# Docker – The Serverless Way (AWS, Azure, and GCP)


**By Ritesh Kumar Singh**

**Email Address: -** riteshkumarsingh9559@gmail.com

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

**GitHub: -** https://github.com/singhritesh85



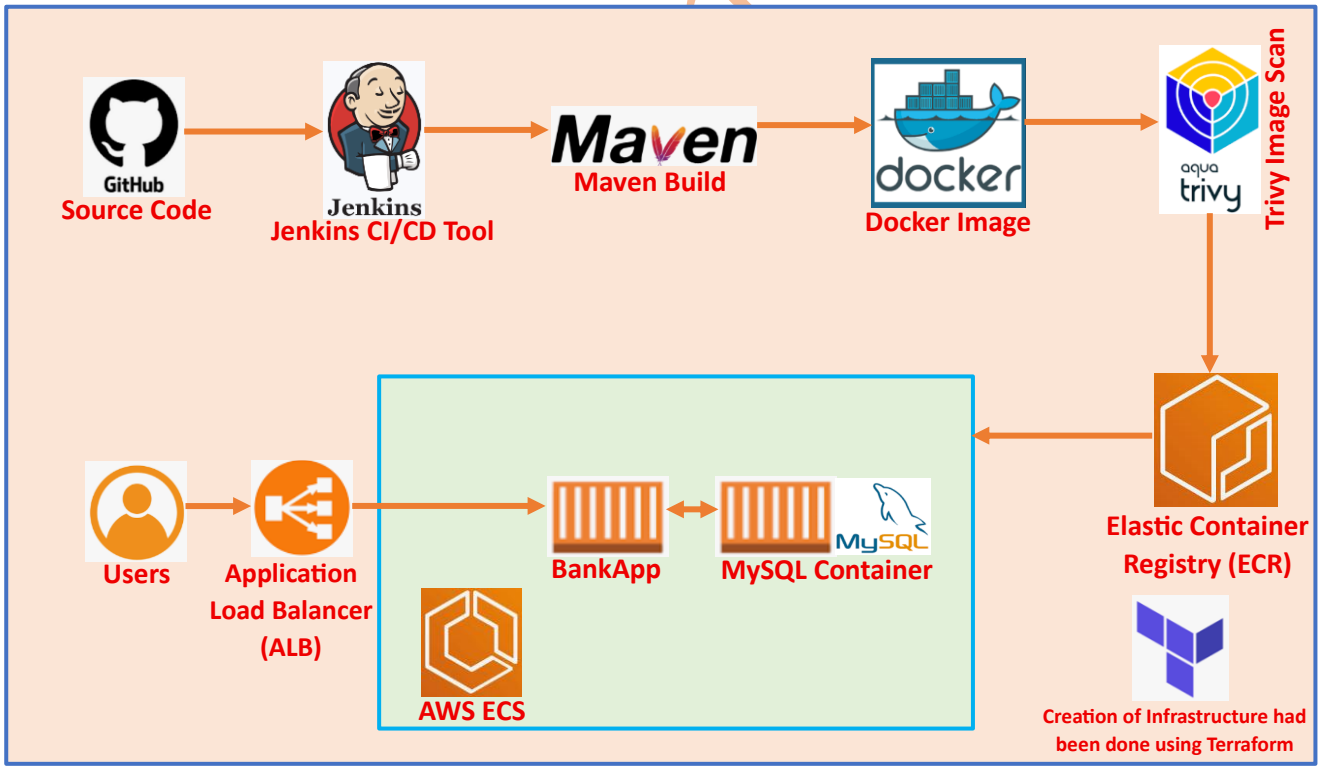
या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

## Introduction

Before starting the discussion, a reader should know why do we move to serverless or what was the need to introduce the serverless concept. In industry when you use light weight applications or microservices, you often use docker container/kubernetes pod. You can either opt the option of creating Kubernetes Cluster on Cloud based VMs (EC2 Instances) then create the pods on the top of it, install Docker on the top of Cloud VMs (EC2 Instances) then run your application inside the container or use the serverless Kubernetes Cluster or Docker Containers. There are some advantages and disadvantages of using the serverless environment. The advantage include organisation do not need Engineers specialised for handling the Docker/Kubernetes environment So it will save the cost as the underlying infrastructure will be taken care the Cloud provider and organisation do not need to worry about it. It also has some disadvantages which includes limited control over underlying hardware so in your project when you need it, you will be unable to access it and you should be aware of the downtime of the Application (may be due to unavailability of the Cloud based service of Docker/Kubernetes) and hence the related SLA. With Serverless you can use predefined options for resources which means if your application needs only 1.5GB of memory then you cannot use it but you need to use the option provided by the cloud provider. It is advisable to use High Availability (HA) across different Availability Zones or different regions when you are going with cloud environment.

