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

By Ritesh Kumar Singh

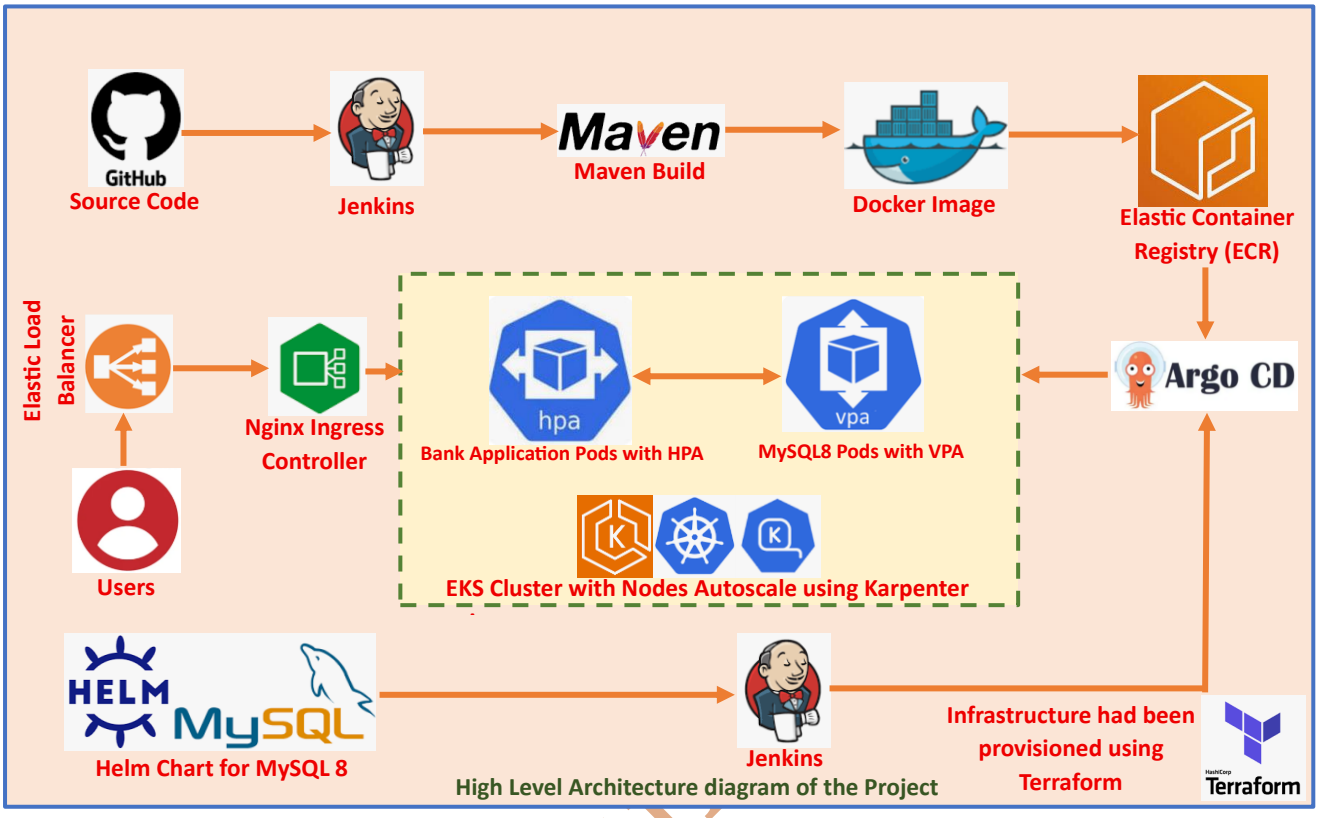
Email Address: - riteshkumarsingh9559@gmail.com

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 निःशेषजाड्यापहा ॥

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_cloudwatch_observability: 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... [03m40s elapsed]
module.bankapp.aws_eks_addon.csi_snapshot_controller: Still creating... [01m30s elapsed]
module.bankapp.aws_eks_addon.core_dns: 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_cloudwatch_observability: 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.[redacted]"
  "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10.[redacted]"
  "Jenkins_ALB_DNS_Name" = "jenkins-ms-[redacted].us-east-2.elb.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "registry_id" = [
    "02[redacted]6",
  ]
  "repository_url" = [
    "02[redacted]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@[redacted] ~]# 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[redacted]6:role/eks-nodegroup-role-dev
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::02[redacted]6:role/karpenter-eks-noderole
      username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2025[redacted]"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "[redacted]"
  uid: d[redacted]b
```

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

```
[root@[redacted] ~]# helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version "1.3.2" --namespace "karpenter" --create-namespac
e --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks\.amazonaws\.com/role-arn=arn:aws:iam::02[redacted]6:role/karpenter-co
ntroller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controll
er.resources.limits.memory=1Gi
```

helm upgrade --install karpenter 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@                    ~]# kubectl get all -n karpenter
NAME                                 READY   STATUS    RESTARTS   AGE
pod/karpenter-8          5-2    f    1/1     Running   0          4m32s
pod/karpenter-8          5-8    v    1/1     Running   0          4m32s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/karpenter    ClusterIP   172.             <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-8          5  2         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.

karpenter.yaml

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

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

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

```
[root@ ~]# kubectl 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\.kubernet
es\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02 6:certificate/ --set controller.service.annotations."s
ervice\.beta\.kubernetes\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-bala
ncer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-type"="elb" --set control
ler.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol"=http --set controller.service.annotations."service\.beta\.kubernete
s\.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)
                  AGE
service/ingress-nginx-controller             LoadBalancer   172.             a                                    .us-east-2.elb.amazonaws.com   80:3183
9/TCP,443:30995/TCP   5m56s
service/ingress-nginx-controller-admission   ClusterIP      172.             <none>                                                                   443/TCP
                  5m56s

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.

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

NAME                                      TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
                  AGE
service/ingress-nginx-controller          LoadBalancer   172.            a                                        .us-east-2.elb.amazonaws.com   80:31278/
TCP,443:31395/TCP
service/ingress-nginx-controller-admission   ClusterIP   172.            <none>                                                                   443/TCP
                  

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

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

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@                ~]# kubectl create namespace argocd
namespace/argocd created
[root@                ~]# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

[root@                ~]# kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
f               g[root@               ~]#
```

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

cat argocd-ingress-rule.yaml

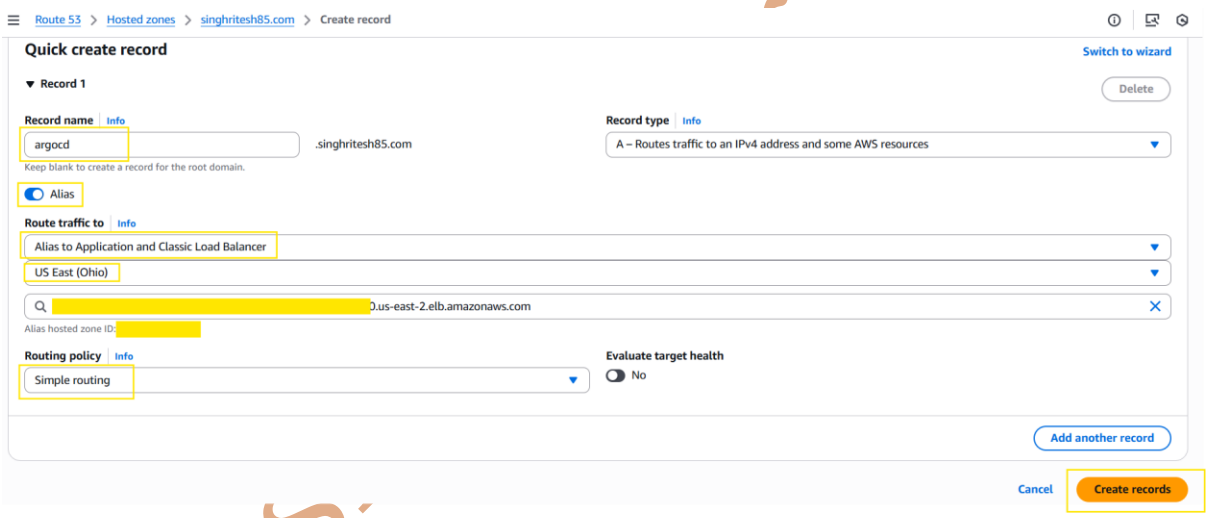
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ### You can use this option for this particular case for ArgoCD but not for all
#    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@                    ~]# kubectl apply -f argocd-ingress-rule.yaml
ingress.networking.k8s.io/minimal-ingress configured
```

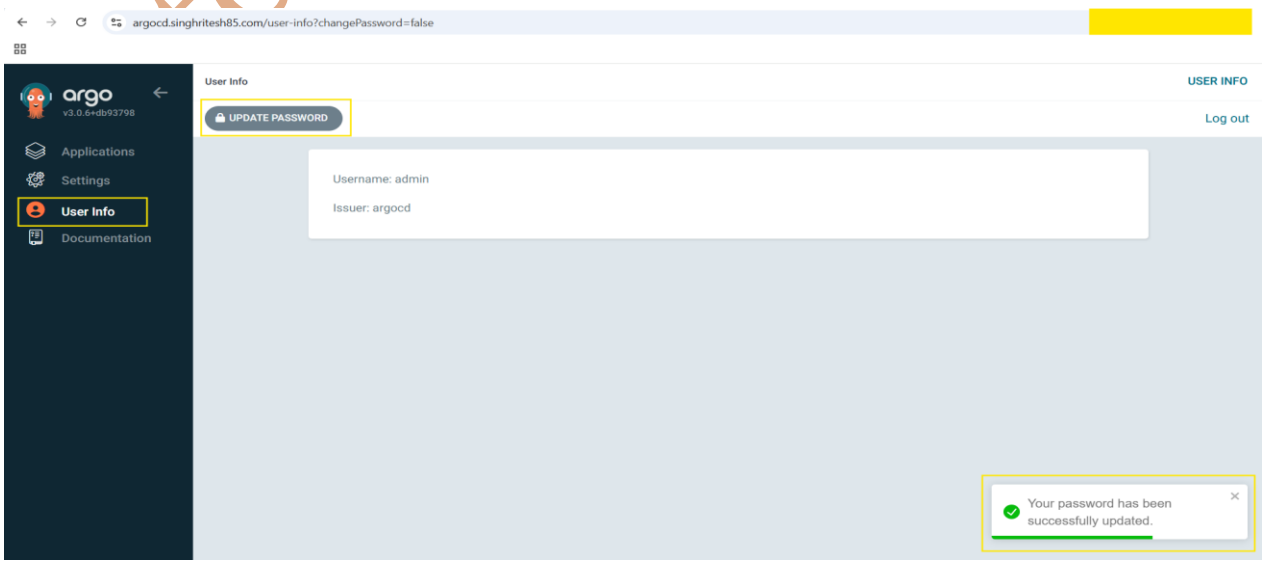
```
[root@                ~]# cat argocd-ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ###  You can use this option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

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

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



I had updated the admin user password in ArgoCD.

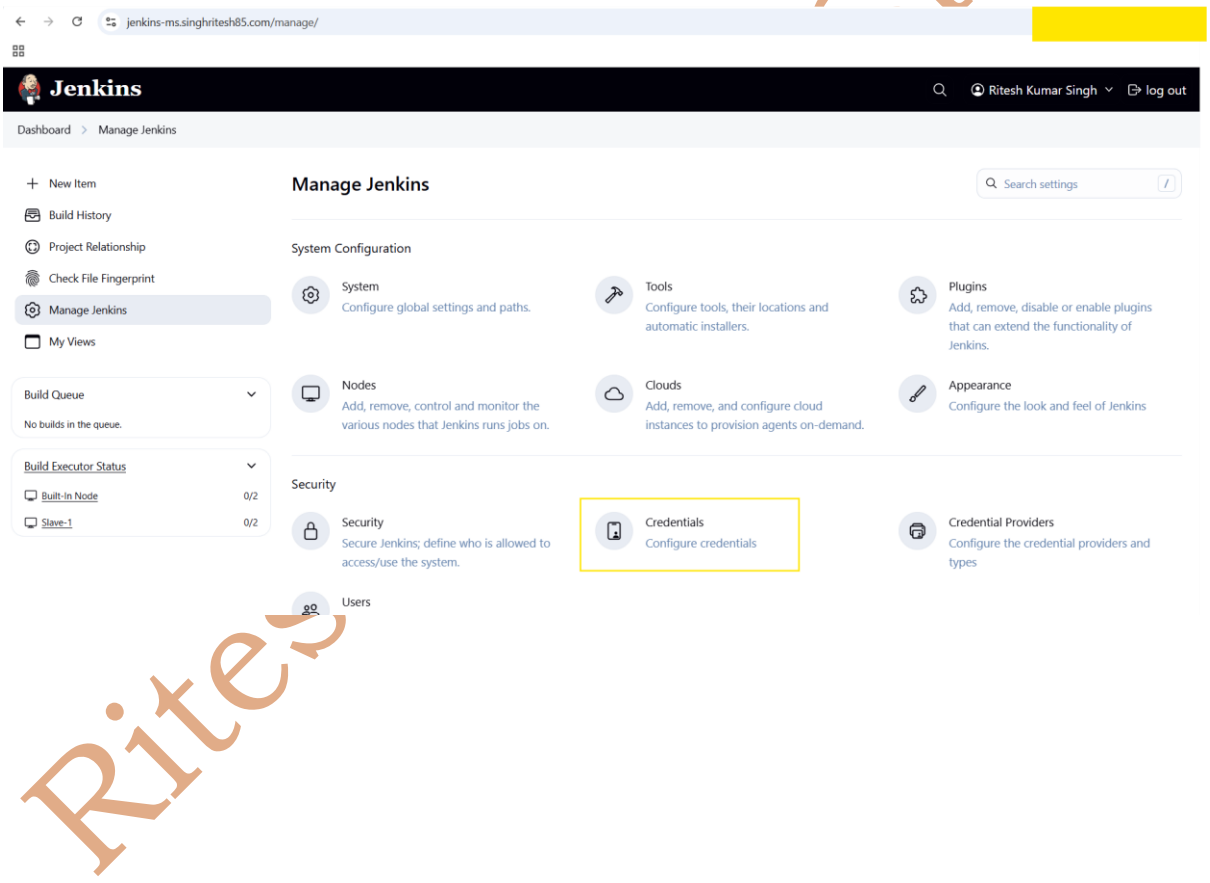


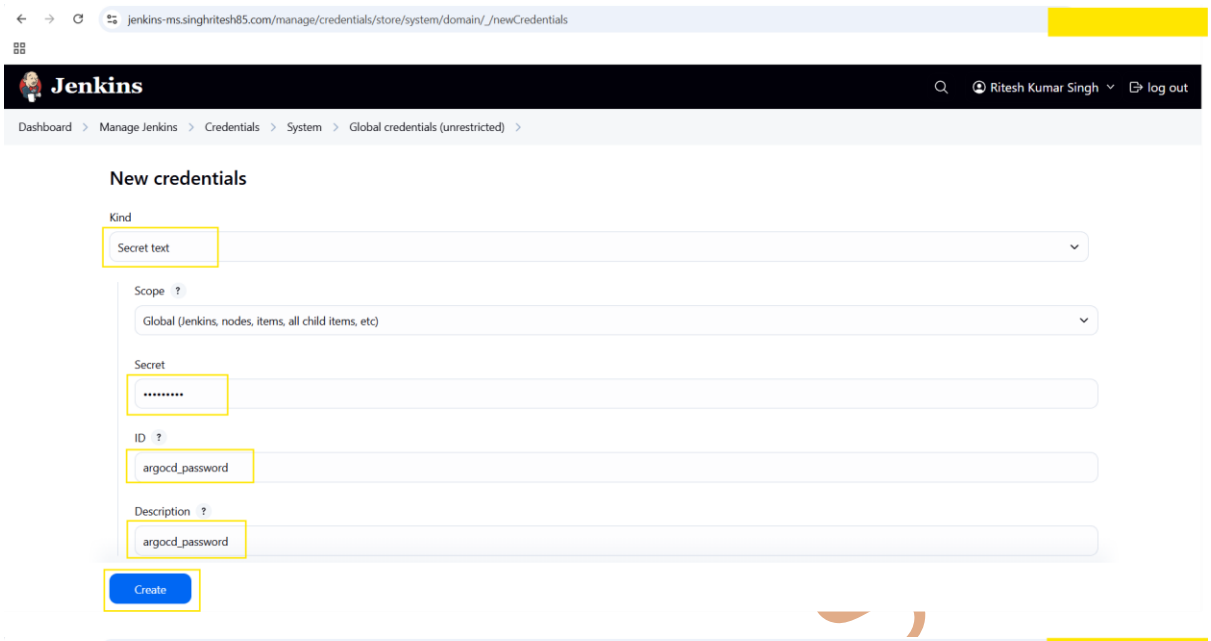
Installation ArgoCD CLI on Jenkins Slave Node

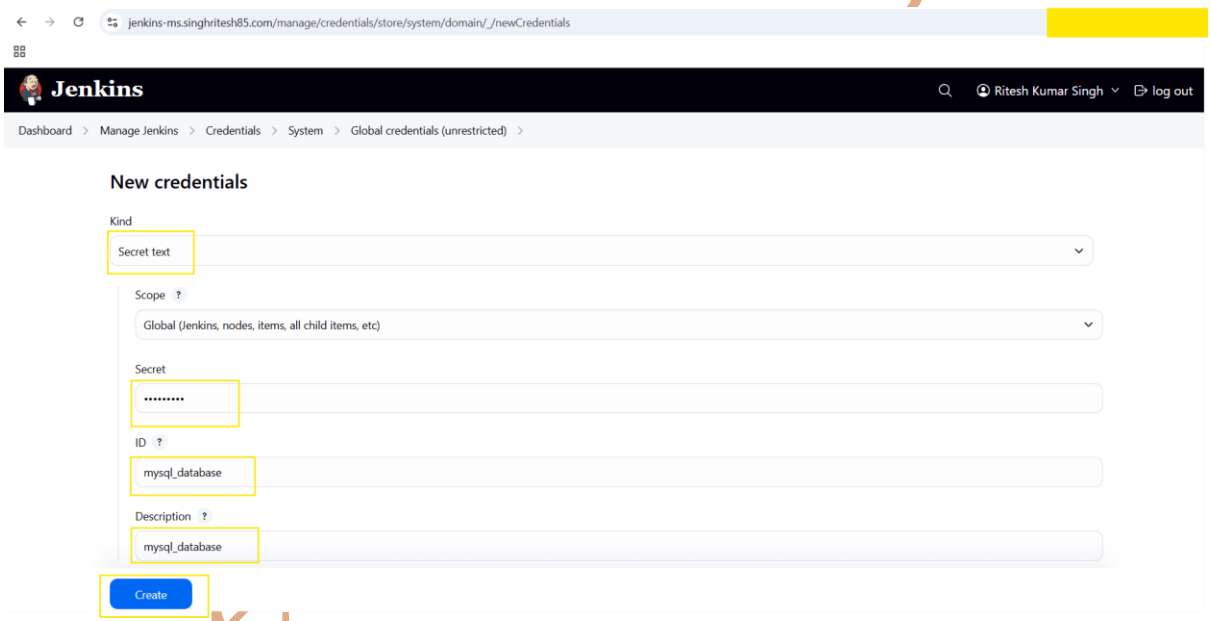
```
curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

rm argocd-linux-amd64
```

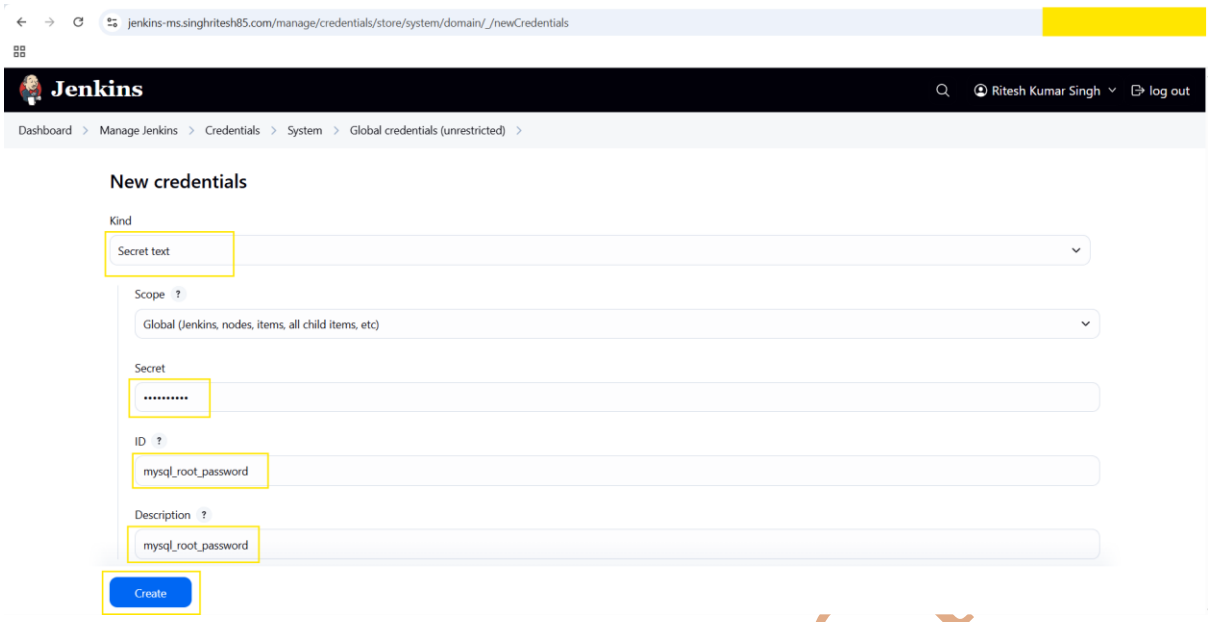
```
[jenkins@          ~]$ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
[jenkins@          ~]$ sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

