce:
            deploy:
              steps:
                checkout, ...onif

```

Pipelines

Recent All Runs

Recently run pipelines

Pipeline	Last run
hpa-vpa	Manually triggered for main 2m ago 22s
bankapp	Manually triggered for main 2m ago 2m 17s
mysql	Manually triggered for main 2m ago 3m 51s

```

[demo@devopsagent-vm ~]$ kubectl get hpa -n bankapp
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
bankapp-hpa  Deployment/bankapp-folo  cpu: 6m/1m  1          3          3          61s
[demo@devopsagent-vm ~]$ kubectl get vpa -n mysql
NAME      MODE      CPU      MEM      PROVIDED      AGE
mysql-vpa  Auto     800m    850Mi   True        73s

```

You can see using HPA the Application Pods had been scaled to 3 pods as shown in the screenshot attached below.

```

[demo@devopsagent-vm ~]$ kubectl get pods -n bankapp
NAME           READY   STATUS    RESTARTS   AGE
bankapp-folo-  1/1     Running   0          2m20s
bankapp-folo-  1/1     Running   0          94m
bankapp-folo-  1/1     Running   0          2m20s

```

Initially when no VPA was not deployed the cpu and memory in mysql8 pod was recorded as shown in the screenshot attached below.

```

[demo@devopsagent-vm ~]$ kubectl describe pod mysql-0 -n mysql

```

```

Containers:
  mysql:
    Container ID:  containerd://[REDACTED]
    Image:          docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:       docker.io/bitnami/mysql@sha256:[REDACTED]
    Port:          3306/TCP
    Host Port:     0/TCP
    SecompProfile: RuntimeDefault
    State:         Running
      Started:    2025-01-10T10:45:25.000000Z
    Ready:         True
    Restart Count: 0
    Limits:
      cpu:        650m
      memory:     650Mi
    Requests:
      cpu:        650m
      memory:     650Mi

```

After I deployed the VPA the cpu and memory of the mysql8 pod is as shown in the screenshot attached below. After deploying VPA the mysql8 pods had been recreated as shown in the screenshot attached below (pods was recreated because I was using update mode in VPA as Auto).

```
[demo@devopsagent-vm ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          [REDACTED]
mysql-1   1/1     Running   0          [REDACTED]
mysql-2   1/1     Running   0          [REDACTED]
mysql-1   1/1     Running   0          [REDACTED]
mysql-1   1/1     Terminating   0          [REDACTED]
mysql-1   1/1     Terminating   0          [REDACTED]
mysql-1   0/1     Completed   0          [REDACTED]
mysql-1   0/1     Completed   0          [REDACTED]
mysql-1   0/1     Completed   0          [REDACTED]
mysql-1   0/1     Pending     0          [REDACTED]
mysql-1   0/1     Pending     0          [REDACTED]
mysql-1   0/1     Init:0/1   0          [REDACTED]
mysql-1   0/1     PodInitializing   0          [REDACTED]
mysql-1   0/1     Running     0          [REDACTED]
mysql-1   0/1     Running     0          [REDACTED]
mysql-1   1/1     Running     0          [REDACTED]
mysql-2   1/1     Running     0          [REDACTED]
mysql-2   1/1     Terminating   0          [REDACTED]
mysql-2   1/1     Terminating   0          [REDACTED]
mysql-2   0/1     Completed   0          [REDACTED]
mysql-2   0/1     Completed   0          [REDACTED]
mysql-2   0/1     Completed   0          [REDACTED]
mysql-2   0/1     Pending     0          [REDACTED]
mysql-2   0/1     Pending     0          [REDACTED]
mysql-2   0/1     Init:0/1   0          [REDACTED]
mysql-2   0/1     PodInitializing   0          [REDACTED]
mysql-2   0/1     Running     0          [REDACTED]
mysql-2   0/1     Running     0          [REDACTED]
mysql-2   1/1     Running     0          [REDACTED]
```