## Module-1

### Docker – The Serverless Way (AWS ECS)




This DevOps Project deals with creation of infrastructure using Terraform and deployment of BankApp using Jenkins CI/CD Pipeline. The creation of MySQL and BankApp containers had been done using Terraform through Jenkins Job. For your reference in GitHub Repo https://github.com/singhritesh85/terraform-docker-the-serverless-way.git I kept the Terraform script at the path **terraform-route53-hosted-zone-with-acm-certificate** to create the Route53 Hosted Zone

and SSL Certificate in Aws Certificate Manager. As it is a well-known fact that you will not get the opportunity to create the Hosted Zone daily So I had provided here Terraform script at the separate path to create the Route53 Hosted Zone and SSL Certificate in AWS Certificate Manager which you can use when needed. I had created the Jenkins Master, Jenkins Slave, ECR Repository and accessed Jenkins Master using the Application LoadBalancer, this entire infrastructure had been created using Terraform Script which is kept at the path **terraform-jenkins-master-slave-ecr**. The terraform script present at the path **terraform-ecs-task-definition-service-ebs-alb** had been used to create the ECS Cluster, Task Definition and Service. I ran Terraform script to create ECS Cluster, Task Definition and Service using Jenkins Job, in this Jenkins Job I provided the URI of ECR as the default vault of Jenkins Declarative Pipeline string parameter and you need to pass the TAG NUMBER for the Docker Image as well. So, whenever a new release comes you will be creating a Docker Image and using this Docker Image you will create a new ECS task definition revision number and hence the ECS Service using revision number of the ECS Task Definition.

Commands as written below had been used to create infrastructure using Terraform.

