DevOps Project BankApp Docker Based Deployment using Jenkins Monitoring and LogAggregartion using CloudWatch

By Ritesh Kumar Singh

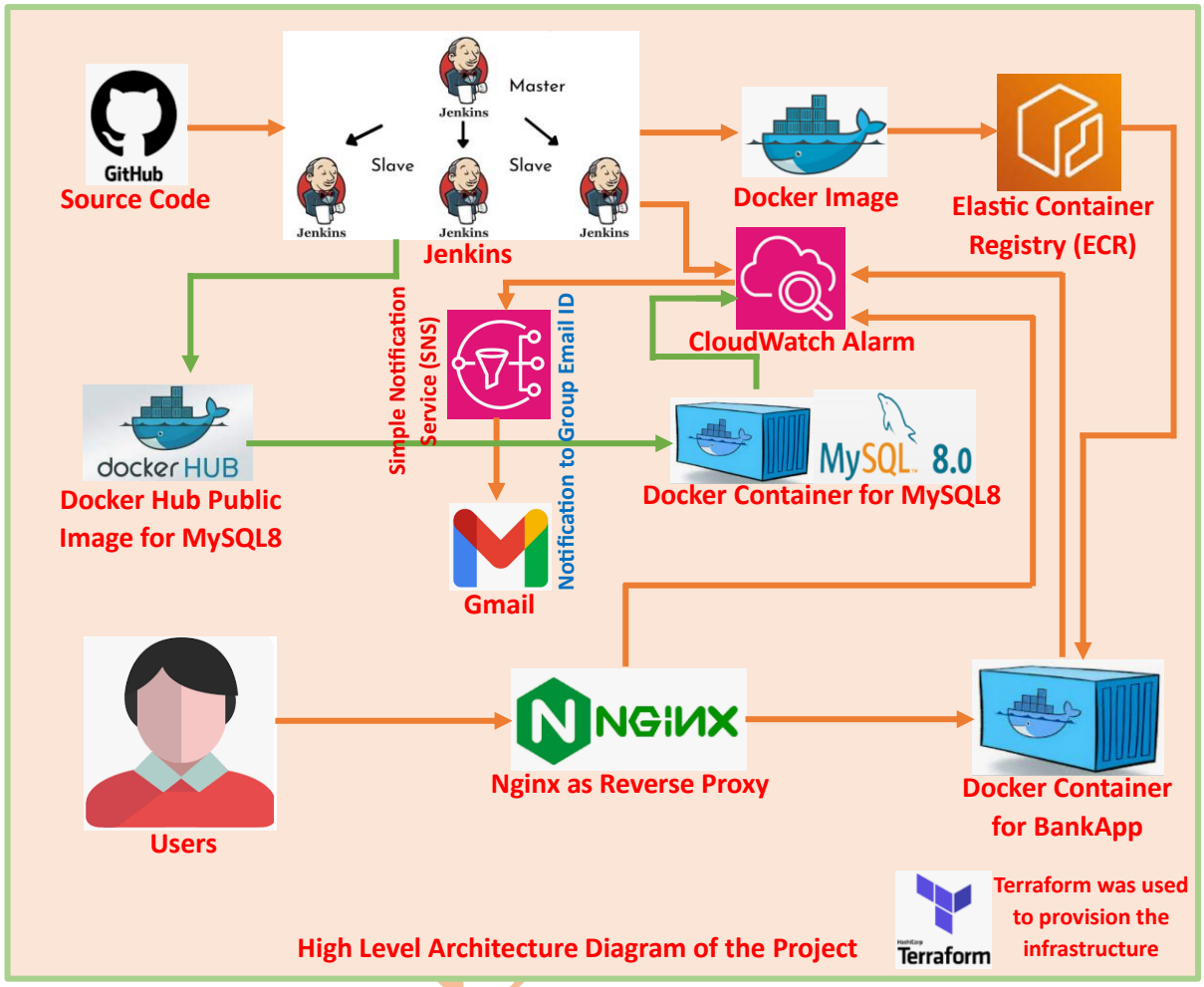
Email Address: - riteshkumarsingh9559@gmail.com

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 ॥

Module-1: Approach with Nginx as Reverse Proxy



The high-level architecture diagram of the project is as shown above. In this project I use **docker-compose** to run multiple containers (bankapp and mysql containers). Using docker-compose we can run multiple containers in one time. The docker-compose.yaml file and source code was present in the GitHub Repo https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch.git. I created the infrastructure using Terraform script which is present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-Docker-Monitoring-LogAggregation-using-CloudWatch.git at the path **terraform-jenkins-docker-nginx**. I am not going to discuss the SonarQube, Sonatype Nexus Artifactory and Trivy Scan here if you are interested then please refer to the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApplication-Multibranch-MultiCloud.git. For CI/CD I had used Jenkins (with Master slave Architecture) and CloudWatch was used for Monitoring and Log Aggregation. Finally, I used Nginx as reverse proxy and generated a URL for the Bank Application to access it. At last, I did the entry of required URL with Nginx Server Public IP Address in Route53 to create the Record Set of A-Type.

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.

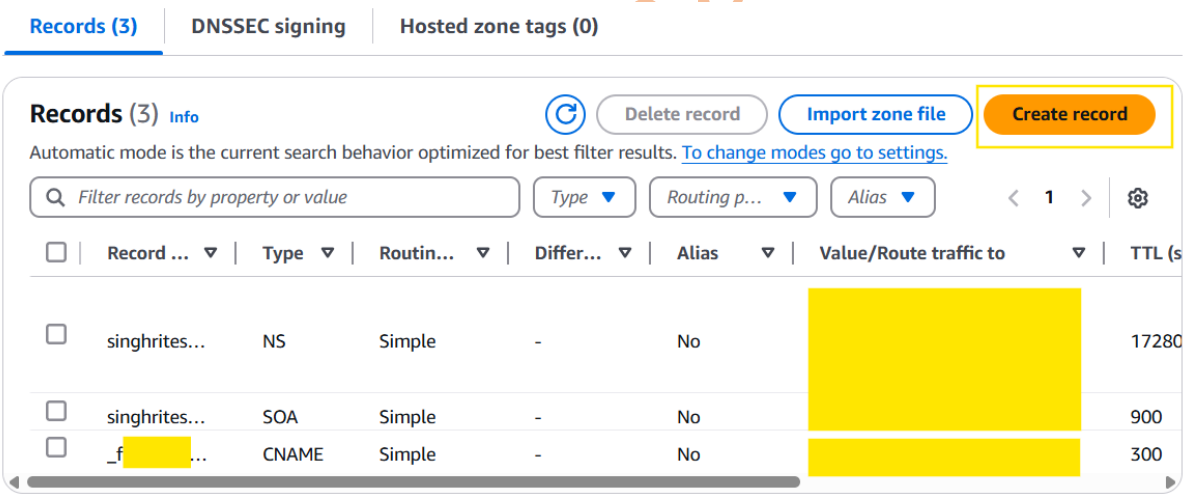
terraform validate --------> verify that terraform configuration file is correct or not

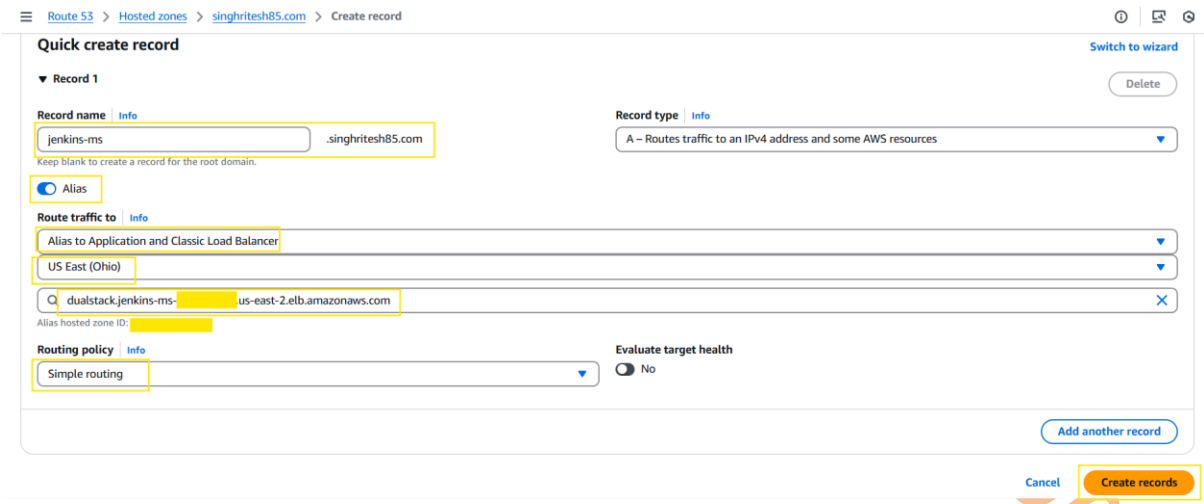
terraform plan ---------> Check which resources are going to be created.

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

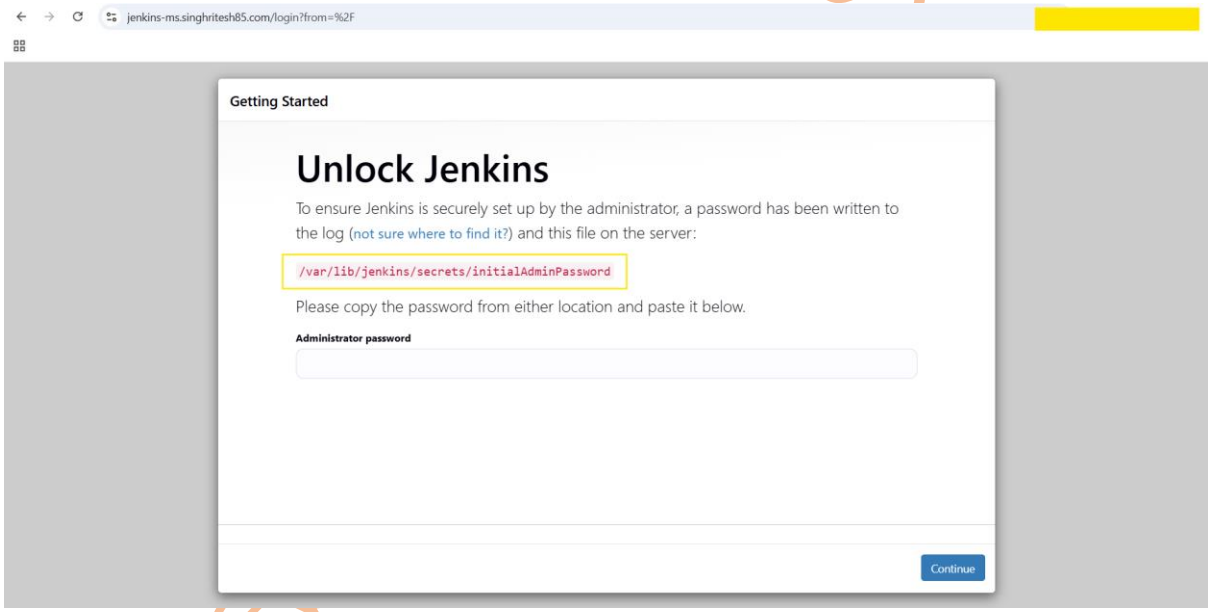
```
module.bankapp.null_resource.nginx_server (remote-exec): amazon-cloudwatch-agent has already been stopped
module.bankapp.null_resource.nginx_server (remote-exec): Created symlink from /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service to /
etc/systemd/system/amazon-cloudwatch-agent.service.
module.bankapp.null_resource.nginx_server: Creation complete after 3m2s [id=[redacted]]
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_cpu_alarm: Creating...
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_memory_alarm: Creating...
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_disk_usage_alarm: Creating...
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_disk_usage_alarm: Creation complete after 0s [id=Nginx-Server-DiskUsageAlarm]
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_cpu_alarm: Creation complete after 0s [id=Nginx-Server-CPUUtilizationAlarm]
module.bankapp.aws_cloudwatch_metric_alarm.nginx_server_memory_alarm: Creation complete after 0s [id=Nginx_Server_Memory_High_Utilization_Alarm]
module.bankapp.aws_rum_app_monitor.bankapp_rum: Creating...
module.bankapp.aws_rum_app_monitor.bankapp_rum: Creation complete after 1s [id=bankapp-rum]
module.bankapp.aws_rum_metrics_destination.rum_destination: Creating...
module.bankapp.aws_iam_policy.bankapp_guest_rum_policy: Creating...
module.bankapp.aws_iam_policy.bankapp_guest_rum_policy: Creation complete after 0s [id=arn:aws:iam::027330342406:policy/bankapp-rum-guest-policy]
module.bankapp.aws_iam_role_policy_attachment.bankapp_guest_rum_policy_attachment: Creating...
module.bankapp.aws_rum_metrics_destination.rum_destination: Creation complete after 0s [id=bankapp-rum]
module.bankapp.aws_iam_role_policy_attachment.bankapp_guest_rum_policy_attachment: Creation complete after 0s [id=bankapp-guest-rum-role-[redacted]
[redacted]]

Apply complete! Resources: 88 added, 0 changed, 0 destroyed.

Outputs:

ecr_ec2_private_ip_alb_dns = {
  "EC2_Instance_Docker_Server_Private_IP_Address" = "10.[redacted]"
  "EC2_Instance_Jenkins_Master_Server_Private_IP_Address" = "10.[redacted]"
  "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10.[redacted]"
  "EC2_Instance_Nginx_Server_Private_IP_Address" = "10.[redacted]"
  "Jenkins_ALB_DNS_Name" = "jenkins-ms-[redacted].us-east-2.elb.amazonaws.com"
  "registry_id" = [
    "02[redacted]6",
  ]
  "repository_url" = [
    "02[redacted]6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev",
  ]
}
```

After creation of Resources in AWS Cloud I created the URL to access Jenkins by creating a Record Set of A-Type with Alias in Route53 as shown in the screenshot attached below.



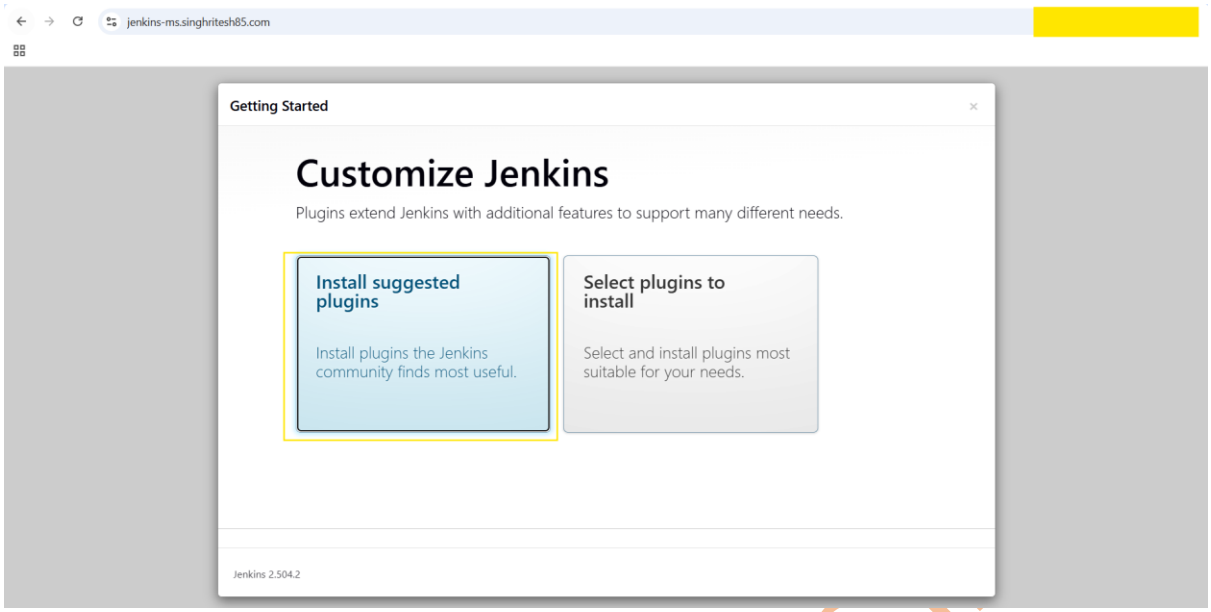


Then I configured Jenkins and Slave for Jenkins as shown in the screenshot attached below. For the first time we you will configure Jenkins then it requires initialAdminPassword which was present at the path **/var/lib/jenkins/secrets/initialAdminPassword**.

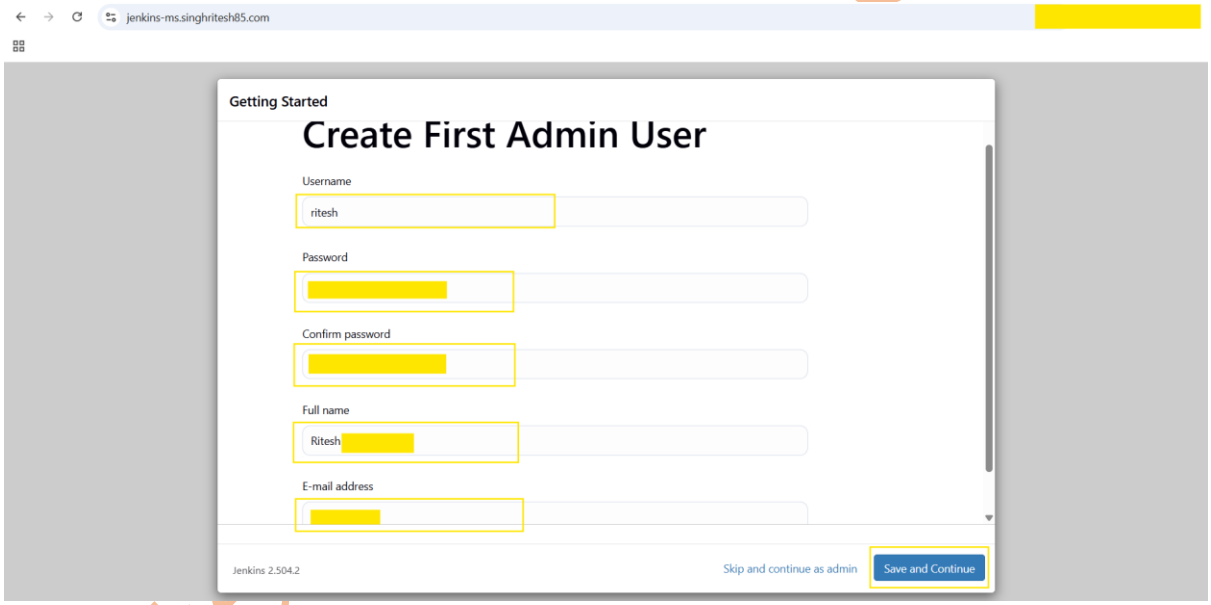


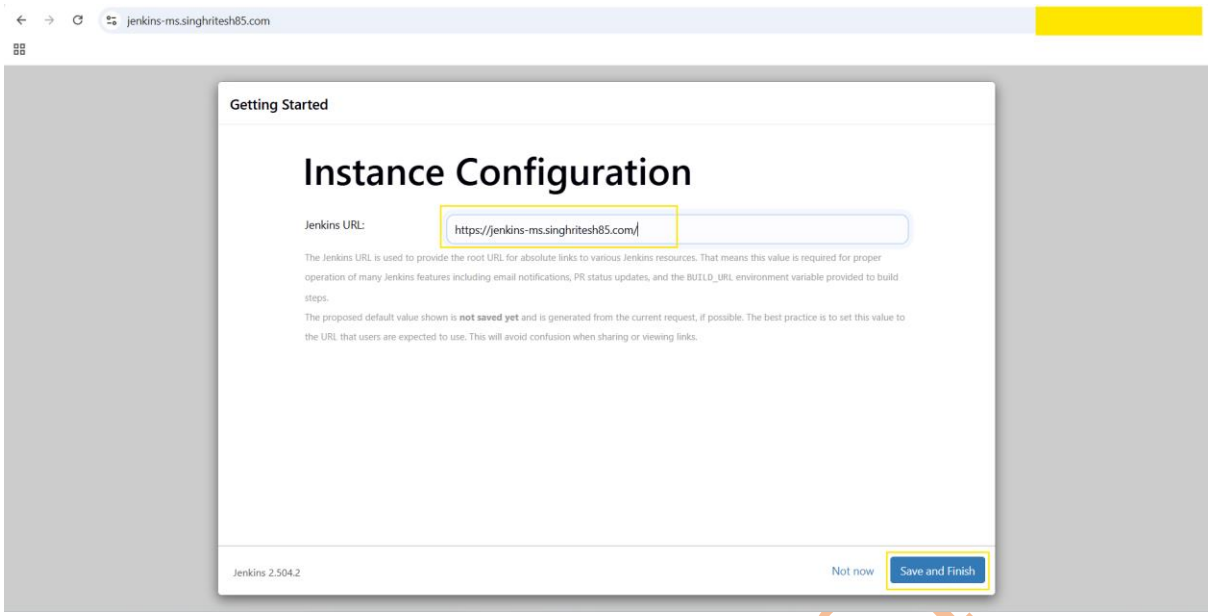
I logged into the Jenkins Master Server and got the initialAdminPassword as shown in the screenshot attached below.