[jenkins@          ~]$ rm argocd-linux-amd64
```

Then 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.singhritesh85.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.singhritesh85.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



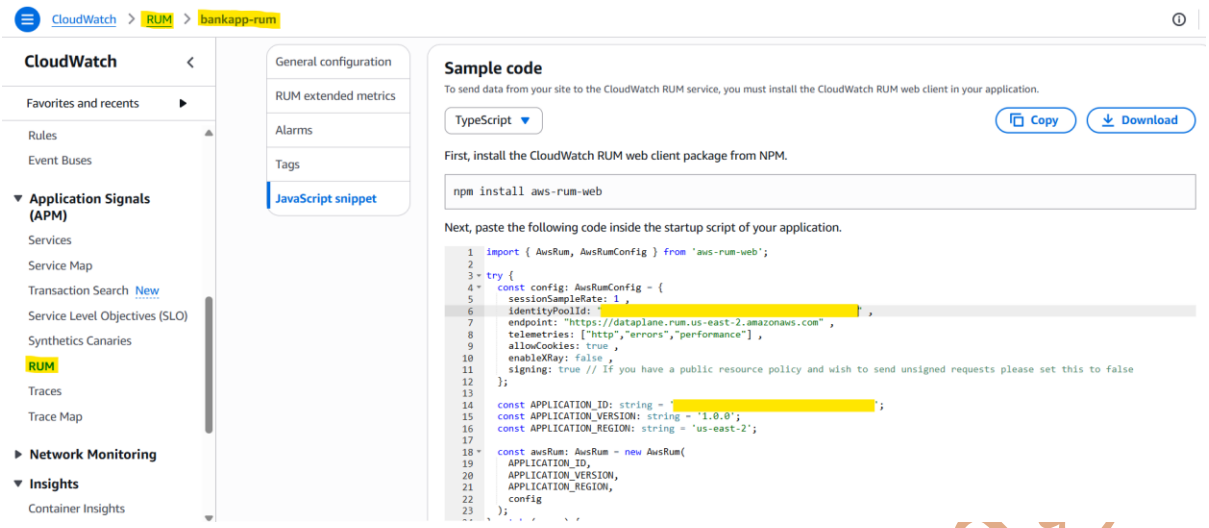
Ran the Jenkins Job for mysql and bankapp, the screenshot for Jenkins Job after its successful execution is as shown in the screenshot attached below. First ran the Jenkins Job for mysql and then for bankapp.

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

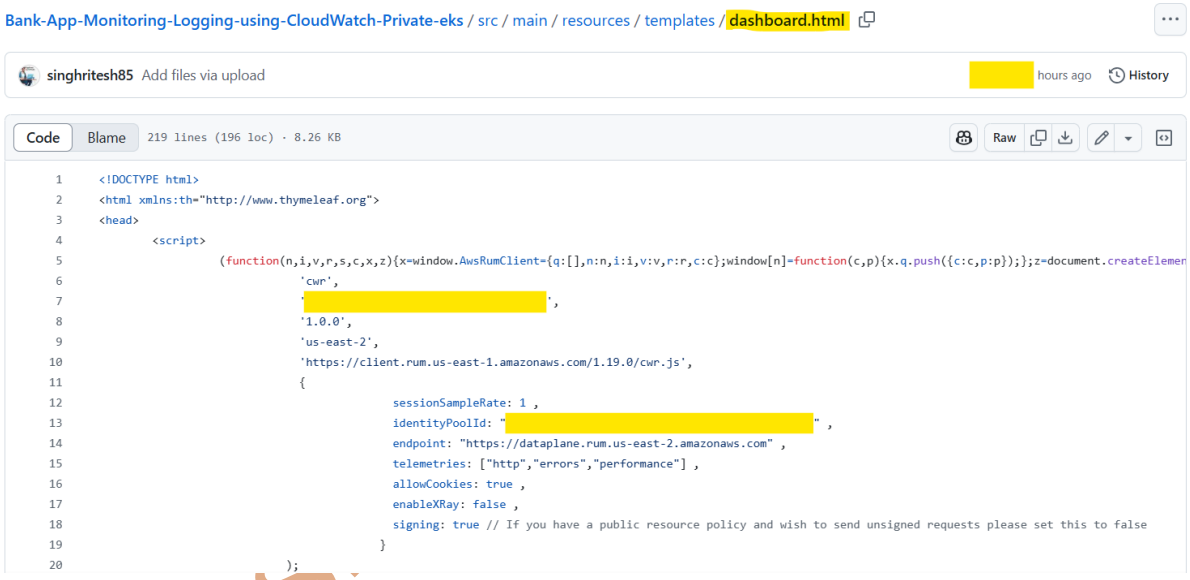
Karpenter was doing its job and spun a new node for EKS to schedule the mysql pods as shown in the screenshot attached below.

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

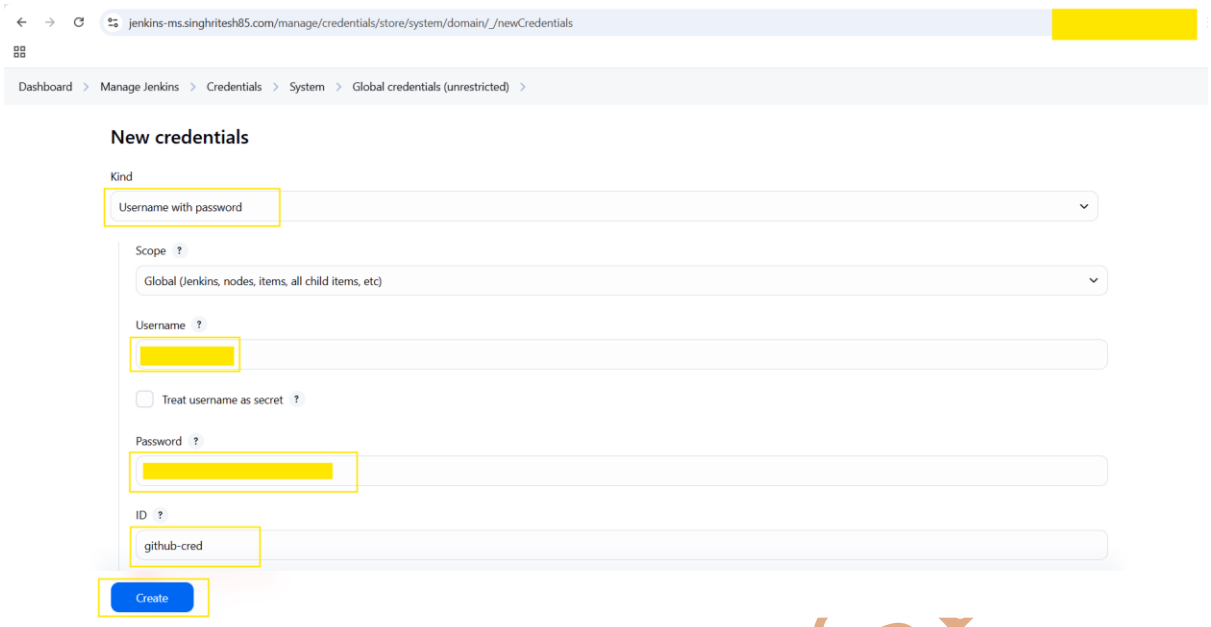
I want to mention here that I wanted to perform Real User Monitoring (RUM) using CloudWatch. So, I got the Script to be inserted into the source code as shown in the screenshot attached below.

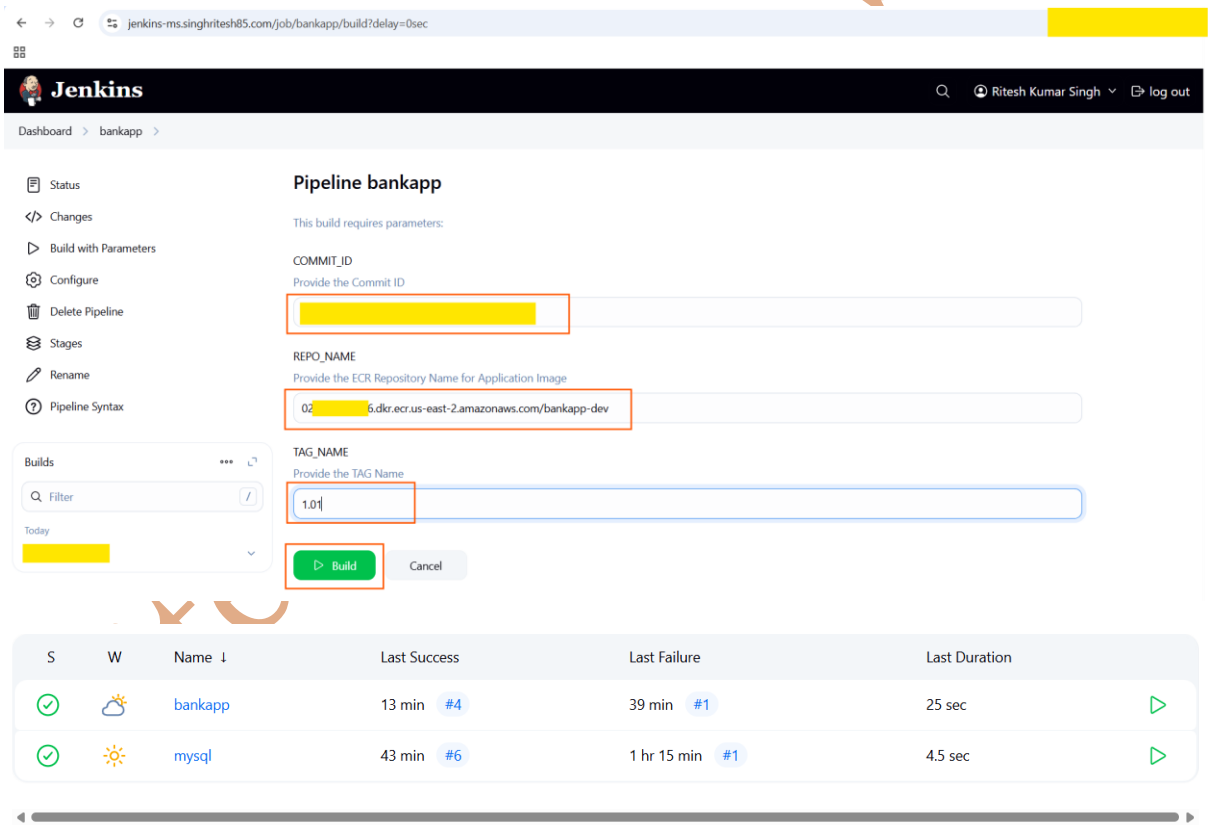


I copied this script and added in the file dashboard.html, login.html, register.html and transactions.html under the line as shown in the screenshot attached below.



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.



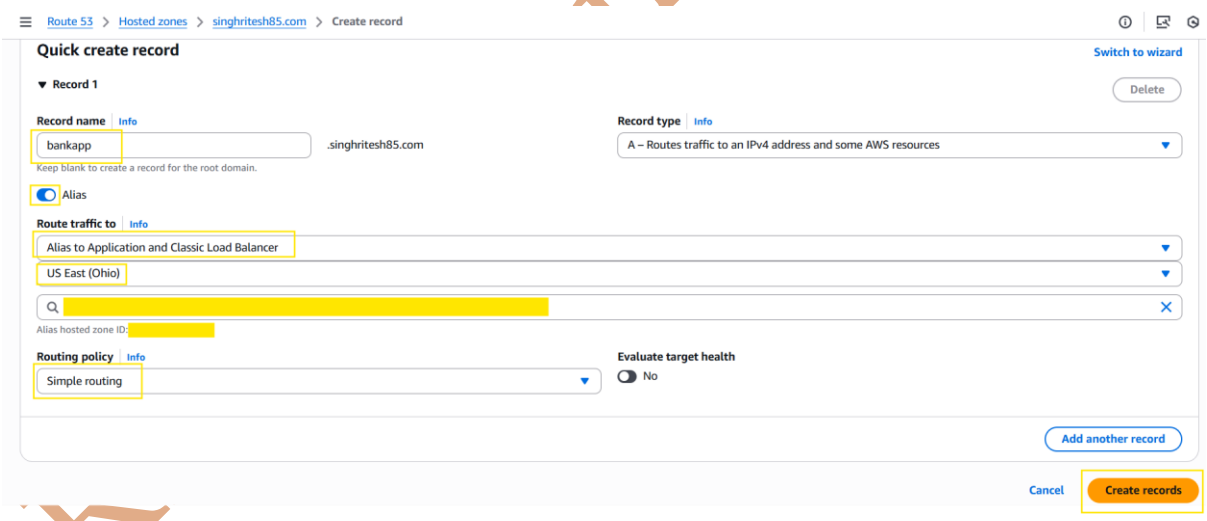


Then I created the ingress rule for bankapp and Record Set of A-Type with Alias in Route53 as shown in the screenshot attached below.

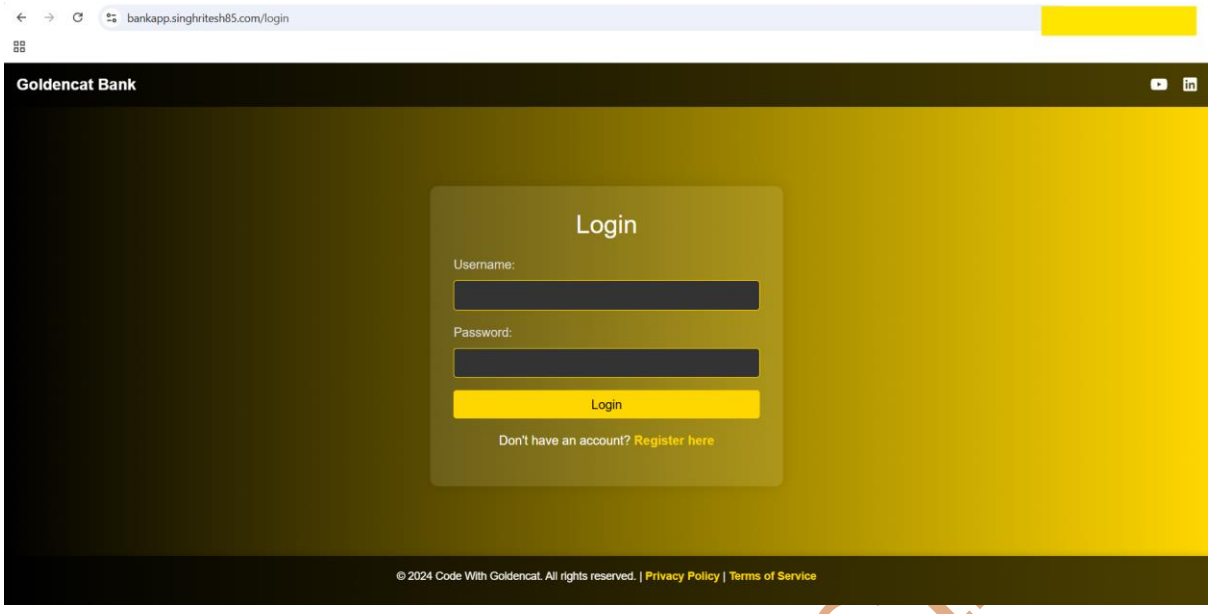
```
[jenkins@                    ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/bankapp-ingress created
```

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

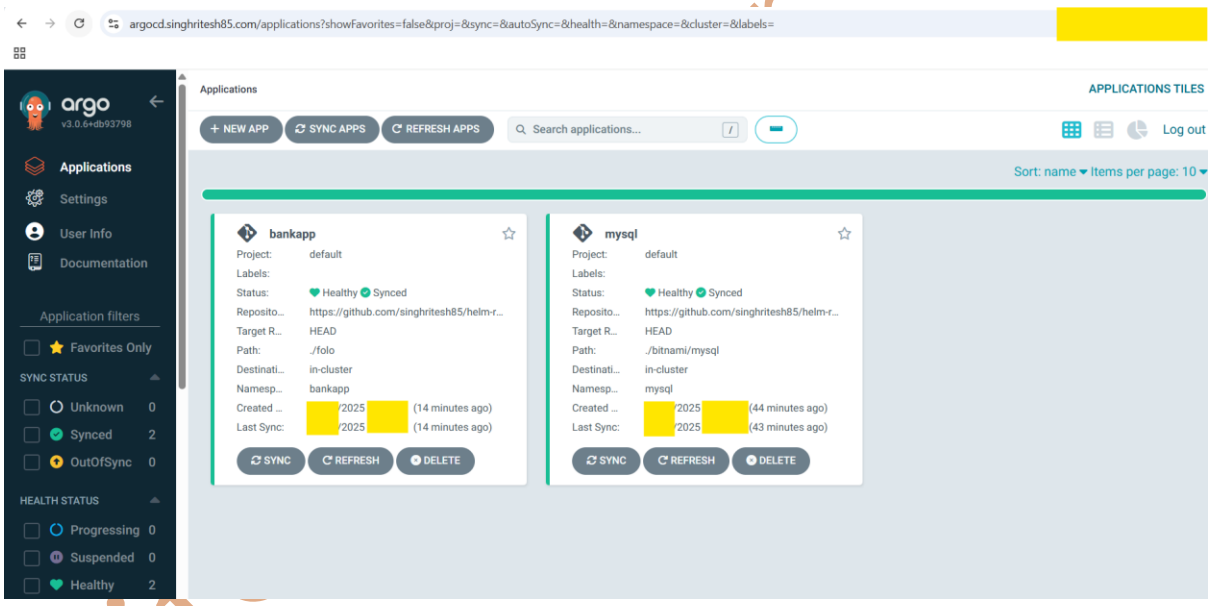
```
[jenkins@                ~]$ cat ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bankapp-ingress
  namespace: bankapp
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: bankapp.singhritesh85.com
    http: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



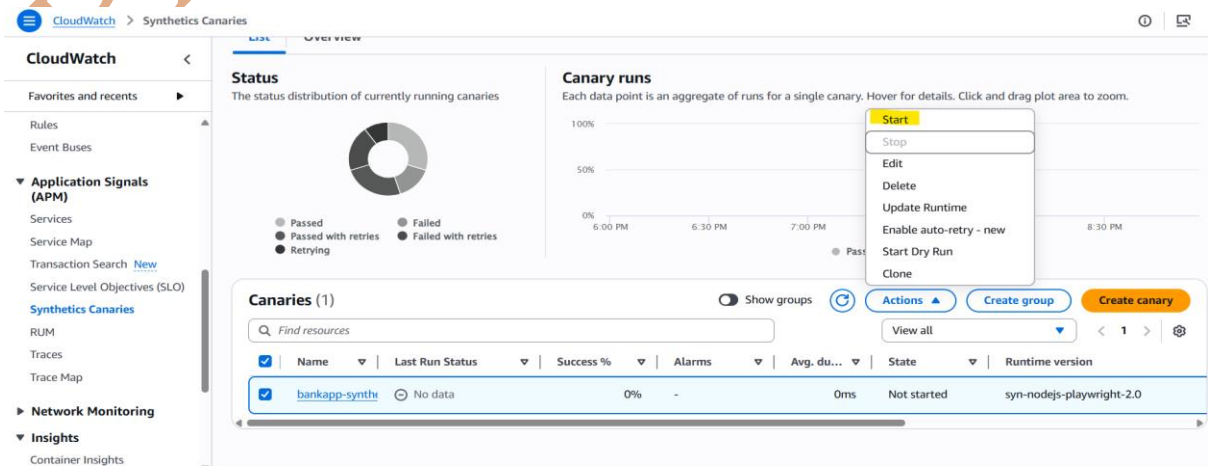
Finally, I was able to access the Application as shown in the screenshot attached below. After I found I was able to access the Bank Application then I confirmed the SNS Topic Subscription for the Group Email ID otherwise unnecessary Emails will be sent to the Group Email ID.

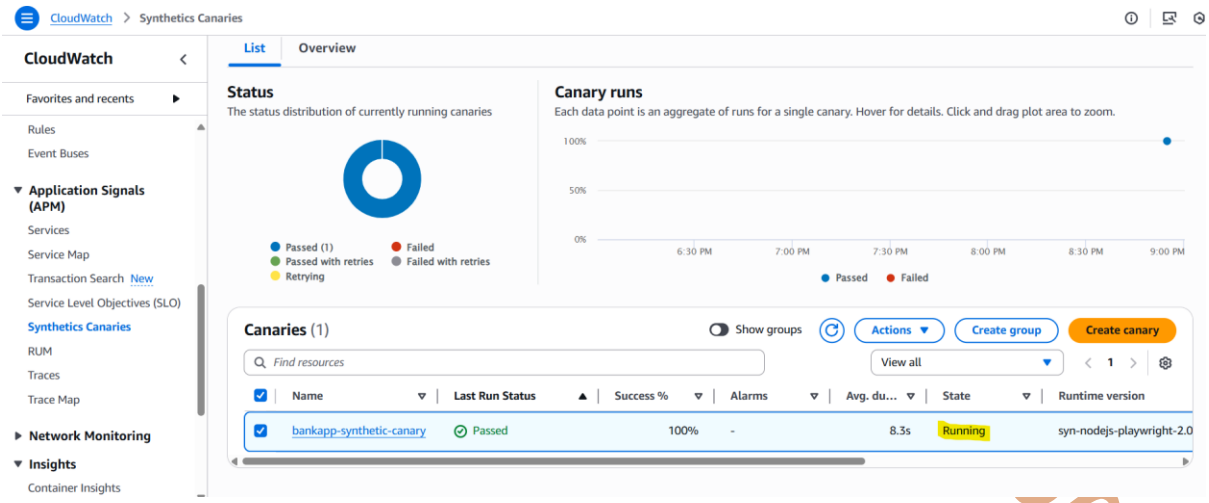


The screenshot for ArgoCD after the successful deployment of pods is as shown in the screenshot attached below.

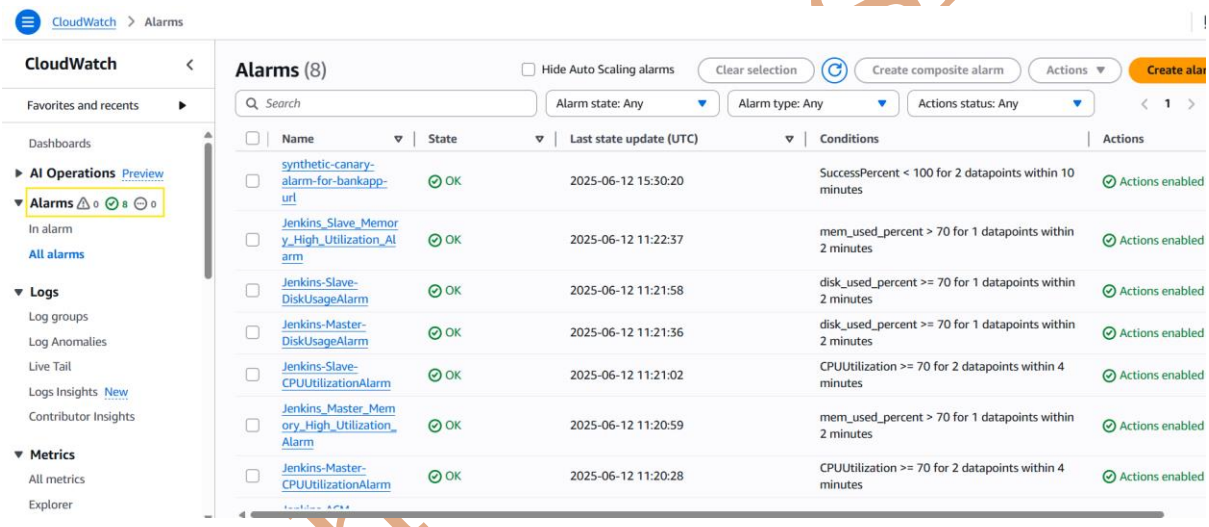


I started the synthetic canary after I found I was able to access the Application through the URL.