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

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

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

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

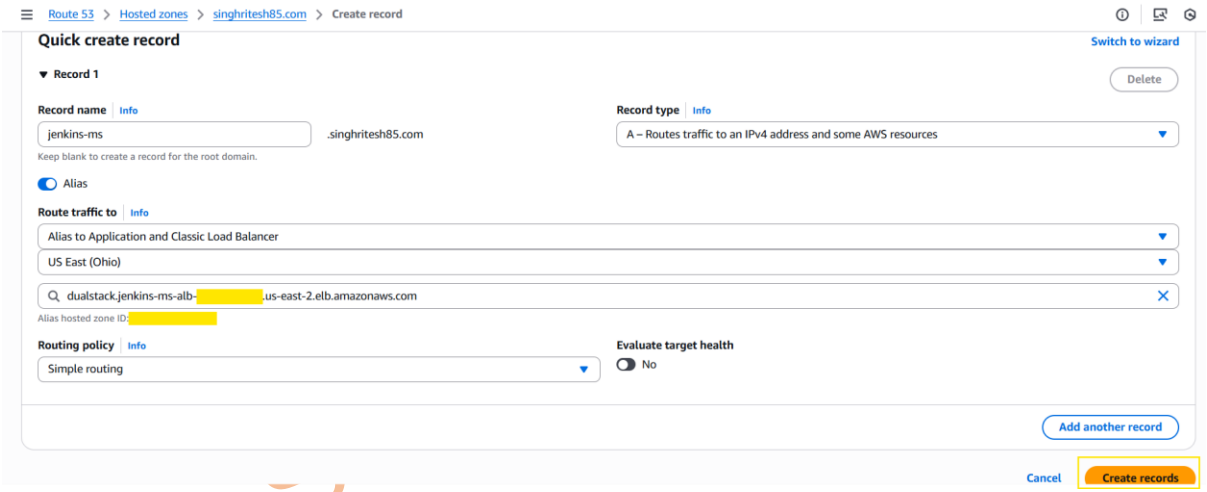
After running the Terraform Script update your domain provider with the generated Nameserver of the Route53 Hosted Zone and hence resources will be created.

```
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Creation complete after 31s [id=                    ]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Creating...
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_route53_record.record_cert_validation: Creation complete after 31s [id=                                                                    s
inghritesh85.com._CNAME]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Creating...
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m40s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [00m50s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m00s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m10s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m20s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m30s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m40s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [01m50s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [02m00s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [02m10s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Still creating... [02m20s elapsed]
module.route53_hosted_zone.aws_acm_certificate_validation.acm_certificate_validation: Creation complete after 2m24s [id=                                  UTC]

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

Outputs:

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

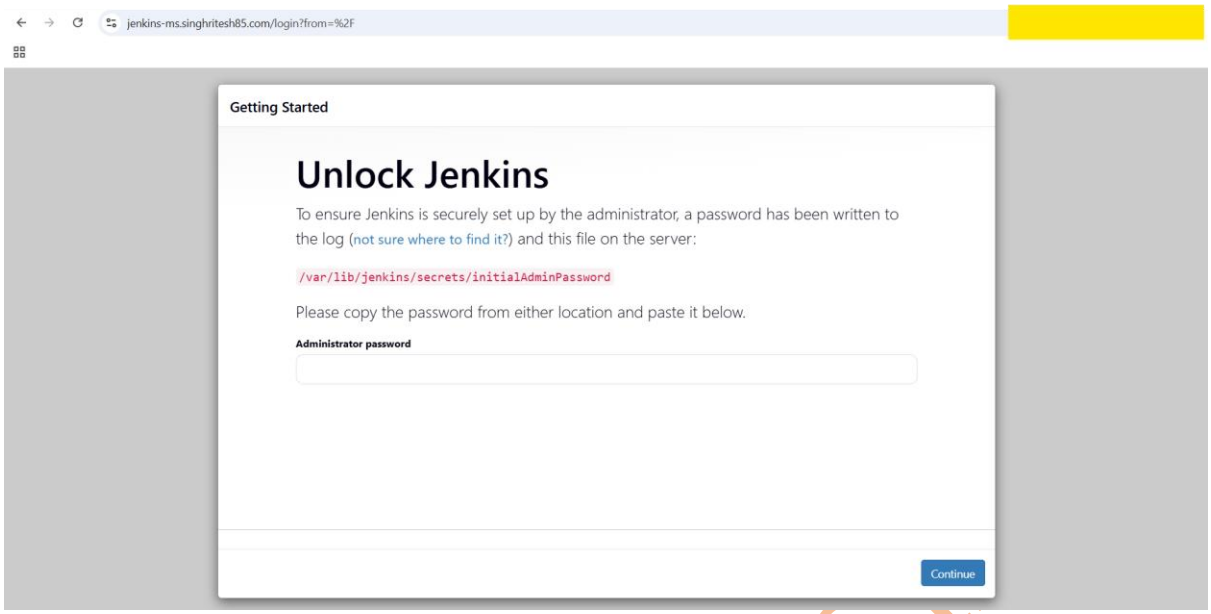
Use this generated certificate ARN in terraform script present at the path **terraform-jenkins-master-slave-ecr** and **terraform-ecs-task-definition-service-ebs-alb** (in the file terraform.tfvars). The result of the screenshot after successful execution of the Terraform Script is as shown below.

```
module.jenkins.aws_route_table.private_route_table_3: Creation complete after 1s [id=                    ]
module.jenkins.aws_route_table.private_route_table_1: Creation complete after 1s [id=                    ]
module.jenkins.aws_route_table_association.private_route_table_association_3: Creating...
module.jenkins.aws_route_table_association.private_route_table_association_1: Creating...
module.jenkins.aws_route_table.private_route_table_2: Creation complete after 1s [id=                    ]
module.jenkins.aws_route_table_association.private_route_table_association_2: Creating...
module.jenkins.aws_route_table_association.private_route_table_association_1: Creation complete after 0s [id=                    ]
module.jenkins.aws_route_table_association.private_route_table_association_3: Creation complete after 0s [id=                    ]
module.jenkins.aws_route_table_association.private_route_table_association_2: Creation complete after 0s [id=                    ]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m00s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m10s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m20s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m30s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m40s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [02m50s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [03m00s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Still creating... [03m10s elapsed]
module.jenkins.aws_lb.test-application-loadbalancer_alb: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:us-east-2:02        6:loadbalancer/ap
p/jenkins-ms-alb/daefca4af0115da6]
module.jenkins.aws_lb_listener.alb_listener_front_end_HTTPS_jenkins: Creating...
module.jenkins.aws_lb_listener.alb_listener_front_end_HTTP_jenkins: Creating...
module.jenkins.aws_lb_listener.alb_listener_front_end_HTTP_jenkins: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:02        6:listene
r/app/jenkins-ms-alb/                                ]
module.jenkins.aws_lb_listener.alb_listener_front_end_HTTPS_jenkins: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:02        6:listen
er/app/jenkins-ms-alb/                                ]

Apply complete! Resources: 40 added, 0 changed, 0 destroyed.