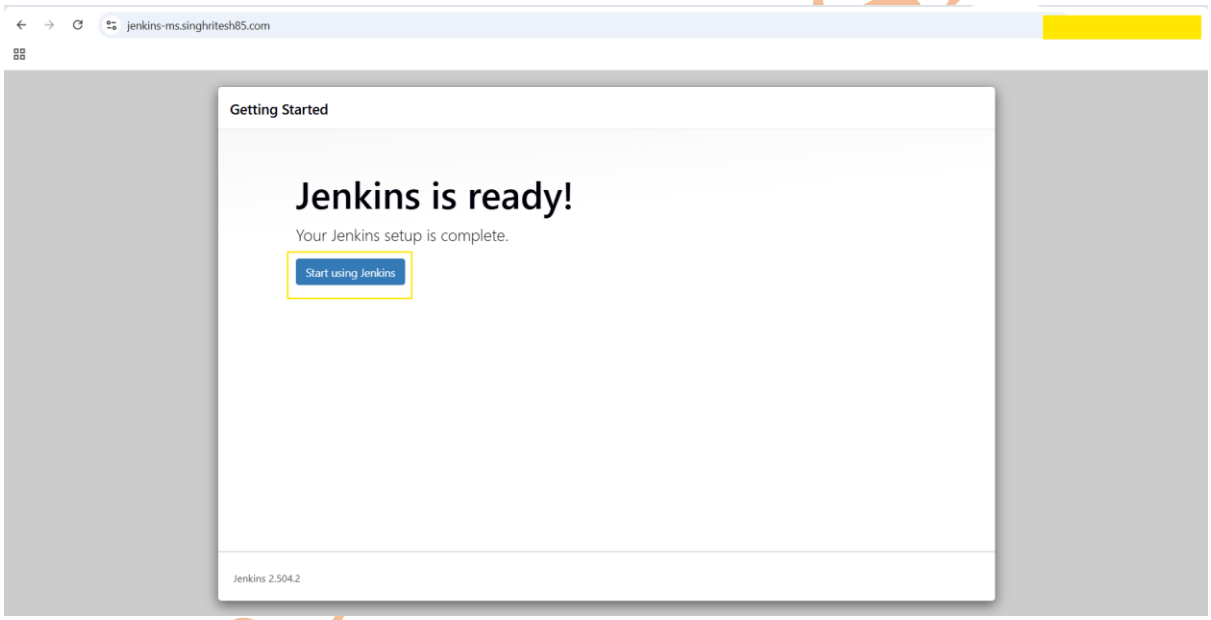
```
[root@                    ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
1                                                    2
```



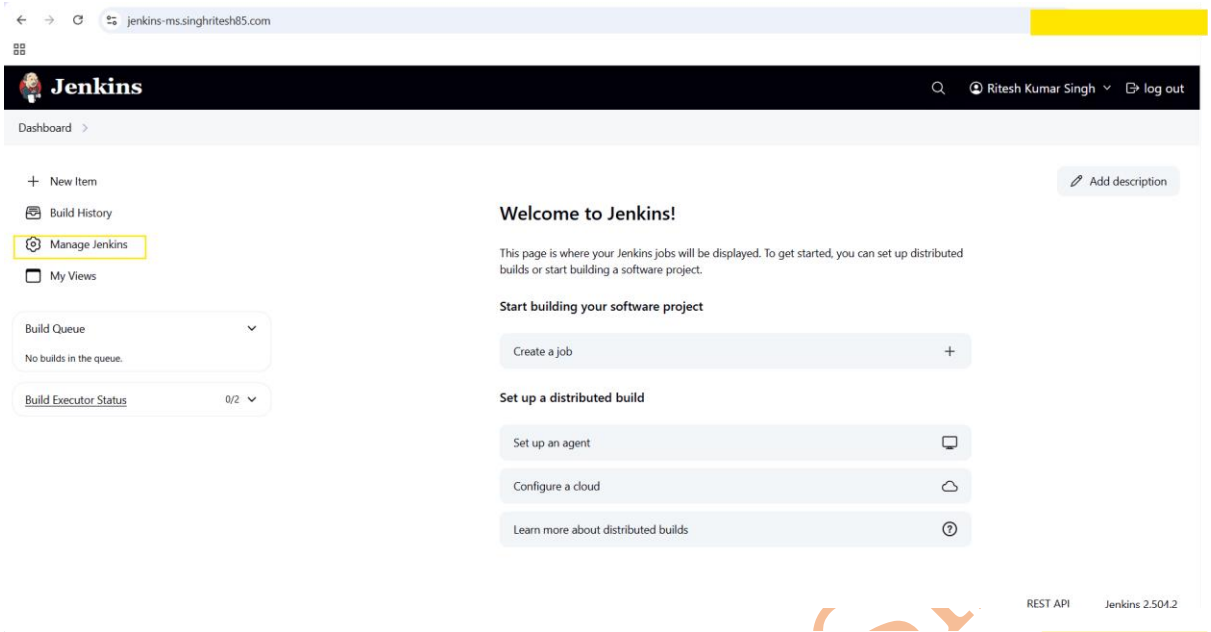
Created the first Admin user for Jenkins as shown in the screenshot attached below.

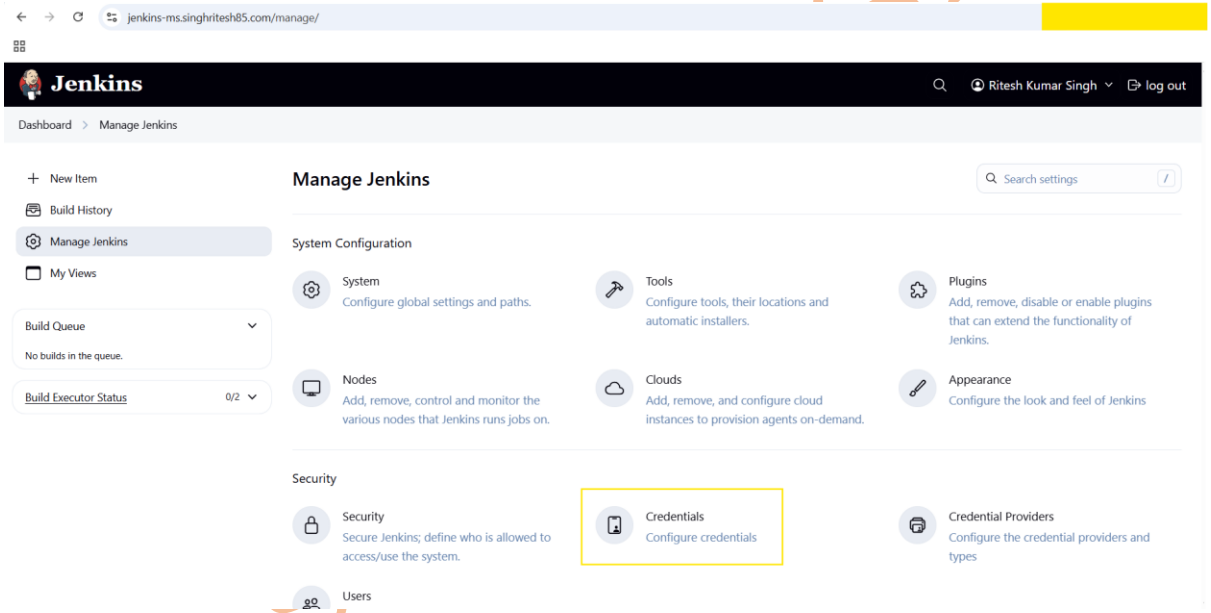


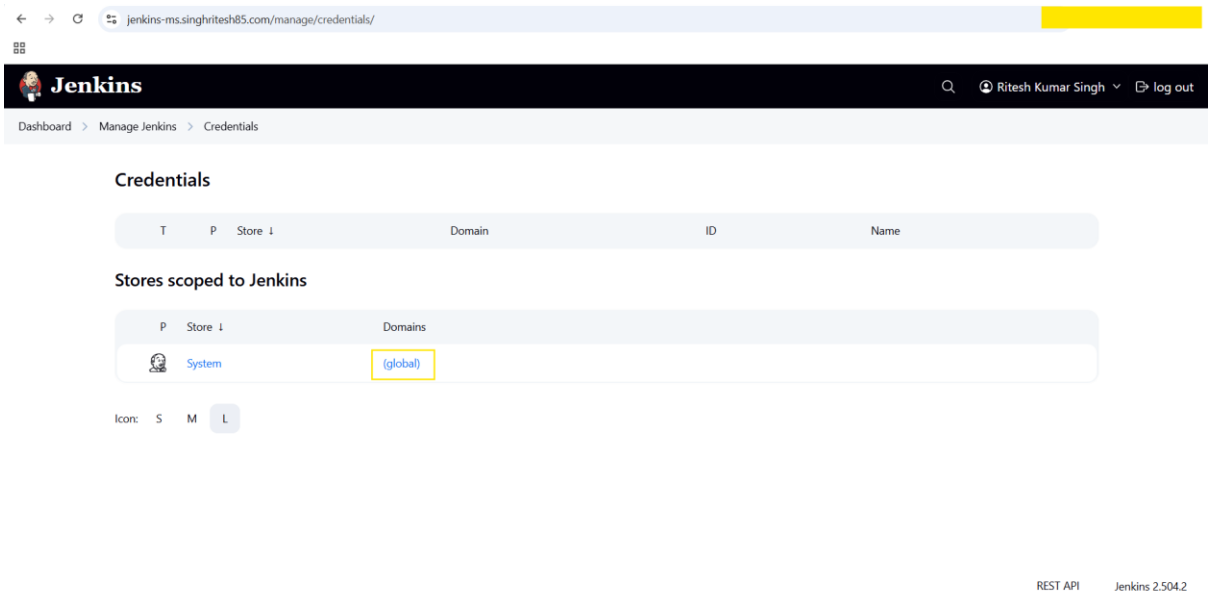


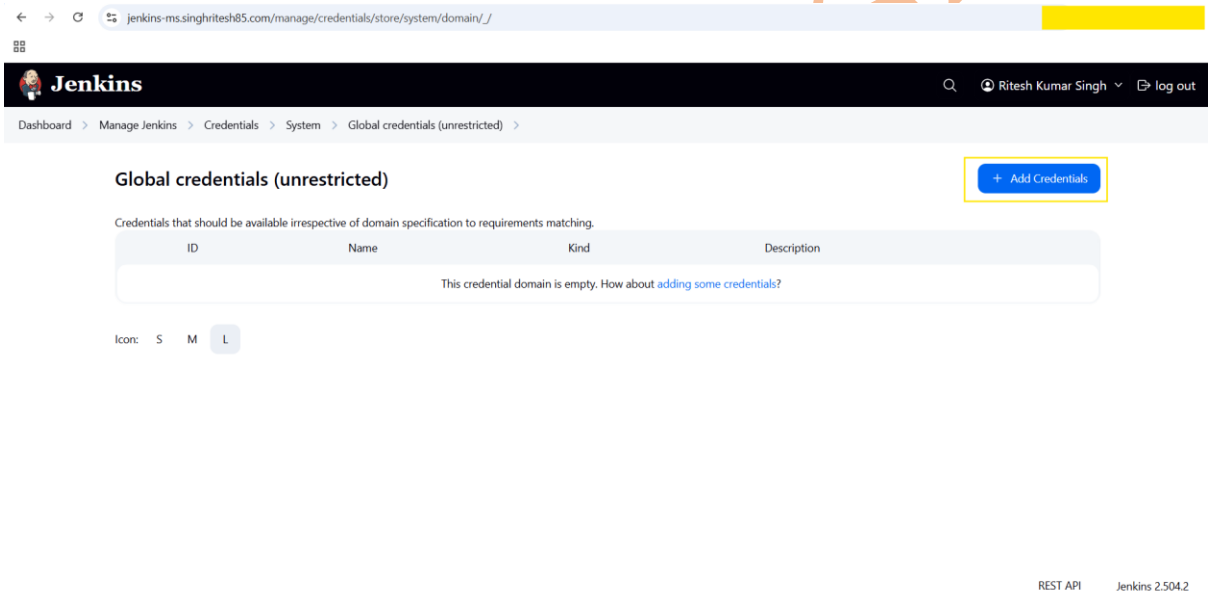


First, I created two credentials in Jenkins of kind username with password one to configure Jenkins Slave and another for GitHub as shown in the screenshot attached below.

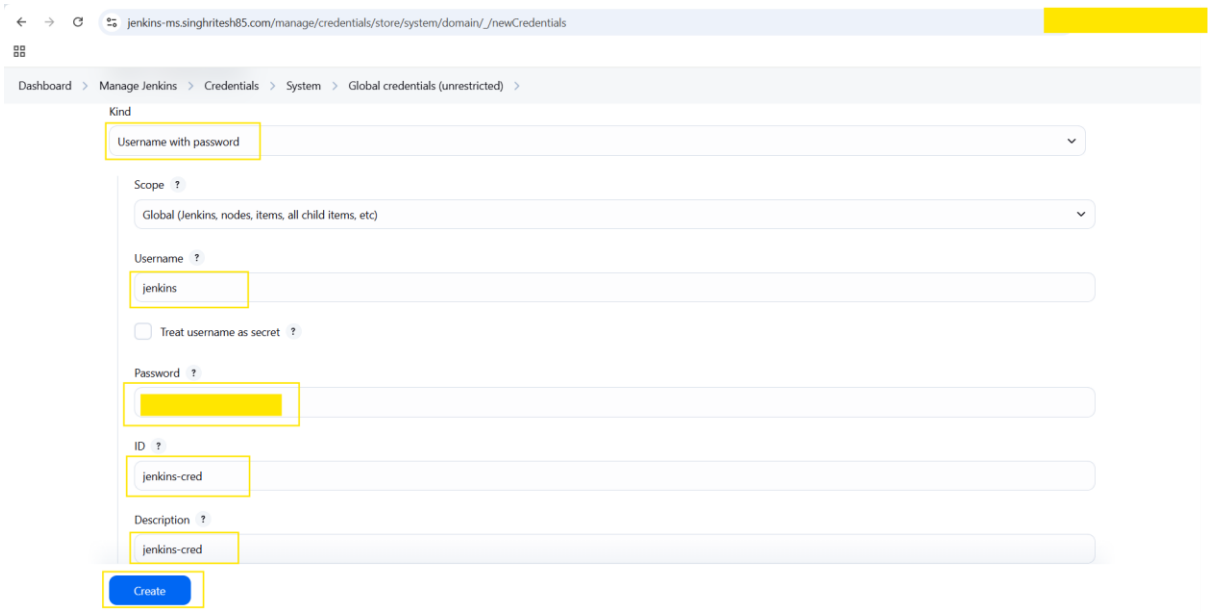

jenkins-ms.singhritesh85.com
Jenkins
Ritesh Kumar Singh
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
0/2
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds
REST API
Jenkins 2.504.2


jenkins-ms.singhritesh85.com/manage/
Jenkins
Ritesh Kumar Singh
log out
Dashboard
Manage Jenkins
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
0/2
Manage Jenkins
Search settings
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Appearance
Configure the look and feel of Jenkins
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users

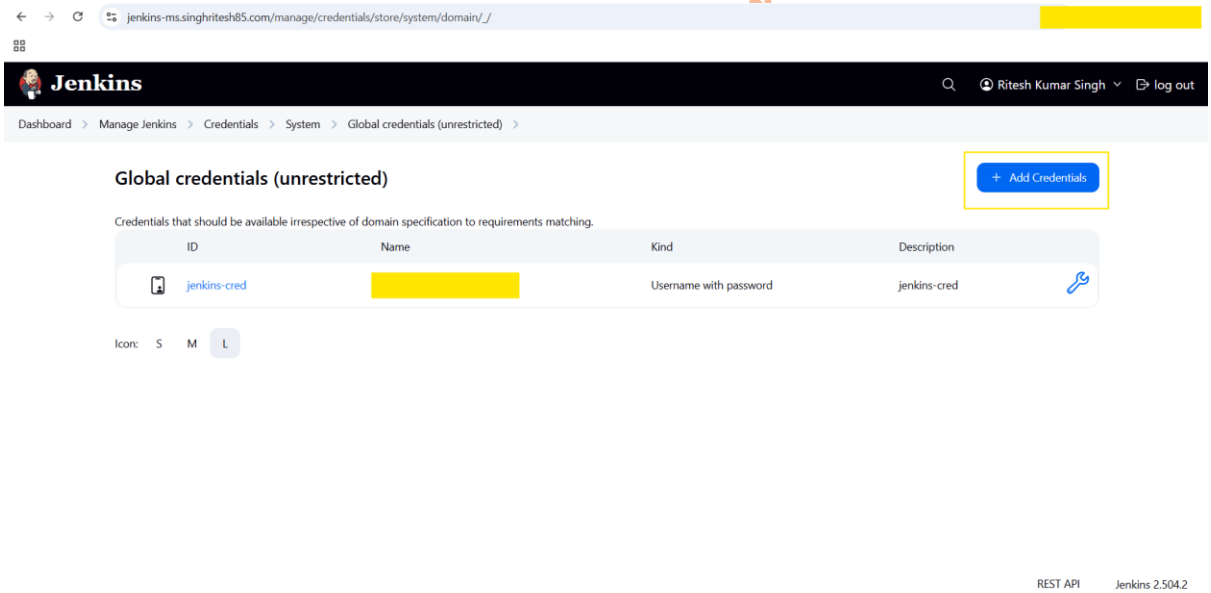


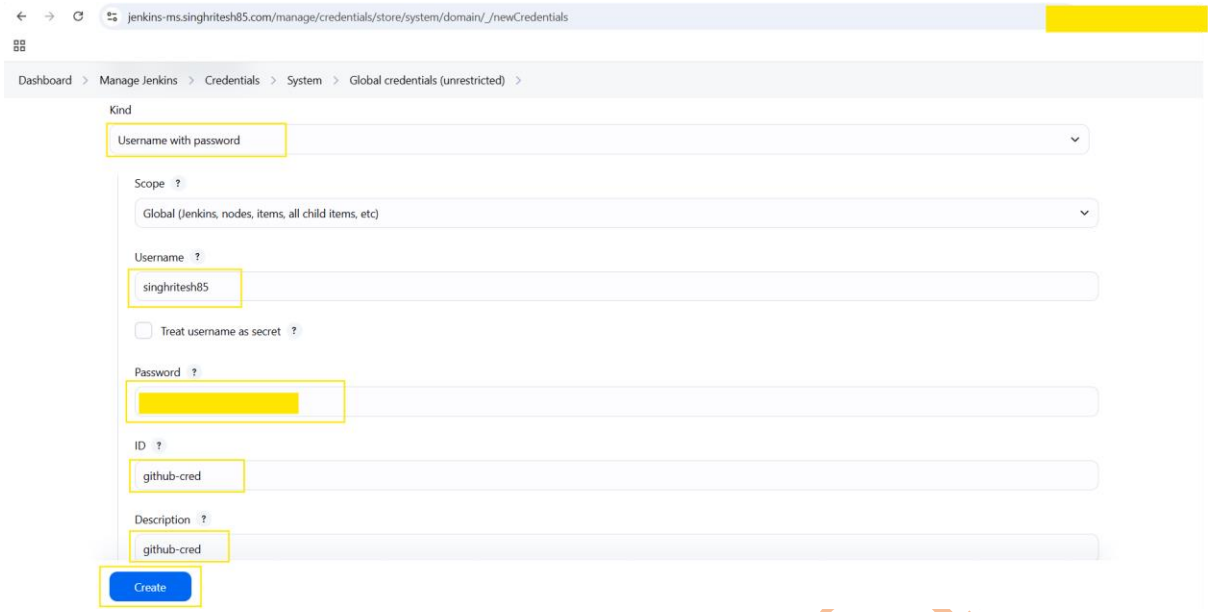


At the time of creation of Jenkins Slave Server using Terraform I used the boot-strap script (user_data_jenkins_slave.sh) to create a user jenkins I used the same user and its password to create credentials for Jenkins.

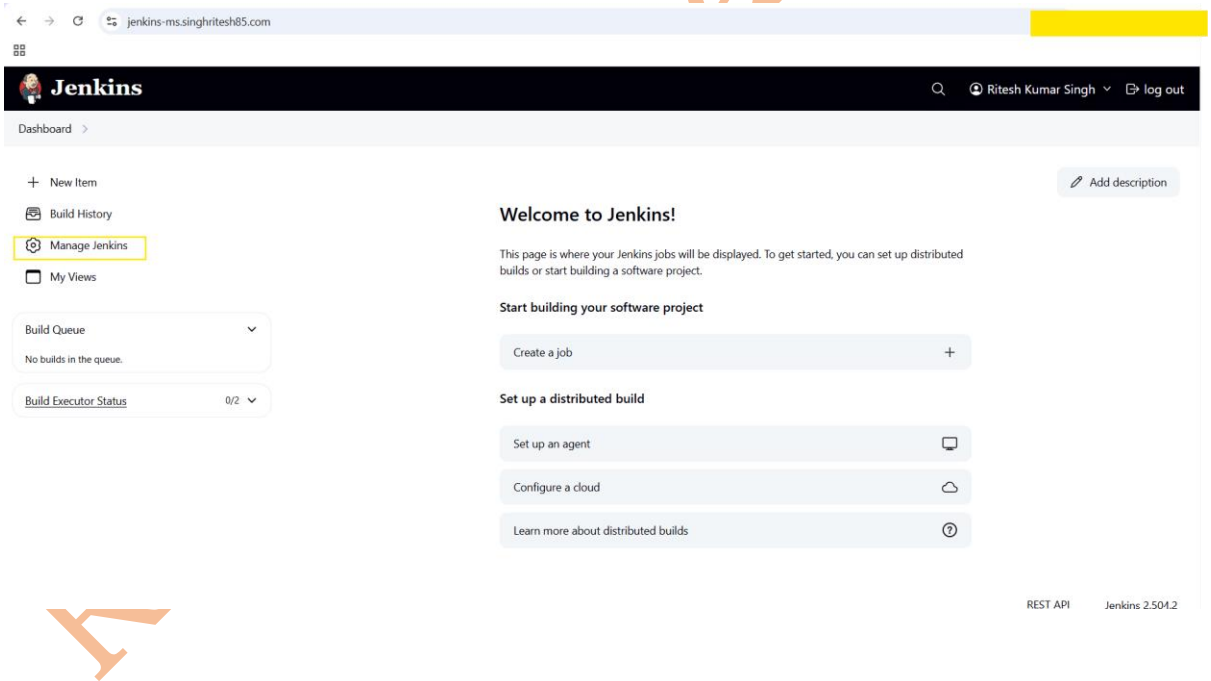


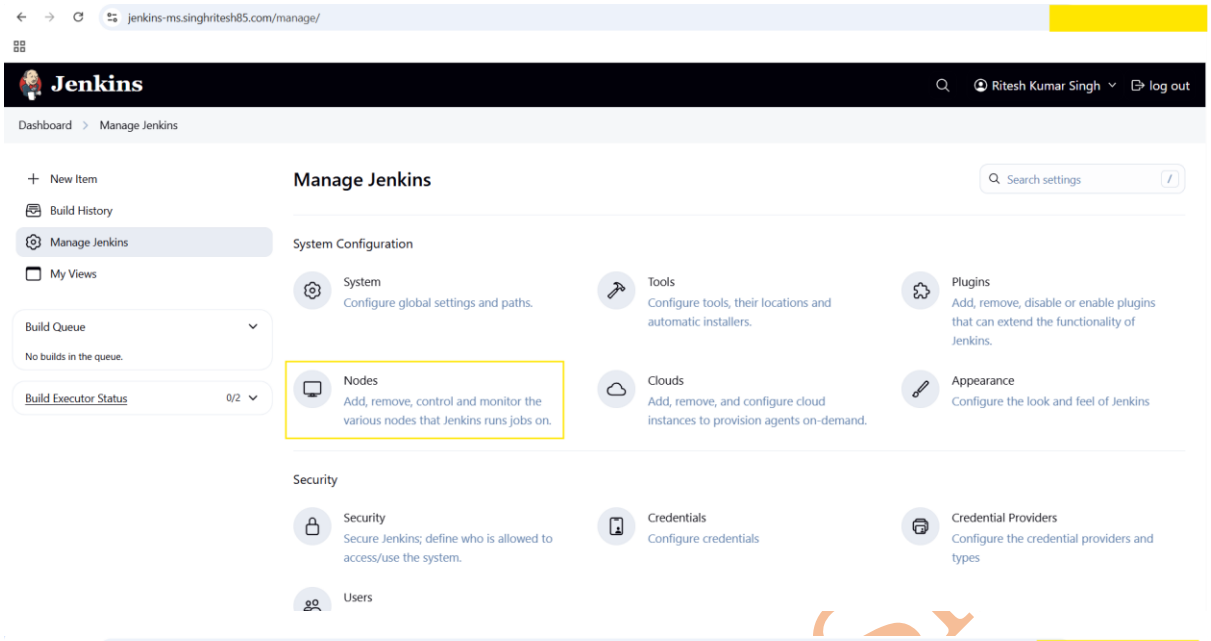
For Configuring credentials for GitHub in Jenkins I used the password as Personal Access Token of your GitHub Account for this Project. You can also use your SSH Keys