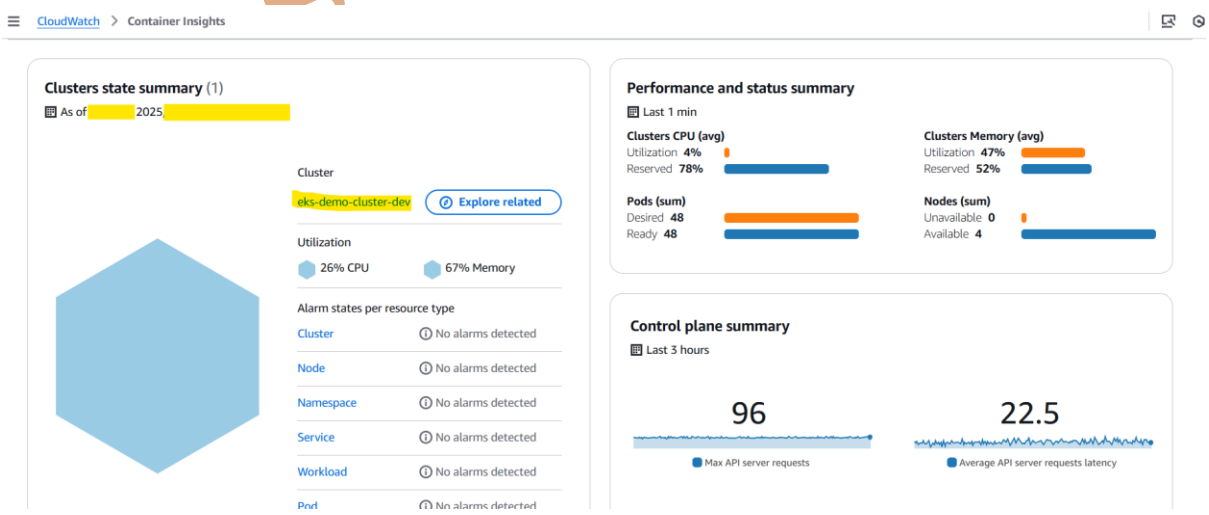


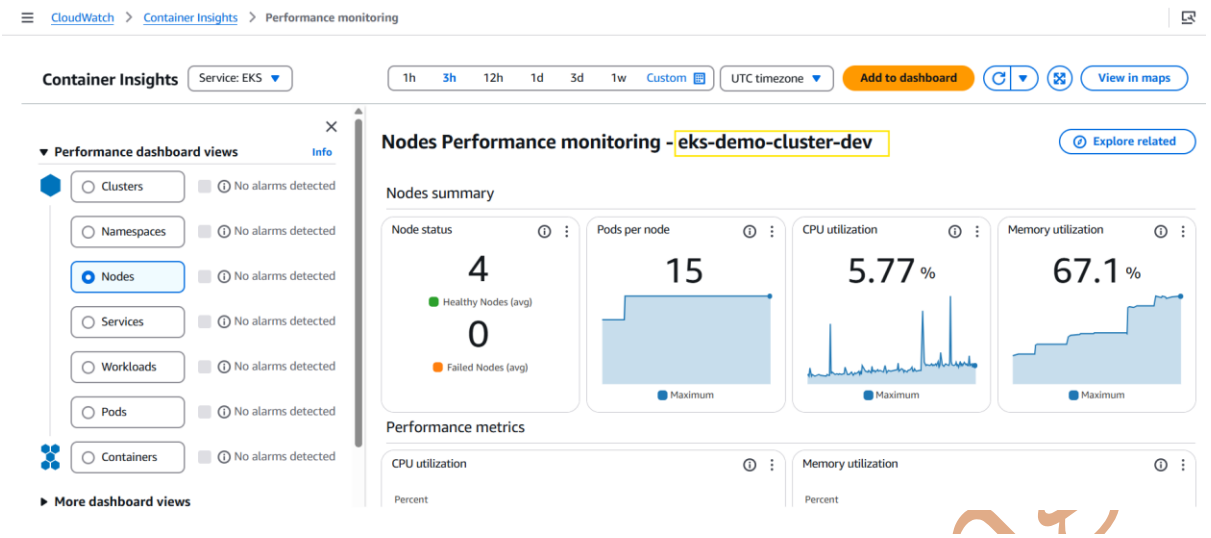


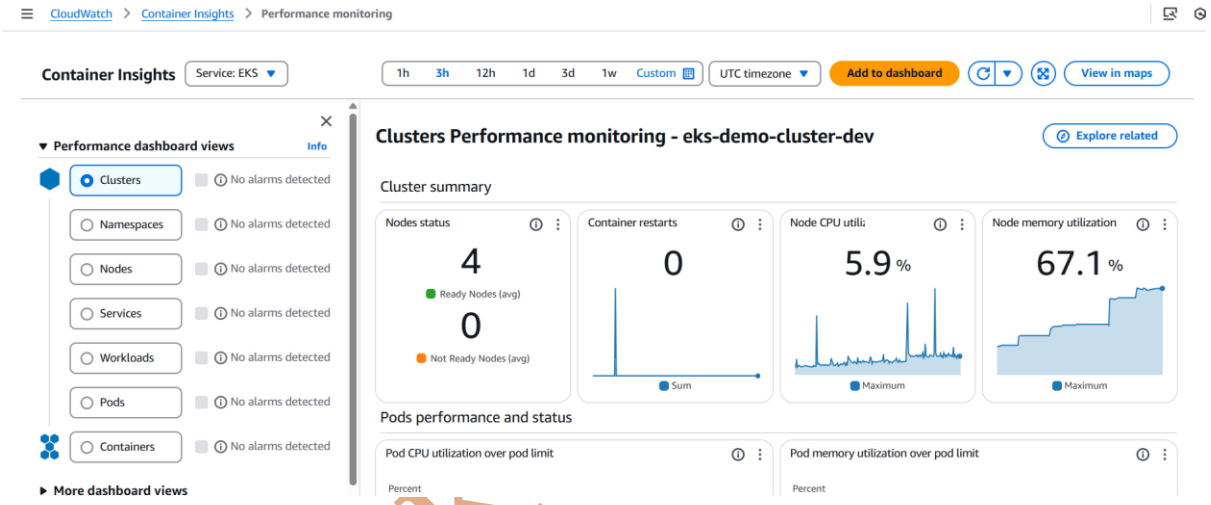
The CloudWatch Alarm was generated using Terraform as shown in the screenshot attached below.



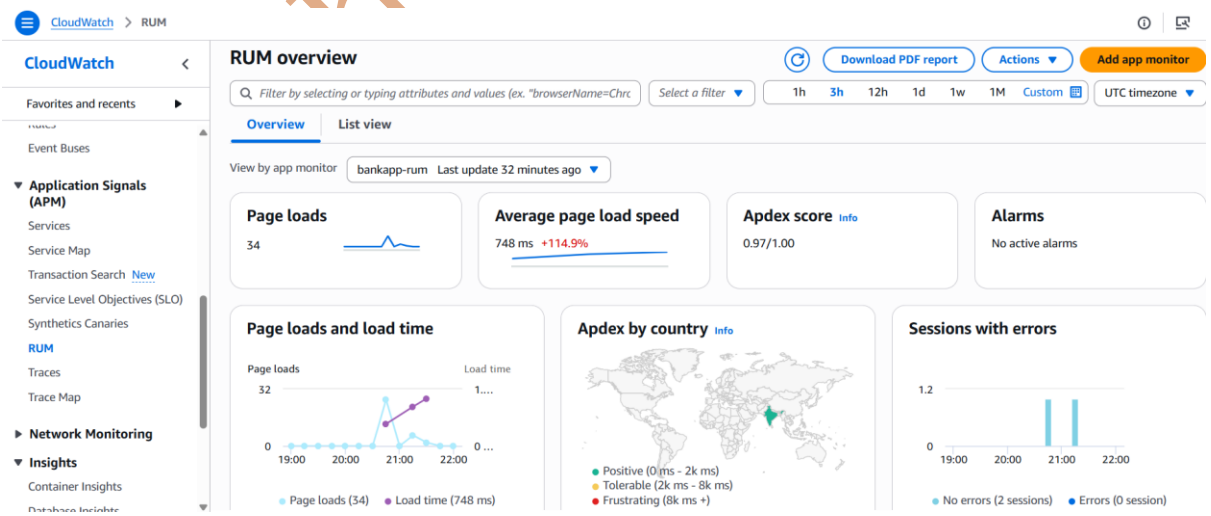
I enabled container Insight as shown in the screenshot attached below.







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.

Bank-App-Monitoring-Logging-using-CloudWatch-Private-eks / Dockerfile-eks

singhritesh85 Add files via upload — hours ago — History

Code | Blame | 17 lines (15 loc) · 700 Bytes | Raw

```
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"]
```

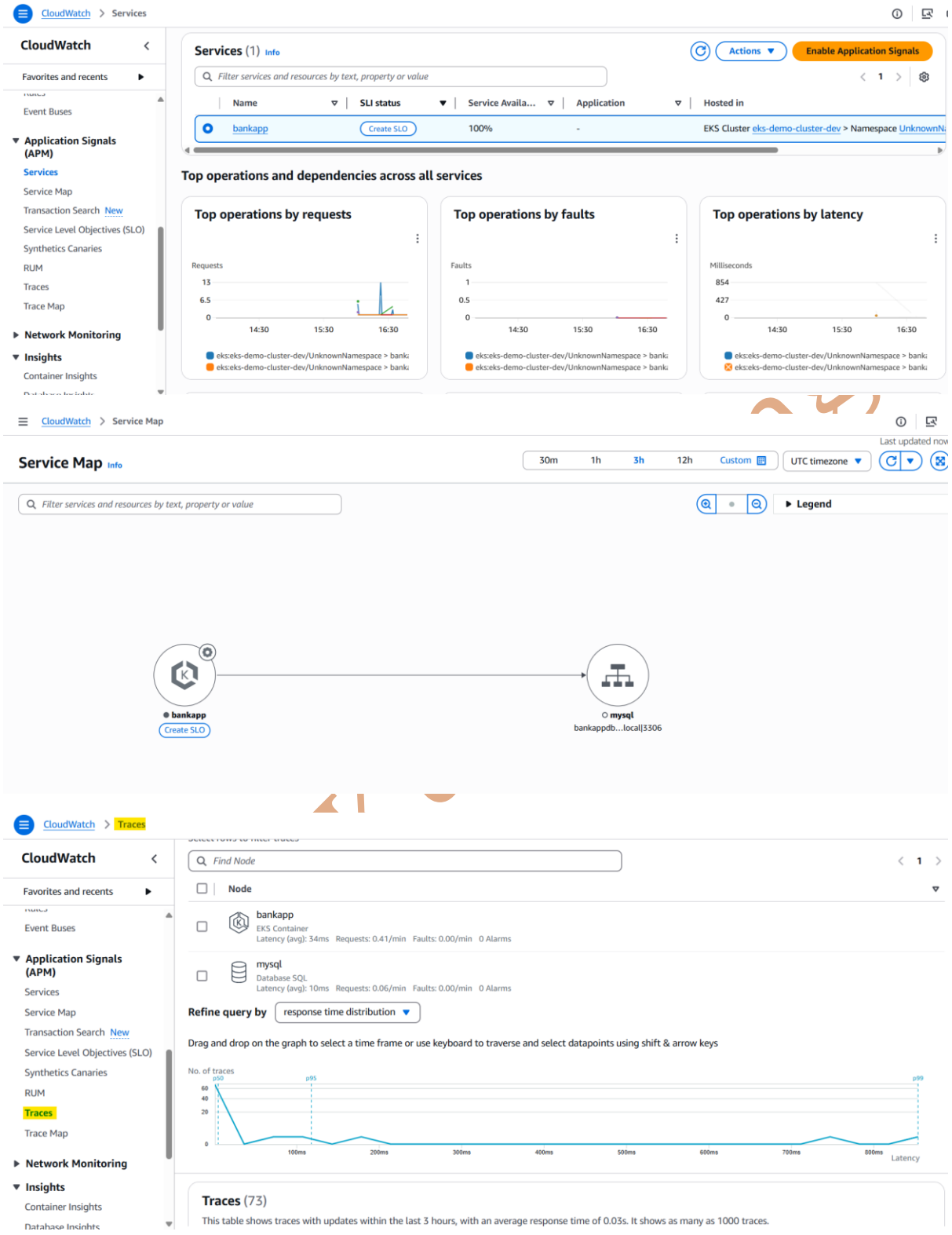
helm-repo-for-ArgoCD-cloudwatch-apm / folo / templates / deployment.yaml

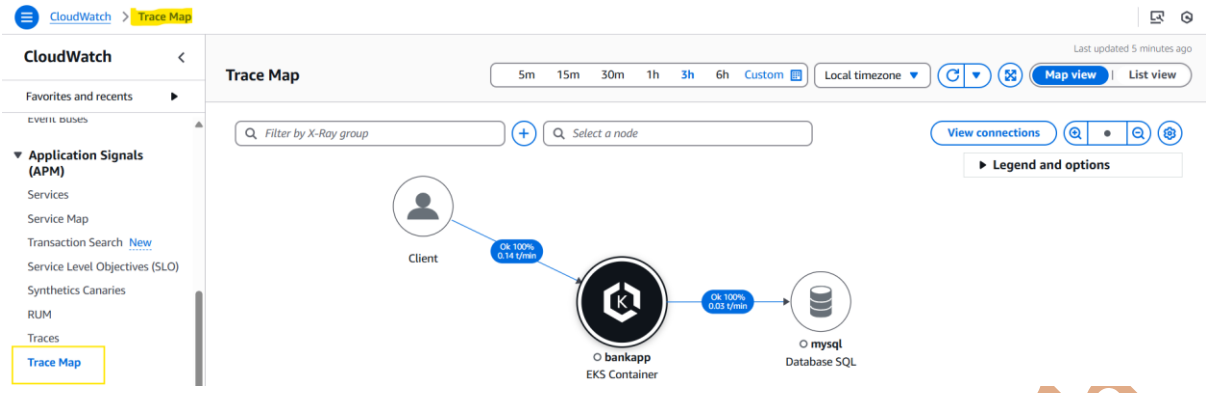
singhritesh85 Add files via upload — History

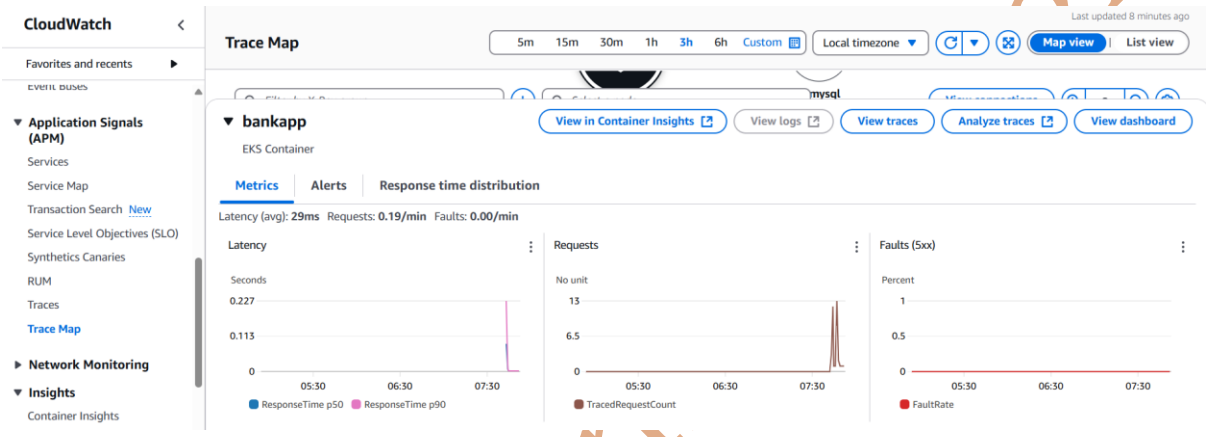
Code | Blame | 54 lines (54 loc) · 1.71 KB | Raw

```
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 recents
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
Services (1) Info
Actions
Enable Application Signals
Filter services and resources by text, property or value
Name
SLI status
Service Availa...
Application
Hosted in
bankapp
Create SLO
100%
EKS Cluster eks-demo-cluster-dev > Namespace UnknownN
Top operations and dependencies across all services
Top operations by requests
Requests
13
6.5
0
14:30
15:30
16:30
Top operations by faults
Faults
1
0.5
0
Top operations by latency
Milliseconds
854
427
CloudWatch > Service Map
Service Map Info
Last updated now
30m
1h
3h
12h
Custom
UTC timezone
Legend
bankapp
Create SLO
mysql
bankappdb...local|3306
CloudWatch > Traces
Find Node
Node
bankapp
EKS Container
Latency (avg): 34ms Requests: 0.41/min Faults: 0.00/min 0 Alarms
mysql
Database SQL
Latency (avg): 10ms Requests: 0.06/min Faults: 0.00/min 0 Alarms
Refine query by
response time distribution
Drag and drop on the graph to select a time frame or use keyboard to traverse and select datapoints using shift & arrow keys
No. of traces
p50
p95
p99
100ms
200ms
300ms
400ms
500ms
600ms
700ms
800ms
Latency
Traces (73)
This table shows traces with updates within the last 3 hours, with an average response time of 0.03s. It shows as many as 1000 traces.





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

```
[root@ 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@ ~]$ 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:
    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**

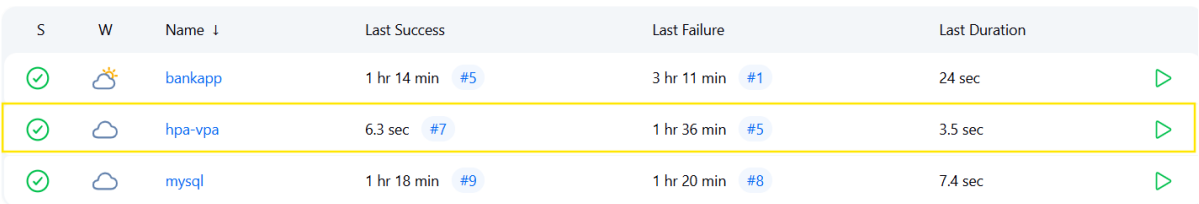
```
---
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: bankapp-hpa
  namespace: bankapp
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: bankapp-folo
  minReplicas: 1
  maxReplicas: 3
  metrics:
  - type: Resource
    resource:
      name: cpu
      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"]
```

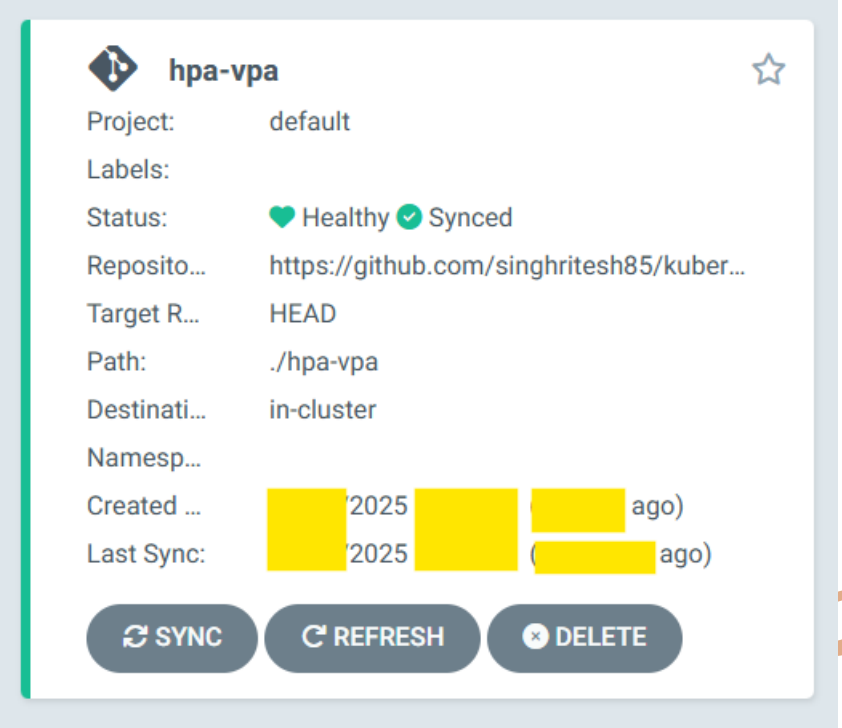
Jenkinsfile-hpa-vpa

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

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

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     Terminating       0          14m
mysql-0   0/1     Terminating       0          14m
mysql-0   0/1     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@                ~]# kubectl get pods -n mysql --watch
NAME      READY   STATUS    RESTARTS   AGE
mysql-0   1/1     Running   0          
mysql-1   1/1     Running   0          
mysql-2   1/1     Running   0          
```

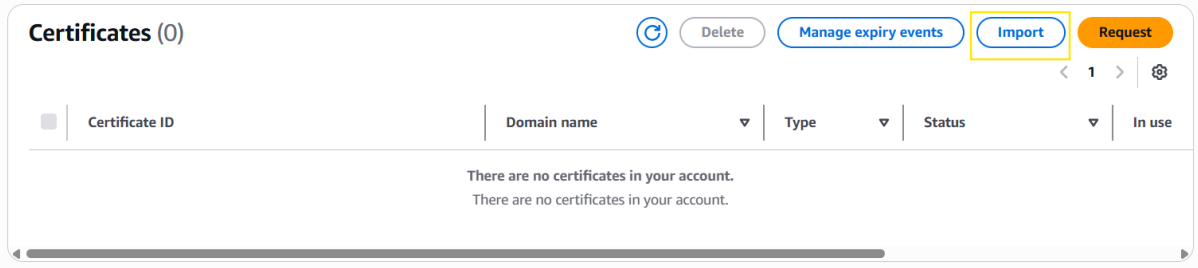
```
Containers:
  mysql:
    Container ID:    containerd://
    Image:           docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:        docker.io/bitnami/mysql@sha256:
    Port:            3306/TCP
    Host Port:       0/TCP
    SeccompProfile:  RuntimeDefault
    State:           Running
 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

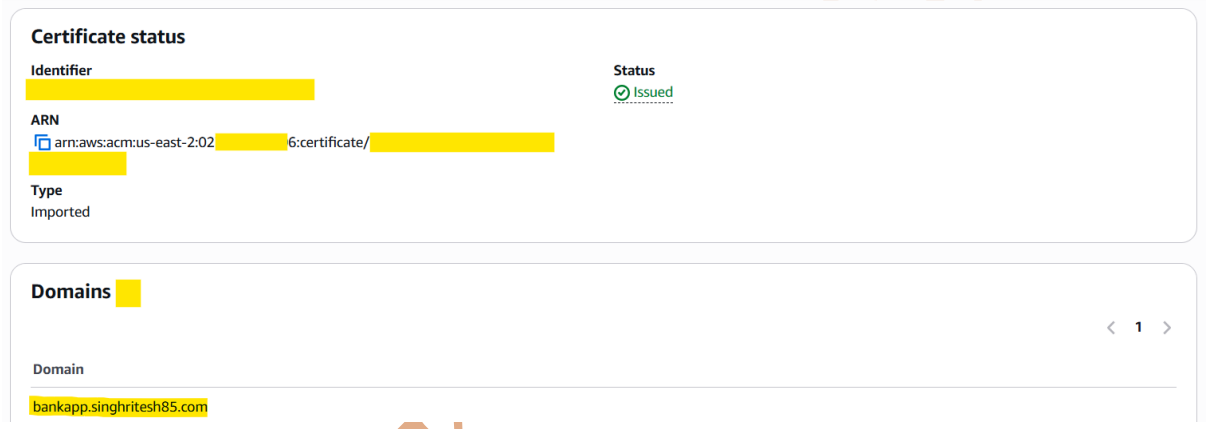
I checked the logs of vpa-admission-controller pod present in the namespace kube-system as found the entry as shown below which matched with the mysql container cpu and memory as shown in the screenshot attached above.