Outputs:

ECR_URI_jenkins_master_and_slave_ec2_private_ip_and_alb_dns_name = {
  "ecr_repository_uri" = "02        6.dkr.ecr.us-east-2.amazonaws.com/bankapp"
  "jenkins_alb_dns_name" = "jenkins-ms-alb-        .us-east-2.elb.amazonaws.com"
  "jenkins_master_and_slave_private_ip" = [
    "172.16.4.  ",
    "172.16.4.  ",
  ]
}
[root@                main]#
```

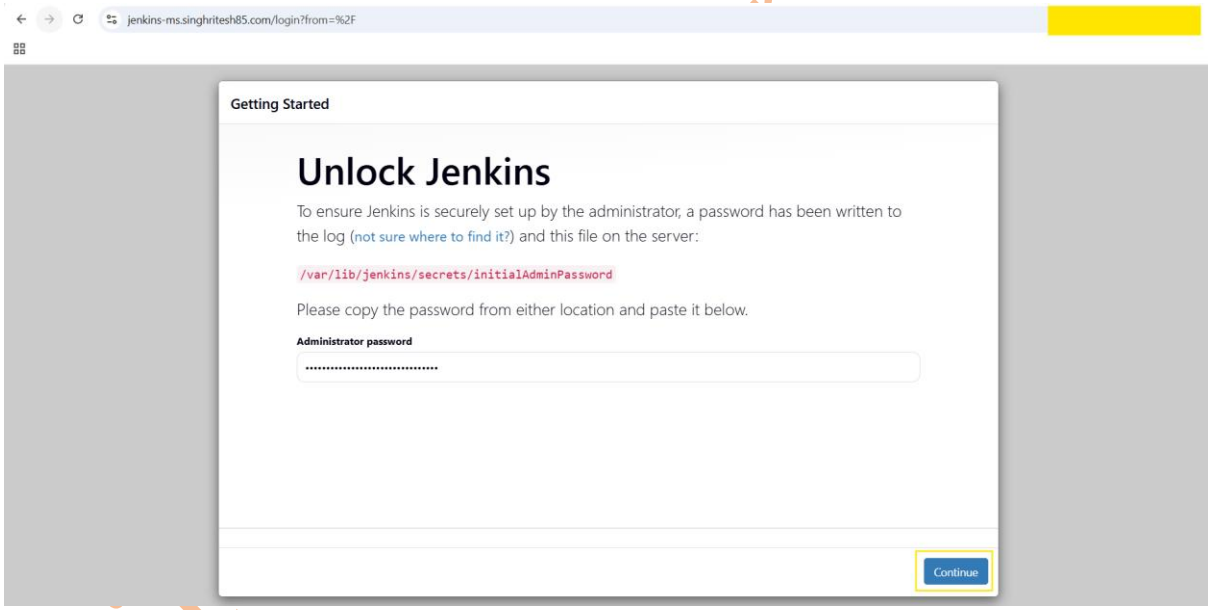
Do the entry for the generated ALB for Jenkins in Route 53 to create the Record Set