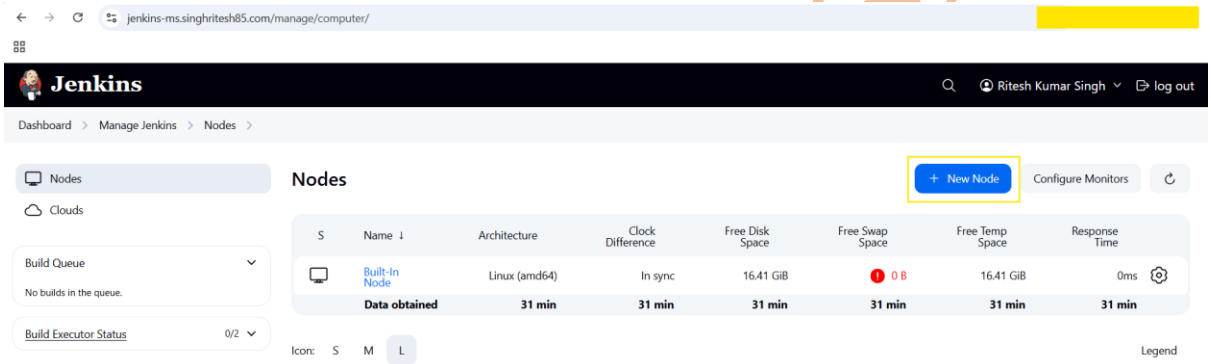


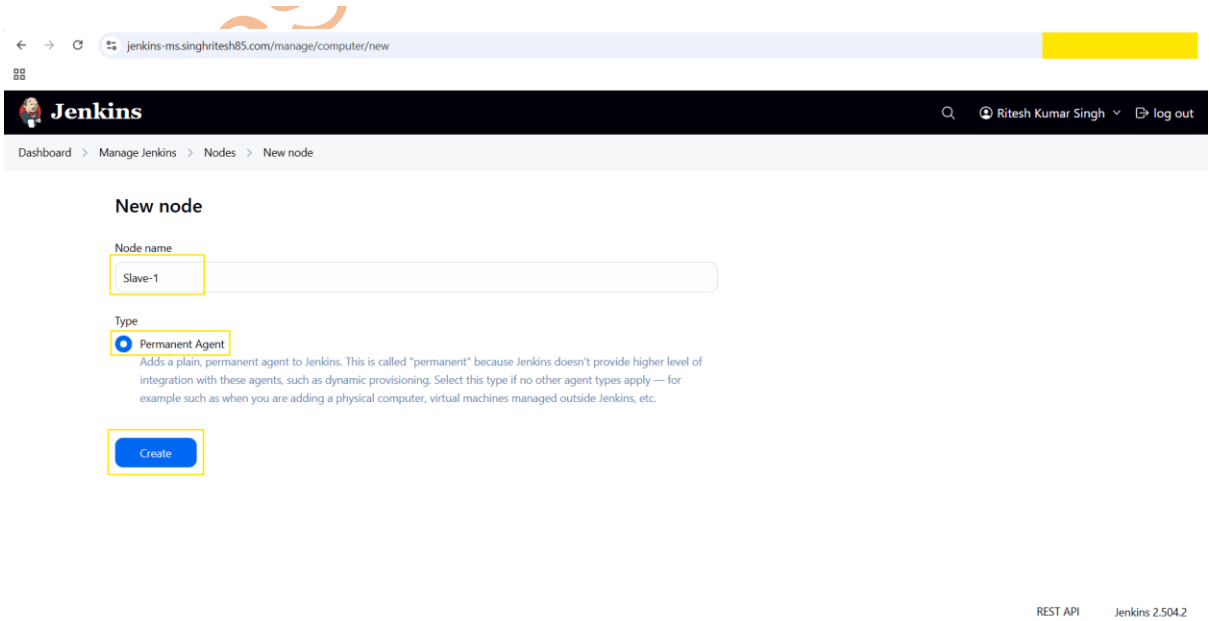


Then I configured the Slave Node for Jenkins as shown in the screenshot attached below.

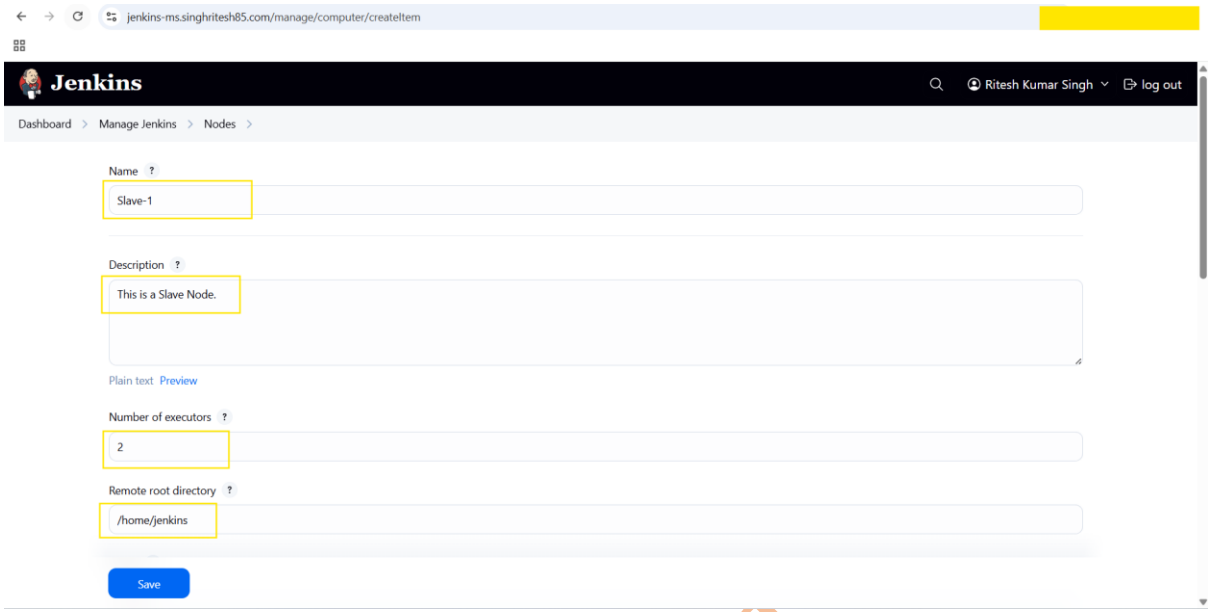


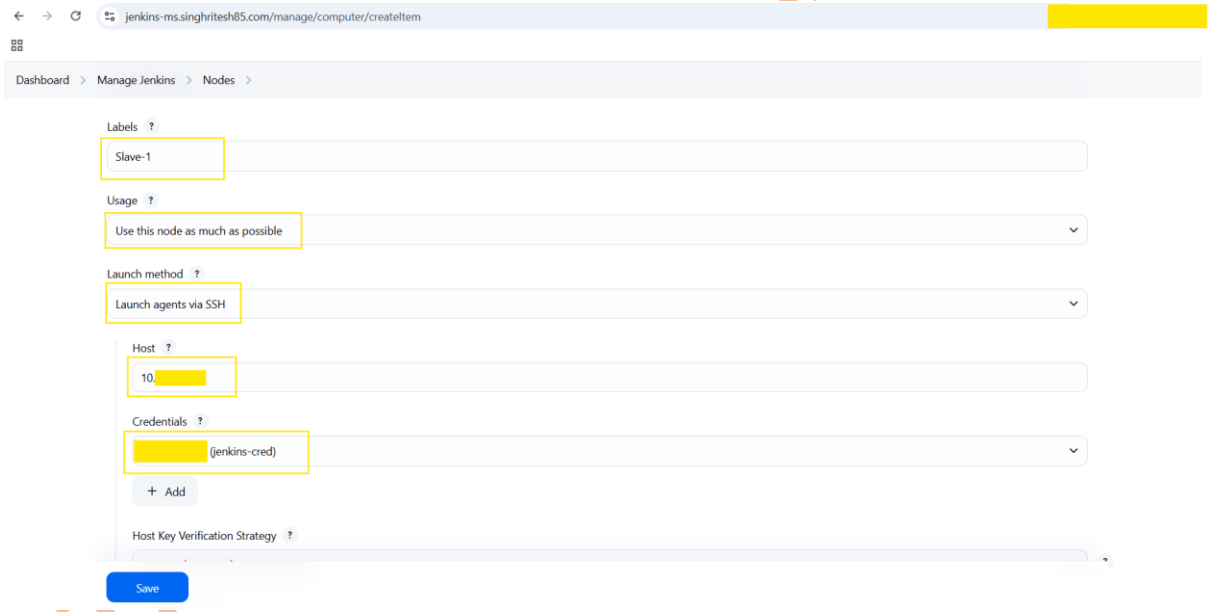

jenkins-ms.singhritesh85.com/manage/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status 0/2
Manage Jenkins
Search settings
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Appearance
Configure the look and feel of Jenkins
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users

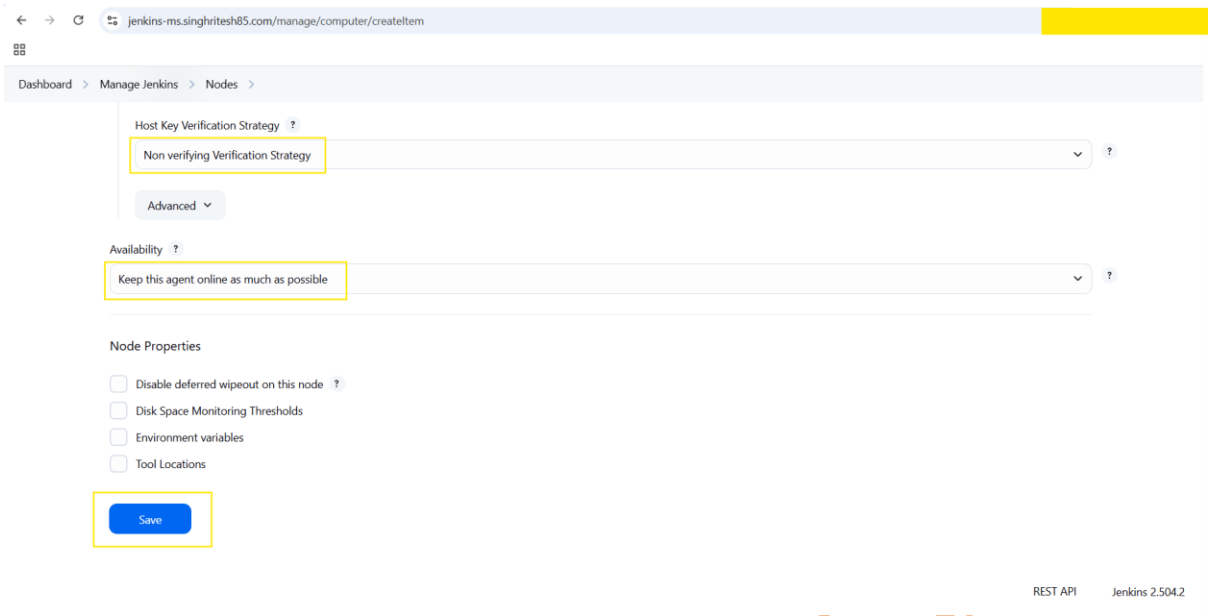

jenkins-ms.singhritesh85.com/manage/computer/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Nodes
Nodes
Clouds
Build Queue
No builds in the queue.
Build Executor Status 0/2
Nodes
+ New Node
Configure Monitors
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
Built-In Node
Linux (amd64)
In sync
16.41 GiB
0 B
16.41 GiB
0ms
Data obtained
31 min
Icon: S M L
Legend
REST API
Jenkins 2.504.2

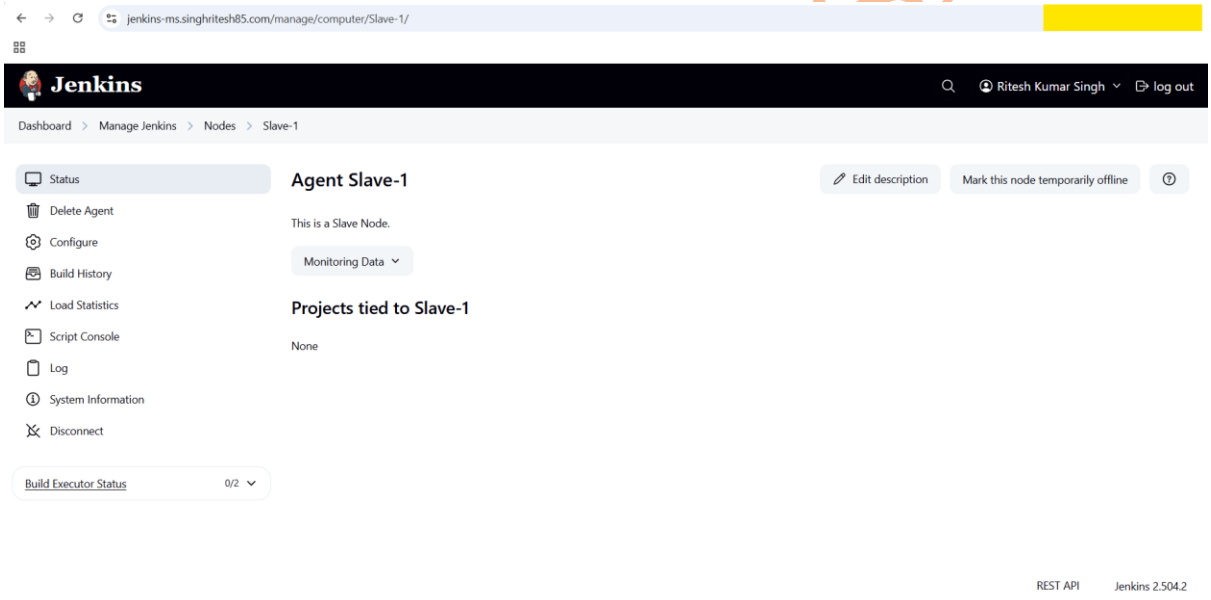

jenkins-ms.singhritesh85.com/manage/computer/new
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Nodes > New node
New node
Node name
Slave-1
Type
Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
Create
REST API
Jenkins 2.504.2

For this project I used Jenkins Master of Instance Type t3.medium but in your organisation you can use Instance Type **m4.4xlarge** or **m5.8xlarge**. Selection of Instance Type depends on your organisation and the project.







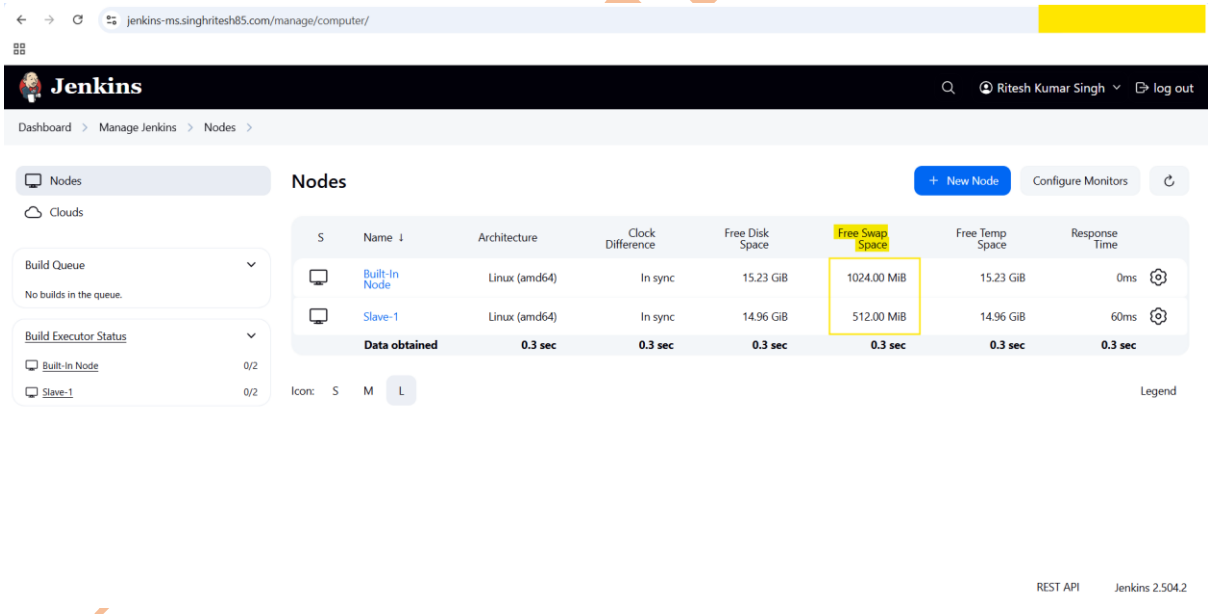


I added 1 GB of swap space on Jenkins Master and 512MB of swap space on Jenkins Slave as shown in the screenshot attached below.

```
[root@                ~]# free -mh
              total        used        free      shared  buff/cache   available
Mem:            1.9G        760M        113M        448K        1.0G        1.0G
Swap:             0B          0B          0B
[root@                ~]# fallocate -l 1G /swapfile
[root@                ~]# chmod 600 /swapfile
[root@                ~]# mkswap /swapfile
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=
[root@                ~]# vim /etc/fstab
[root@                ~]# swapon -a
```

```
[root@                ~]# cat /etc/fstab
#
UUID=                                    /           xfs    defaults,noatime  1   1

/swapfile swap swap defaults 0 0
[root@                ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           1.9G        761M        107M        448K        1.0G        1.0G
Swap:          1.0G          0B        1.0G
```

```
[root@                ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           1.9G        310M         75M        492K        1.5G        1.4G
Swap:            0B          0B          0B
[root@                ~]# fallocate -l 512M /swapfile
[root@                ~]# chmod 600 /swapfile
[root@                ~]# mkswap /swapfile
Setting up swapspace version 1, size = 512 MiB (536866816 bytes)
no label, UUID=
[root@                ~]# vim /etc/fstab
[root@                ~]# cat /etc/fstab
#
UUID=                                    /           xfs    defaults,noatime  1   1

/swapfile swap swap default 0 0
[root@                ~]# swapon -a
[root@                ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           1.9G        312M         69M        492K        1.5G        1.4G
Swap:          511M          0B        511M
```



Before proceeding further, I established the password less authentication between **Jenkins-Slave** and **Docker-Server**. On **Jenkins-Slave** run the commands as shown in the screenshot attached below.

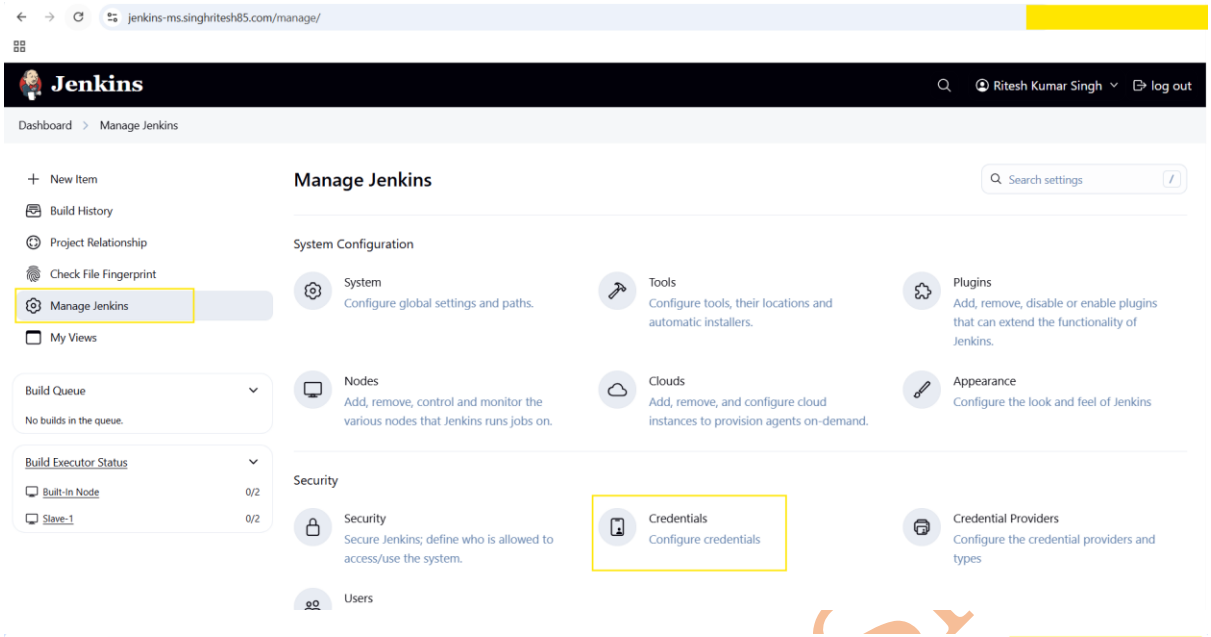
```
[root@ ~]# su - jenkins
Last login: 2025 on pts/0
[jenkins@ ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jenkins/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jenkins/.ssh/id_rsa.
Your public key has been saved in /home/jenkins/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:E Y jenkins@ .us-east-2.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|+..      o.|
|                |
|                |
|                |
|  .+ o          |
|      .         |
|                |
|                |
+----[SHA256]-----+
[jenkins@ ~]$ ssh-copy-id docker-admin@
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jenkins/.ssh/id_rsa.pub"
The authenticity of host ' ( )' can't be established.
ECDSA key fingerprint is SHA256:m                                o.
ECDSA key fingerprint is MD5:7                                  0.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
docker-admin@ 's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'docker-admin@ '"
and check to make sure that only the key(s) you wanted were added.
```