```
[demo@devopsagent-vm ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          1d
mysql-1   1/1     Running   0          1d
mysql-2   1/1     Running   0          1d
mysql-0   1/1     Running   0          1d
mysql-0   1/1     Terminating   0          1d
mysql-0   1/1     Terminating   0          1d
mysql-0   0/1     Completed   0          1d
mysql-0   0/1     Completed   0          1d
mysql-0   0/1     Completed   0          1d
mysql-0   0/1     Pending     0          1d
mysql-0   0/1     Pending     0          1d
mysql-0   0/1     Init:0/1   0          1d
mysql-0   0/1     PodInitializing   0          1d
mysql-0   0/1     Running     0          1d
mysql-0   0/1     Running     0          1d
mysql-0   1/1     Running     0          1d
```

```
[demo@devopsagent-vm ~]$ kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          66s
mysql-1   1/1     Running   0          3m5s
mysql-2   1/1     Running   0          2m6s
```

```
[demo@devopsagent-vm ~]$ kubectl describe pod mysql-0 -n mysql
```

```
Containers:
  mysql:
    Container ID:  containerd://[REDACTED]
    Image:         docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:      docker.io/bitnami/mysql@sha256:[REDACTED]
    Port:          3306/TCP
    Host Port:    0/TCP
    SecompProfile: RuntimeDefault
    State:         Running
      Started:    2025-01-11T10:40:25Z
    Ready:        True
    Restart Count: 0
    Limits:
      cpu:  800m
      memory:  850Mi
    Requests:
      cpu:  800m
      memory:  850Mi
```

I checked the VPA-Controller pod logs and observed the same as shown in the screenshot attached below.

```
[root@[REDACTED] ~]# kubectl logs -f vpa-admission-controller-[REDACTED] -n kube-system
```

```

I0622 11:50:08.471262      1 handler.go:81] "Admitting pod" pod="mysql/mysql-1"
I0622 11:50:08.471354      1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-1"
I0622 11:50:08.471431      1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":"Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit"}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":"mysql"}]
I0622 11:50:30.237663      1 reflector.go:946] "Watch close" reflector="pkg/mod/k8s.io/client-go@v0.33.0/tools/cache/reflector.go:285" type="*v1.VerticalPodAutoscaler" totalItems=7
I0622 11:51:05.782577      1 reflector.go:946] "Watch close" reflector="pkg/mod/k8s.io/client-go@v0.33.0/tools/cache/reflector.go:285" type="*v1.ReplicaSet" totalItems=20
I0622 11:51:07.745096      1 handler.go:81] "Admitting pod" pod="mysql/mysql-2"
I0622 11:51:07.745185      1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-2"
I0622 11:51:07.745255      1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":"Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit"}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":"mysql"}]
I0622 11:51:23.408270      1 reflector.go:946] "Watch close" reflector="pkg/mod/k8s.io/client-go@v0.33.0/tools/cache/reflector.go:285" type="*v1.Job" totalItems=10
I0622 11:51:54.344021      1 reflector.go:946] "Watch close" reflector="pkg/mod/k8s.io/client-go@v0.33.0/tools/cache/reflector.go:285" type="*v1.ReplicationController" totalItems=7
I0622 11:52:07.918170      1 handler.go:81] "Admitting pod" pod="mysql/mysql-0"
I0622 11:52:07.918248      1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-0"
I0622 11:52:07.918324      1 server.go:114] "Sending patches" patches=[{"op":"add","path":"/spec/containers/0/resources/requests/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/spec/containers/0/resources/limits/cpu","value":"800m"}, {"op":"add","path":"/spec/containers/0/resources/limits/memory","value":"850Mi"}, {"op":"add","path":"/metadata/annotations/vpaUpdates","value":"Pod resources updated by mysql-vpa: container 0: cpu request, memory request, cpu limit, memory limit"}, {"op":"add","path":"/metadata/annotations/vpaObservedContainers","value":"mysql"}]

```

You observed here that mysql8 pods was recreated after you implemented the VPA for mysql8 pods which involves recreation of pods. To refrain from this situation, you can use

InPlacePodVerticalScaling. In InPlacePodVerticalScaling you can increase the resource (CPU or memory) of the pod without recreating it. Using InPlacePodVerticalScaling I changed the CPU and memory of a running mysql8 pod from 650m and 650Mi to 1 and 1Gi (requested value and 1.5 and 1.5Gi as limited value) respectively using **kubectl edit pod mysql-0 -n mysql --subresource=resize** as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl edit pod mysql-0 -n mysql --subresource=resize
```

```

resizePolicy:
- resourceName: cpu
  restartPolicy: NotRequired
- resourceName: memory
  restartPolicy: NotRequired
resources:
  limits:
    cpu: 800m
    memory: 850Mi
  requests:
    cpu: 800m
    memory: 850Mi