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

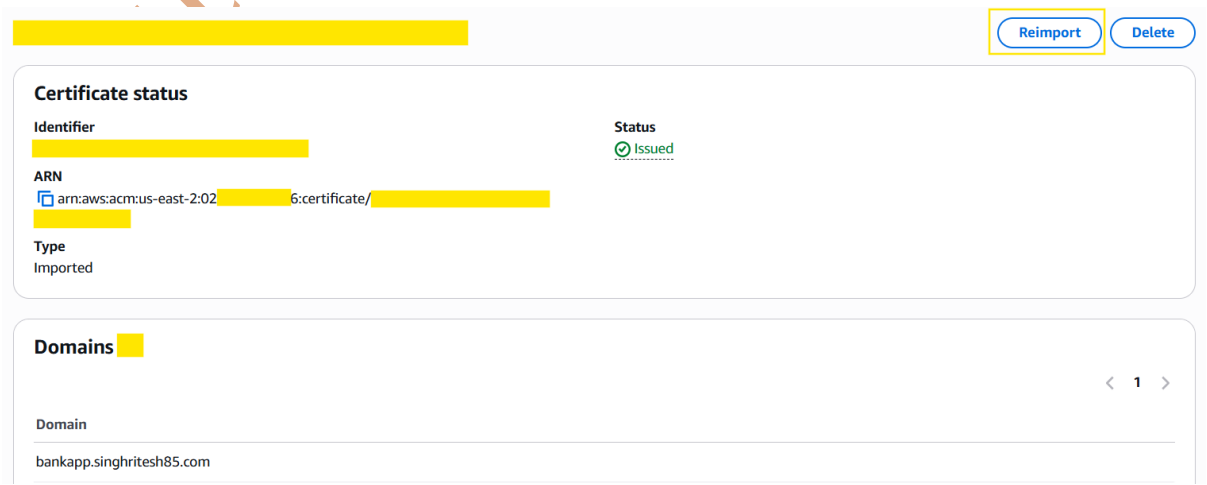
```
I0615  1 handler.go:95] "Processing vpa" vpa={"kind":"VerticalPodAutoscaler","apiVersion":"autoscaling.k8s.io/v1","metadata":{"name":"mysql-vpa","namespace":"mysql","creationTimestamp":null,"annotations":{"kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"autoscaling.k8s.io/v1\",\"kind\":\"VerticalPodAutoscaler\",\"metadata\":{\"annotations\":{},\"name\":\"mysql-vpa\",\"namespace\":\"mysql\"},\"spec\":{\"resourcePolicy\":{\"containerPolicies\":[{\"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\"}}}\n"},"managedFields":[{"manager":"kubectl-client-side-apply","operation":"Update","apiVersion":"autoscaling.k8s.io/v1","time":"2025 ","fieldsType":"FieldsV1","fieldsV1":{"f:metadata":{"f:annotations":{".":{},"f:kubectl.kubernetes.io/last-applied-configuration":{}}},"f:spec":{".":{},"f:resourcePolicy":{".":{},"f:containerPolicies":{}},"f:targetRef":{},"f:updatePolicy":{".":{},"f:updateMode":{}}}}}]},"spec":{"targetRef":{"kind":"StatefulSet","name":"mysql","apiVersion":"apps/v1"},"updatePolicy":{"updateMode":"Auto"},"resourcePolicy":{"containerPolicies":[{"containerName":"mysql","minAllowed":{"cpu":"800m","memory":"850Mi"},"maxAllowed":{"cpu":"1024m","memory":"1Gi"},"controlledResources":["cpu","memory"]}]}},"status":{}}

I0615  1 handler.go:81] "Admitting pod" pod="mysql/mysql-1"
I0615  1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-1"
I0615  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/requests/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  1 handler.go:81] "Admitting pod" pod="mysql/mysql-2"
I0615  1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-2"
I0615  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/requests/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  1 handler.go:81] "Admitting pod" pod="mysql/mysql-0"
I0615  1 recommendation_provider.go:121] "Updating requirements for pod" pod="mysql-0"
I0615  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/requests/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.





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.



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", "1.29.8-20241109", "1.30.4-20241109", "1.31.7-20250610", "1.32.3-20250610", "1.33.0-20250610"]
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_guardduty = ["v1.10.0-eksbuild.2", "v1.10.0-eksbuild.1", "v1.8.1-eksbuild.2", "v1.4.1-eksbuild.2", "v1.4.0-eksbuild.1", "v1.3.1-eksbuild.1", "v1.2.0-eksbuild.3"]
addon_version_kubeproxy = ["v1.30.9-eksbuild.3", "v1.28.15-eksbuild.9", "v1.27.10-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" "eksnode" {
  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-[redacted]", "subnet-[redacted]", "subnet-[redacted]"]    #### 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@[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
```

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@[redacted] ~]# 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
```

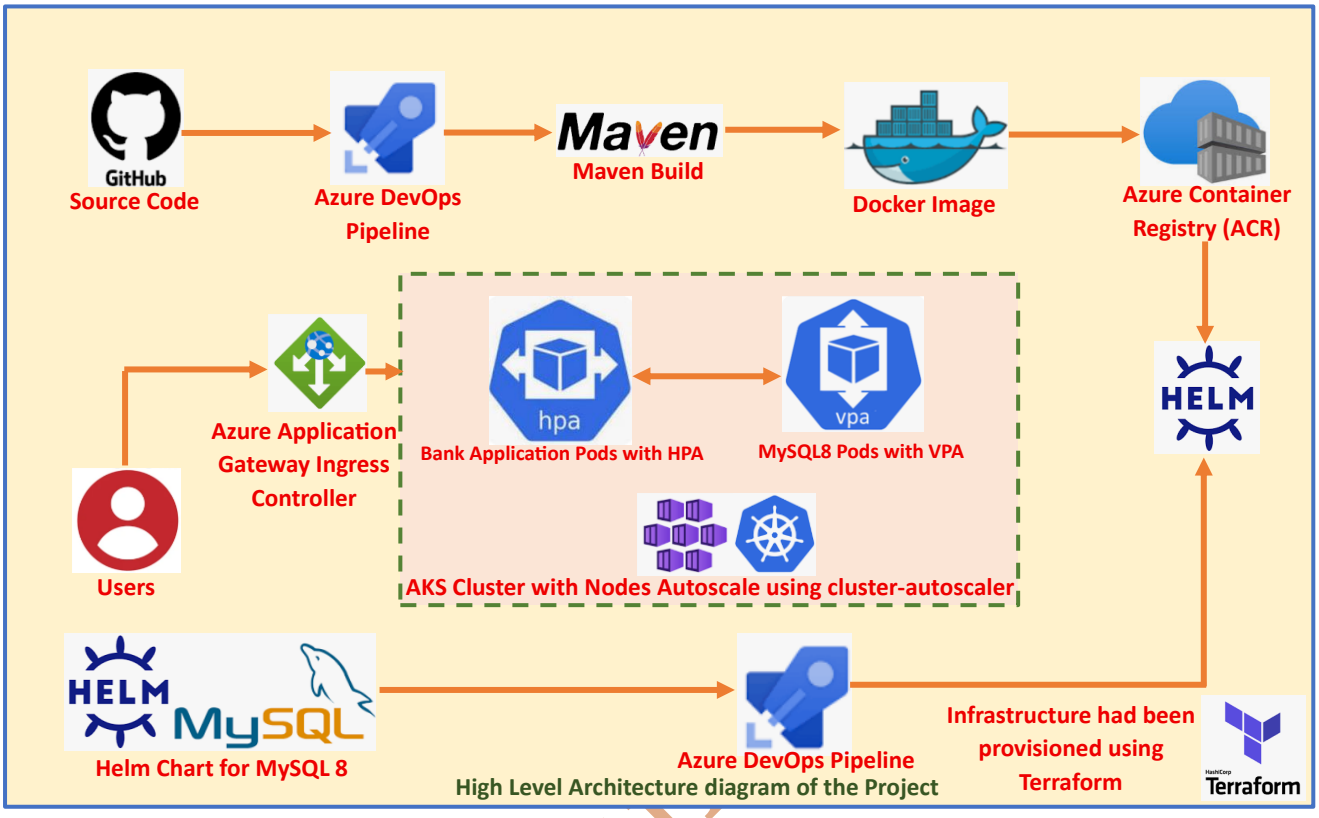
```
[root@                ~]# kubectl edit pod mysql-0 -n mysql --subresource=resize
pod/mysql-0 edited
```

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@                ~]# 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



High Level Architecture diagram of the Project

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 Autoscale 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

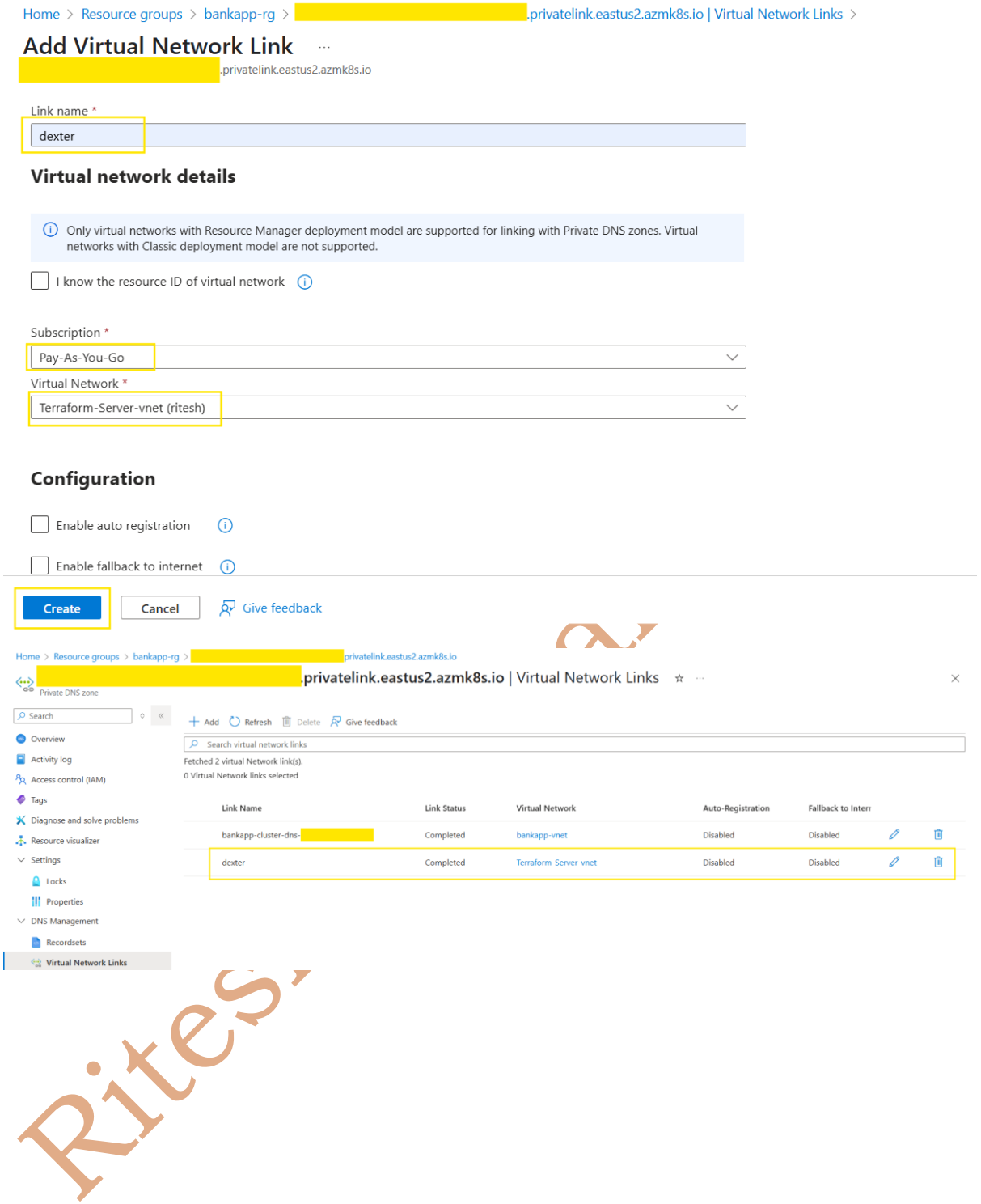
terraform plan ---------> Check which resources are going to be created.

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

```
module.aks.azurerm_monitor_metric_alert.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.alert_rule1: Still creating... [01m40s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule2: Still creating... [01m50s elapsed]
module.aks.azurerm_monitor_metric_alert.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.alert_rule1: Still creating... [01m50s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule2: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule3: Still creating... [02m00s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule2: Creation complete after 2m2s [id=/subscriptions/[redacted]/resourceGroups/banka
pp-rg/providers/Microsoft.Insights/metricAlerts/bankapp-alert-rule2]
module.aks.azurerm_monitor_metric_alert.alert_rule3: Creation complete after 2m2s [id=/subscriptions/[redacted]/resourceGroups/banka
pp-rg/providers/Microsoft.Insights/metricAlerts/bankapp-alert-rule3]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Creation complete after 2m2s [id=/subscriptions/[redacted]/resourceGroups/banka
pp-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]/reso
urceGroups/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-clust
er"
  "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 > .privatelink.eastus2.azmk8s.io | Virtual Network Links >
Add Virtual Network Link
.privatelink.eastus2.azmk8s.io
Link name *
dexter
Virtual network details
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
Subscription *
Pay-As-You-Go
Virtual Network *
Terraform-Server-vnet (ritesh)
Configuration
Enable auto registration
Enable fallback to internet
Create
Cancel
Give feedback
Home > Resource groups > bankapp-rg > privatelink.eastus2.azmk8s.io
.privatelink.eastus2.azmk8s.io | Virtual Network Links
Private DNS zone
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Locks
Properties
DNS Management
Recordsets
Virtual Network Links
Add
Refresh
Delete
Give feedback
Search virtual network links
Fetched 2 virtual Network link(s).
0 Virtual Network links selected
Link Name
Link Status
Virtual Network
Auto-Registration
Fallback to Interr
bankapp-cluster-dns-
Completed
bankapp-vnet
Disabled
Disabled
dexter
Completed
Terraform-Server-vnet
Disabled
Disabled

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 * peer-2

Virtual network deployment model ⓘ
- (•) Resource manager
- ( ) Classic

I know my resource ID ⓘ [ ]

Subscription * Pay-As-You-Go

Virtual network * bankapp-vnet (bankapp-rg)

Remote virtual network peering settings

Allow 'bankapp-vnet' to access 'Terraform-Server-vnet' ⓘ [x]

Add | Cancel

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

Add peering

Terraform-Server-vnet

Local virtual network summary

Peering link name * peer-1

Local virtual network peering settings

Allow 'Terraform-Server-vnet' to access 'bankapp-vnet' ⓘ [x]

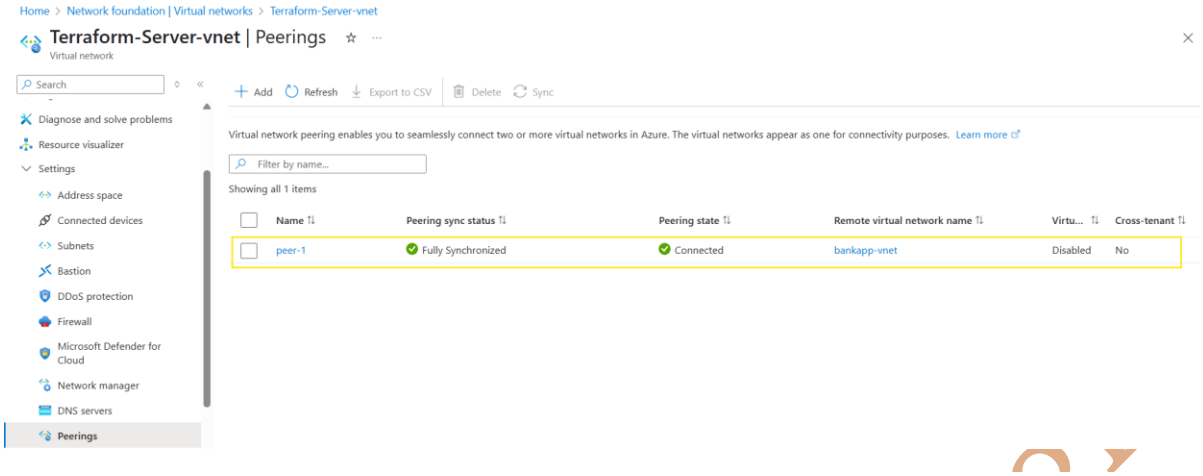
Allow 'Terraform-Server-vnet' to receive forwarded traffic from 'bankapp-vnet' ⓘ [ ]

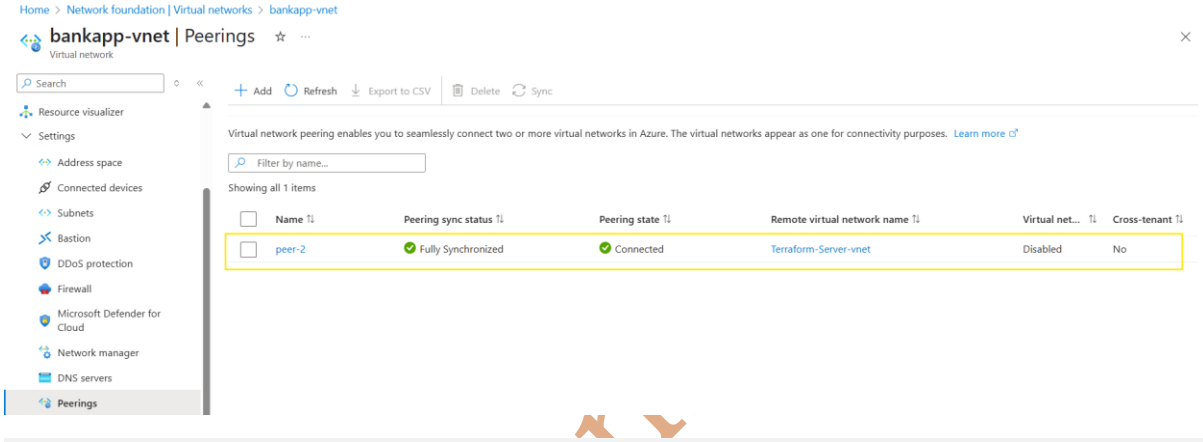
Allow gateway or route server in 'Terraform-Server-vnet' to forward traffic to 'bankapp-vnet' ⓘ [ ]

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

Add | Cancel

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



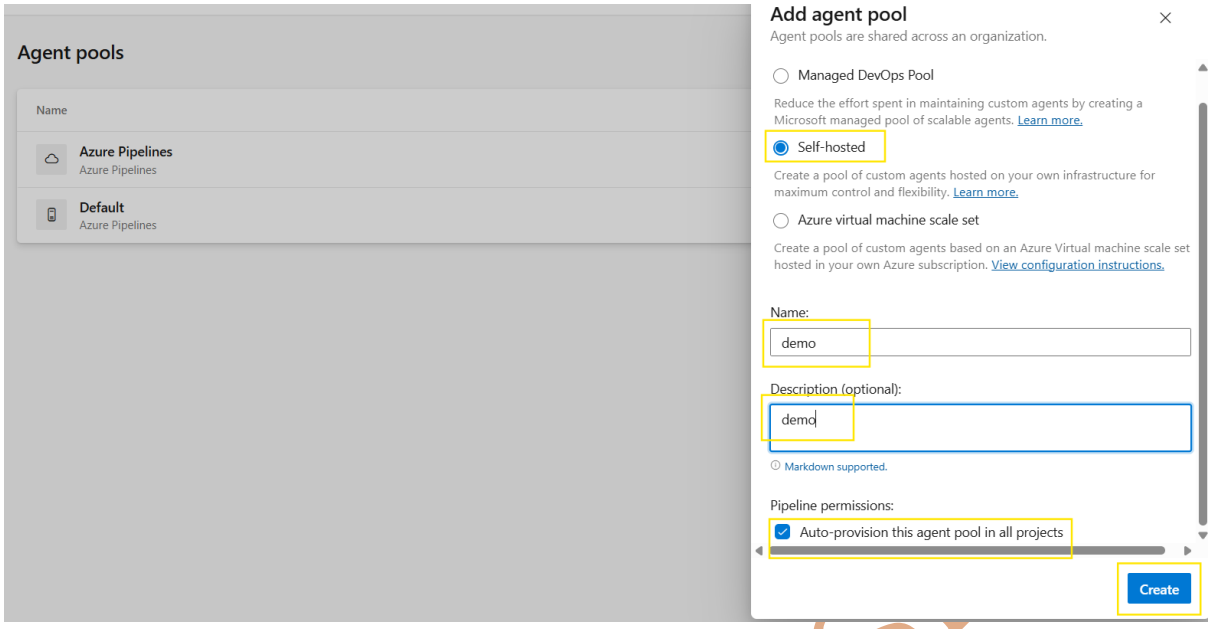



```
[root@Terraform-Server ~]# kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
aks-agentpool-[redacted]-vmss000000   Ready    <none>   94m   v1.33.0
aks-userpool-[redacted]-vmss000000    Ready    <none>   87m   v1.33.0
```

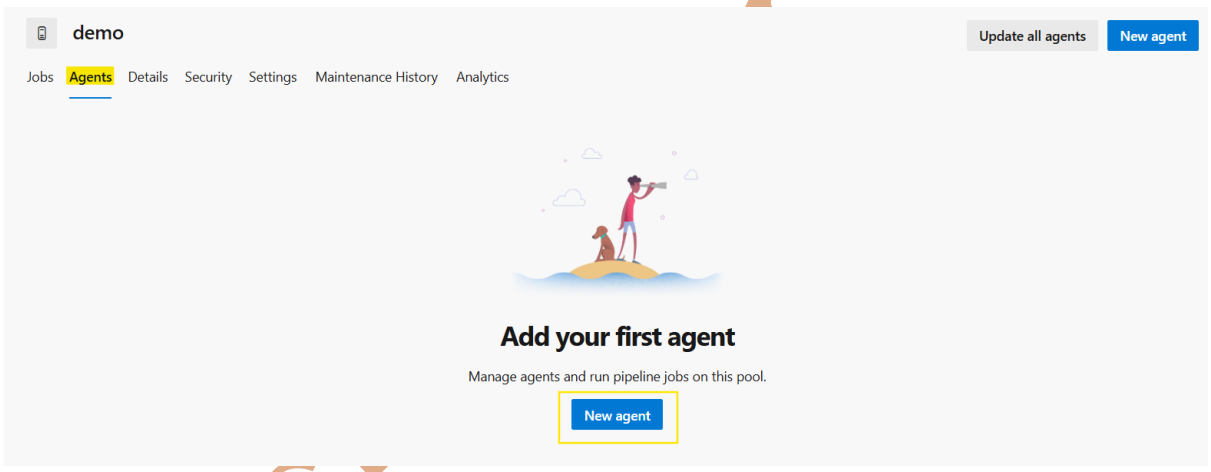
Installation of Self-hosted Azure DevOps Agent

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-  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)... , , , ...
Connecting to download.agent.dev.azure.com (download.agent.dev.azure.com)| |: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 (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
```

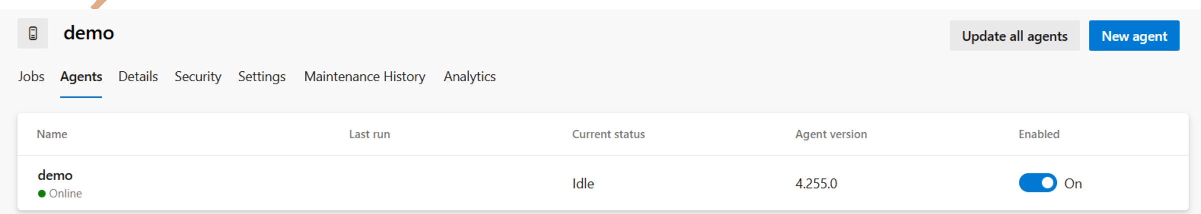
```
[root@devopsagent-vm myagent]# chown -R demo:demo /opt/myagent/
[root@devopsagent-vm myagent]# ./bin/installdependencies.sh
```

```
[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 sou
rces from Git repositories.