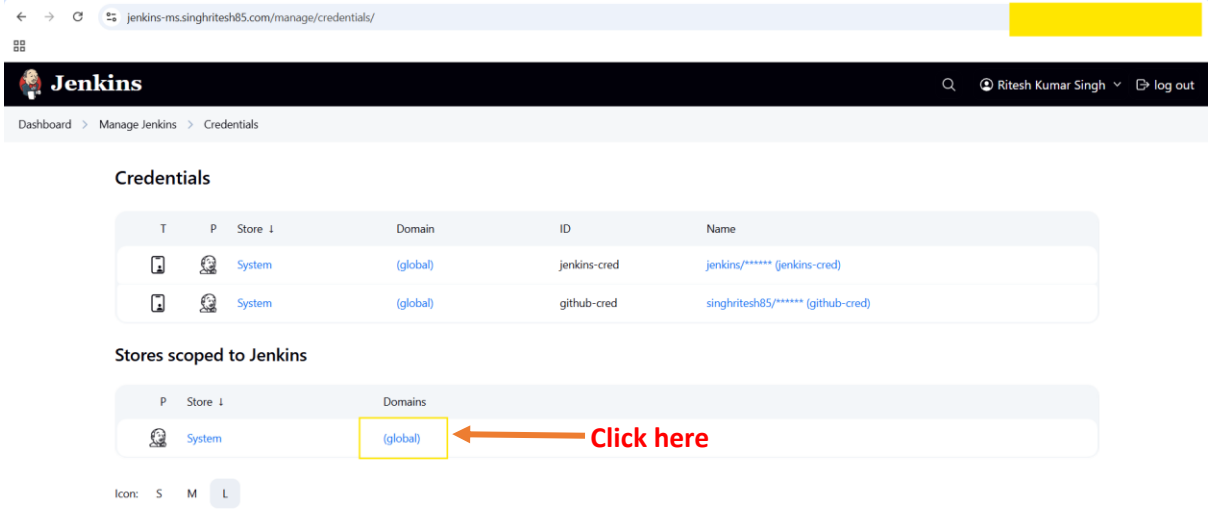
```
[jenkins@ ~]$ ssh 'docker-admin@10.10.4.97'
Last login: 2025
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[docker-admin@ ~]$ sudo -i
[root@ ~]# logout
[docker-admin@ ~]$ logout
Connection to  closed.
[jenkins@ ~]$
```

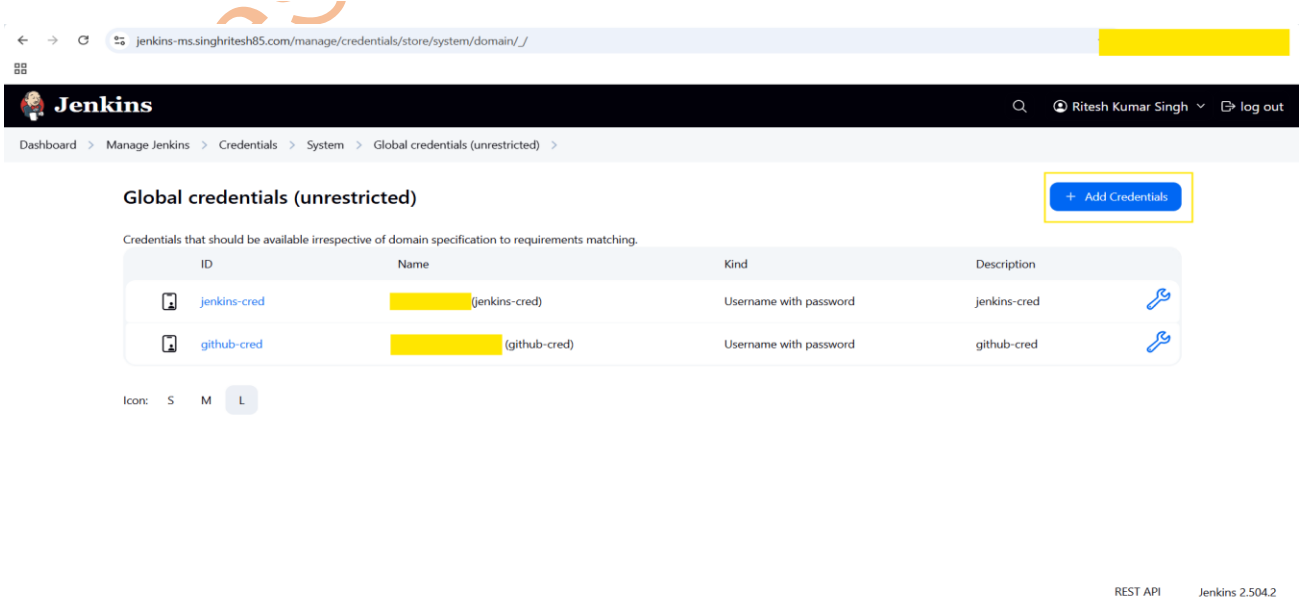
As shown in the screenshot attached above, we achieved the password less authentication between jenkins-slave and docker-server.

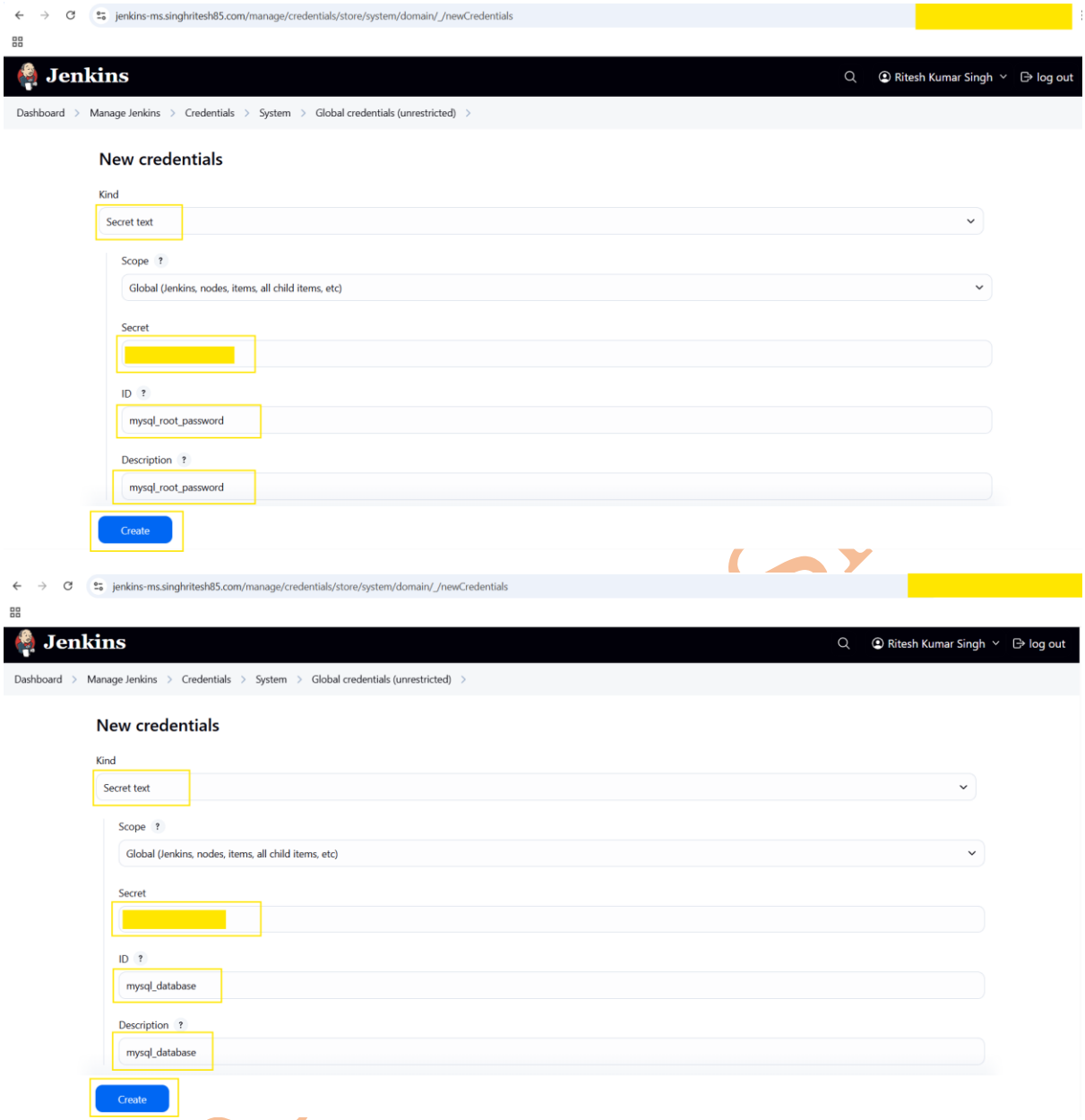
Then I created two Jenkins Credentials named as MYSQL_ROOT_PASSWORD and MYSQL_DATABASE of **Kind secret text** as shown in the screenshot attached below.

Go to Manage Jenkins > Credentials and create the Credentials.

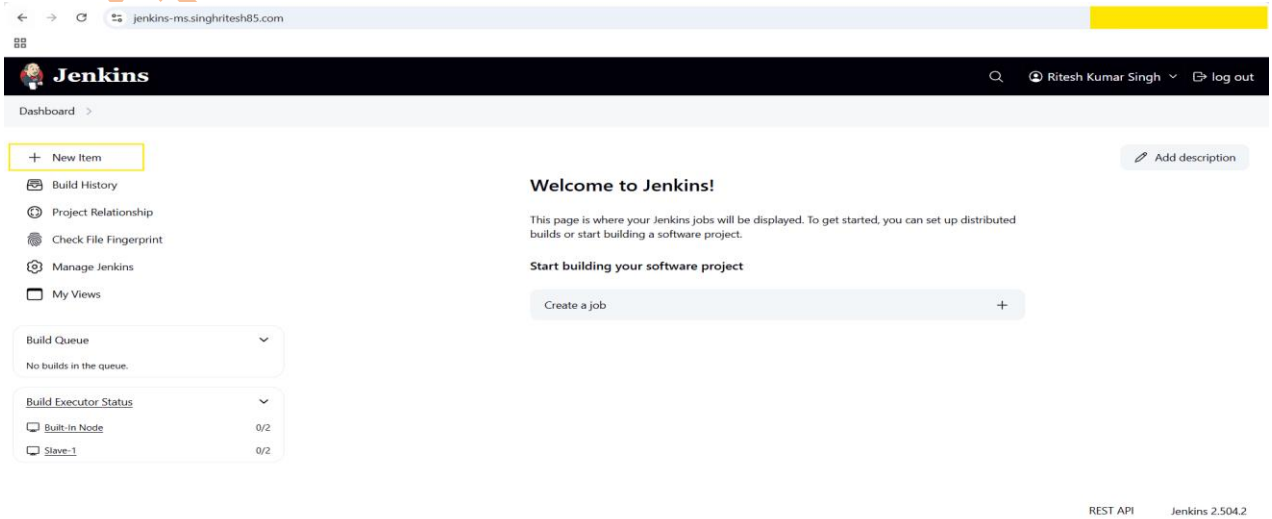

jenkins-ms.singhritesh85.com/manage/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins
New Item
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node 0/2
Slave-1 0/2
Manage Jenkins
Search settings
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Appearance
Configure the look and feel of Jenkins
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users

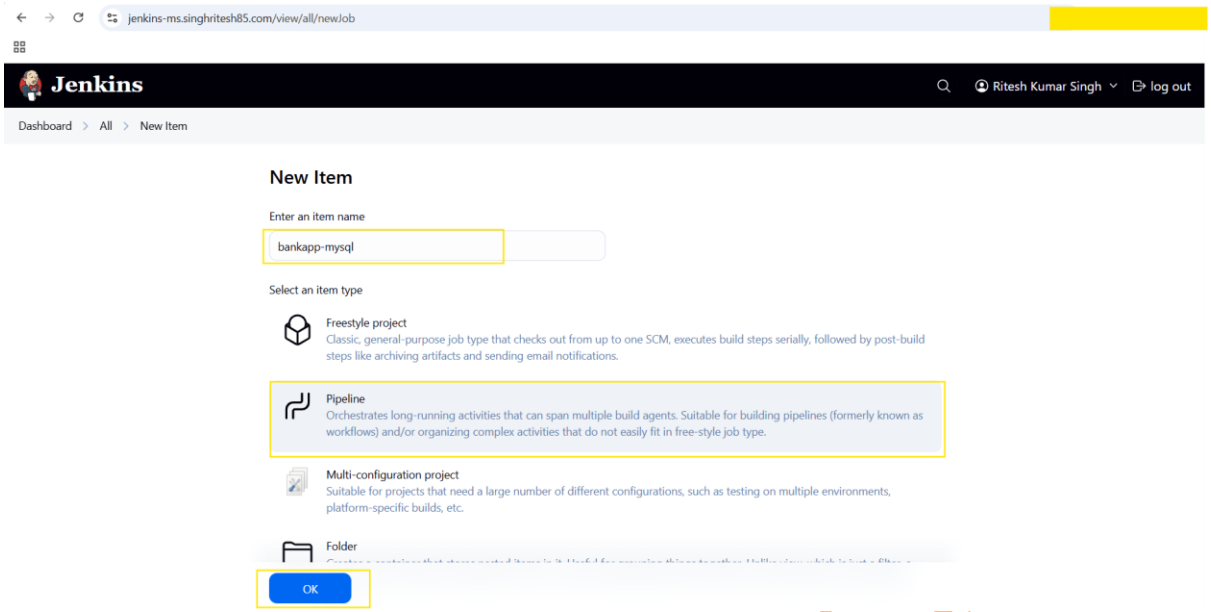

jenkins-ms.singhritesh85.com/manage/credentials/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Credentials
Credentials
T P Store ↓ Domain ID Name
System (global) jenkins-cred jenkins/****** (jenkins-cred)
System (global) github-cred singhritesh85/****** (github-cred)
Stores scoped to Jenkins
P Store ↓ Domains
System (global)
Click here
Icon: S M L
REST API Jenkins 2.504.2


jenkins-ms.singhritesh85.com/manage/credentials/store/system/domain/_/
Jenkins
Ritesh Kumar Singh
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID Name Kind Description
jenkins-cred (jenkins-cred) Username with password jenkins-cred
github-cred (github-cred) Username with password github-cred
Icon: S M L
REST API Jenkins 2.504.2

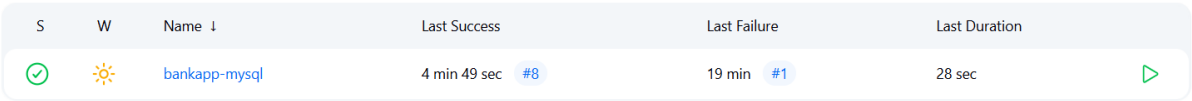


Now I proceed with the deployment using Jenkins Job as shown in the screenshot attached below.





Provided the Jenkinsfile and ran the Jenkins Job as shown in the screenshot attached below. Before running the Jenkins Job make sure two Log Groups should be created with the name of **bankapp** and **mysql**.



For your reference I kept the Jenkinsfile in GitHub Repo https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch.git.

Jenkinsfile

```
def DOCKER_SERVER="10.10.4.197"  //Provide Docker Server Public IP/Private IP Address.

pipeline {
   agent {
      label{
         label "Slave-1"
         customWorkspace "/home/jenkins/bankapp"
      }
   }
   environment{
      JAVA_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
      PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
      MYSQL_ROOT_PASSWORD=credentials('mysql_root_password')
      MYSQL_DATABASE=credentials('mysql_database')
   }
   parameters {
      string(name: 'COMMIT_ID', defaultValue: '', description: 'Provide the Commit ID')
      string(name: 'REPO_NAME', defaultValue: '', description: 'Provide the ECR Repository Name for Application Image')
      string(name: 'TAG_NAME', defaultValue: '', description: 'Provide the TAG Name')
   }
   stages{
      stage("clone-code"){
         steps{
            cleanWs()
            checkout scmGit(branches: [[name: '${COMMIT_ID}']], extensions: [], userRemoteConfigs: [[credentialsId: 'github-cred', url: 'https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-CloudWatch.git']])
         }
      }
      stage("Build and DockerImage"){
         steps{
            sh 'mvn clean install'
```

```
            sh 'wget https://github.com/aws-observability/aws-otel-java-instrumentation/releases/download/v2.11.0/aws-opentelemetry-agent.jar'
            sh "sed -i -e ''s/##MYSQL_ROOT_PASSWORD##/${MYSQL_ROOT_PASSWORD}/g'' ./.env"
            sh "sed -i -e ''s/##MYSQL_DATABASE##/${MYSQL_DATABASE}/g'' ./.env"
            sh "sed -i -e ''s@##REPO_NAME##@${REPO_NAME}@g'' ./.env"
            sh "sed -i -e ''s/##TAG_NAME##/${TAG_NAME}/g'' ./.env"
            sh "scp -rv Dockerfile-Project-1 docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv target docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv aws-opentelemetry-agent.jar docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv docker-compose.yaml docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv .env docker-admin@${DOCKER_SERVER}:~/"
            sh 'docker system prune -f'
            sh "docker build -t ${REPO_NAME}:${TAG_NAME} -f Dockerfile-Project-1 ."
            sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 027330342406.dkr.ecr.us-east-2.amazonaws.com'
            sh "docker push ${REPO_NAME}:${TAG_NAME}"
        }
    }
    stage("Deployment"){
        steps{
            sh """ssh docker-admin@$DOCKER_SERVER <<-EOF
            docker-compose -p bankapp down     ### do not use docker-compose down -v here oherwise docker volume will be deleted.
            docker system prune -f
            sudo docker-compose -p bankapp up -d
            exit
            EOF
            """
        }
    }
  }
}
```

.env

MYSQL_ROOT_PASSWORD=##MYSQL_ROOT_PASSWORD##

MYSQL_DATABASE=##MYSQL_DATABASE##

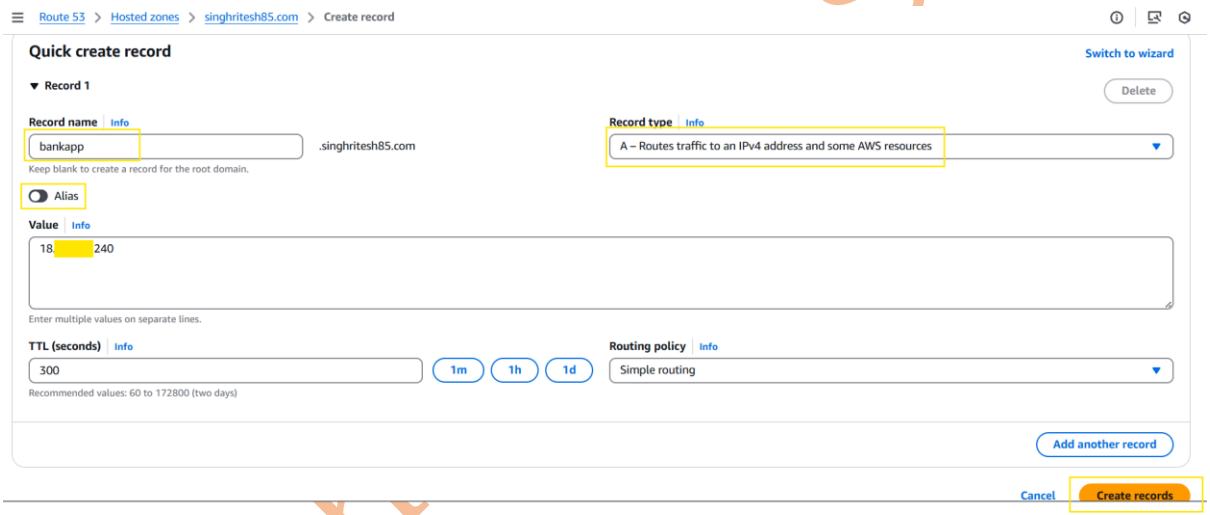
REPO_NAME=##REPO_NAME##

TAG_NAME=##TAG_NAME##

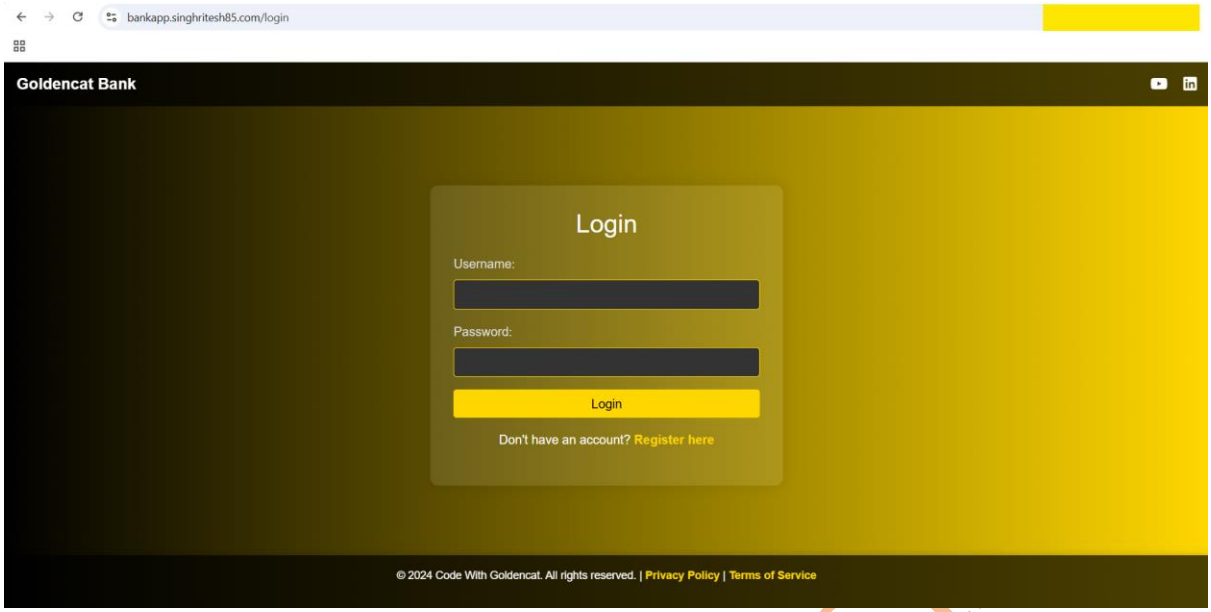
After running the Jenkins Job, the docker containers will be created as shown in the screenshot attached below.

```
[docker-admin@                ~]$ docker ps
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED          STATUS          PORTS
                    NAMES
              02          6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev:1.22   "/__cacert_entrypoin…"   19 seconds ago   Up 17 seconds   0.0.0.0:8080->8080/tc
p, :::8080->8080/tcp   bankapp
              mysql:8.0.0                                              "docker-entrypoint.s…"   19 seconds ago   Up 18 seconds   0.0.0.0:3306->3306/tc
p, :::3306->3306/tcp   mysql
```

Then I provided the Public IP of Nginx Server in Route53 to create the record set of A-Type (without Alias) as shown in the screenshot attached below.

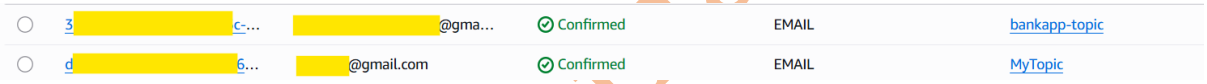


At last, I checked bankapp application was up and running as shown in the screenshot attached below.