of A-Type as shown in the screenshot attached below.

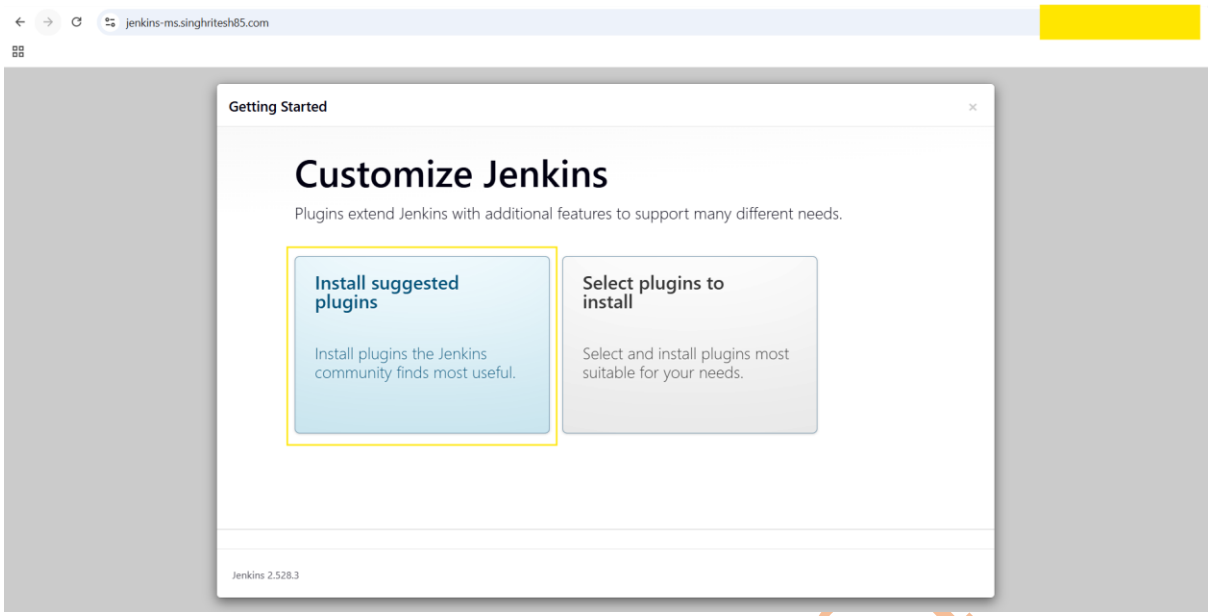


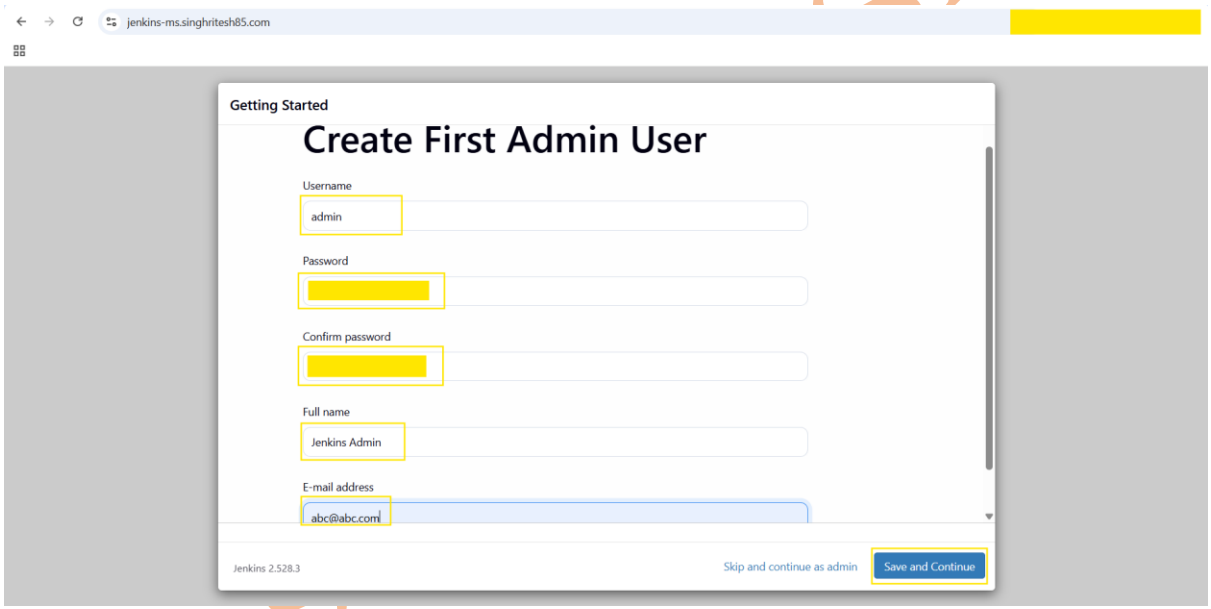

Then open the URL as shown in the screenshot attached below.



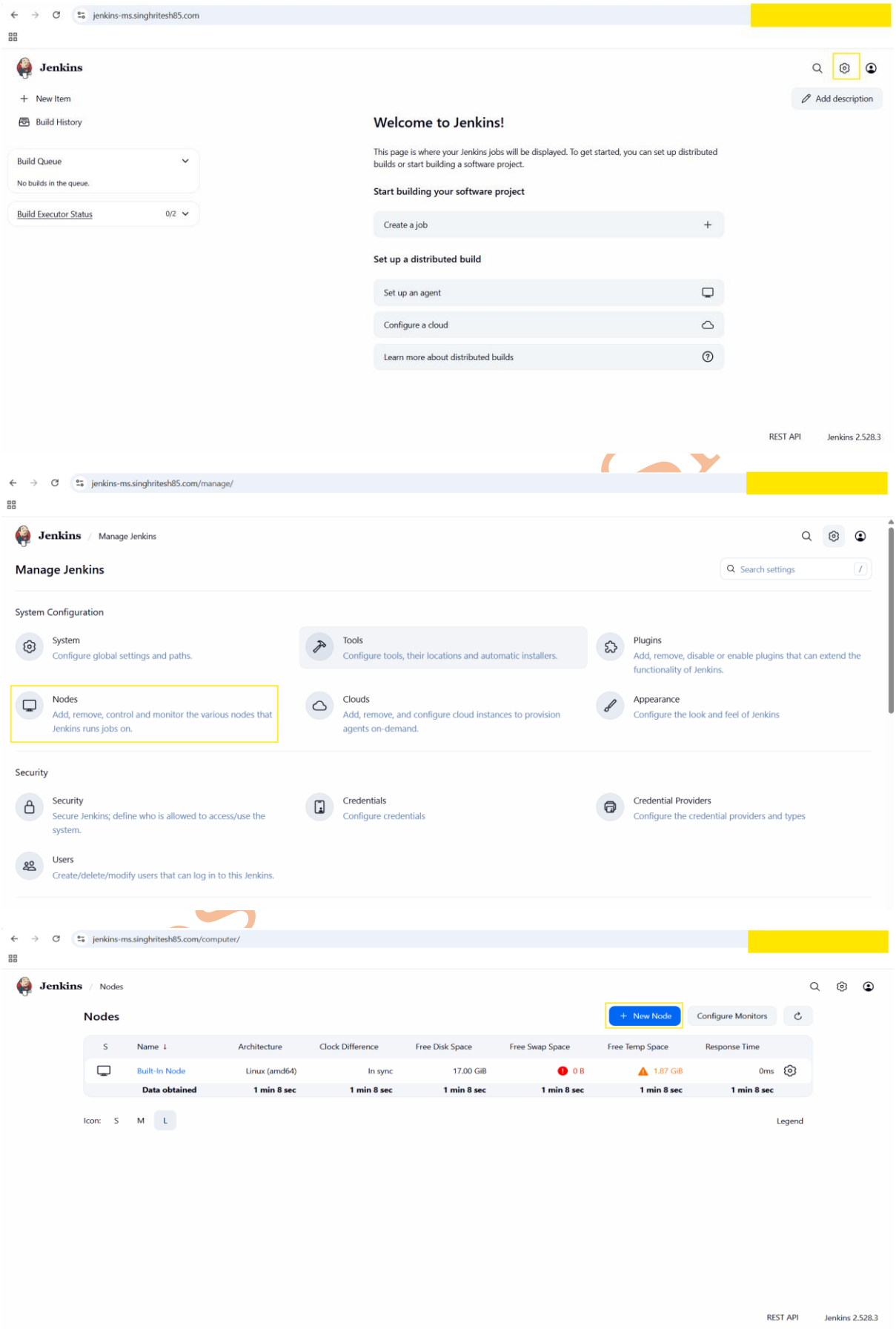