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

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

>> Connect:

Enter server URL > 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.
```

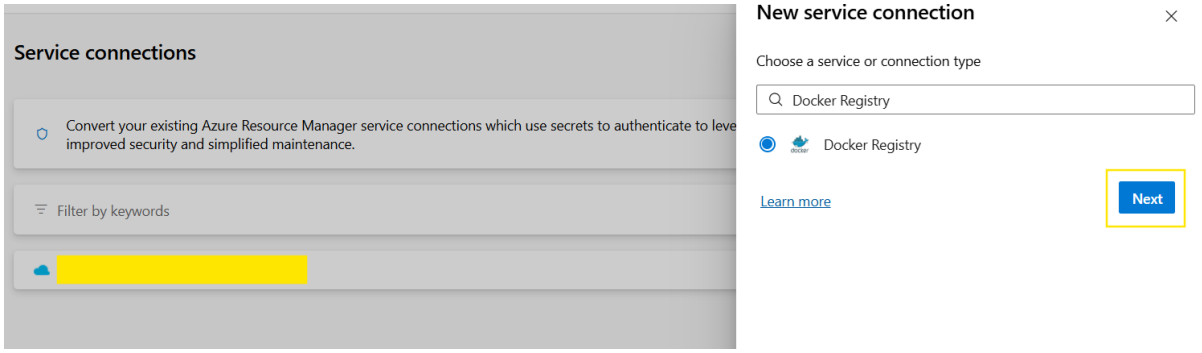
```
Enter work folder (press enter for _work) >
2025[redacted]: Settings Saved.
[demo@devopsagent-vm myagent]$ sudo ./svc.sh install
Creating launch agent in /etc/systemd/system/vsts.agent.[redacted].demo.demo.service
Run as user: demo
Run as uid: 1001
gid: 1001
/sbin/sestatus
SELinux status:                 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
           ├─14089 /bin/bash /opt/myagent/runsvc.sh
           └─14093 ./externals/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:/bin:/opt/sonar-scanner/bin:/opt/apache-m…r/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.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] 4s ago
 Main PID: [redacted] (runsvc.sh)
    Tasks: 25 (limit: [redacted])
   Memory: 60.8M
   CGroup: /system.slice/vsts.agent.[redacted].demo.demo.service
           ├─14089 /bin/bash /opt/myagent/runsvc.sh
```

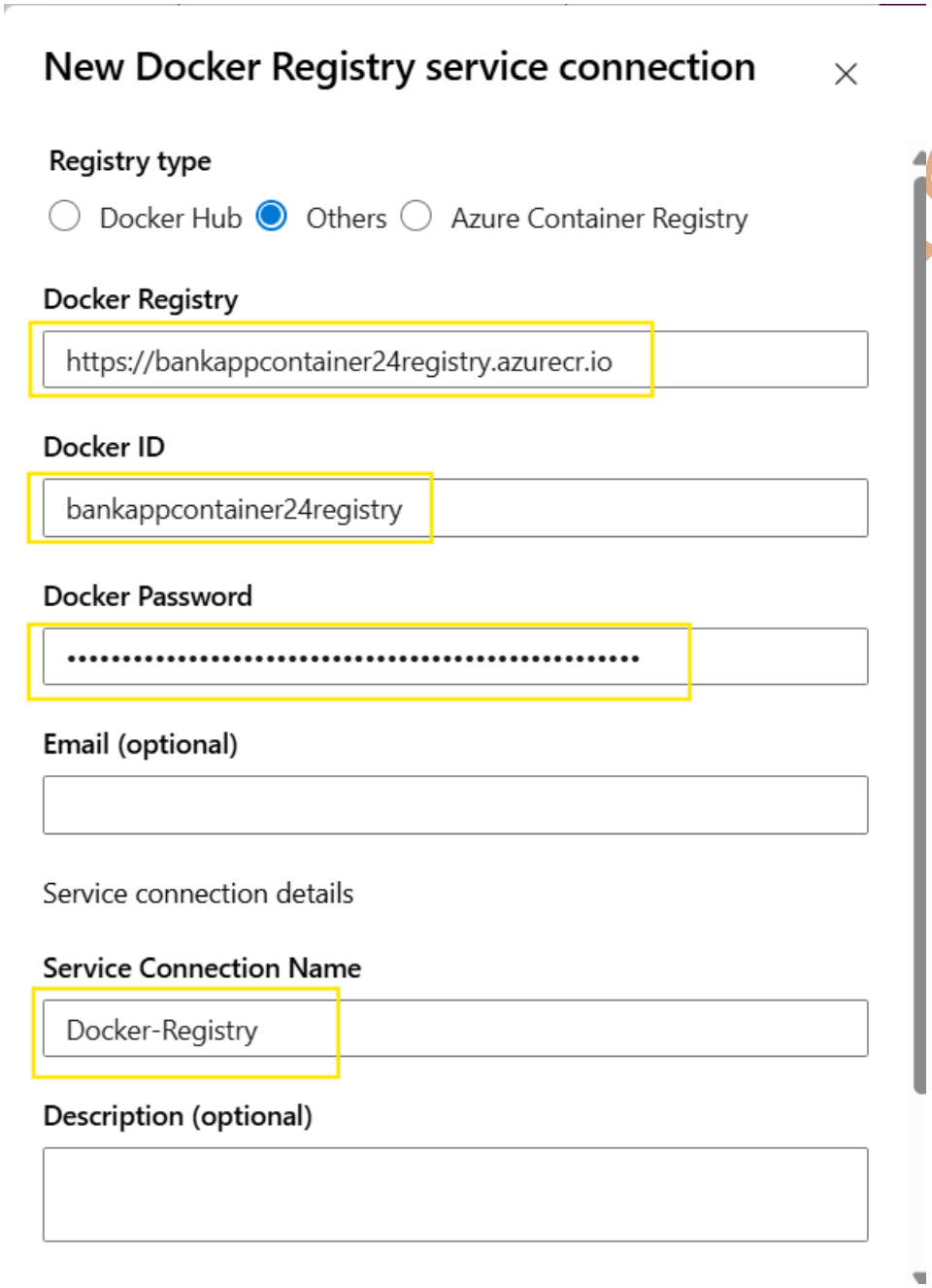
```
[demo@devopsagent-vm myagent]$ ./env.sh
[demo@devopsagent-vm myagent]$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/opt/sonar-scanner/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/opt/dependency-check/bin:/usr/local
/bin
```

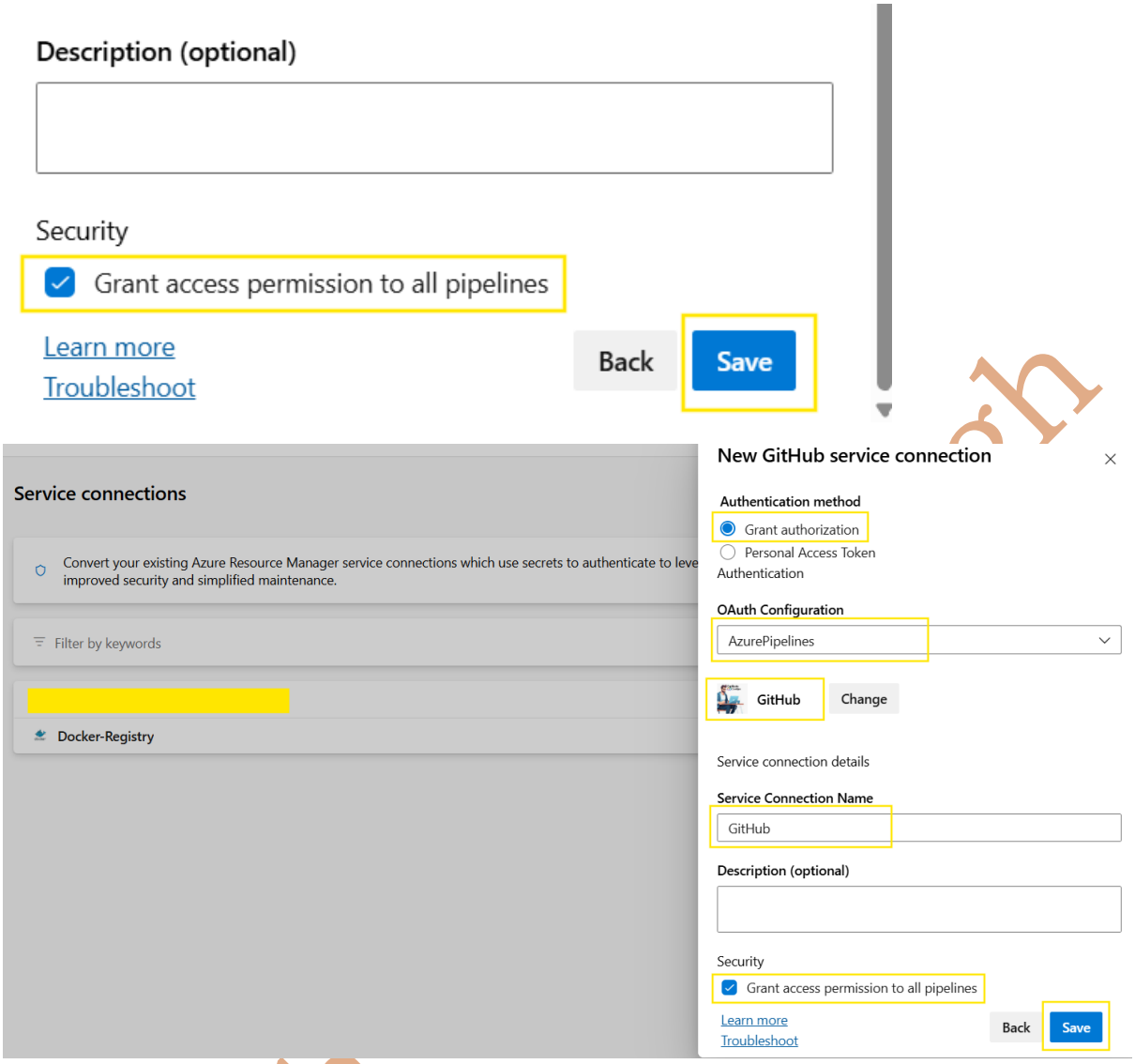
Now the Self-hosted Azure DevOps Agent is in Online mode as shown in the screenshot attached below.



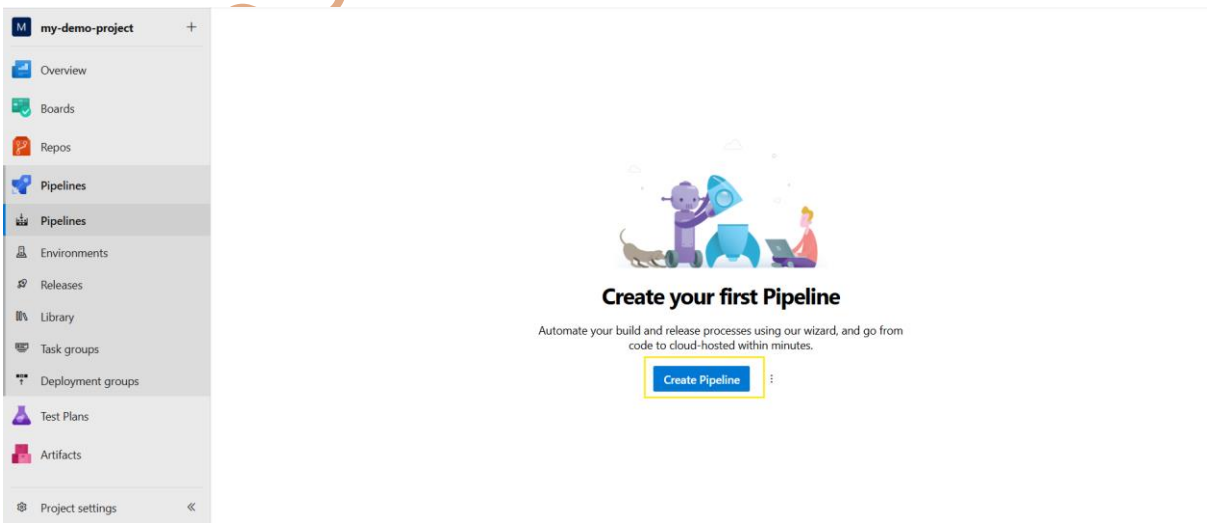
Then I went to Project in the **Azure DevOps Organization > Project Settings > Service connections** and created a new service connection for Azure Container Registry and GitHub Account as shown in the screenshot attached below.

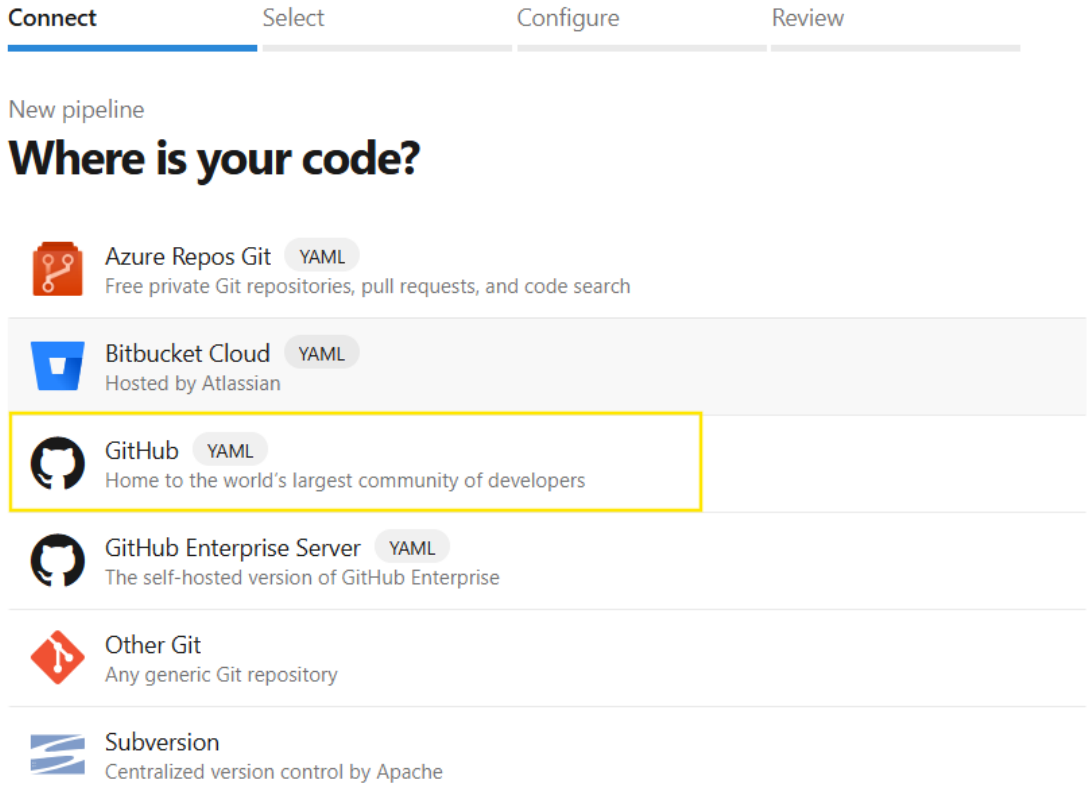

Service connections
Convert your existing Azure Resource Manager service connections which use secrets to authenticate to leve
improved security and simplified maintenance.
Filter by keywords
New service connection
Choose a service or connection type
Docker Registry
Docker Registry
Learn more
Next

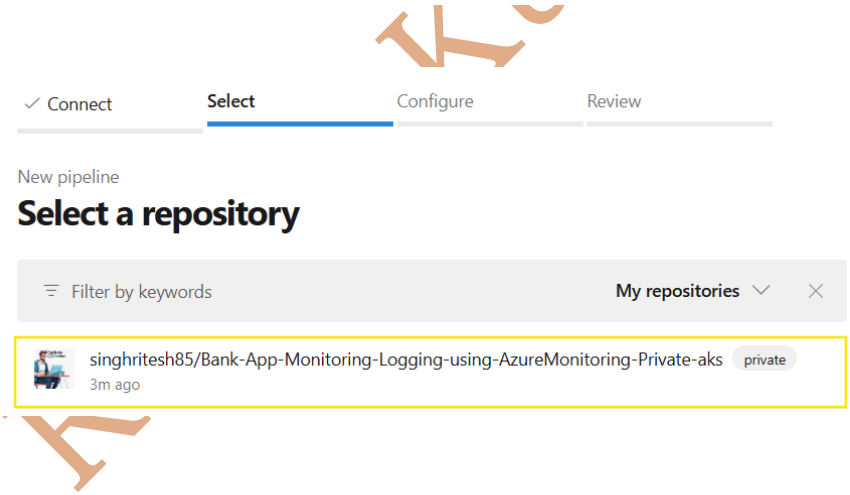

New Docker Registry service connection
Registry type
Docker Hub
Others
Azure Container Registry
Docker Registry
https://bankappcontainer24registry.azurecr.io
Docker ID
bankappcontainer24registry
Docker Password
Email (optional)
Service connection details
Service Connection Name
Docker-Registry
Description (optional)

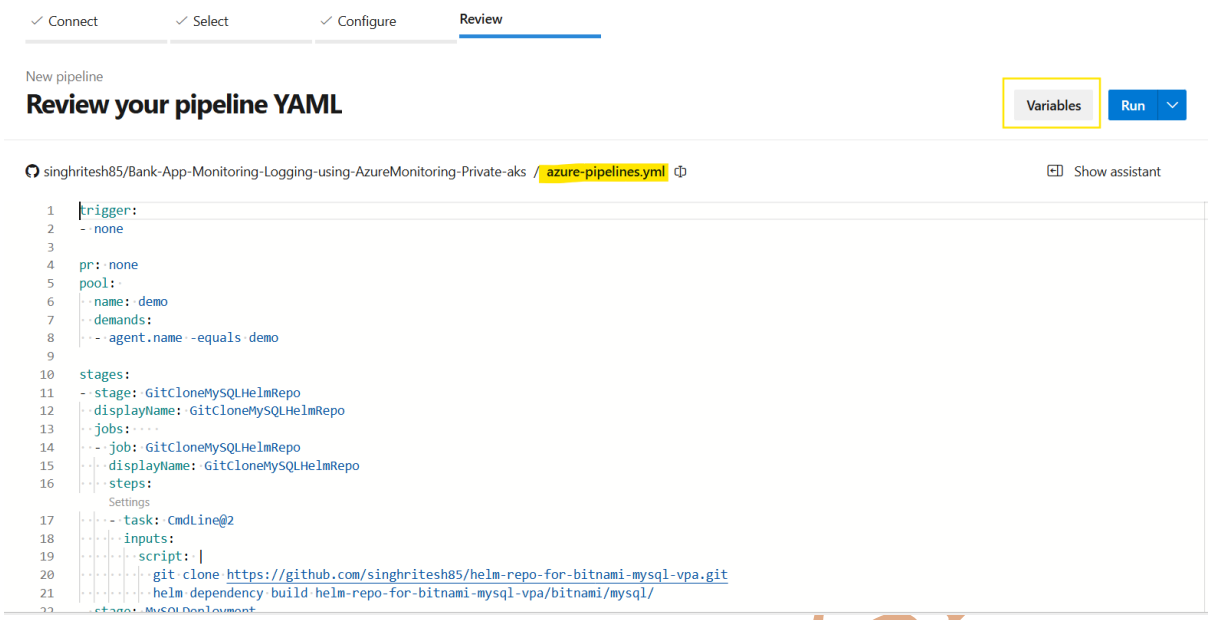


Then I had created the Azure DevOps Pipeline first for mysql and then for bankapp and executed in them in the same order.

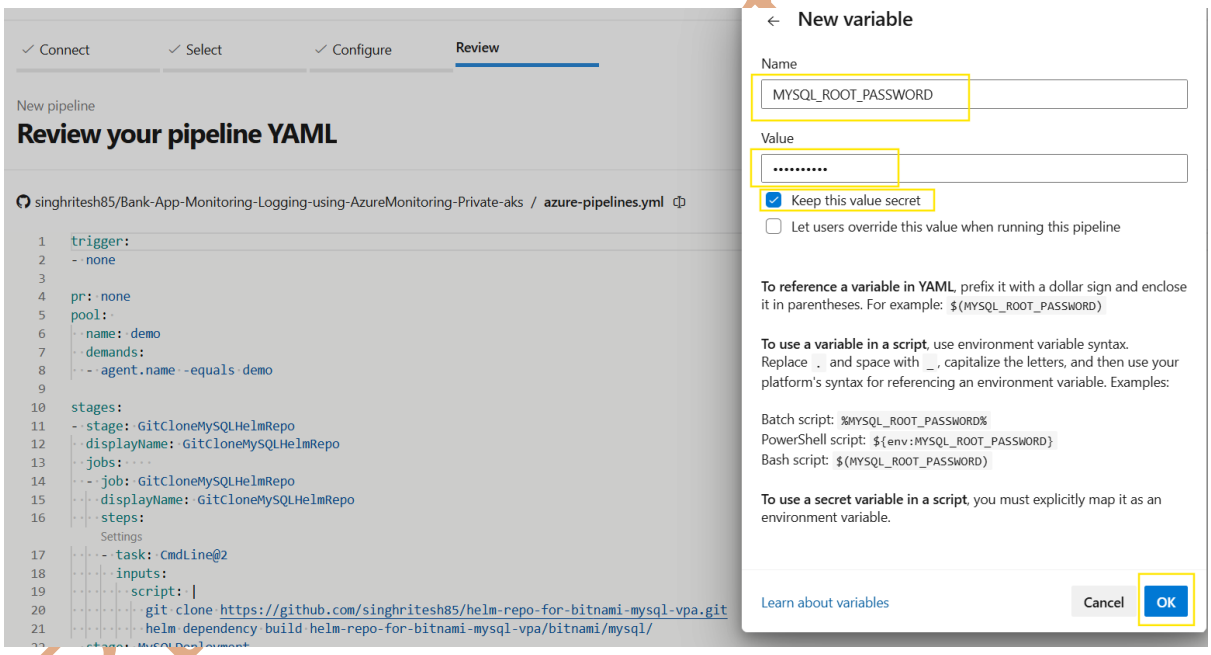


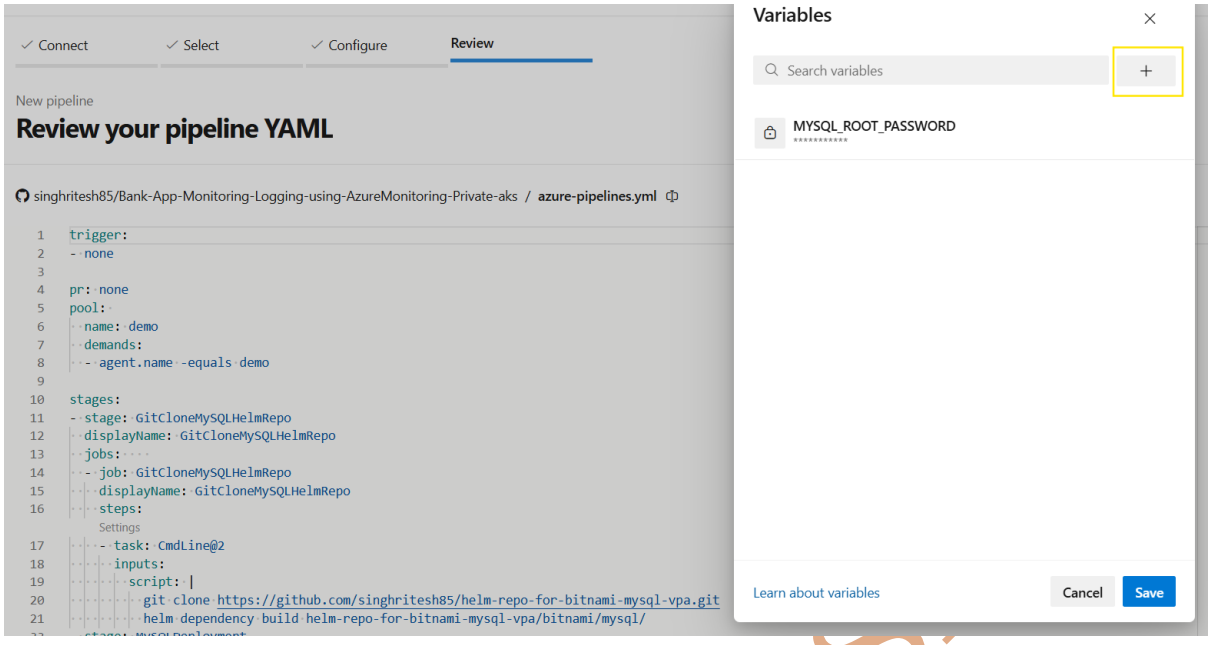

Connect
Select
Configure
Review
New pipeline
Where is your code?
Azure Repos Git YAML
Free private Git repositories, pull requests, and code search
Bitbucket Cloud YAML
Hosted by Atlassian
GitHub YAML
Home to the world's largest community of developers
GitHub Enterprise Server YAML
The self-hosted version 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.

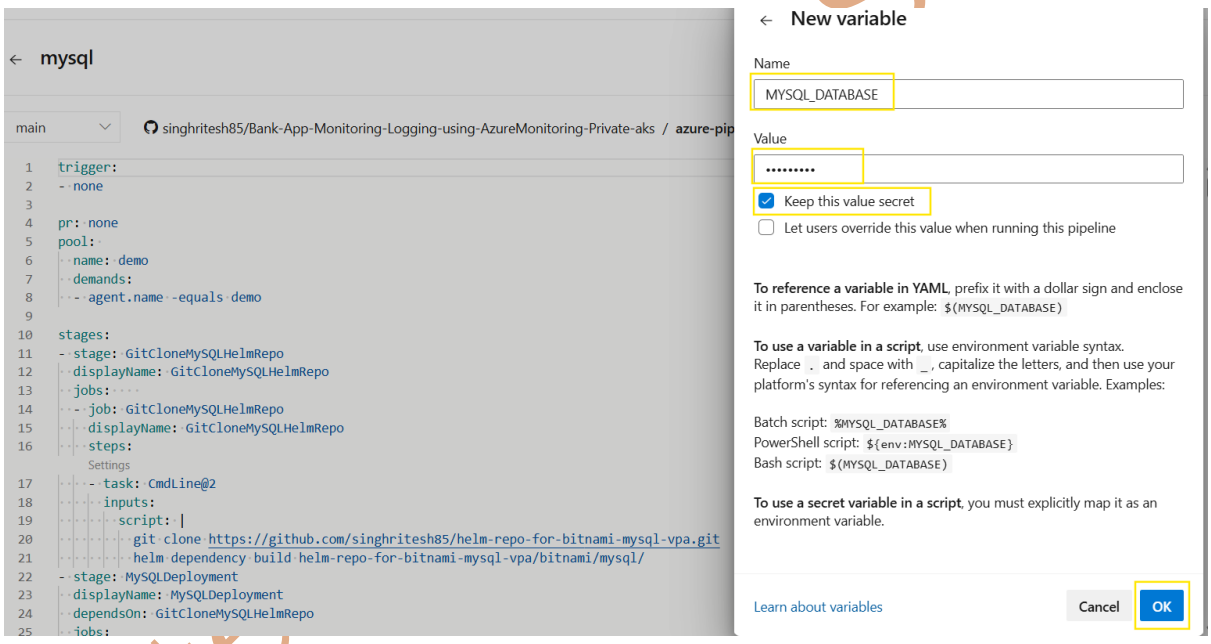

Connect
Select
Configure
Review
New pipeline
Select a repository
Filter by keywords
My repositories
singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks
private
3m ago

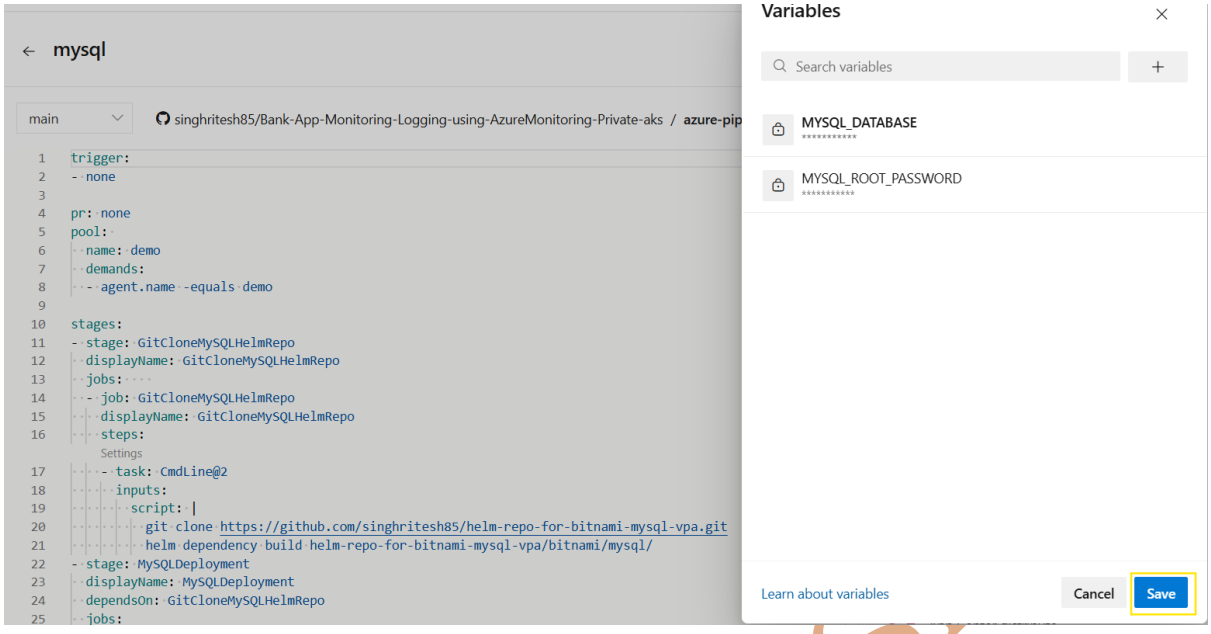


Before running this pipeline, I created two variables MYSQL_ROOT_PASSWORD and MYSQL_DATABASE as shown in the screenshot attached below.

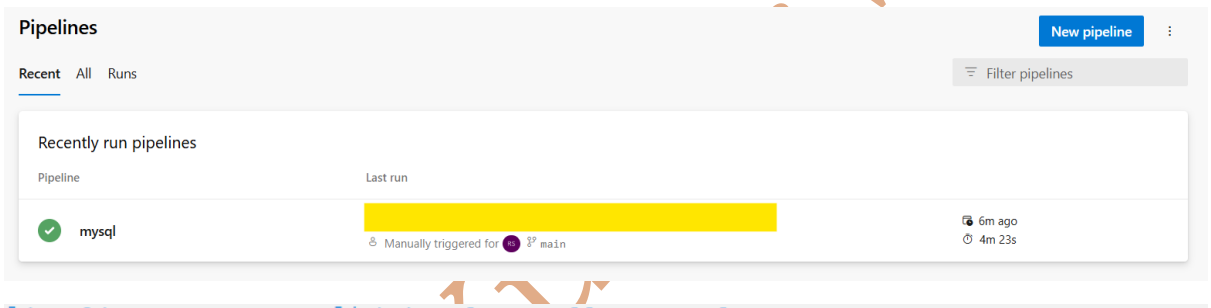



Connect
Select
Configure
Review
New pipeline
Review your pipeline YAML
singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks / azure-pipelines.yml
trigger:
- none
pr: none
pool:
name: demo
demands:
- agent.name -equals demo
stages:
- stage: GitCloneMySQLHelmRepo
displayName: GitCloneMySQLHelmRepo
jobs:
- job: GitCloneMySQLHelmRepo
displayName: GitCloneMySQLHelmRepo
steps:
Settings
- 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/
Variables
Search variables
MYSQL_ROOT_PASSWORD
Learn about variables
Cancel
Save


mysql
main
singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks / azure-pip
trigger:
- none
pr: none
pool:
name: demo
demands:
- agent.name -equals demo
stages:
- stage: GitCloneMySQLHelmRepo
displayName: GitCloneMySQLHelmRepo
jobs:
- job: GitCloneMySQLHelmRepo
displayName: GitCloneMySQLHelmRepo
steps:
Settings
- 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:
New variable
Name
MYSQL_DATABASE
Value
Keep this value secret
Let users override this value when running this pipeline
To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example: $(MYSQL_DATABASE)
To use a variable in a script, use environment variable syntax. Replace . and space with _, capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:
Batch script: %MYSQL_DATABASE%
PowerShell script: ${env:MYSQL_DATABASE}
Bash script: $(MYSQL_DATABASE)
To use a secret variable in a script, you must explicitly map it as an environment variable.
Learn about variables
Cancel
OK



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