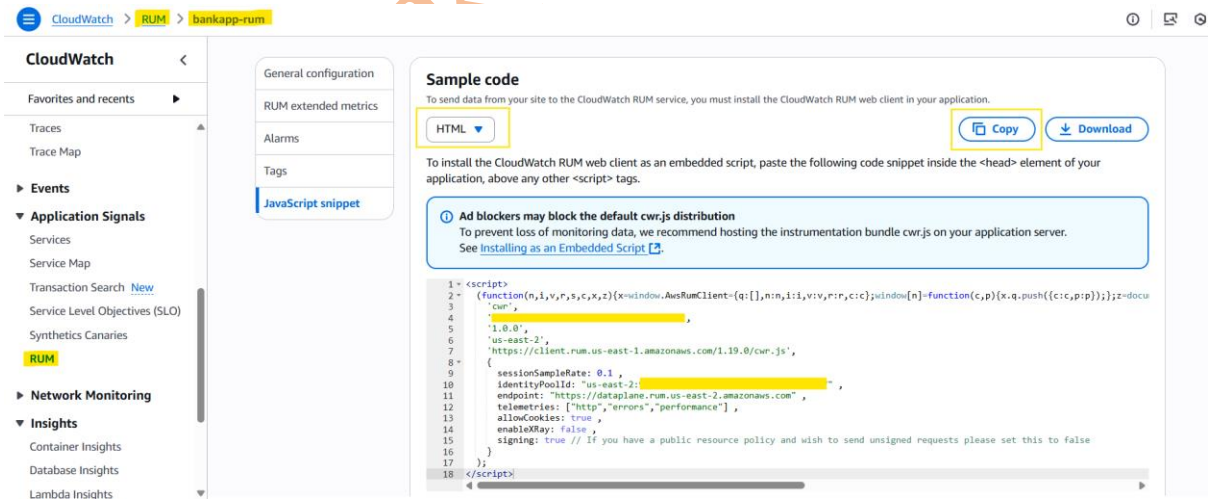


Monitoring and Log Aggregation with CloudWatch

After successful deployment of containers and creation of bankapp application URL in Route53 with the creation of Record Set I confirmed the subscription for SNS Notification. Please confirm the subscription at this stage otherwise unnecessary emails will be sent to your group email ID.



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 <head> as shown in the screenshot attached below.

Bank-App-Monitoring-Logging-using-CloudWatch-Private / src / main / resources / templates / dashboard.html

singhritesh85 Update dashboard.html [redacted] 25 minutes ago History

Code Blame 219 lines (196 loc) · 8.05 KB Raw

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
        <script>
                (function(n,i,v,r,s,c,x,z){x=window.AwsRumClient={q:[],n:n,i:i,v:v,r:r,c:c};window[n]=function(c,p){x.q.push({c:c,p:p});};z=document.createElemer
                        'cwr',
                        '[redacted]',
                        '1.0.0',
                        'us-east-2',
                        'https://client.rum.us-east-1.amazonaws.com/1.19.0/cwr.js',
                        {
                                sessionSampleRate: 1 ,
                                identityPoolId: "[redacted]" ,
                                endpoint: "https://dataplane.rum.us-east-2.amazonaws.com" ,
                                telemetries: ["http","errors","performance"] ,
                                allowCookies: true ,
                                enableXRay: false ,
                                signing: true // If you have a public resource policy and wish to send unsigned requests please set this to false
                        }
                );
        </script>
    <title>Dashboard - Goldencat Bank</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            font-family: 'Arial', sans-serif;
```

In the similar way I added in more three files **login.html**, **register.html** and **transactions.html**. After doing these changes proceed with the deployment and after the deployment containers had been created and I checked the log of the docker container as shown in the screenshot attached below.

```
[docker-admin@[redacted] ~]$ docker ps
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED         STATUS         PORTS
                NAMES
[redacted]   02[redacted]6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev:1.01   "/__cacert_entrypoin…"   8 seconds ago   Up 7 seconds   0.0.0.0:8080->8080/tcp,
 :::8080->8080/tcp   bankapp
[redacted]   mysql:8.0.0                                              "docker-entrypoint.s…"   8 seconds ago   Up 7 seconds   0.0.0.0:3306->3306/tcp,
 :::3306->3306/tcp   mysql
[docker-admin@[redacted] ~]$ docker logs -f 0[redacted]e
```

```
Hibernate: select a1_0.id,a1_0.balance,a1_0.password,a1_0.username from account a1_0 where a1_0.username=?
Hibernate: select a1_0.id,a1_0.balance,a1_0.password,a1_0.username from account a1_0 where a1_0.username=?
Hibernate: select a1_0.id,a1_0.balance,a1_0.password,a1_0.username from account a1_0 where a1_0.username=?
Hibernate: select a1_0.id,a1_0.balance,a1_0.password,a1_0.username from account a1_0 where a1_0.username=?
[otel.javaagent 2025-06-02 04:18:27:852 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created G
auge for jvm.classes.loaded
[otel.javaagent 2025-06-02 04:18:27:857 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created C
ounter for jvm.gc.collections.count
[otel.javaagent 2025-06-02 04:18:27:857 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created C
ounter for jvm.gc.collections.elapsed
[otel.javaagent 2025-06-02 04:18:27:858 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created G
auge for jvm.memory.heap.init
[otel.javaagent 2025-06-02 04:18:27:858 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created G
auge for jvm.memory.heap.used
[otel.javaagent 2025-06-02 04:18:27:859 +0000] [jmx_bean_finder] INFO io.opentelemetry.javaagent.shaded.instrumentation.jmx.engine.MetricRegistrar - Created G
auge for jvm.memory.heap.committed
```

The same logs you can check from the CloudWatch Log Group as shown in the screenshot attached below.

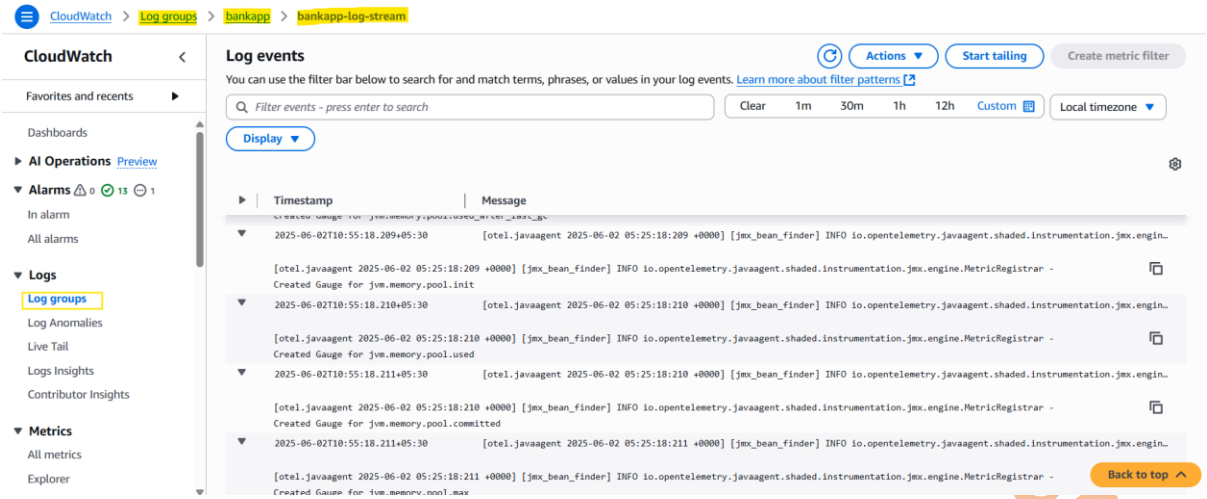
Log groups (4)

Actions | View in Logs Insights | Start tailing | Create log group

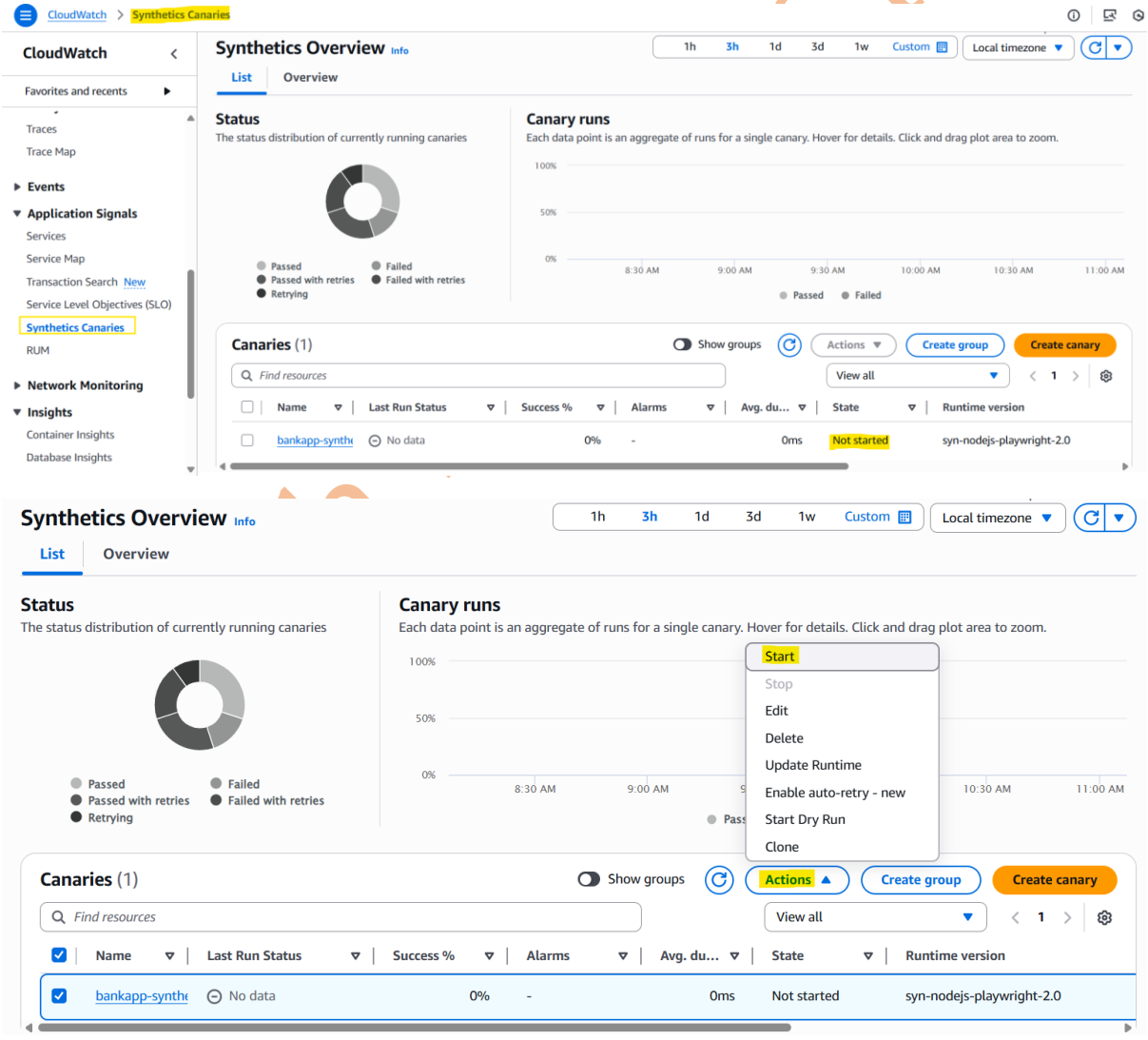
By default, we only load up to 10000 log groups.

Filter log groups or try prefix search | Exact match | 1

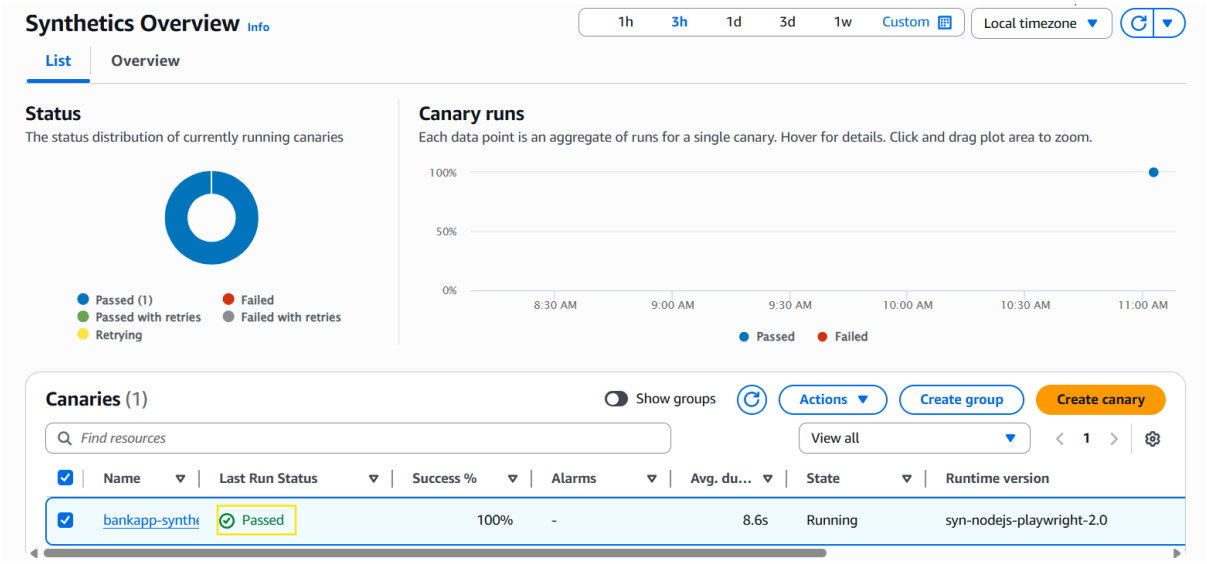
Log group	Log class	Anomaly d...	Data pr...	Sensitiv...	Retention	Metric fi...
/aws/application-signals/data	Standard	Configure	-	-	Never expire	-
/aws/vendedlogs/RUMService_bankapp-rum[redacted]	Standard	Configure	-	-	1 month	-
bankapp	Standard	Configure	-	-	Never expire	-
mysql	Standard	Configure	-	-	Never expire	-

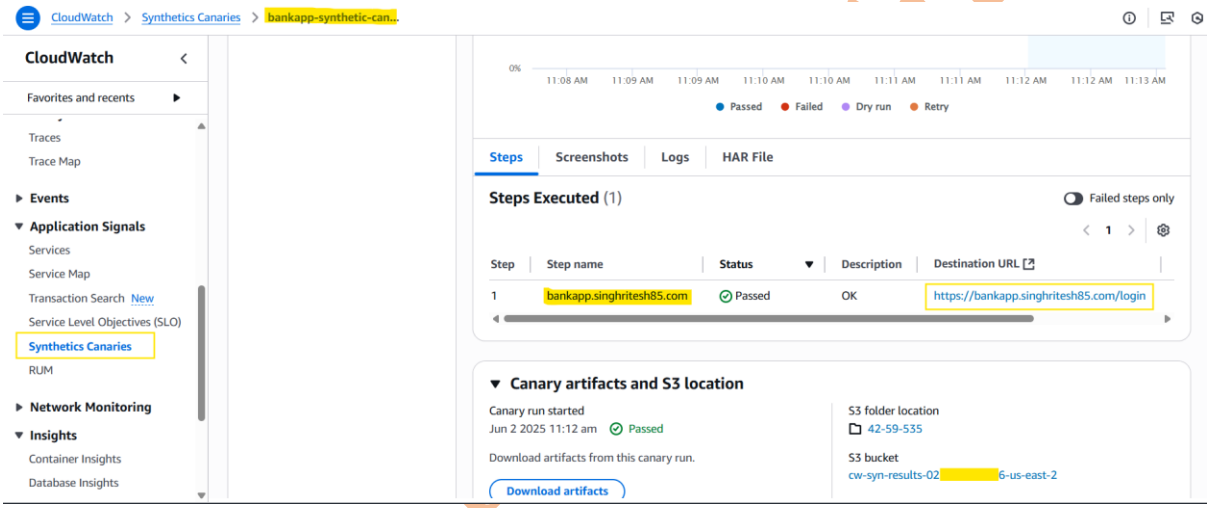


The CloudWatch Synthetic Canary had been configured using Terraform but initially it was not started you need to start it as shown in the screenshot attached below.

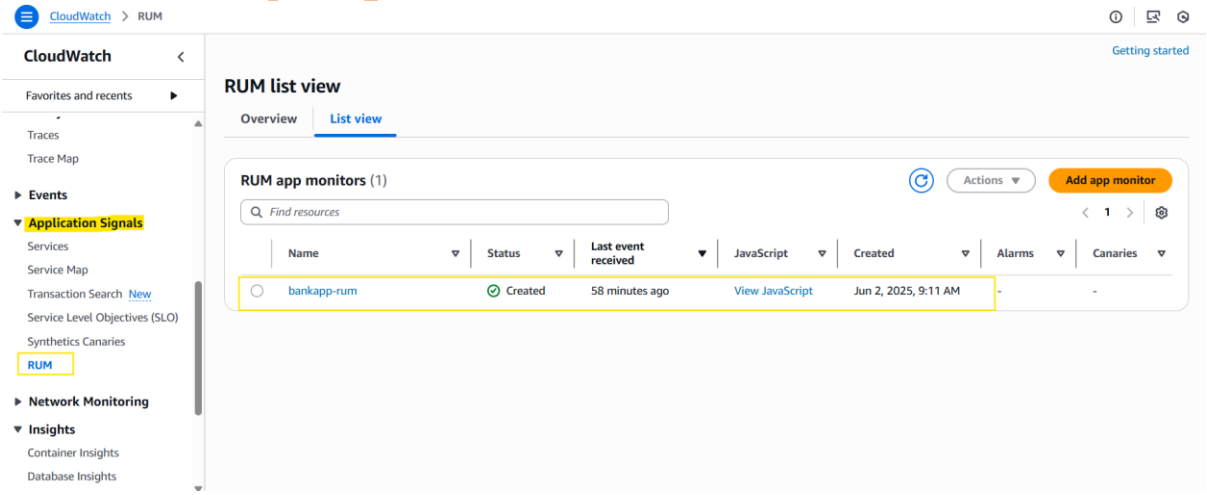


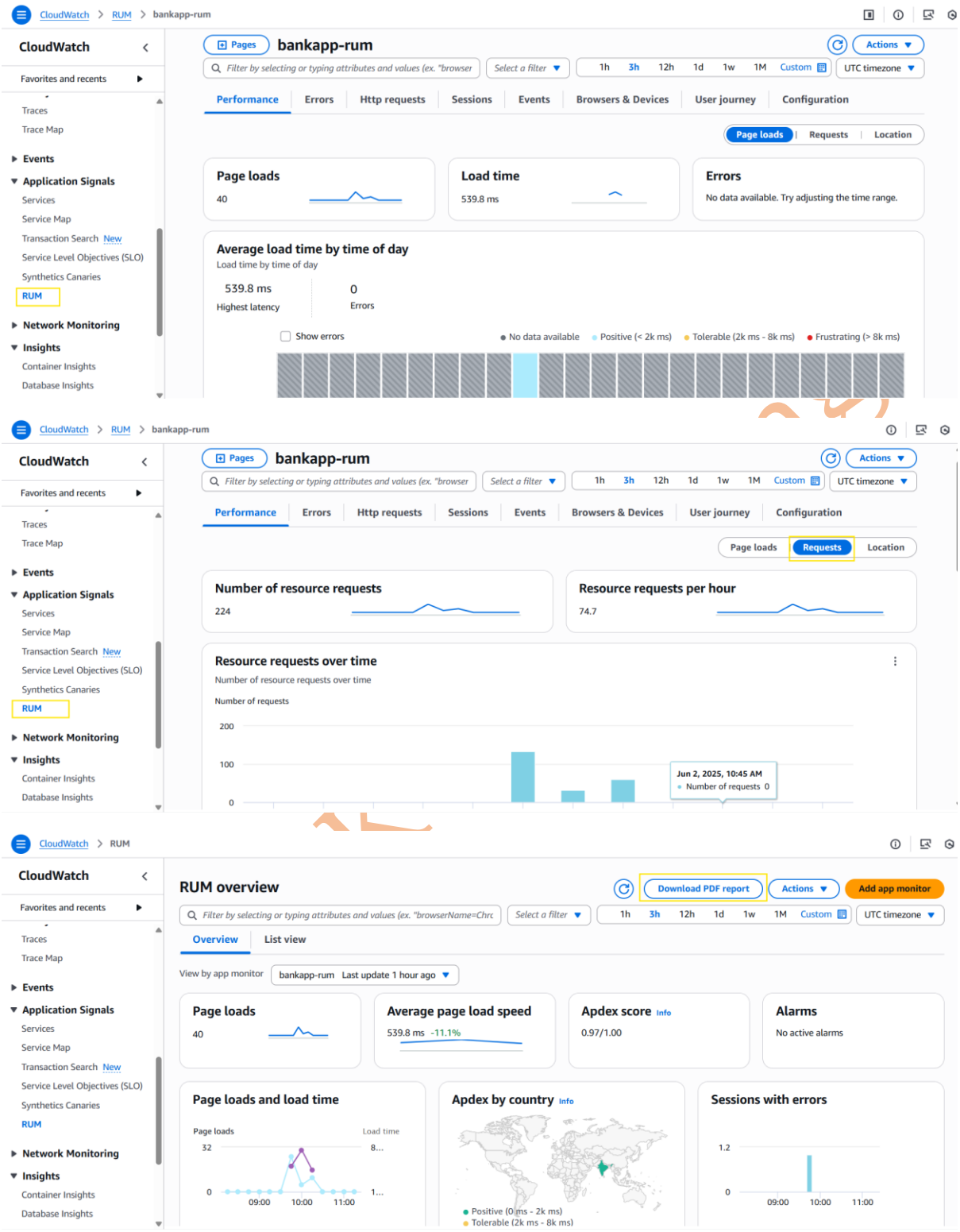
After starting it wait for 10 secs and then if the Bank Application URL was UP then for synthetic canary (synthetic monitoring) you will see the Run Status Passed as shown in the screenshot attached below.





I configured the Real User Monitoring (RUM) using Terraform for BankApp as shown in the screenshot attached below.





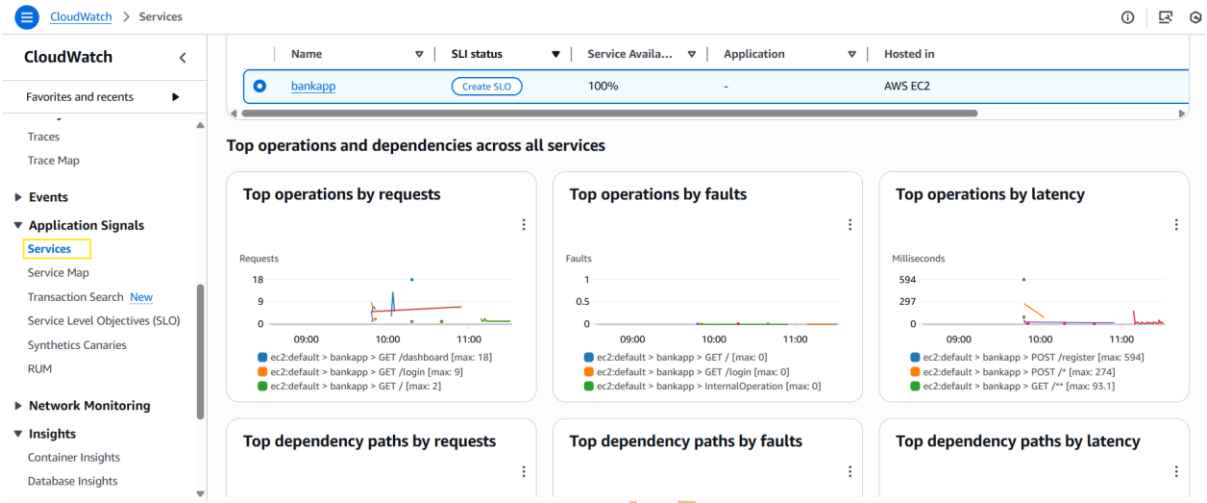
If you want you can download the Report as shown in the screenshot attached above.

As shown in the screenshot attached above the Appdex Score is 0.97 which represents the satisfaction level of Users. It ranges from 0 to 1. Where 0 represents maximum frustration.

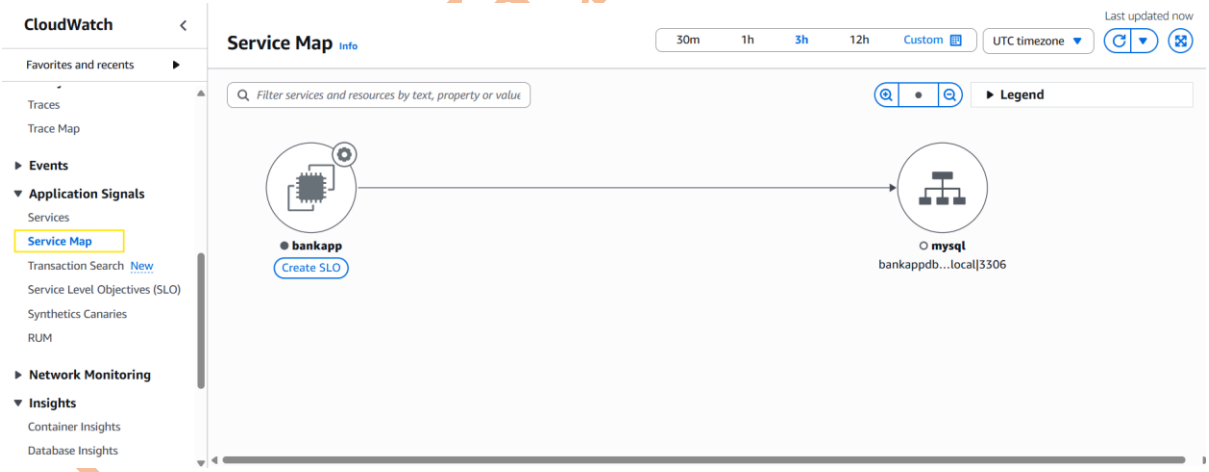
$$Appdex\ Score = \frac{Satisfied\ Request + (Tolerating\ Request)/2}{Total\ Request}$$

Where Total Request = Satified Request + Tolerating Request + Frustrated Request.

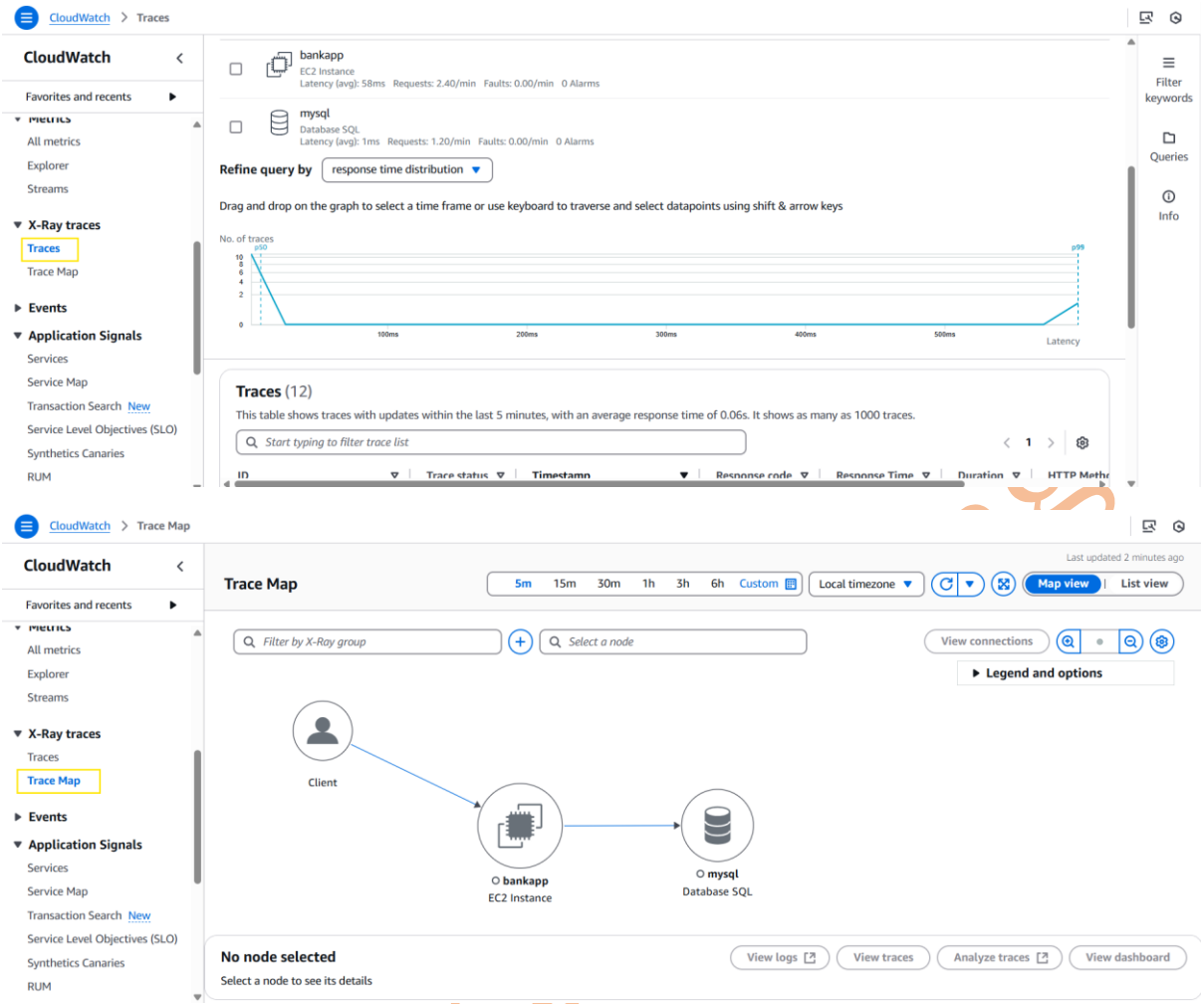
You can see the service with the same name which I provided in the Dockerfile which was bankapp.



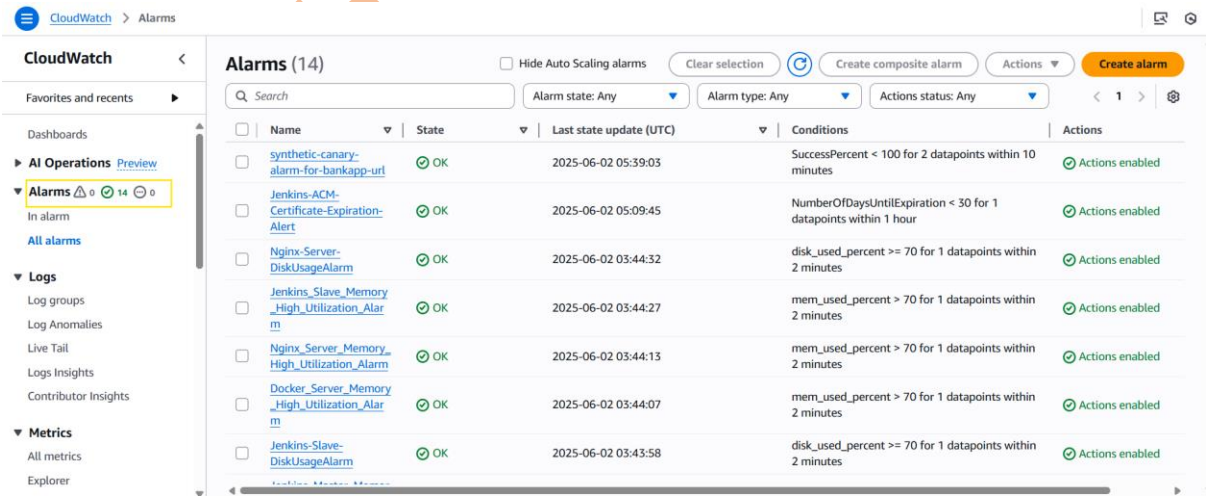
As shown in the screenshot attached above, you can observe the Latency which is basically the waiting time for a request to be handled.



You can observer the X-Ray traces and Trace Map as shown in the screenshot attached below.



Finally, I observed the CloudWatch Alarm and found that all the CloudWatch Alarms were in OK state as shown in the screenshot attached below.

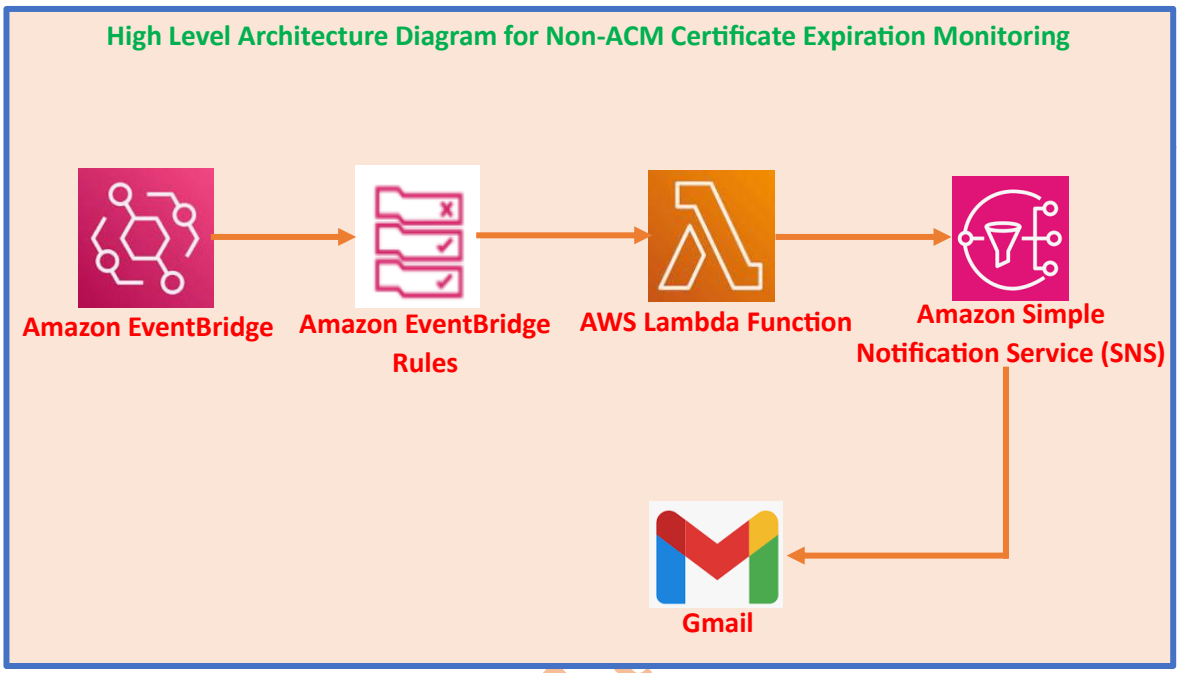


As shown in the above screenshots I performed the Application Performance Monitoring (APM) using the AWS CloudWatch.

SSL Monitoring

I performed the SSL Monitoring of AWS Certificate Manager Certificates using the CloudWatch.

For expiration of Non-ACM (Non-AWS Certificate Manager Certificates) SSL Certificate monitoring you can use **Lambda Function**, **CloudWatch Event Rule** (now it became **Amazon Event Bridge Rule**), and **SNS Topic**.



However, for this project I had used Bash shell script for Monitoring of Non-ACM SSL Certificate regarding its expiration, which will send email to the group Email ID using sendmail. For this project I configured sendmail without using AWS SES, I used Gmail username and its App Password for configuration of sendmail. Below is the shell script which I used for SSL Monitoring Expiration.

Configuration of sendmail with Gmail (username as Gmail username and Password as App Password)

yum install -y sendmail sendmail-cf m4 mailx

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@ip-10-XX-X-XXX auth]# ls

gmail-auth gmail-auth.db

cd /etc/mail/

vim genericstable

ritesh abc@gmail.com

ritesh@ip-10-XX-X-XXX.us-east-2.compute.internal abc@gmail.com

makemap hash genericstable < genericstable

ls

genericstable.db ----------> **File will be generated**

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(`genericstable',`hash -o /etc/mail/genericstable.db')dnl

FEATURE(`authinfo',`hash -o /etc/mail/authinfo/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
```

```
[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/genericstable
[root@            mail]# ls
Makefile  access.db       domaintable     helpfile          mailertable.db  sendmail.mc          submit.mc      virtusertable.db
\         aliasesdb-stamp  domaintable.db  local-host-names  make            sendmail.mc.rpmsave  trusted-users
access    auth            genericstable   mailertable       sendmail.cf     submit.cf            virtusertable
[root@            mail]# makemap hash genericstable < genericstable
[root@            mail]# ls
Makefile  access.db       domaintable     genericstable.db  mailertable     sendmail.cf          submit.cf      virtusertable
\         aliasesdb-stamp  domaintable.db  helpfile          mailertable.db  sendmail.mc          submit.mc      virtusertable.db
access    auth            genericstable   local-host-names  make            sendmail.mc.rpmsave  trusted-users
```

```
[root@                    ~]# cat /etc/mail/genericstable
ritesh                    @gmail.com
ritesh@ip-10-          .us-east-2.compute.internal                    @gmail.com
```

```
[root@                    ~]# vim /etc/mail/sendmail.mc
```