```



```

resizePolicy:
- resourceName: cpu
  restartPolicy: NotRequired
- resourceName: memory
  restartPolicy: NotRequired
resources:
  limits:
    cpu: 1.5
    memory: 1.5Gi
  requests:
    cpu: 1
    memory: 1Gi

```

```
[demo@devopsagent-vm ~]$ kubectl edit pod mysql-0 -n mysql --subresource=resize
pod/mysql-0 edited
```

Then I checked the status of cpu and memory of the pod mysql-0 and found that it was changed from 850m and 850Mi to 1 and 1Gi (required cpu and memory; 1.5 and 1.5Gi limited cpu and memory) as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl describe pod mysql-0 -n mysql
```

```
Containers:
  mysql:
    Container ID:  containerd://[REDACTED]
    Image:          docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:       docker.io/bitnami/mysql@sha256:[REDACTED]
    Port:          3306/TCP
    Host Port:     0/TCP
    SeccompProfile: RuntimeDefault
    State:         Running
      Started:   2025-01-11T10:45:25Z
    Ready:        True
    Restart Count: 0
    Limits:
      cpu:        1500m
      memory:     1536Mi
    Requests:
      cpu:        1
      memory:     1Gi
```

I would like to mention here that I am using AKS Cluster with version 1.33 in which InPlacePodVerticalScaling is enabled which I checked using the command **kubectl get --raw /metrics | grep kubernetes_feature_enabled** as shown in the screenshot attached below.

```
[root@[REDACTED] ~]# kubectl get --raw /metrics | grep kubernetes_feature_enabled | grep InPlacePodVerticalScaling
kubernetes_feature_enabled{name="InPlacePodVerticalScaling",stage="BETA"} 1
kubernetes_feature_enabled{name="InPlacePodVerticalScalingAllocatedStatus",stage="DEPRECATED"} 0
kubernetes_feature_enabled{name="InPlacePodVerticalScalingExclusiveCPUs",stage="ALPHA"} 0
```

Synthetic Monitoring of BankApp URL using Bash Shell Script

For Application Insight, Availability Test (Standard Test) I had selected the test frequency as 5 minutes (which means every five minutes a ping test will be run) to confirm that the site is available. However, in this section I had described the monitoring of Bank Application URL and the SSL certificate (associated with BankApp URL) monitoring using Bash Shell Script.

Monitoring of BankApp URL using Bash Shell Script

To monitor BankApp URL using Bash Shell script first I configured the Sendmail with Gmail. I had mentioned the below steps which you can follow to install and configure the Sendmail with Gmail. You install Sendmail and can create and run the crontab from any one of the Linux Server in your organisation's environment. However, for the current project I am using the Azure DevOps Agent which I created with the help of Terraform Script ran initially.

```
yum install -y sendmail sendmail-cf m4 mailx make cyrus-sasl-lib cyrus-sasl-plain cyrus-sasl-md5 cyrus-sasl-ntlm
cyrus-sasl-gssapi
```

```
mkdir /etc/mail/auth
```

```
vim /etc/mail/auth/gmail-auth
```

```
AuthInfo: "U:root" "I:abc@gmail.com" "P:XXXX XXXX XXXX XXXX" <---- Gmail App Password
```

```
cd /etc/mail/auth/
```

```
makemap hash gmail-auth < gmail-auth
```

```
[root@devopsagent-vm auth]# ls
```

```
gmail-auth gmail-auth.db
```

```
vim /etc/mail/sendmail.mc
```

```
dnl ##### Deleted last three lines or commented by just adding dnl in front of them
```