```
[demo@devopsagent-vm ~]$ kubectl get all -n mysql
NAME              READY   STATUS    RESTARTS   AGE
pod/mysql-0       1/1     Running   0          3m34s
pod/mysql-1       1/1     Running   0          2m15s
pod/mysql-2       1/1     Running   0          61s

NAME                       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/mysql              ClusterIP   10.0.66.27   <none>        3306/TCP   3m35s
service/mysql-headless     ClusterIP   None         <none>        3306/TCP   3m35s

NAME                       READY   AGE
statefulset.apps/mysql     3/3     3m35s
```

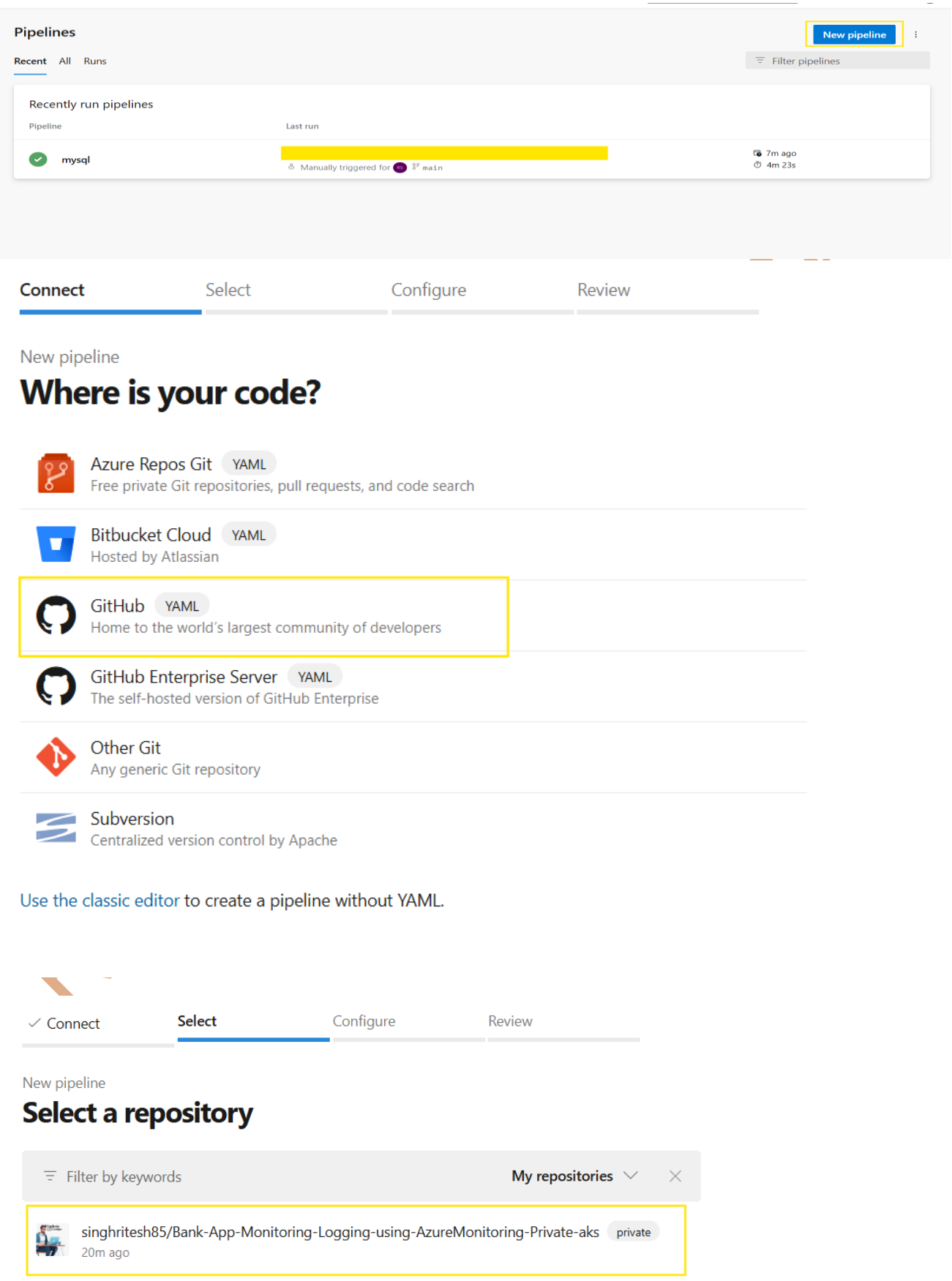
Before creating and running the Azure DevOps Pipeline I create kubernetes secretes for ACR (Azure Container Registry) as shown in the screenshot attached below.

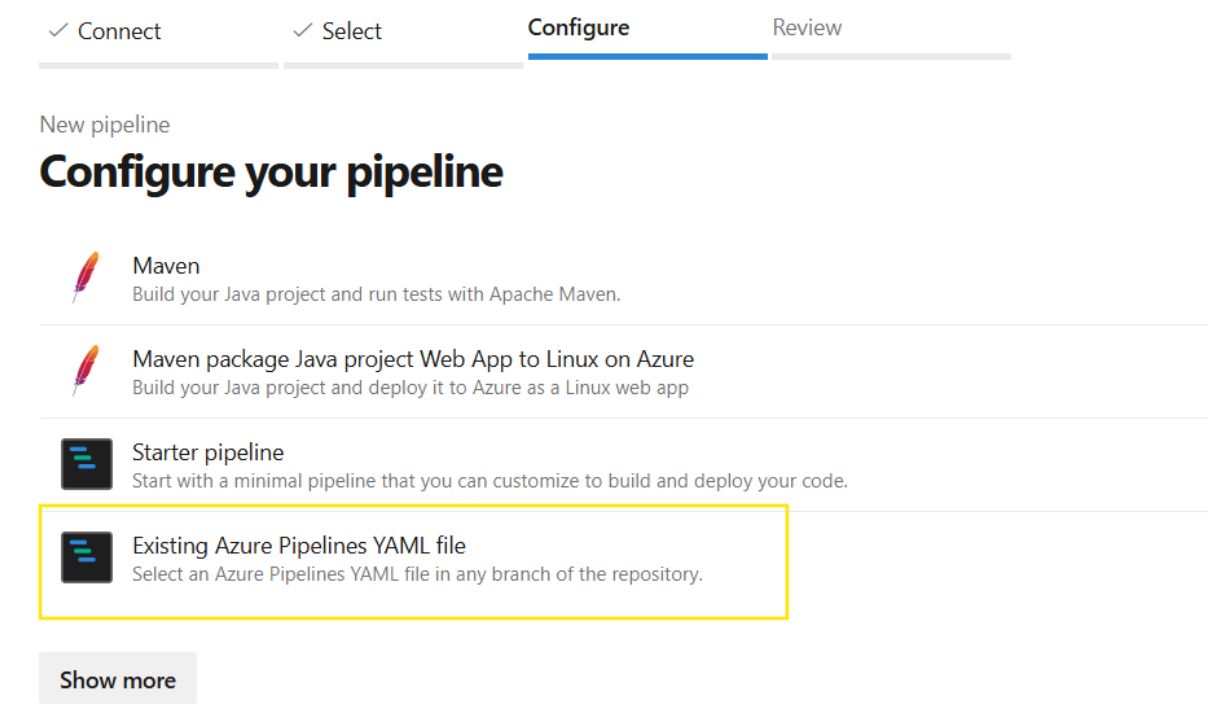
kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappcontainer24registry.azurecr.io --docker-username=

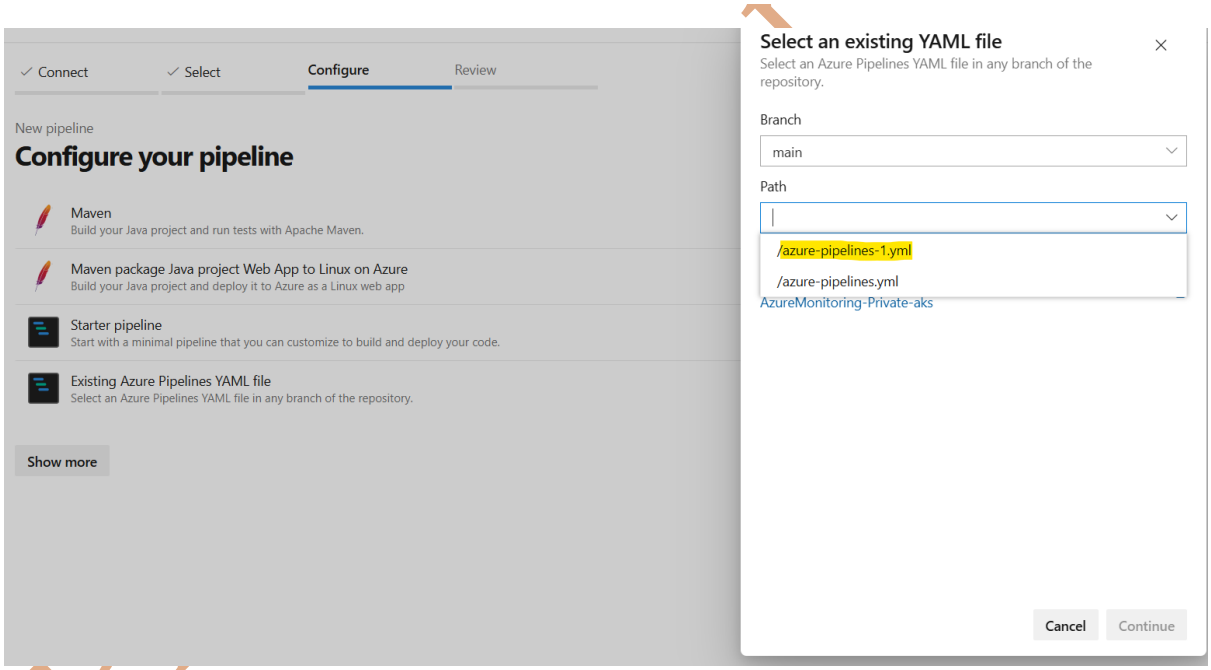
bankappcontainer24registry --docker-password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -n bankapp

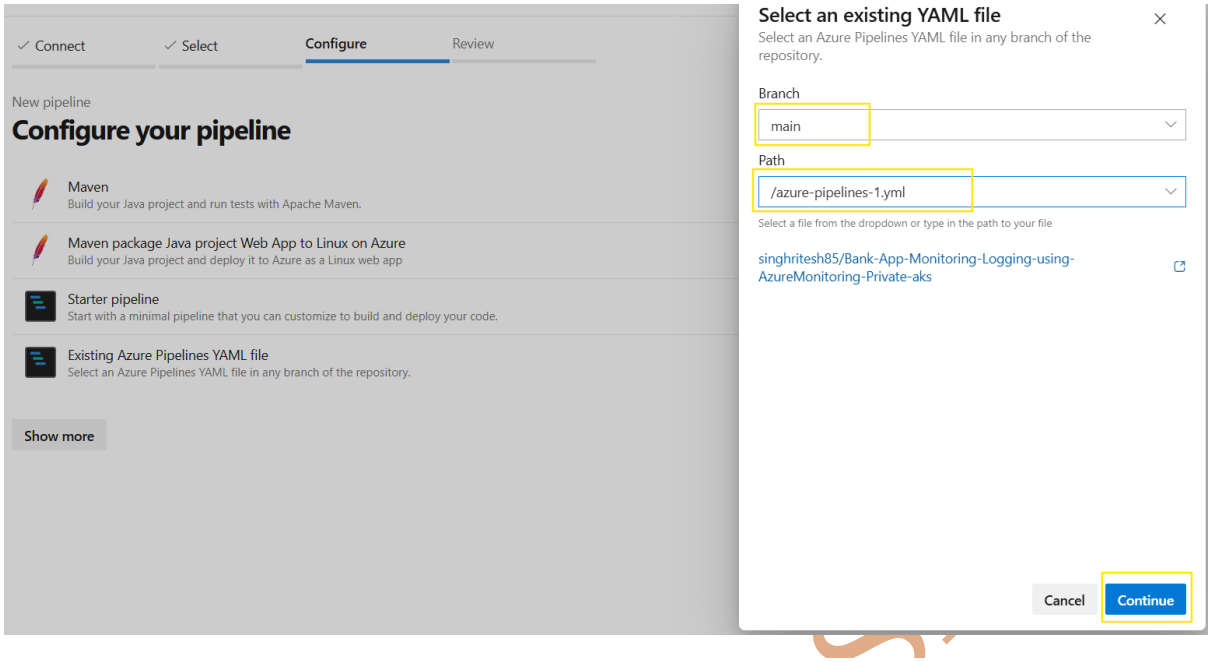
```
[root@                ~]# kubectl create ns bankapp
namespace/bankapp created
[root@                ~]# kubectl create secret docker-registry bankapp-auth --docker-server=https://bankappcontainer24registry.azurecr.io --docker-username=
bankappcontainer24registry --docker-password=               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

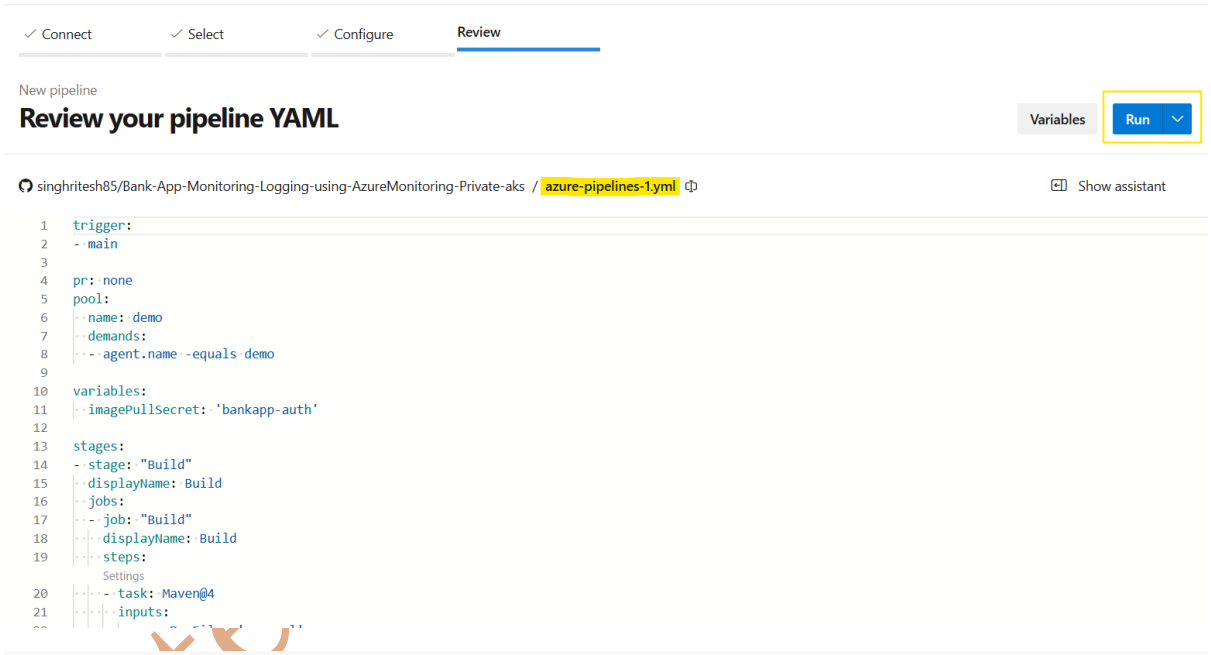
Then I created and ran the Azure DevOps Pipeline for BankApp as shown in the screenshot attached below.