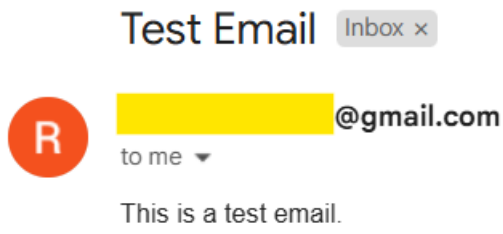
```
dnl ###MAILER(smtp)dnl
dnl ###MAILER(procmail)dnl
dnl ###MAILER(cyrusv2)dnl
define(`SMART_HOST',`[smtp.gmail.com]')dnl
define(`RELAY_MAILER_ARGS', `TCP $h 587')dnl
define(`ESMTP_MAILER_ARGS', `TCP $h 587')dnl
define(`confAUTH_OPTIONS', `A p')dnl
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
FEATURE(`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
```

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

```
[root@                ~]# systemctl restart sendmail.service
[root@                ~]# systemctl status sendmail.service
● sendmail.service - Sendmail Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/sendmail.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2025                5s ago
```

Sent a test mail and found sendmail was configured successfully with Gmail as shown in the screenshot attached below.

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



Monitoring SSL Expiration and docker container running status using Bash Shell Script

```
[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 $DAYS_LEFT days." | mail -s "30 days left for SSL to be expired"                @gmail.com
else
  if [ "$DAYS_LEFT" -eq 15 ]; then
    echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -s "15 days left for SSL to be expired"                @gmail.com
    exit 1
  fi
  if [ "$DAYS_LEFT" -eq 7 ]; then
    echo "Caution. SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -s "7 days left for SSL to be expired"                @gmail.com
  else
    echo "SSL certificate for $HOST is valid for $DAYS_LEFT days." | mail -s "SSL Expiration Notification"                @gmail.com
  fi
fi
```

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

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 -s "30 days left for SSL to be expired"
abc@gmail.com

else

 if [ "$DAYS_LEFT" -eq 15 ]; then

   echo "SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -s "15 days left for SSL to be expired"
abc@gmail.com

   exit 1

 fi

 if [ "$DAYS_LEFT" -eq 7 ]; then

   echo "Caution. SSL certificate for $HOST expires in $DAYS_LEFT days." | mail -s "7 days left for SSL to be expired"
abc@gmail.com

 else

   echo "SSL certificate for $HOST is valid for $DAYS_LEFT days." | mail -s "SSL Expiration Notification" abc@gmail.com

 fi

fi
```

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

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

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

I am also monitoring docker containers using the bash shell-script as provided below.

```
[root@              ~]# cat monitoring-docker-containers.sh
#!/bin/bash

CONTAINER_NAME_ONE=mysql
CONTAINER_NAME_TWO=bankapp
EMAIL_TO=                @gmail.com

if ! docker ps | grep $CONTAINER_NAME_ONE > /dev/null; then
    echo "Container $CONTAINER_NAME_ONE is not running on $(hostname)." | mail -s "$CONTAINER_NAME_ONE Container not running" $EMAIL_TO
fi
if ! docker ps | grep $CONTAINER_NAME_TWO > /dev/null; then
    echo "Container $CONTAINER_NAME_TWO is not running on $(hostname)." | mail -s "$CONTAINER_NAME_TWO Container not running" $EMAIL_TO
else
    echo "Both the containers $CONTAINER_NAME_ONE and $CONTAINER_NAME_TWO are running."
fi
```

```
[root@                    ~]# chmod +x monitoring-docker-containers.sh
```

cat monitoring-docker-containers.sh

#!/bin/bash

CONTAINER_NAME_ONE=mysql

CONTAINER_NAME_TWO=bankapp

EMAIL_TO=abc@gmail.com

if ! docker ps | grep $CONTAINER_NAME_ONE > /dev/null; then

echo "Container $CONTAINER_NAME_ONE is not running on $(hostname)." | mail -s "$CONTAINER_NAME_ONE Container not running" $EMAIL_TO

fi

if ! docker ps | grep $CONTAINER_NAME_TWO > /dev/null; then

echo "Container $CONTAINER_NAME_TWO is not running on $(hostname)." | mail -s "$CONTAINER_NAME_TWO Container not running" $EMAIL_TO

else

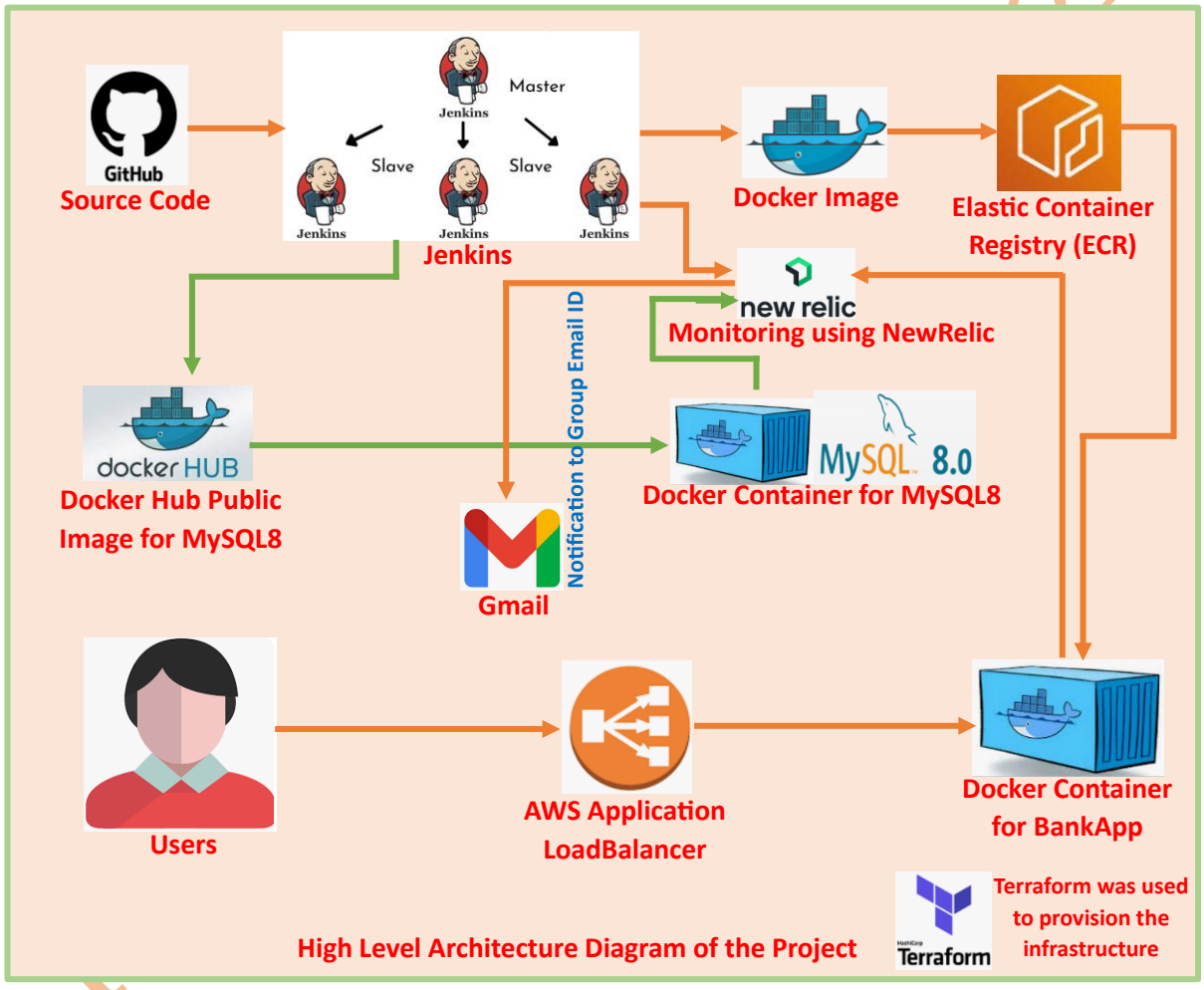
echo "Both the containers $CONTAINER_NAME_ONE and $CONTAINER_NAME_TWO are running."

fi

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

Every minute I am monitoring the running status of containers using bash shell-script through crontab if it is not running then an email will be triggered to the group Email ID.

Module-2: Approach with Application LoadBalancer to access BankApp



This project is similar as I explained in **Module 1**. However, in this project (**Module 2**) I used NewRelic as Monitoring and Log Aggregation Tool and instead of Nginx Server as a Reverse Proxy I used Application LoadBalancer to access the Bank Application. I used Terraform to provision the infrastructure.

You can run the below commands to provision the infrastructure using Terraform.

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

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

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

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

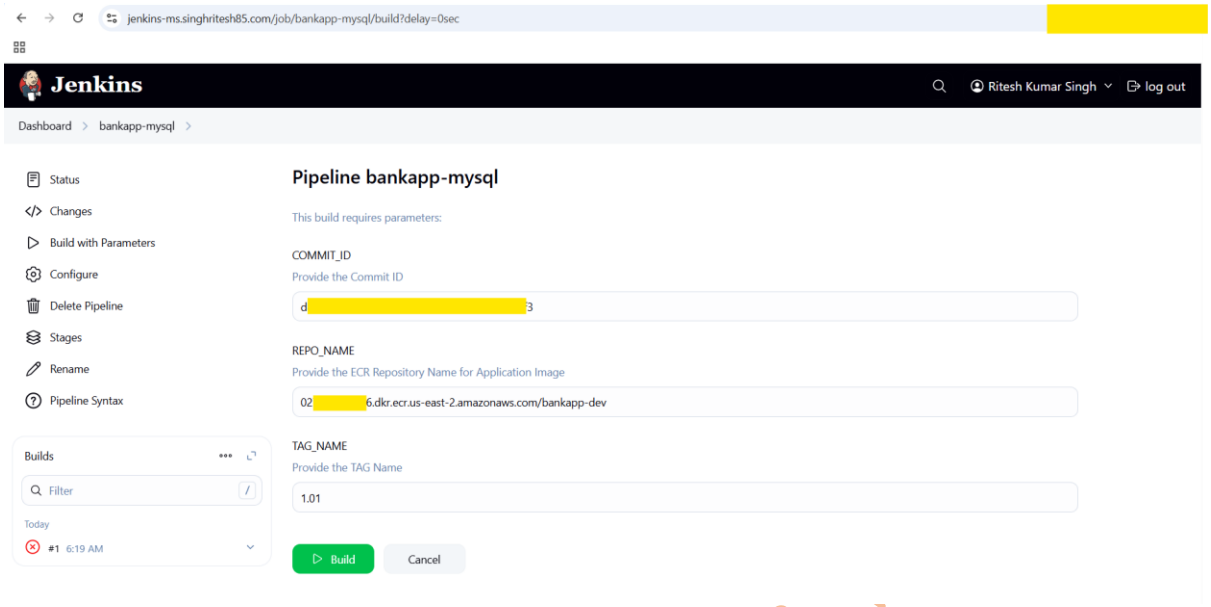
```
module.bankapp.null_resource.jenkins_master (remote-exec):   ✔ Logs Integration  (installed)