```
dnl ###MAILER(smtp)dnl
```

```
dnl ###MAILER(procmail)dnl
```

```
dnl ###MAILER(cyrusv2)dnl
```

```
define(`SMART_HOST', `smtp.gmail.com')dnl
```

```
define(`RELAY_MAILER_ARGS', `TCP $h 587')dnl
```

```
define(`ESMTP_MAILER_ARGS', `TCP $h 587')dnl
```

```
define(`confAUTH_OPTIONS', `A p')dnl
```

```
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
```

```
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
```

```
FEATURE(`authinfo', `hash -o /etc/mail/auth/gmail-auth.db')dnl
```

```
MAILER(smtp)dnl
```

```
MAILER(local)dnl
```

```
make -C /etc/mail
```

```
systemctl restart sendmail
```

```
systemctl enable sendmail
```

```
systemctl status sendmail
```

```
[root@] ~]# yum install -y sendmail sendmail-cf m4 mailx make cyrus-sasl-lib cyrus-sasl-plain cyrus-sasl-md5 cyrus-sasl-ntlm cyrus-sasl-gssapi
[root@] ~]# mkdir /etc/mail/auth
[root@] ~]# vim /etc/mail/auth/gmail-auth

[root@] ~]# cat /etc/mail/auth/gmail-auth
AuthInfo: "U:root" "I:[REDACTED]@gmail.com" "P:[REDACTED]"

[root@] ~]# cd /etc/mail/auth/
[root@] auth]# makemap hash gmail-auth < gmail-auth
auth]# ls
gmail-auth  gmail-auth.db

[root@] mail]# vim /etc/mail/sendmail.mc
dnl ##### Deleted last three lines or commented by just adding dnl in front of them
dnl ###MAILER(smtp)dnl
dnl ###MAILER(procmail)dnl
dnl ###MAILER(cyrusv2)dnl
define(`SMART_HOST',`[smtp.gmail.com]')dnl
define(`RELAY_MAILER_ARGS', `TCP $h 587')dnl
define(`ESMTP_MAILER_ARGS', `TCP $h 587')dnl
define(`confAUTH_OPTIONS', `A p')dnl
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl FEATURE(`genericstable','hash -o /etc/mail/genericstable.db')dnl ← Commented Line
FEATURE(`authinfo','hash -o /etc/mail/auth/gmail-auth.db')dnl
MAILER(smtp)dnl
MAILER(local)dnl

[root@] mail]# make -C /etc/mail
make: Entering directory '/etc/mail'
make: Leaving directory '/etc/mail'

[root@] mail]# systemctl restart sendmail
[root@] mail]# systemctl enable sendmail
Created symlink /etc/systemd/system/multi-user.target.wants/sendmail.service → /usr/lib/systemd/system/sendmail.service.
Created symlink /etc/systemd/system/multi-user.target.wants/sm-client.service → /usr/lib/systemd/system/sm-client.service.
[root@] mail]# systemctl status sendmail
● sendmail.service - Sendmail Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/sendmail.service; enabled; vendor preset: disabled)
   Active: active (running) since [REDACTED] 2025-01-11 UTC; 1s ago
```

Then a test mail was sent to the group email ID as shown in the screenshot attached below.

```
[root@] ~]# echo "This is a Test Mail" | mail -r [REDACTED]@gmail.com -s "Test Email" [REDACTED]@gmail.com
```

Test Email

 [REDACTED]@gmail.com

This is a Test Mail

```
[root@] ~]# vim synthetic-monitor-url.sh
[root@] ~]# chmod +x synthetic-monitor-url.sh
[root@] ~]# crontab -e
```

```
[root@yellow ~]# crontab -l
*/1 * * * * sh synthetic-monitor-url.sh
```

```
[root@yellow ~]# cat synthetic-monitor-url.sh
#!/bin/bash

URL="https://bankapp.singhritesh85.com/login" ###Provide the URL to be monitored
EMAIL="yellow@gmail.com"

RESPONSE_CODE=$(curl -k -s -o /dev/null -w "%{http_code}" "$URL")

TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
if [ "$RESPONSE_CODE" -eq 200 ]; then
    echo "$TIMESTAMP - $URL is UP (HTTP $RESPONSE_CODE)"
else
    echo "$TIMESTAMP - $URL is DOWN (HTTP $RESPONSE_CODE)"
    echo "Critical Issue: $URL is Down, Please check and resolve the issue" | mail -s "URL https://bankapp.singhritesh85.com is Down" $EMAIL
fi
```

Synthetic Monitoring notification always be on higher priority and whenever the concerned team will get the notification on their group email ID immediately, they need to investigate the RCA (Root Cause Analysis) and try to resolve the issue in the stipulated time interval (Service Level Agreement). As for this issue I removed the entry of Public IP and HOST for the record set of bankapp URL in Azure DNS Zone. So, the concerned team should investigate it as a part of their RCA and do the entry into Azure DNS Zone to create the record set for bankapp with HOST and Public IP which you got from the Ingress Rule. However, for this project I created this scenario in dev environment and provided the severity as 3 but in production it should be assigned severity 0 (critical issue) or severity 1 task in Azure. If a URL (website) is down which means end users will be affected and hence organisation business also gets affected.

Monitoring of SSL Certificate of BankApp URL

I had monitored the SSL Certificate of the BankApp URL using the Bash shell script daily in non-production hours. A notification will be sent on group email Id for the current expiration status of the SSL Certificate.

```
[root@yellow ~]# cat ssl-expiration-monitor.sh
#!/bin/bash

HOST=bankapp.singhritesh85.com
PORT=443

EXPIRY_DATE=$(openssl s_client -servername $HOST -connect $HOST:$PORT </dev/null | openssl x509 -noout -enddate)
EXPIRY_DATE=$(echo "$EXPIRY_DATE" | cut -d'=' -f2)

# Get the current date in seconds
CURRENT_DATE=$(date +%s)

# Convert the expiry date to seconds
EXPIRY_DATE_SECONDS=$(date -d "$EXPIRY_DATE" +%s)

# Calculate the remaining days
DAYS_LEFT=$(( ($EXPIRY_DATE_SECONDS - $CURRENT_DATE) / (60 * 60 * 24) ))

# Check the certificate expiration and notify if days left 30, 15 or 7 days.
if [ "$DAYS_LEFT" -eq 30 ]; then
    echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r yellow@gmail.com -s "30 days left for SSL to be expired" yellow@gmail.com
else
    if [ "$DAYS_LEFT" -eq 15 ]; then
        echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r yellow@gmail.com -s "15 days left for SSL to be expired" yellow@gmail.com
        exit 1
    fi
    if [ "$DAYS_LEFT" -eq 7 ]; then
        echo "Caution. SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r yellow@gmail.com -s "7 days left for SSL to be expired" yellow@gmail.com
    else
        echo "SSL certificate for $HOST is valid for $DAYS_LEFT days." | mail -r singhritesh8515@gmail.com -s "SSL Expiration Notification" yellow@gmail.com
    fi
fi
```

I ran this shell-script using the crontab as shown in the screenshot attached below.

```
[root@yellow ~]# chmod +x ssl-expiration-monitor.sh

[root@yellow ~]# crontab -e
crontab: installing new crontab
[root@yellow ~]# crontab -l
*/1 * * * * sh synthetic-monitor-url.sh
5 0 * * * sh ssl-expiration-monitor.sh
```

This shell script will be executed daily at 12:05AM UTC which is 5:35AM IST.