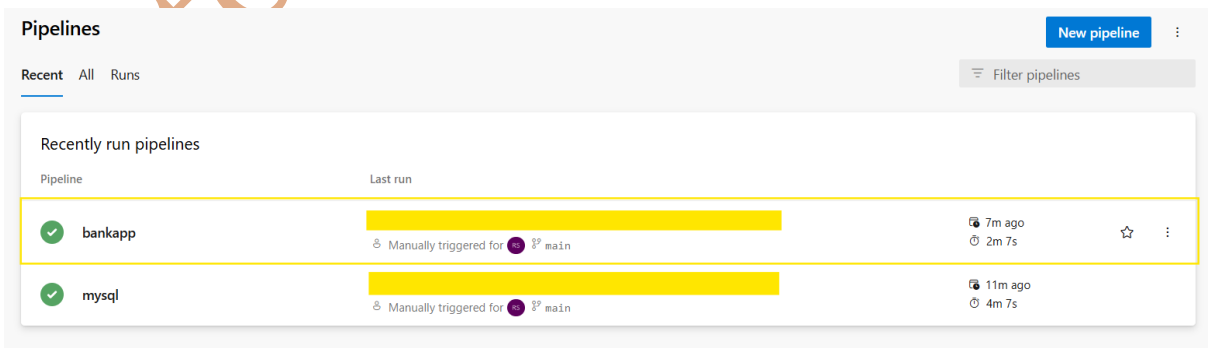



✓ Connect
✓ Select
Configure
Review
New pipeline
Configure your pipeline
Maven
Build your Java project and run tests with Apache Maven.
Maven package Java project Web App to Linux on Azure
Build your Java project and deploy it to Azure as a Linux web app
Starter pipeline
Start with a minimal pipeline 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


✓ 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


✓ 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
Select a file from the dropdown or type in the path to your file
singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks
Cancel
Continue


✓ Connect
✓ Select
✓ Configure
Review
New pipeline
Review your pipeline YAML
Variables
Run
singhritesh85/Bank-App-Monitoring-Logging-using-AzureMonitoring-Private-aks / azure-pipelines-1.yml
Show assistant
trigger:
- main
pr: none
pool:
name: demo
demands:
- agent.name -equals demo
variables:
imagePullSecret: 'bankapp-auth'
stages:
- stage: "Build"
displayName: Build
jobs:
- job: "Build"
displayName: Build
steps:
Settings
- task: Maven@4
inputs:


Pipelines
New pipeline
Recent
All
Runs
Filter pipelines
Recently run pipelines
Pipeline
Last run
bankapp
Manually triggered for
main
7m ago
2m 7s
mysql
Manually triggered for
main
11m ago
4m 7s

```
[demo@devopsagent-vm ~]$ kubectl get all -n bankapp
NAME                                READY   STATUS    RESTARTS   AGE
pod/bankapp-folo-                   1/1     Running   0          2m48s

NAME                    TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/bankapp-folo    ClusterIP   10.           <none>        80/TCP    2m48s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bankapp-folo    1/1     1            1           2m48s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/bankapp-folo-              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 secretes and Ingress Rule as shown in the screenshot attached below.

```
[root@                ~]# kubectl create secret tls ingress-tls --key mykey.key --cert STAR_singhritesh85_com.crt --namespace bankapp --context=bankapp-clust
er
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                  80, 443   7s
```

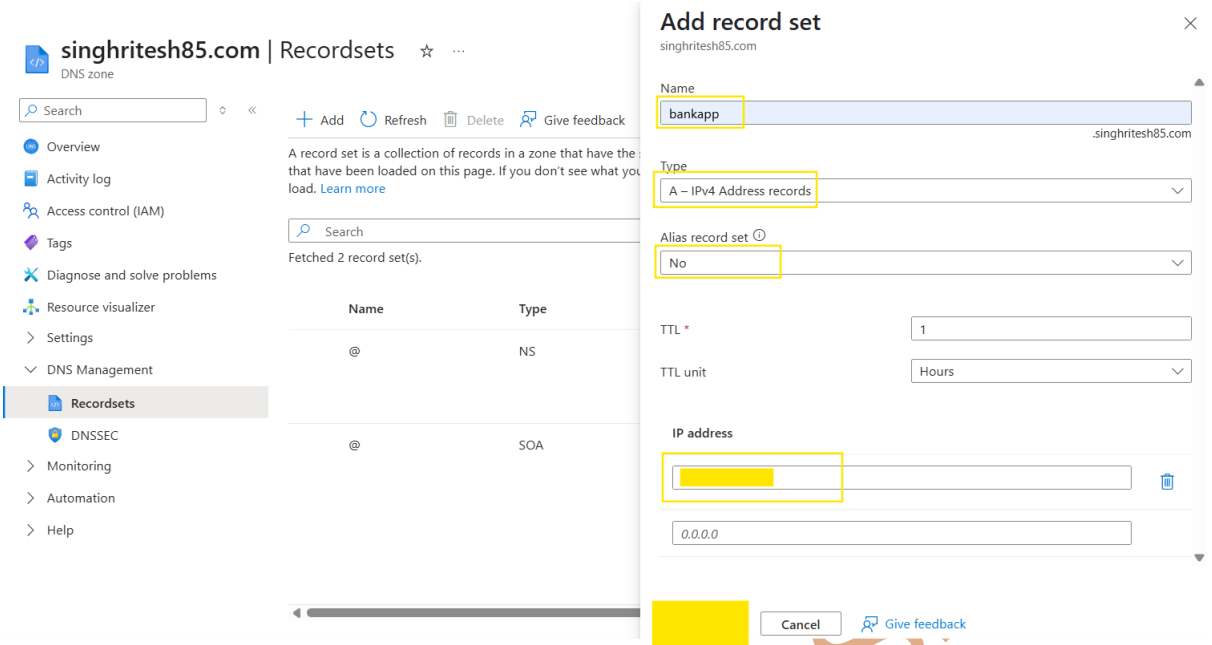
cat ingress-rule-azure.yaml

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

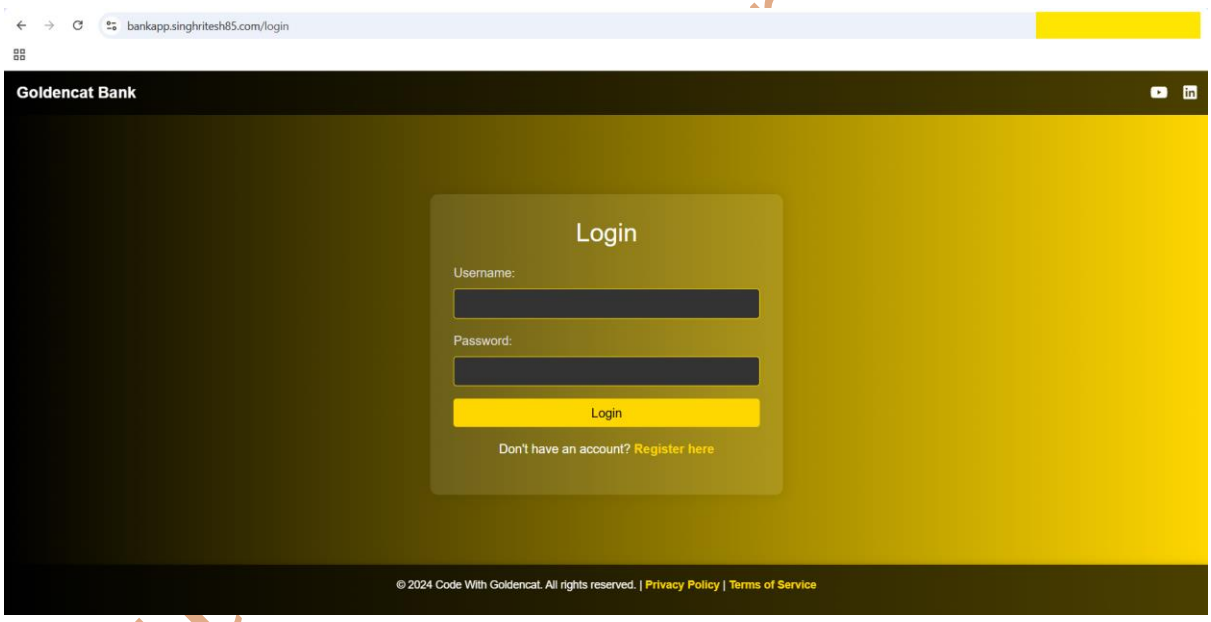
---

```
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
```

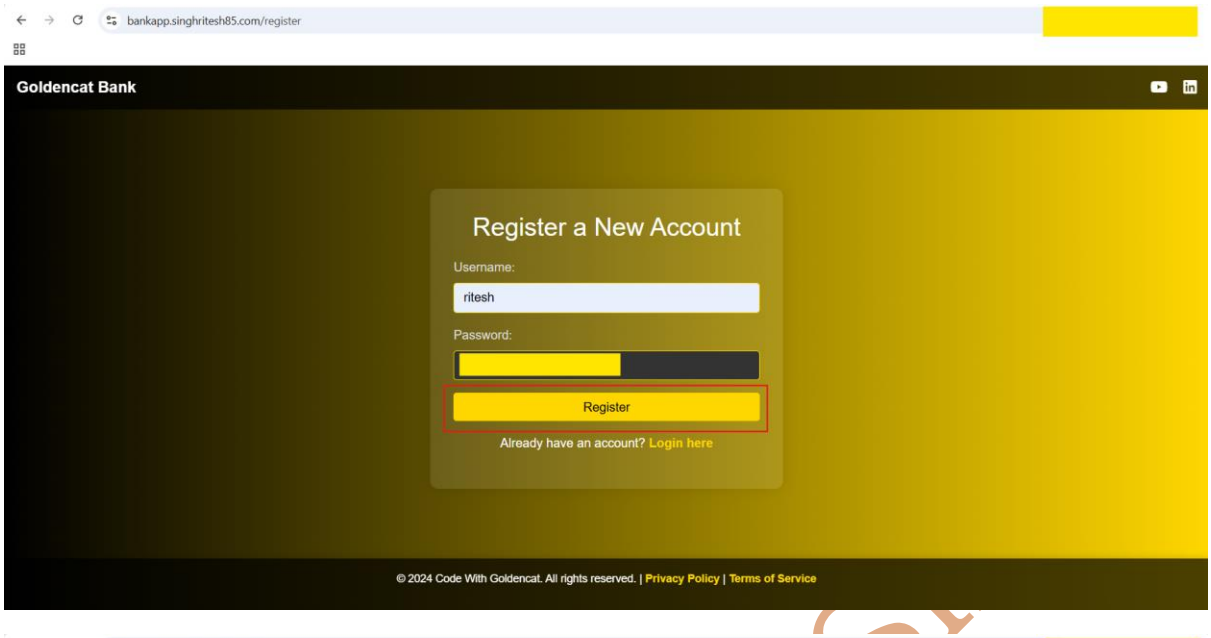
I create the URL to access the bank application by providing the HOST and Public IP Address in Azure DNS Zone to create the record set of A-Type as shown in the screenshot attached below.

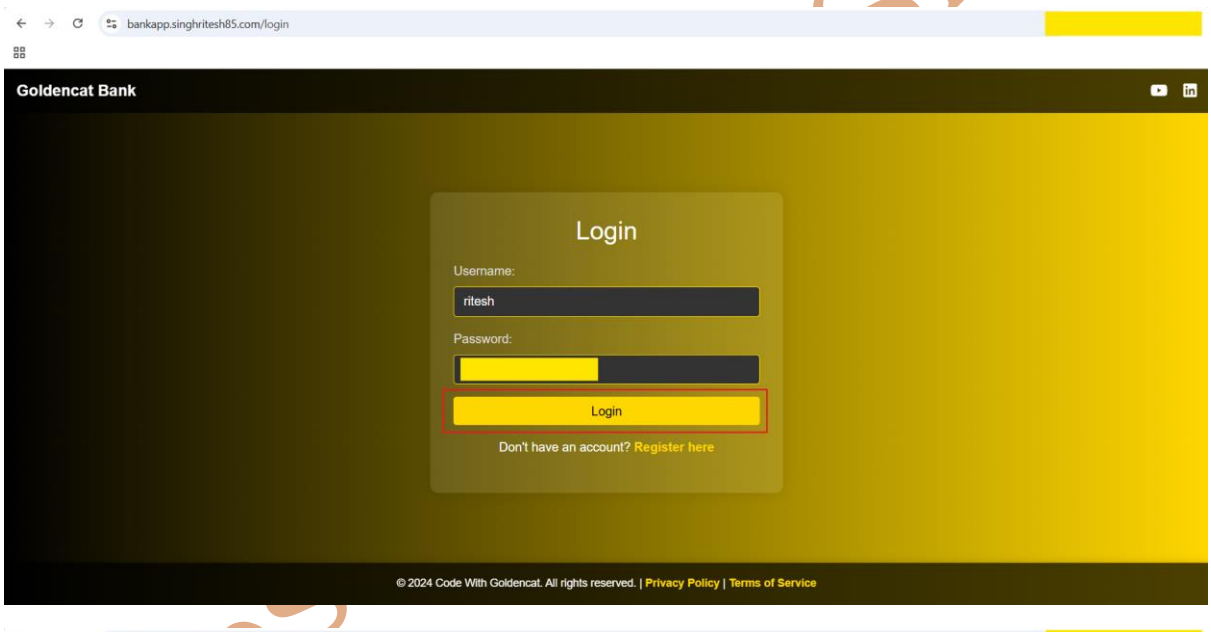


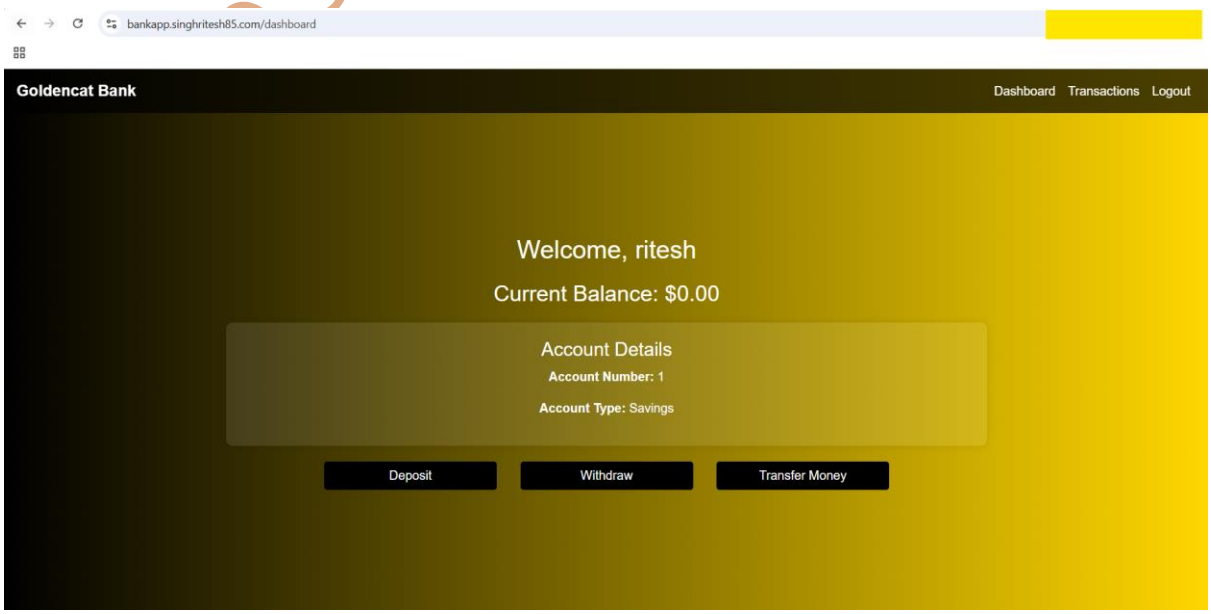
At last, I was able to access the application using the URL as shown in the screenshot attached below.



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.


bankapp.singhritesh85.com/register
Goldencat Bank
Register a New Account
Username:
ritesh
Password:
Register
Already have an account? Login here
© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service


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


bankapp.singhritesh85.com/dashboard
Goldencat Bank
Dashboard Transactions Logout
Welcome, ritesh
Current Balance: $0.00
Account Details
Account Number: 1
Account Type: Savings
Deposit
Withdraw
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 |                                                                | 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

singhritesh85 Add files via upload — 60ee1c9 · 31 minutes ago — History

Code | Blame | 215 lines (192 loc) · 13.2 KB — Raw

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <script type="text/javascript">
!(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="disableExceptionTracki
  src: "https://js.monitor.azure.com/scripts/b/ai.3.gbl.min.js",
  // name: "appInsights", // Global SDK Instance name defaults to "appInsights" when not supplied
  // 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,
  // useXhr: 1, // Use XHR instead of fetch to report failures (if available),
  // dle: true, // Prevent the SDK from reporting load failure log
  crossOrigin: "anonymous", // When supplied this will add the provided value as the cross origin attribute on the script tag
  // onInit: null, // Once the application insights instance has loaded and initialized this callback function will be called with 1 argument -- the sdk instance (
  // sri: false, // Custom optional value to specify whether fetching the snippet from integrity file and do integrity check
  cfg: { // Application Insights Configuration
    connectionString: "InstrumentationKey=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX;IngestionEndpoint=https://eastus2-3.in.applicationinsights.azure.com/;LiveEndpoint=h
}});
</script>
    <title>Dashboard - Goldencat Bank</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <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.

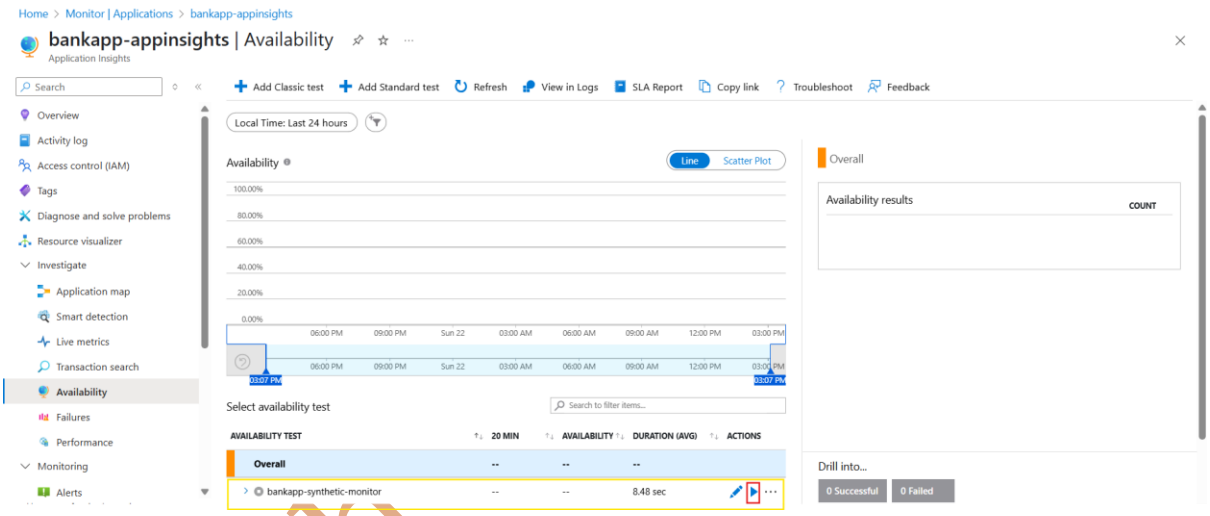
Bank-App-Monitoring-Logging-using-AzureMonitoring-aks / Dockerfile-Project-1

singhritesh85 Add files via upload 60ee1c9 · 34 minutes ago History

Code Blame 10 lines (8 loc) · 546 Bytes Raw

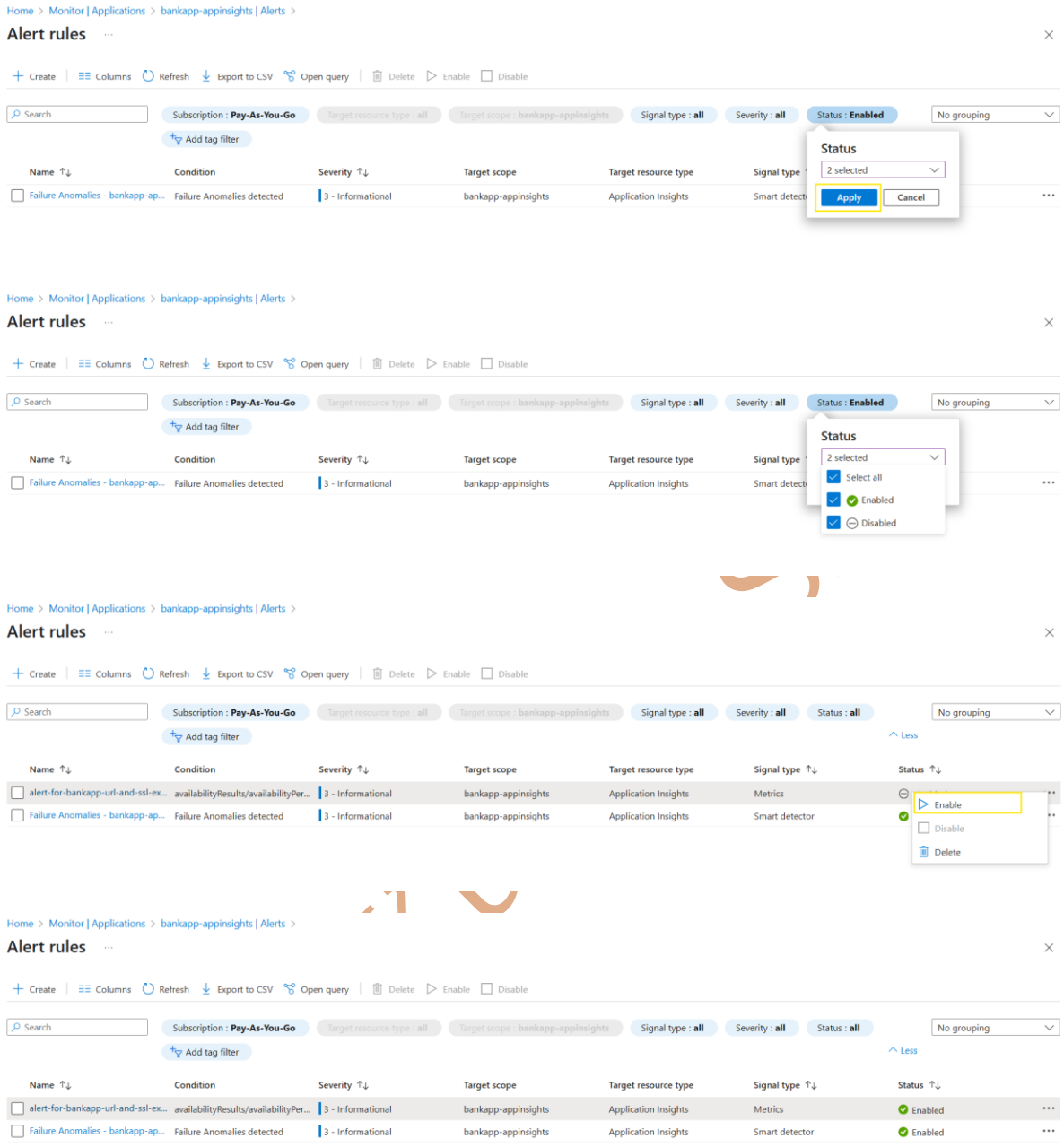
```
1   FROM eclipse-temurin:17-jdk-alpine
2   MAINTAINER "Dexter"
3
4   ENV APPLICATIONINSIGHTS_CONNECTION_STRING=InstrumentationKey=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX;IngestionEndpoint=https://eastus2-3.in.applicationinsights.azure.
5
6   WORKDIR /usr/src/app
7   COPY target/*.jar ./app.jar
8   COPY ./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.