module.bankapp.null_resource.jenkins_master (remote-exec):   View your data at the link below:
module.bankapp.null_resource.jenkins_master (remote-exec):   https://onenr.io/

module.bankapp.null_resource.jenkins_master (remote-exec):   View your logs at the link below:
module.bankapp.null_resource.jenkins_master (remote-exec):   https://onenr.io/

module.bankapp.null_resource.jenkins_master (remote-exec):   --------------------

module.bankapp.null_resource.jenkins_master (remote-exec): ● newrelic-infra.service - New Relic Infrastructure Agent
module.bankapp.null_resource.jenkins_master (remote-exec):    Loaded: loaded (/etc/systemd/system/newrelic-infra.service; enabled; vendor preset: disabled)
module.bankapp.null_resource.jenkins_master (remote-exec):    Active: active (running) since Mon 2025
module.bankapp.null_resource.jenkins_master (remote-exec):  Main PID: 10764 (newrelic-infra-)
module.bankapp.null_resource.jenkins_master (remote-exec):    Memory: 30.2M (limit: 1.0G)
module.bankapp.null_resource.jenkins_master (remote-exec):    CGroup: /system.slice/newrelic-infra.service
module.bankapp.null_resource.jenkins_master (remote-exec):            ├─10764 /usr/bin/newrelic-...
module.bankapp.null_resource.jenkins_master (remote-exec):            ├─10774 /usr/bin/newrelic-...
module.bankapp.null_resource.jenkins_master (remote-exec):            └─10947 /opt/fluent-bit/bi...
module.bankapp.null_resource.jenkins_master: Creation complete after 3m30s [id=]

Apply complete! Resources: 78 added, 0 changed, 0 destroyed.

Outputs:

ecr_ec2_private_ip_alb_dns = {
  "BankApp_ALB_DNS_Name" = "bankapp-.us-east-2.elb.amazonaws.com"
  "EC2_Instance_Docker_Server_Private_IP_Address" = "10."
  "EC2_Instance_Jenkins_Master_Server_Private_IP_Address" = "10."
  "EC2_Instance_Jenkins_Slave_Server_Private_IP_Address" = "10."
  "Jenkins_ALB_DNS_Name" = "jenkins-ms-.us-east-2.elb.amazonaws.com"
  "registry_id" = [
    "02      6",
  ]
  "repository_url" = [
    "02      6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev",
  ]
}
```

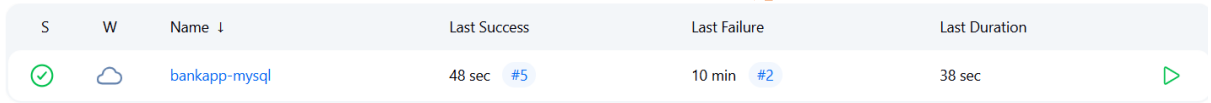
I configured Jenkins, added Jenkins Slave Node, and created credentials in Jenkins as explained in the **Module-1**. Apart from the credentials which I created in **Module-1**, I create another Jenkins Credentials of type **secret text** and named as **newrelic_license_key** as shown in the screenshot attached below.

Then I created the Jenkins Job to deploy the Docker Containers. For your reference I kept the Jenkinsfile in GitHub Repo https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-NewRelic.git. For docker containers deployment I used docker-compose because it leverages me to create multiple containers at a single time. The Jenkins Job I created with the name of **bankapp-mysql**, is as shown in the screenshot attached below after its successful execution. Before running I established the password less authentication between **jenkins-slave** and **docker-server** as shown in the screenshot attached below.

```
[root@jenkins-slave ~]# su - jenkins
[jenkins@jenkins-slave ~]$ ssh-keygen
Generating public/private rsa key pair.

Enter file in which to save the key (/home/jenkins/.ssh/id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jenkins/.ssh/id_rsa.
Your public key has been saved in /home/jenkins/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:u                                                    0 jenkins@jenkins-slave
The key's randomart image is:
+---[RSA 2048]----+
|.                |
|+                |
|+                |
|.                |
|.                |
|o                |
|.                |
|                 |
|.. .   .+.o . .. |
+----[SHA256]-----+
[jenkins@jenkins-slave ~]$ ssh-copy-id docker-admin@10.
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/jenkins/.ssh/id_rsa.pub"
The authenticity of host '10.        (10.        )' can't be established.
ECDSA key fingerprint is SHA256:J                                        8.
ECDSA key fingerprint is MD5:c                                           c.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
docker-admin@10.        's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'docker-admin@10.        '"
and check to make sure that only the key(s) you wanted were added.
```

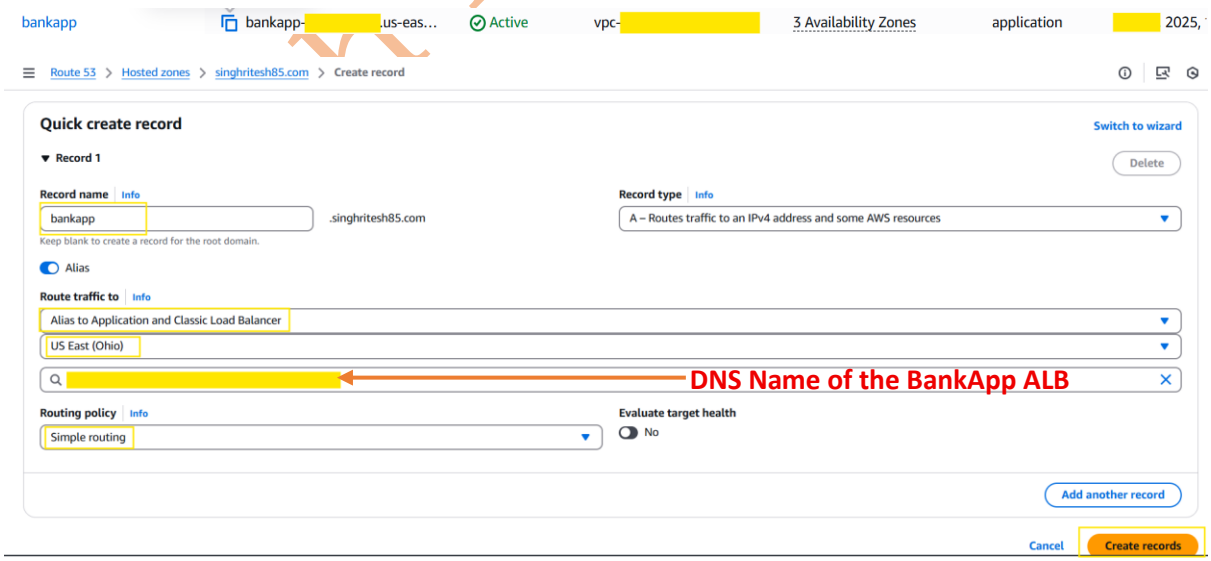
To the run the Jenkins Job I had to provide the string parameters as shown in the screenshot attached below.



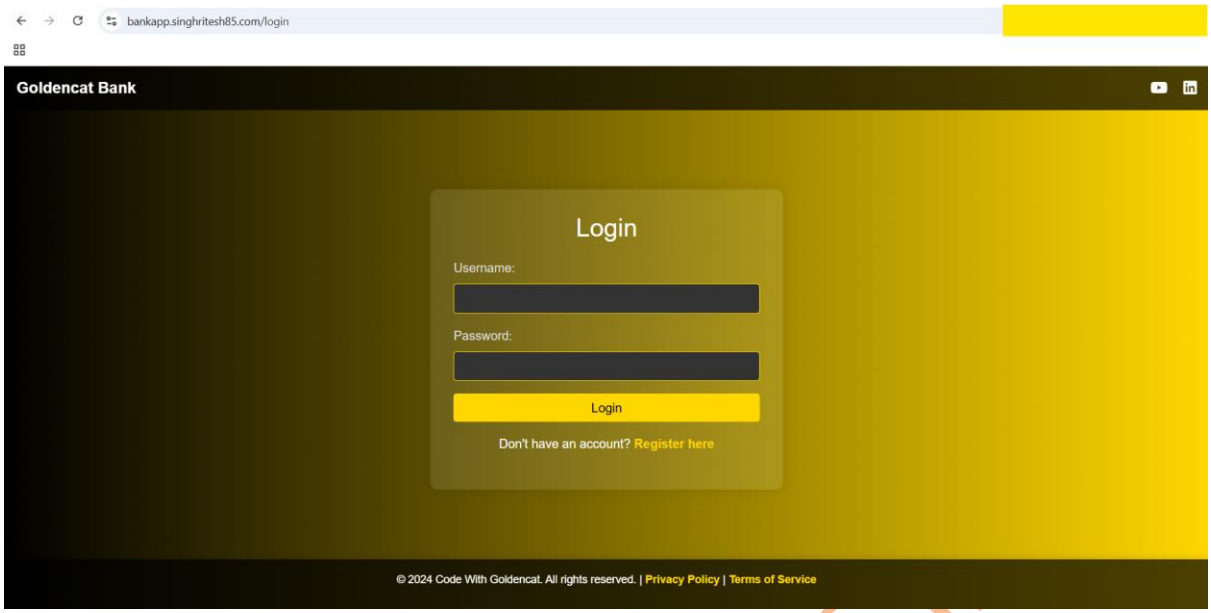



```
[docker-admin@docker-server ~]$ docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED          STATUS          PORTS
                NAMES
9f5    2c40d   02         6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev:1.01   "/__cacert_entrypoin…"   5 minutes ago   Up 5 minutes   0.0.0.0:8080->8080/tcp,
 :::8080->8080/tcp   bankapp
946    5b990   mysql:8.0.0                                            "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   0.0.0.0:3306->3306/tcp,
 :::3306->3306/tcp   mysql
```

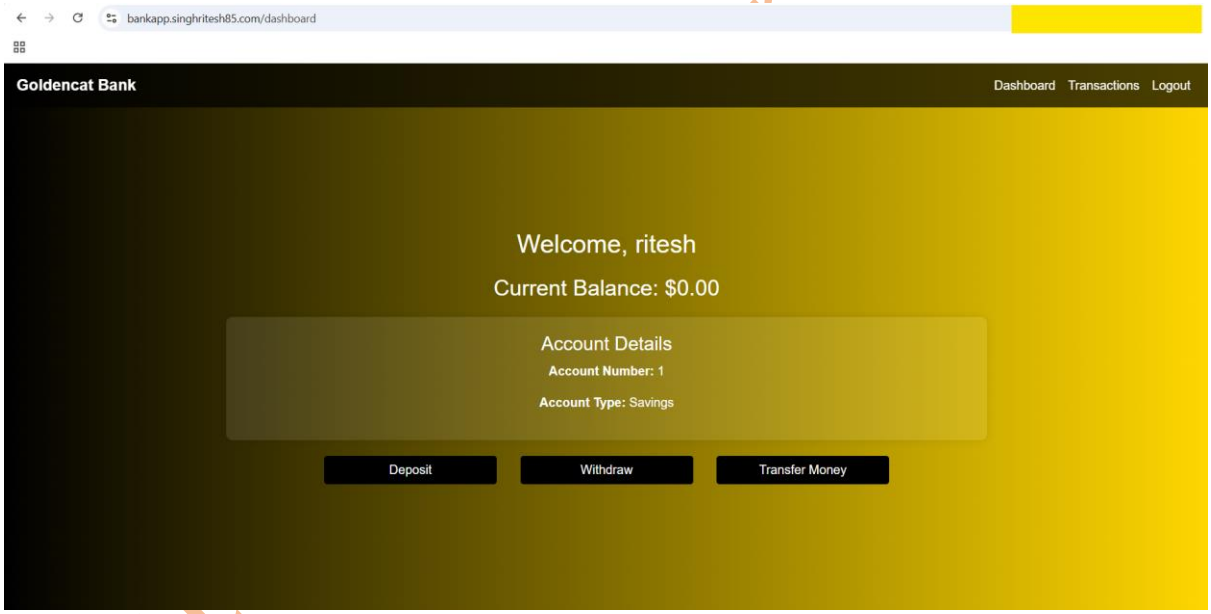
I did the entry of bankapp ALB DNS Name in Route53 to create the Recode Set of A-Type with Alias as shown in the screenshot attached below.



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



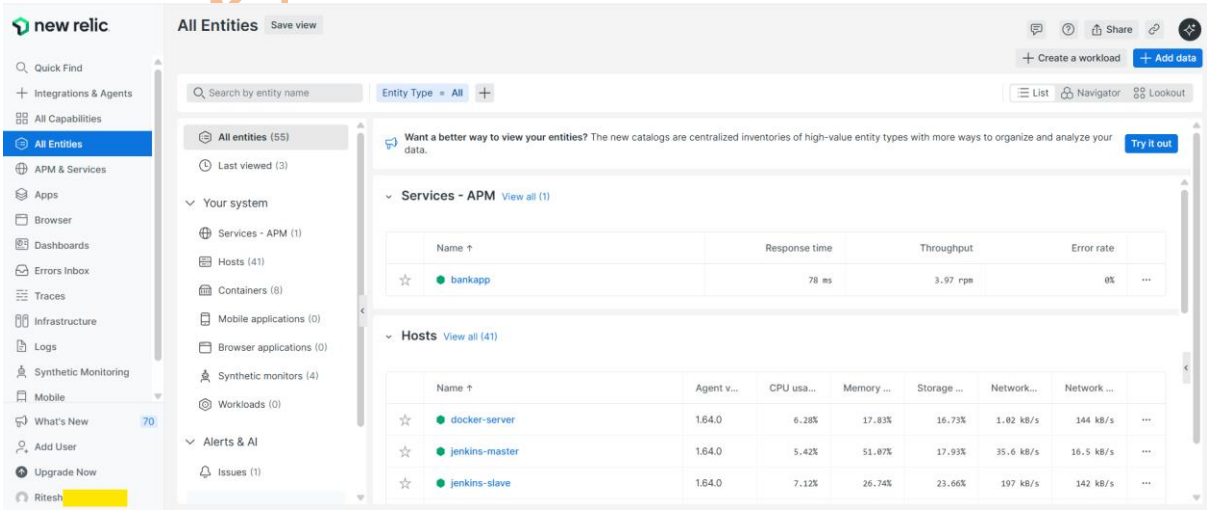
I created a user to login and same user got created in mysql container as shown in the screenshot attached below.



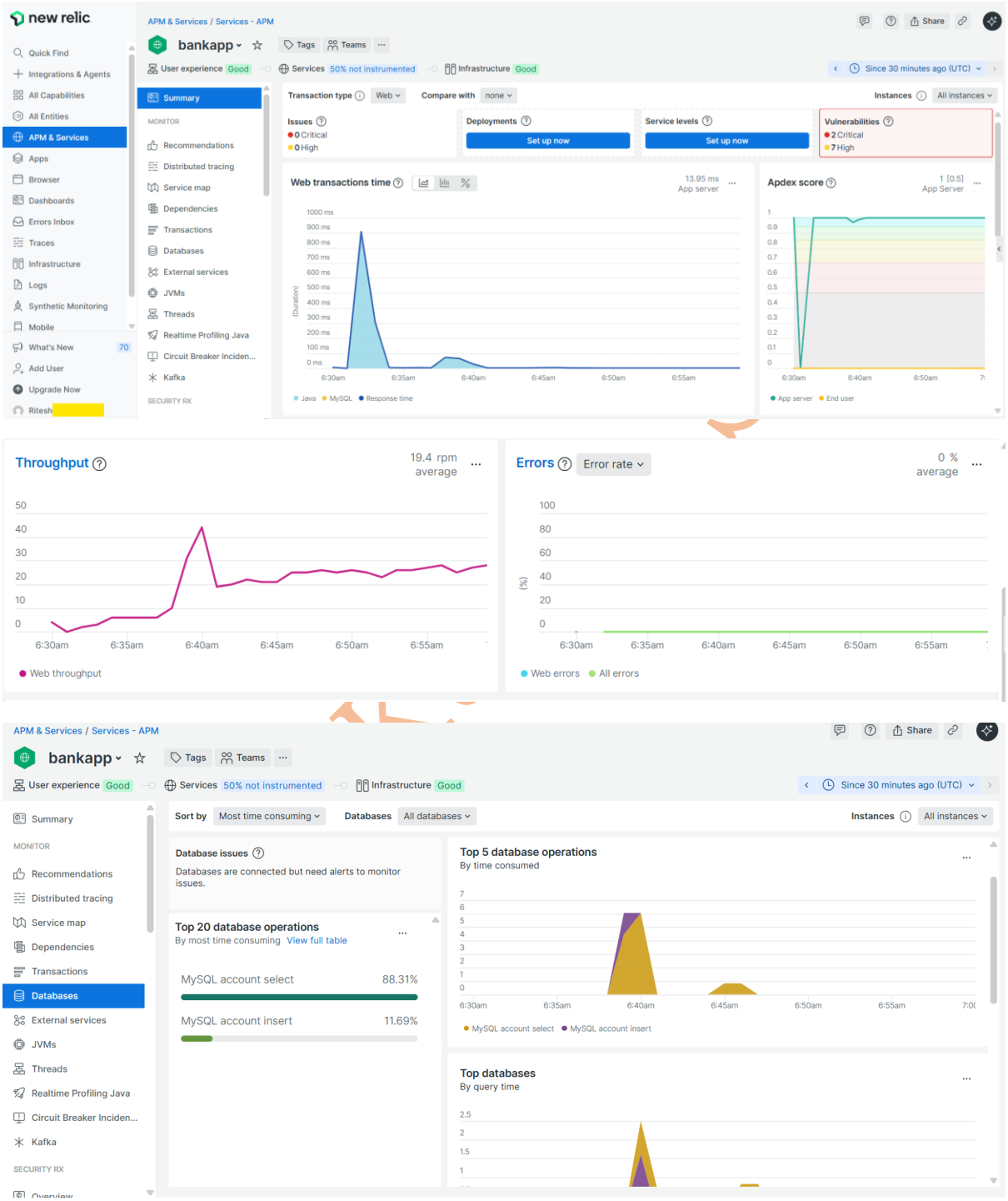
```
[docker-admin@docker-server ~]$ docker ps
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED         STATUS         PORTS                    NAMES
9f5[redacted]c40d   02[redacted]6.dkr.ecr.us-east-2.amazonaws.com/bankapp-dev:1.01   "/__cacert_entrypoin…"   5 minutes ago   Up 5 minutes   0.0.0.0:8080->8080/tcp,
 :::8080->8080/tcp   bankapp
946[redacted]b990   mysql:8.0.0                                              "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   0.0.0.0:3306->3306/tcp,
 :::3306->3306/tcp   mysql
[docker-admin@docker-server ~]$ docker exec -it 946[redacted]990  bash
root@946[redacted]90:/# mysql -h localhost -u root --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.0-dmr MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

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> show databases;
+--------------------+
| Database           |
+--------------------+
| bankappdb          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

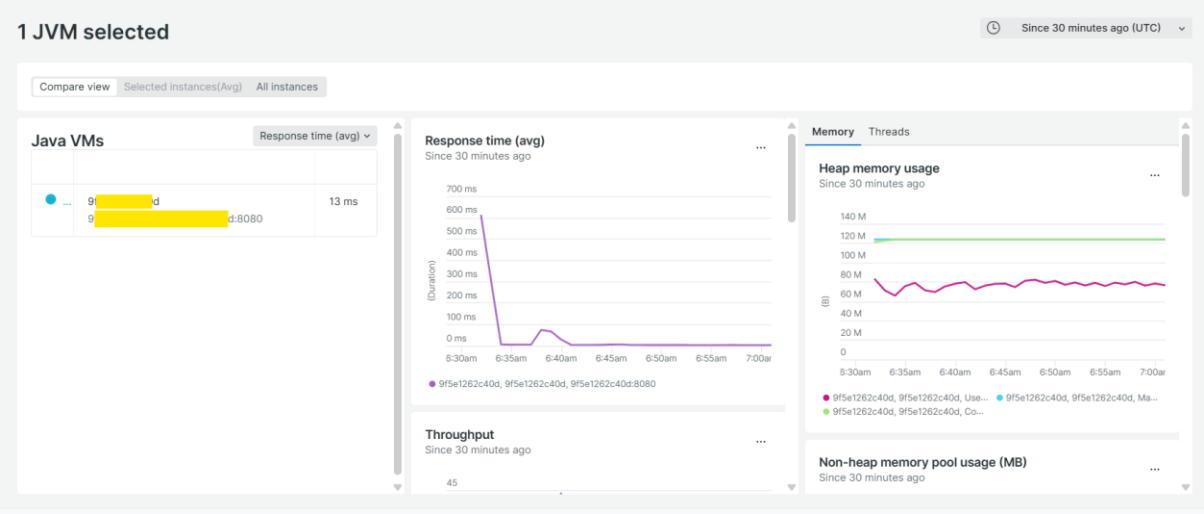
```
mysql> use bankappdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------+
| Tables_in_bankappdb |
+----------------------+
| account              |
| transaction          |
+----------------------+
2 rows in set (0.00 sec)

mysql> select * from account;
+----+---------+--------------------------------------------------------------+----------+
| id | balance | password                                                     | username |
+----+---------+--------------------------------------------------------------+----------+
|  1 |    0.00 | $[redacted]C | ritesh   |
+----+---------+--------------------------------------------------------------+----------+
1 row in set (0.00 sec)
```

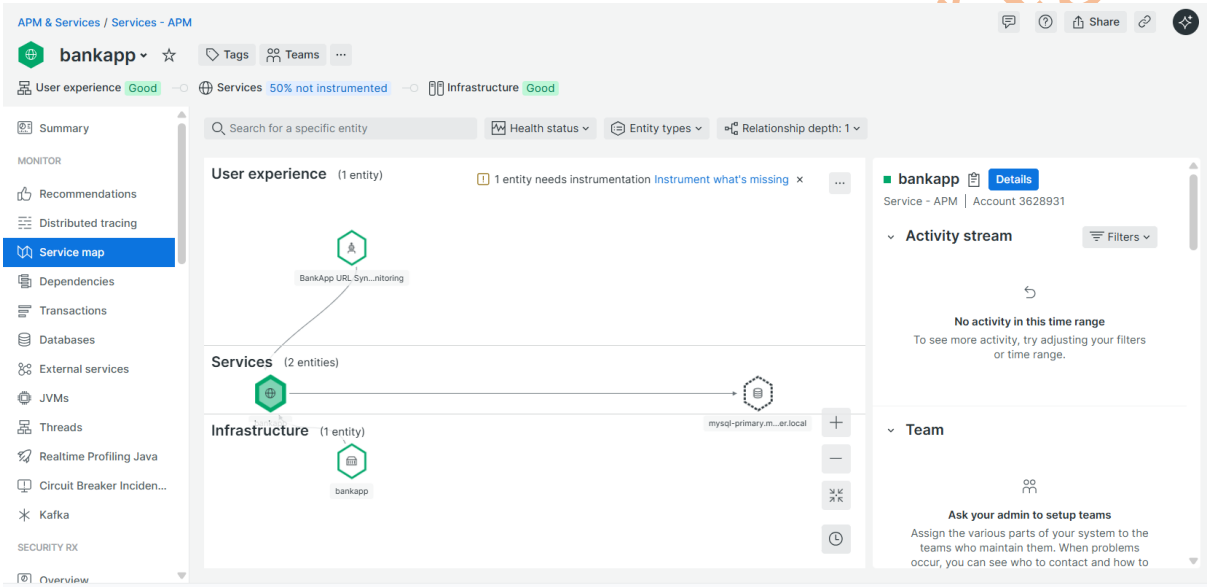
The NewRelic Web Console is as shown in the screenshot attached below.