```

cat ssl-expiration-monitor.sh
#!/bin/bash

HOST=bankapp.singhritesh85.com
PORT=443

EXPIRY_DATE=$(openssl s_client -servername $HOST -connect $HOST:$PORT </dev/null | openssl x509 -noout -enddate)
EXPIRY_DATE=$(echo "$EXPIRY_DATE" | cut -d'=' -f2)

# Get the current date in seconds
CURRENT_DATE=$(date +%s)

# Convert the expiry date to seconds
EXPIRY_DATE_SECONDS=$(date -d "$EXPIRY_DATE" +%s)

# Calculate the remaining days
DAYS_LEFT=$((($EXPIRY_DATE_SECONDS - $CURRENT_DATE) / (60 * 60 * 24)))

# Check the certificate expiration and notify if days left 30, 15 or 7 days.
if [ "$DAYS_LEFT" -eq 30 ]; then
    echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r abc@gmail.com -s "30 days left for SSL to be expired" xyz@gmail.com
else
    if [ "$DAYS_LEFT" -eq 15 ]; then
        echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r abc@gmail.com -s "15 days left for SSL to be expired" xyz@gmail.com
        exit 1
    fi
    if [ "$DAYS_LEFT" -eq 7 ]; then
        echo "Caution. SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r abc@gmail.com -s "7 days left for SSL to be expired" xyz@gmail.com
    else
        echo "SSL certificate for $HOST is valid for $DAYS_LEFT days." | mail -r abc@gmail.com -s "SSL Expiration Notification" xyz@gmail.com
    fi
fi

```

In the above provided Bash Shell script abc@gmail.com is the email Id which I used during configuration of Sendmail and xyz@gmail.com is the Group Email ID on which you want to send the notification. It is also possible that you can keep the email Id abc@gmail.com and xyz@gmail.com same.

Always remember when your team is doing any activity which may affects the Application URL then it is suggested to do the activity during non-production hours and disable the Alerts during the activity and do not forget to enable it after the activity.

The last part of this project is configuration of Log Rotate for crontab logs. It is also possible to archive these log files and send them to the Azure Storage Account Container but for current project I used Log Rotate which runs daily and dropped the archived log files older than 15 days. As I am running crontab every minute for monitoring BankApp URL So, I configured the Log Rotate here. For Azure DevOps Agent I used Alma Linux 8 and Log rotate was already installed with the version as shown in the screenshot attached below.

```
[root@[REDACTED] ~]# logrotate --version
logrotate 3.14.0

Default mail command:      /bin/mail
Default compress command:  /bin/gzip
Default uncompress command: /bin/gunzip
Default compress extension: .gz
Default state file path:   /var/lib/logrotate/logrotate.status
ACL support:                yes
SELinux support:            yes

[root@[REDACTED] logrotate.d]# cat cronjob
/var/log/cron {
    daily
    missingok
   notifempty
    rotate 15
    copytruncate
    olddir /var/log/cronlog
    compress
    dateext
    dateformat %d%m%Y
    postrotate
        echo "Log Rotation for file /var/log/cron just took place" | mail -r [REDACTED]@gmail.com -s "Log Rotated" [REDACTED]@gmail.com
    endscript
}

[root@[REDACTED] logrotate.d]# mkdir /var/log/cronlog
[root@[REDACTED] logrotate.d]# crontab -e
crontab: installing new crontab
[root@[REDACTED] logrotate.d]# crontab -l
* * * * * sh synthetic-monitor-url.sh

5 0 * * * sh ssl-expiration-monitor.sh

0 2 * * * logrotate /etc/logrotate.d/cronjob
```

Log will be rotated daily at 2:00 AM UTC and an email will be sent to the Group Email ID. Logs older than 15 days will be dropped. If you want to rotate log forcefully then uses the command **logrotate -f /etc/logrotate.d/cronjob**.

For Module 1

Source Code: <https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch-eks.git>

GitHub Repo: <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git>

Helm Chart: <https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git>

<https://github.com/singhritesh85/helm-repo-for-ArgoCD-cloudwatch-apm.git>

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

Terraform Script: <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git>

For Module 2

Source Code: <https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-aks.git>

GitHub Repo: <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git>

Helm Chart: <https://github.com/singhritesh85/helm-repo-for-bitnami-mysql-vpa.git>

<https://github.com/singhritesh85/helm-repo-for-ArgoCD.git>

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

Terraform Script: <https://github.com/singhritesh85/DevOps-Project-Kubernetes-AutoScale-Karpenter-ClusterAutoScaler-HPA-VPA-AWS-Azure-Monitoring-Logging.git>

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

<https://github.com/singhritesh85/Autoscale.git>