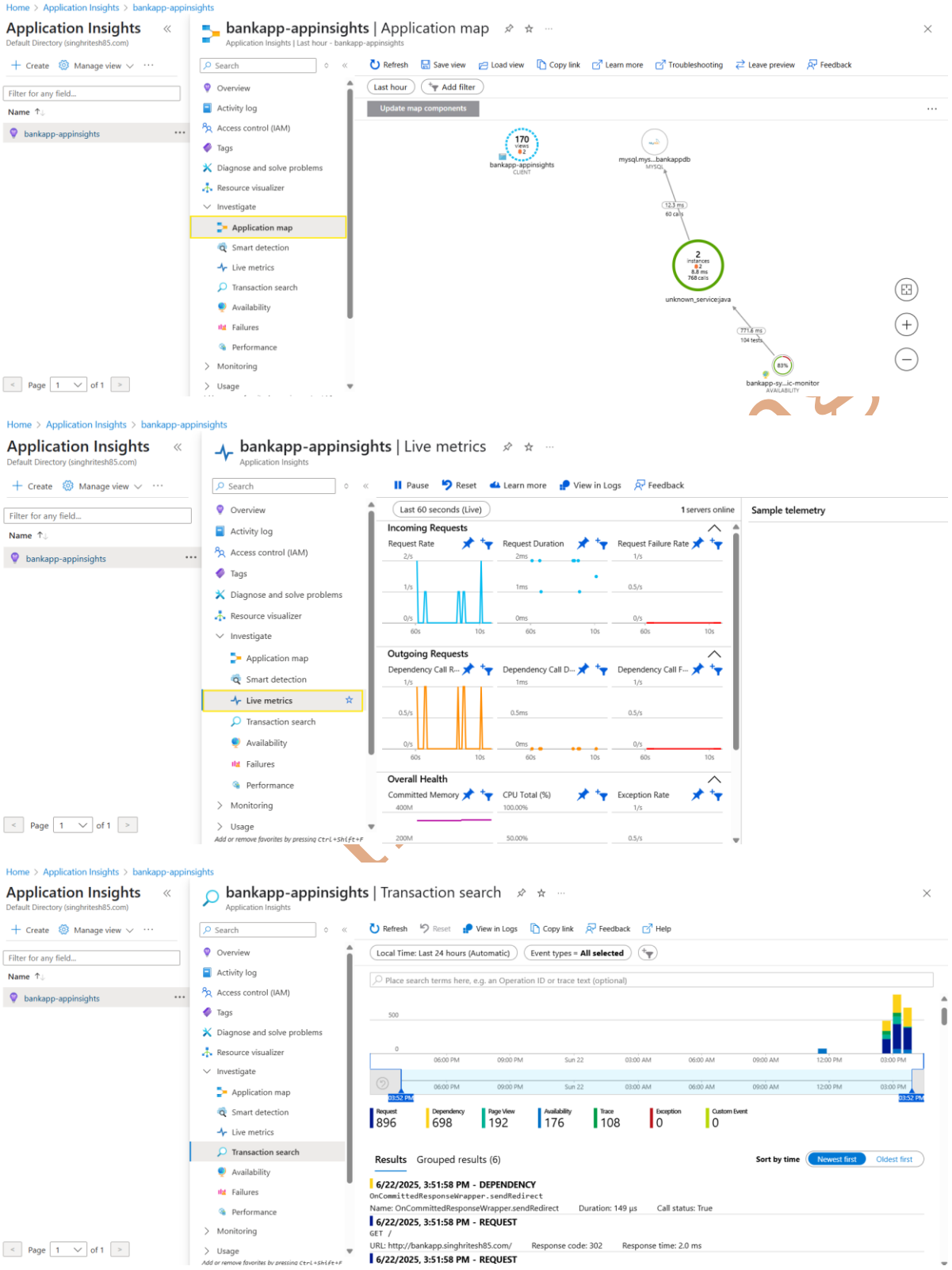


The Application is enabled now as shown in the screenshot attached below.

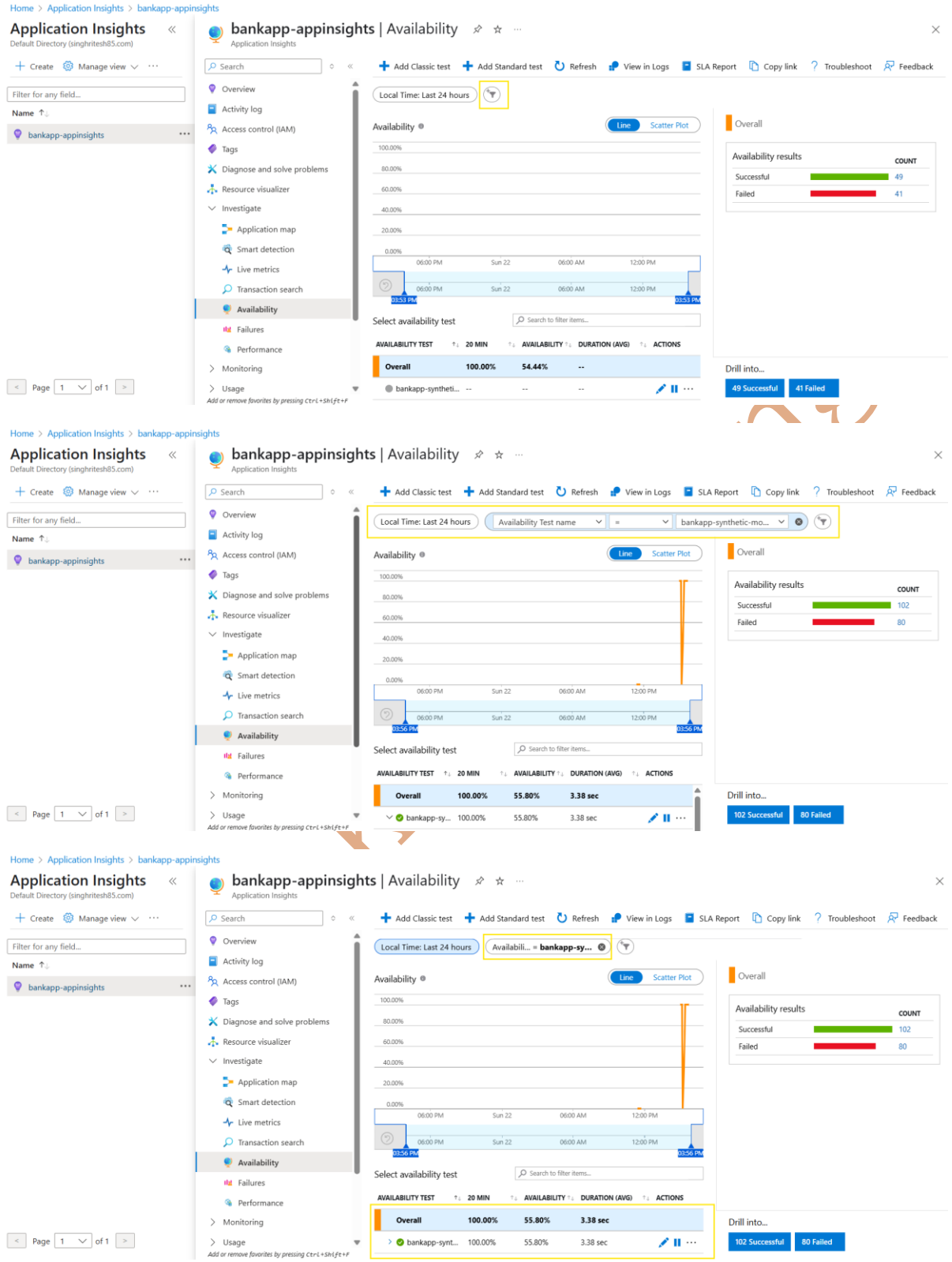
Then I enabled the Alert for Application Insight as shown in the screenshot attached below.



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



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

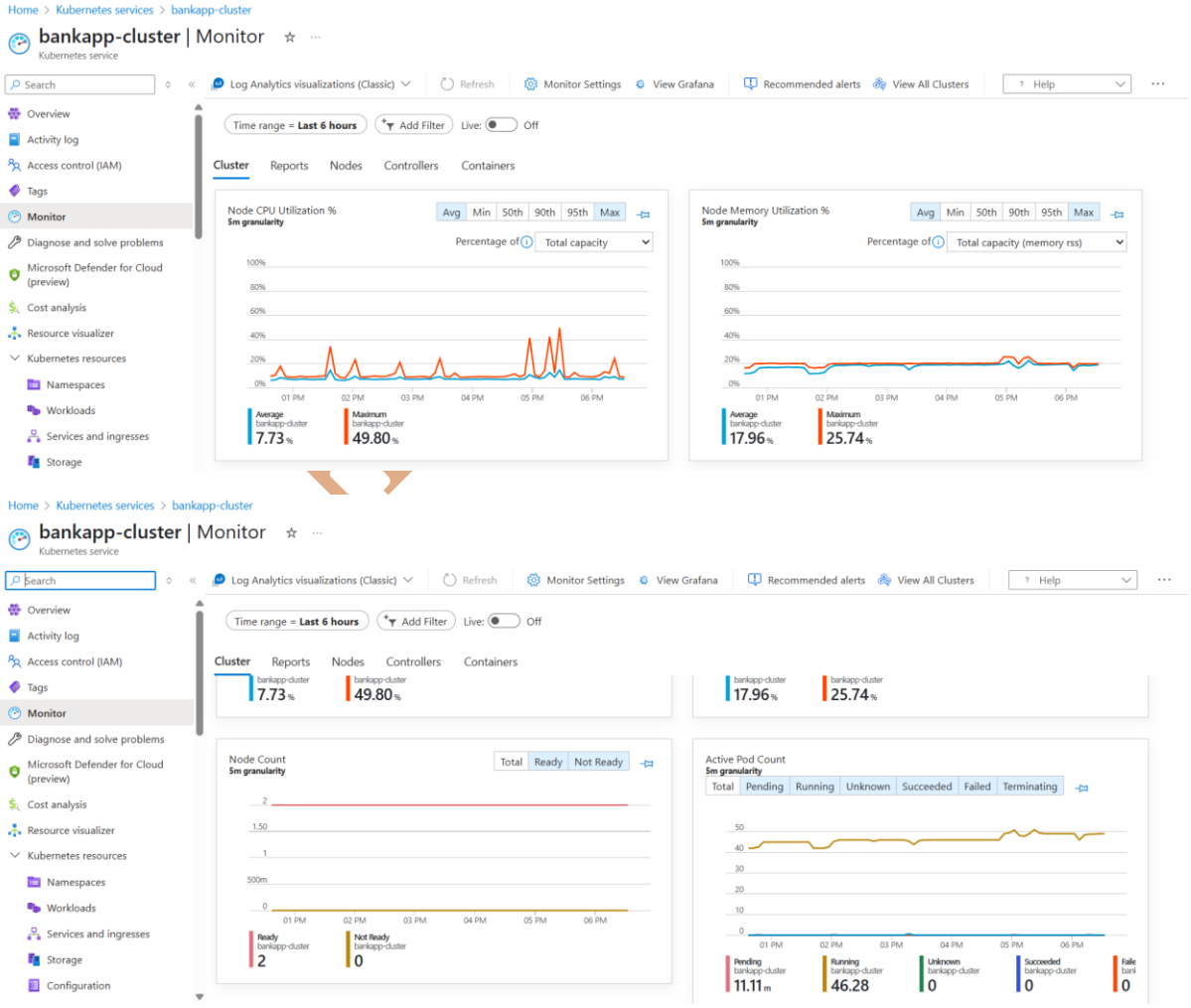


Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/ || GitHub: - https://github.com/singhritesh85

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/

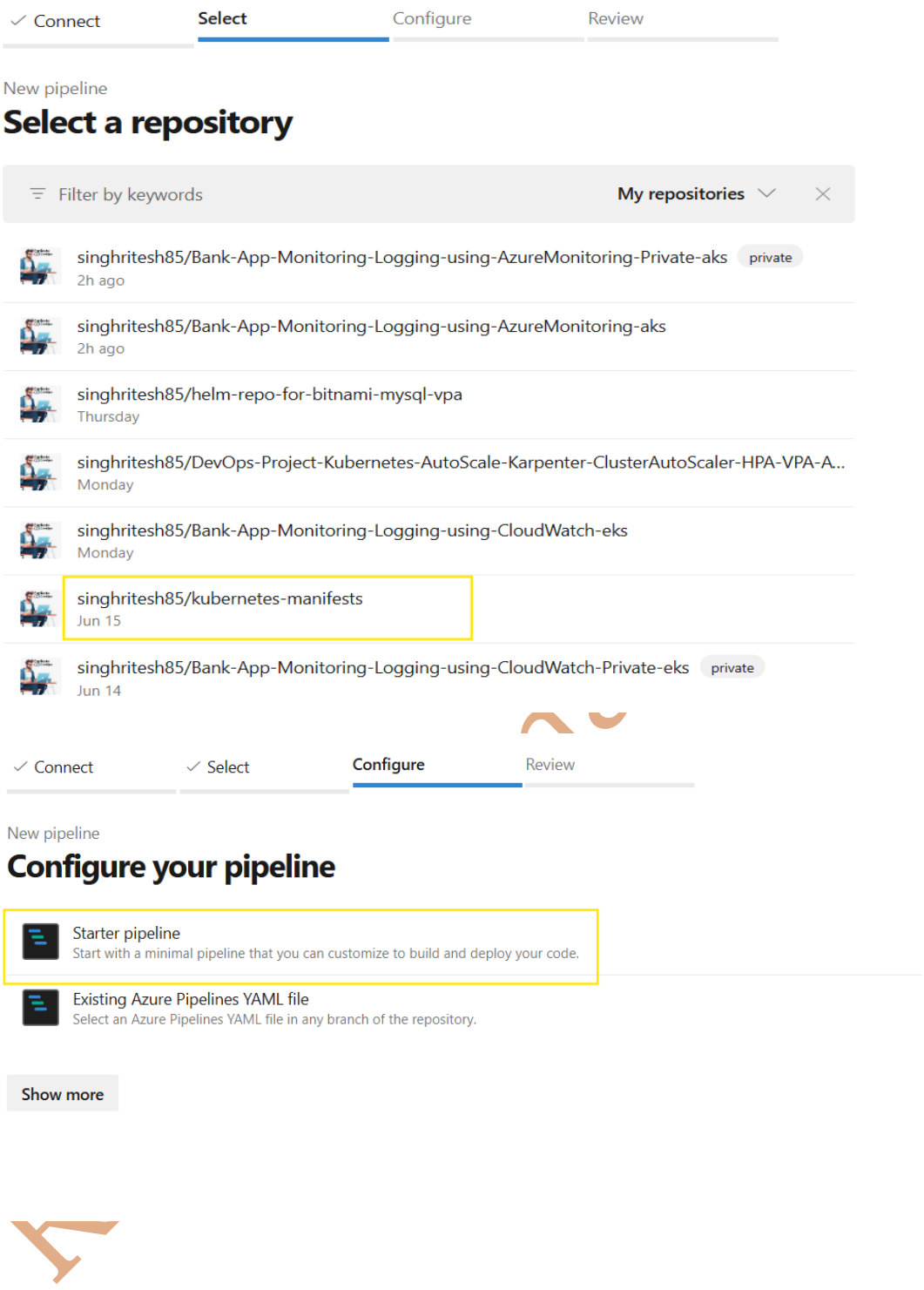
./vpa-up.sh

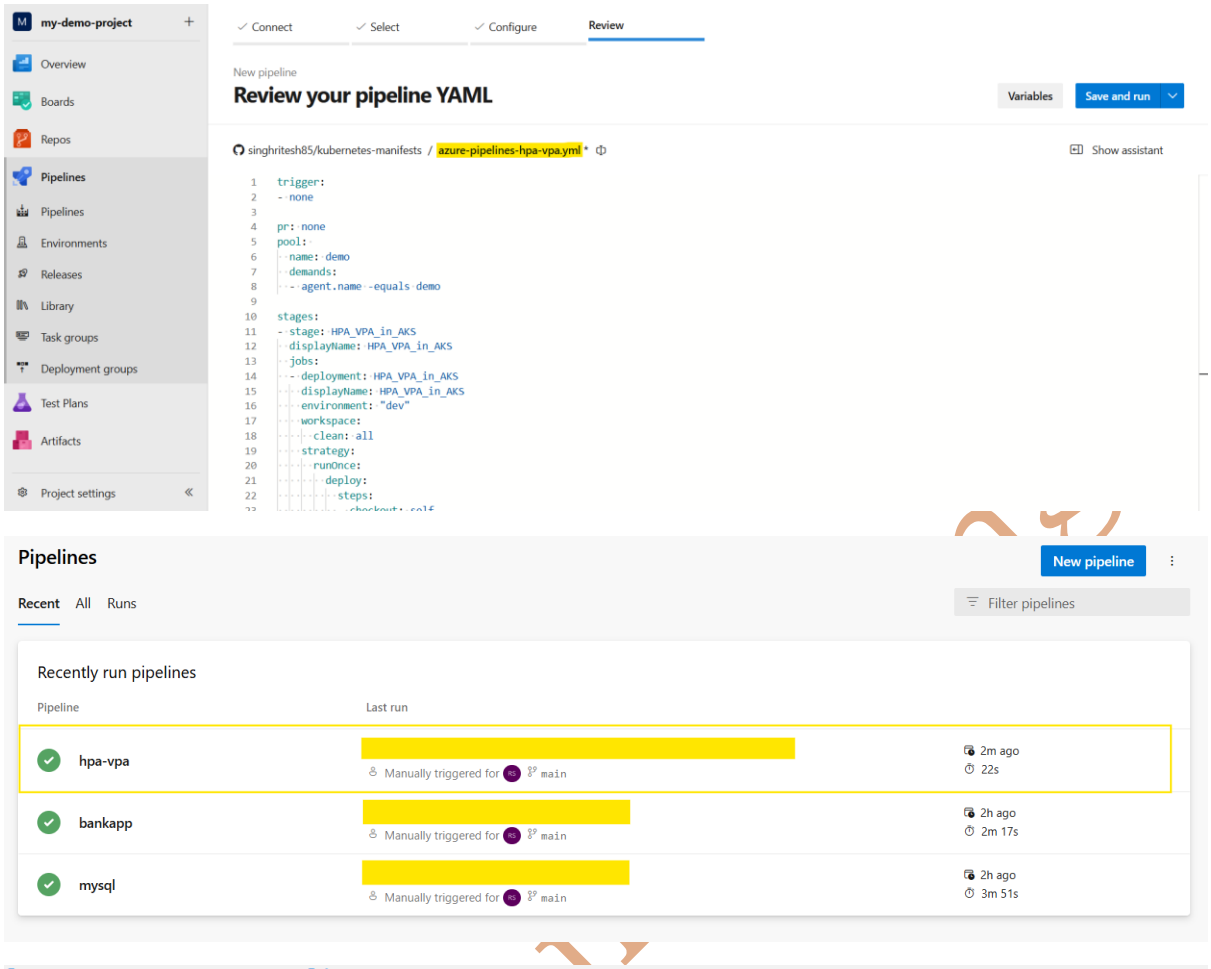
The requirement to run the above commands is OpenSSL must be 1.1.1 or later version.

```
[root@                ~]# 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@                ~]# cd autoscaler/
.git/                          LICENSE                        addon-resizer/                 code-of-conduct.md
.github/                       OWNERS                         balancer/                      hack/
.gitignore                     OWNERS_ALIASES                 builder/                       multidimensional-pod-autoscaler/
.pre-commit-config.yaml        README.md                      charts/                        vertical-pod-autoscaler/
CONTRIBUTING.md                SECURITY_CONTACTS              cluster-autoscaler/
[root@                ~]# cd autoscaler/vertical-pod-autoscaler/hack/
[root@                hack]# ./vpa-up.sh
```

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




```
[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-[redacted]             1/1     Running   0          2m20s
bankapp-folo-[redacted]             1/1     Running   0          94m
bankapp-folo-[redacted]             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://
    Image:           docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:        docker.io/bitnami/mysql@sha256:
    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
```

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
mysql-1   1/1     Running   0
mysql-2   1/1     Running   0
mysql-1   1/1     Running   0
mysql-1   1/1     Terminating   0
mysql-1   1/1     Terminating   0
mysql-1   0/1     Completed     0
mysql-1   0/1     Completed     0
mysql-1   0/1     Completed     0
mysql-1   0/1     Pending       0
mysql-1   0/1     Pending       0
mysql-1   0/1     Init:0/1      0
mysql-1   0/1     PodInitializing   0
mysql-1   0/1     Running           0
mysql-1   0/1     Running           0
mysql-1   1/1     Running           0
mysql-2   1/1     Running           0
mysql-2   1/1     Terminating       0
mysql-2   1/1     Terminating       0
mysql-2   0/1     Completed         0
mysql-2   0/1     Completed         0
mysql-2   0/1     Completed         0
mysql-2   0/1     Pending           0
mysql-2   0/1     Pending           0
mysql-2   0/1     Init:0/1          0
mysql-2   0/1     PodInitializing   0
mysql-2   0/1     Running           0
mysql-2   0/1     Running           0
mysql-2   1/1     Running           0
```

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

```
[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://
    Image:           docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:        docker.io/bitnami/mysql@sha256:
    Port:            3306/TCP
    Host Port:       0/TCP
    SeccompProfile:  RuntimeDefault
    State:           Running
      Started:       2025
    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@ ~]# kubectl logs -f vpa-admission-controller- -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/requests/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 updat
ed 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.VerticalPodA
utoscaler" 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/requests/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 updat
ed 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" totalIt
ems=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.ReplicationC
ontroller" 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/requests/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 updat
ed 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://
    Image:           docker.io/bitnami/mysql:8.4.0-debian-12-r3
    Image ID:        docker.io/bitnami/mysql@sha256:
    Port:            3306/TCP
    Host Port:       0/TCP
    SeccompProfile:  RuntimeDefault
    State:           Running
      Started:       2025
    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@ ~]# 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 infornt of then

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: @gmail.com" "P: "
```

```
[root@ ~]# cd /etc/mail/auth/
[root@ auth]# makemap hash gmail-auth < gmail-auth
[root@ 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 infornt of then
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
FEATURE(`authinfo',`hash -o /etc/mail/auth/gmail-auth.db')dnl
MAILER(smtp)dnl
MAILER(local)dnl
```

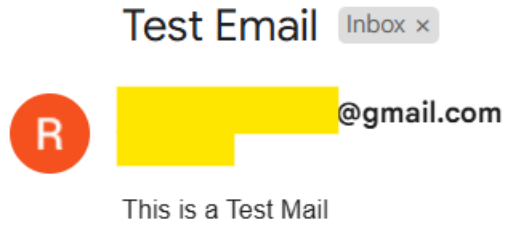
Commented Line

```
[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  2025  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 @gmail.com -s "Test Email" @gmail.com
```




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

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

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

URL="https://bankapp.singhritesh85.com/login"  ###Provide the URL to be monitored
EMAIL="               @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@                ~]# 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 $DAYS_LEFT days." | mail -r               @gmail.com -s "30 days left for SSL to be expired"               @gma
il.com
else
  if [ "$DAYS_LEFT" -eq 15 ]; then
    echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r               @gmail.com -s "15 days left for SSL to be expired"               @g
mail.com
    exit 1
  fi
  if [ "$DAYS_LEFT" -eq 7 ]; then
    echo "Caution. SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -r               @gmail.com -s "7 days left for SSL to be expired"
      @gmail.com
  else
    echo "SSL certificate for $HOST is valid for $DAYS_LEFT days." | mail -r singhritesh8515@gmail.com -s "SSL Expiration Notification"               @gmail.
com
  fi
fi
```

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

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

```
[root@                    ~]# crontab -e
crontab: installing new crontab
[root@                    ~]# 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 xyx@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@                ~]# 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@              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              @gmail.com -s "Log Rotated"              @gmail.com
    endscript
}
```

```
[root@              logrotate.d]# mkdir /var/log/cronlog
```

```
[root@              logrotate.d]# crontab -e
crontab: installing new crontab
[root@              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