I am monitoring CPU Utilization, Disk Utilization and Memory Utilization of Docker-Server, Jenkins-Master, and Jenkins-Slave. I performed Application Performance Monitoring (APM) using NewRelic as shown in the screenshot attached above. I kept the newrelic.yml in the GitHub Repo https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-NewRelic.git.







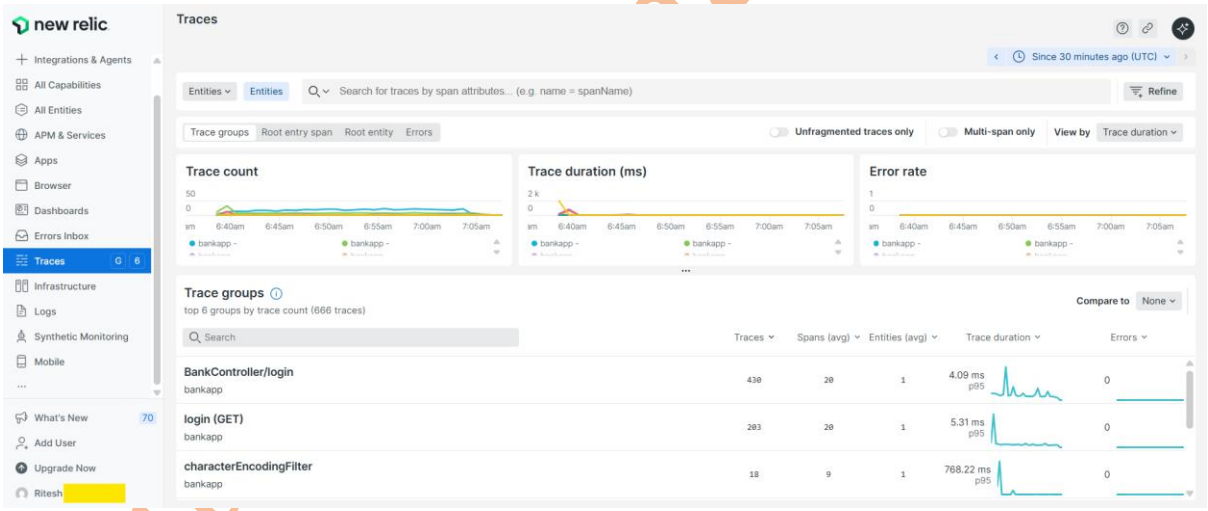
The **Jenkinsfile** is as shown in the screenshot attached below.

```
def DOCKER_SERVER="10.XX.X.XXX"  //Provide Docker Server Public IP/Private IP Address.
pipeline {
   agent {
      label{
         label "Slave-1"
         customWorkspace "/home/jenkins/bankapp"
      }
   }
   environment{
      JAVA_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
      PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
      MYSQL_ROOT_PASSWORD=credentials('mysql_root_password')
      MYSQL_DATABASE=credentials('mysql_database')
      newrelic_license_key=credentials('newrelic_license_key')
   }
   parameters {
      string(name: 'COMMIT_ID', defaultValue: '', description: 'Provide the Commit ID')
      string(name: 'REPO_NAME', defaultValue: '', description: 'Provide the ECR Repository Name for Application Image')
      string(name: 'TAG_NAME', defaultValue: '', description: 'Provide the TAG Name')
   }
   stages{
      stage("clone-code"){
         steps{
            cleanWs()
            checkout scmGit(branches: [[name: '${COMMIT_ID}']], extensions: [], userRemoteConfigs: [[credentialsId: 'github-cred', url: 'https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-NewRelic-Private.git']])
         }
      }
      stage("Build and DockerImage"){
         steps{
```

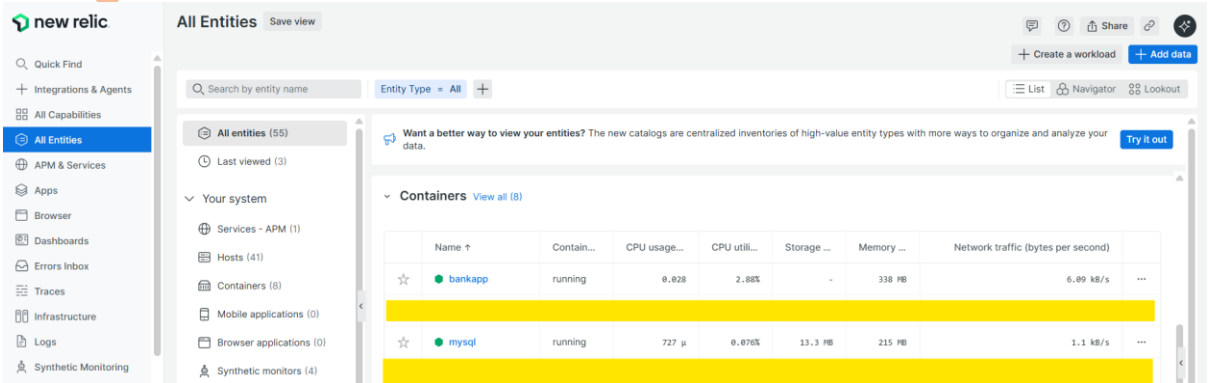
```
sh 'mvn clean install'
            sh 'wget https://download.newrelic.com/newrelic/java-agent/newrelic-agent/current/newrelic-java.zip'
            sh 'unzip newrelic-java.zip'
            sh "yes|cp ./newrelic.yml ./newrelic/"
            sh "sed -i -e ''s/##MYSQL_ROOT_PASSWORD##/${MYSQL_ROOT_PASSWORD}/g'' ./.env"
            sh "sed -i -e ''s/##MYSQL_DATABASE##/${MYSQL_DATABASE}/g'' ./.env"
            sh "sed -i -e ''s@##REPO_NAME##@${REPO_NAME}@g'' ./.env"
            sh "sed -i -e ''s/##TAG_NAME##/${TAG_NAME}/g'' ./.env"
            sh "sed -i -e ''s/'<%= license_key %>'/$newrelic_license_key/g'' ./newrelic/newrelic.yml"
            sh "sed -i -e ''s/'app_name: My Application'/'app_name: bankapp'/g'' ./newrelic/newrelic.yml"
            sh "echo -e ''enable_process_metrics: true'' >> ./newrelic/newrelic.yml"
            sh "echo -e ''status_server_enabled: true'' >> ./newrelic/newrelic.yml"
            sh "echo -e ''status_server_port: 18003'' >> ./newrelic/newrelic.yml"
            sh "echo -e ''license_key: $newrelic_license_key'' >> ./newrelic/newrelic.yml"
            sh "echo -e ''custom_attributes:'' >> ./newrelic/newrelic.yml"
            sh "echo -e '  nr_deployed_by: bankapp' >> ./newrelic/newrelic.yml"
            sh "scp -rv Dockerfile-Project-1 docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv target docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv docker-compose.yaml docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv .env docker-admin@${DOCKER_SERVER}:~/"
            sh "scp -rv newrelic docker-admin@${DOCKER_SERVER}:~/"
            sh 'docker system prune -f'
            sh "docker build -t ${REPO_NAME}:${TAG_NAME} -f Dockerfile-Project-1 ."
            sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 027330342406.dkr.ecr.us-east-2.amazonaws.com'
            sh "docker push ${REPO_NAME}:${TAG_NAME}"
        }
    }
    stage("Deployment"){
        steps{
```

```
        sh """ssh docker-admin@$DOCKER_SERVER <<-EOF
        docker-compose -p bankapp down     ### do not use docker-compose down -v here oherwise docker volume will be deleted.
        docker system prune -f
        sudo docker-compose -p bankapp up -d
        exit
        EOF
        """
      }
    }
  }
}
```

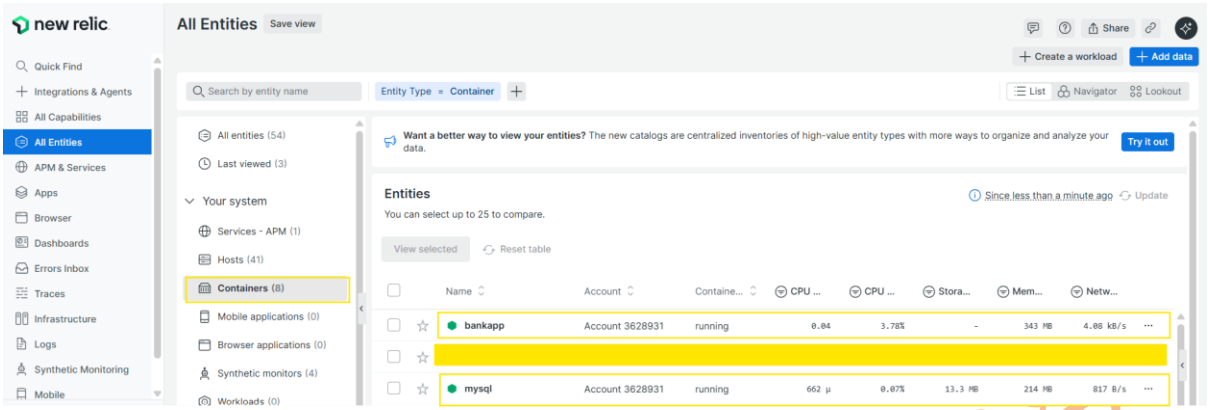
The traces can be found as shown in the screenshot attached below.



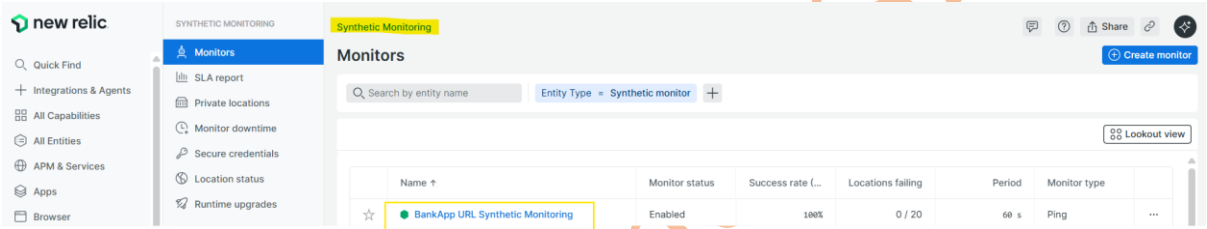
The Dockerfile for this project is kept in the GitHub Repo https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-NewRelic.git.



The docker containers I am monitoring using the NewRelic as shown in the screenshot attached below.



I performed synthetic monitoring of Bank Application URL using NewRelic as shown in the screenshot attached below.



Destination, Workflow, Alert Policy and Alert Condition had been created using the Terraform Script present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-Docker-Monitoring-LogAggregation-using-CloudWatch.git at the path **terraform-jenkins-docker-alb**.

Log Aggregation using NewRelic

For Log Aggregation in NewRelic you need to check the file **/etc/newrelic-infra/logging.d/logging.yml** as shown in the screenshot attached below.

```
[root@docker-server ~]# cat /etc/newrelic-infra/logging.d/logging.yml
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
```

For aggregating docker logs into NewRelic I did the configuration change in the file logging.yml as shown in the screenshot attached below.

```
[root@docker-server ~]# vim /etc/newrelic-infra/logging.d/logging.yml
```

```
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
  - name: containers
    file: /var/lib/docker/containers/*/*.log
```

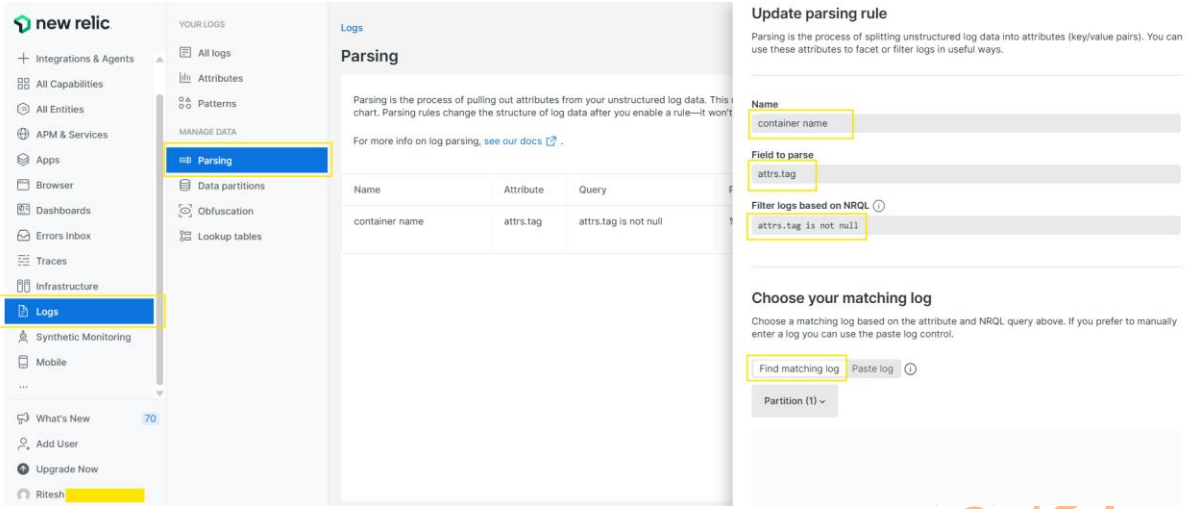
Then restarted the newrelic-infra service as shown in the screenshot attached below.

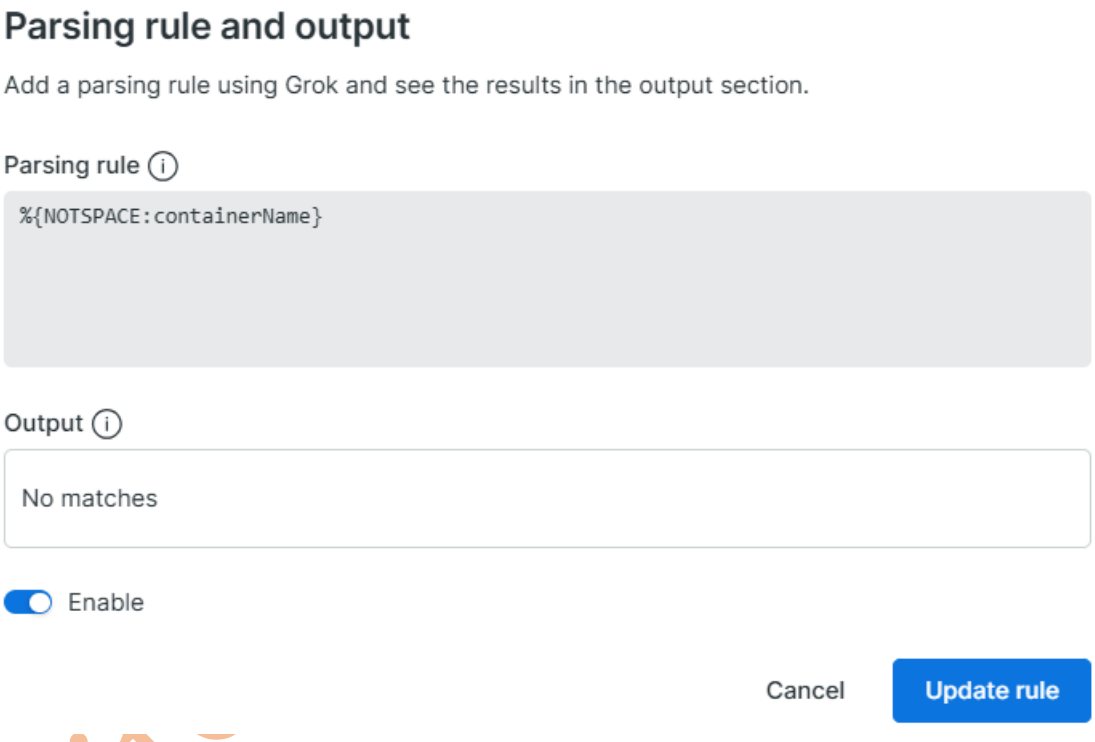
```
[root@docker-server ~]# systemctl restart newrelic-infra.service
[root@docker-server ~]# systemctl status newrelic-infra.service
● newrelic-infra.service - New Relic Infrastructure Agent
   Loaded: loaded (/etc/systemd/system/newrelic-infra.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2025
```

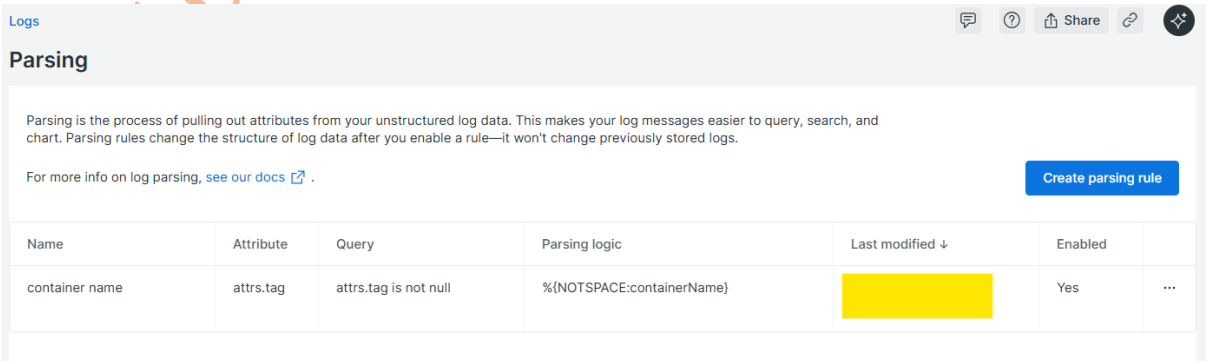
In docker-compose.yaml I mentioned below lines.

```
51  x-default-logging: &logging     ### For extracting Containers Logs
52    driver: "json-file"
53    options:
54      max-size: "5m"
55      max-file: "2"
56      tag: "{{.Name}}"
```

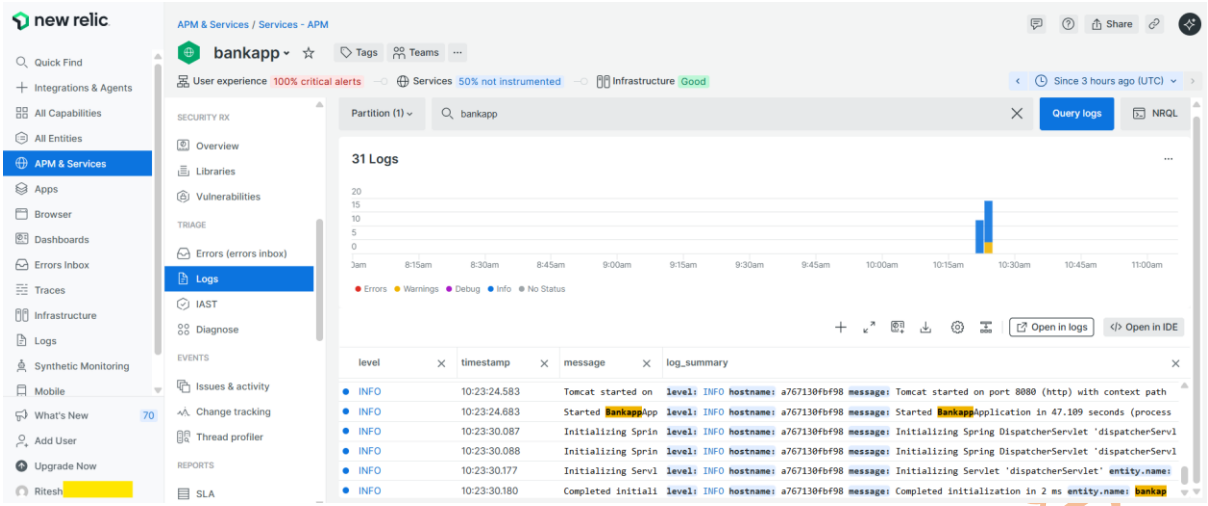
Then I created the parsing rule in newrelic logs as shown in the screenshot attached below.







Finally, you will be able to see the container logs inside the container tab in NewRelic. You can also see the logs from logs tab in NewRelic and in the logs tab present under APM as shown in the screenshot attached below.



Source Code: https://github.com/singhritesh85/Bank-App-Monitoring-Logging-using-NewRelic.git

GitHub Repo: https://github.com/singhritesh85/DevOps-Project-BankApp-Docker-Monitoring-LogAggregation-using-CloudWatch.git

Terraform Script: https://github.com/singhritesh85/DevOps-Project-BankApp-Docker-Monitoring-LogAggregation-using-CloudWatch.git

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