```
[root@jenkins-master ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
a83                                e69
```

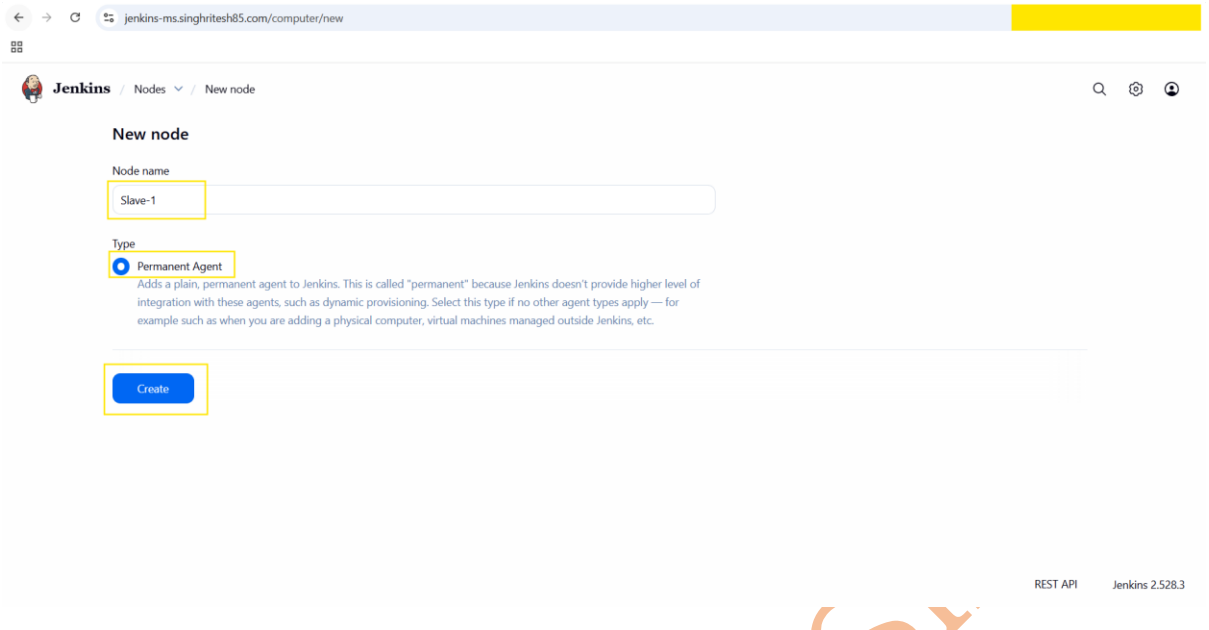


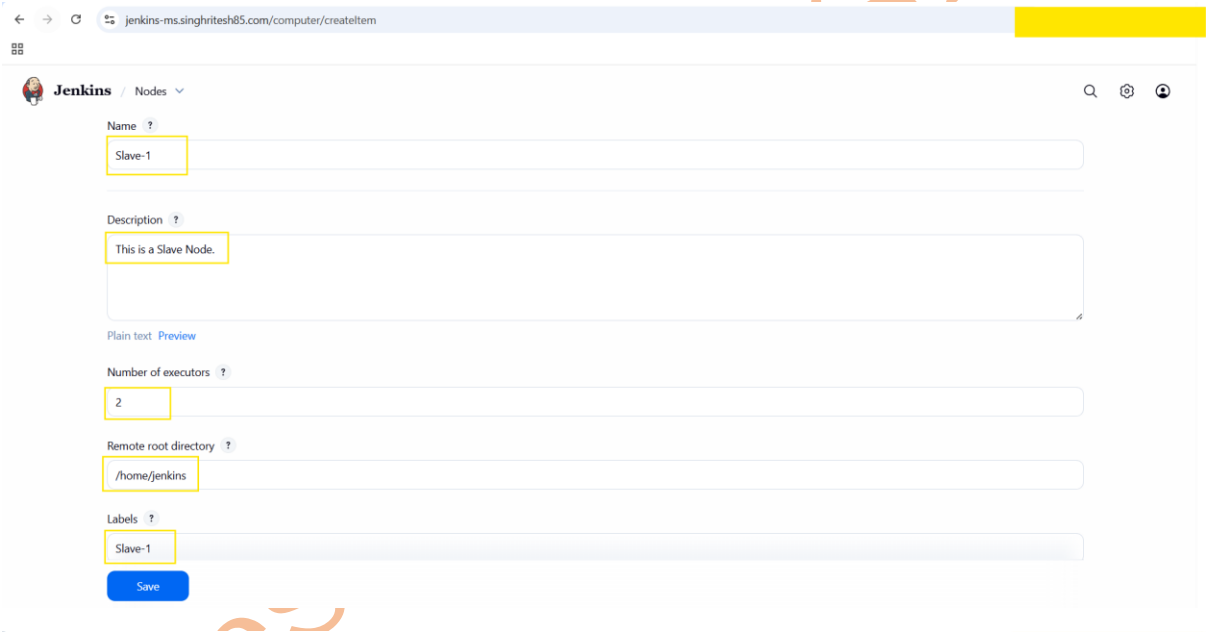


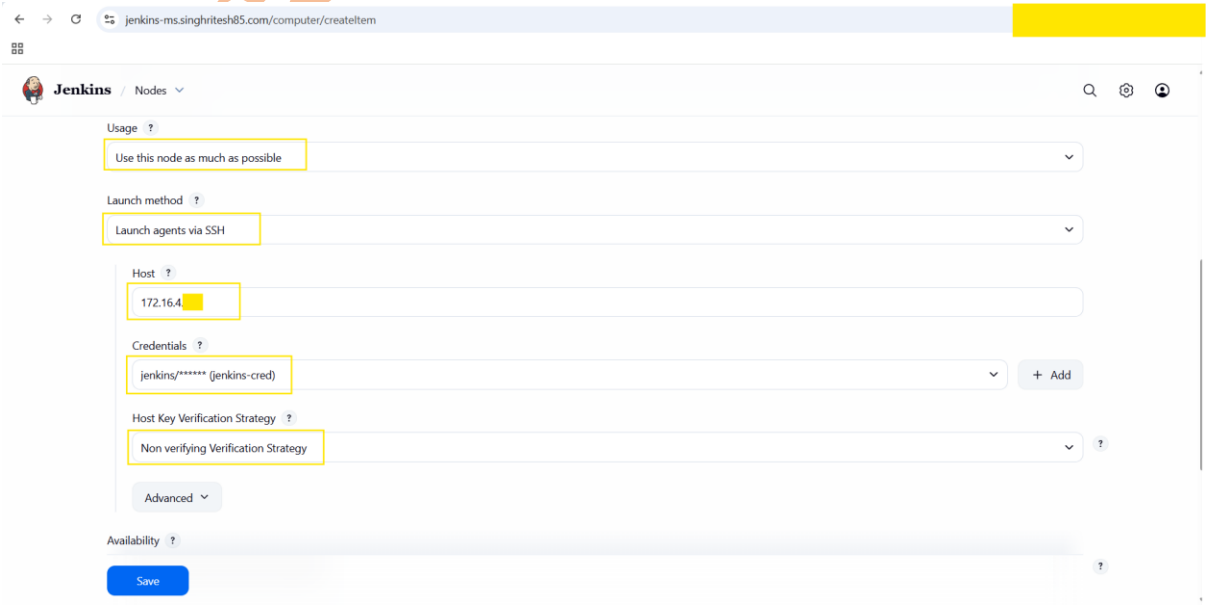



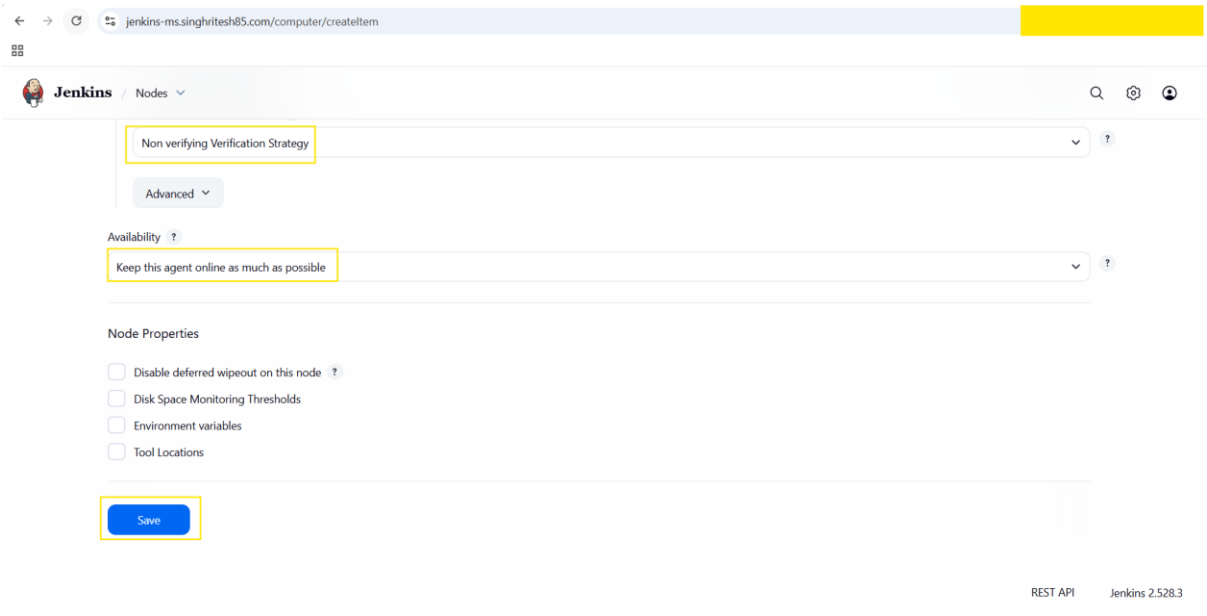

Go to Manage Jenkins and add the Jenkins Slave as shown in the screenshot attached below.


jenkins-ms.singhritesh85.com
Jenkins
New Item
Build History
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 2.528.3
jenkins-ms.singhritesh85.com/manage/
Jenkins / Manage Jenkins
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
Create/delete/modify users that can log in to this Jenkins.
jenkins-ms.singhritesh85.com/computer/
Jenkins / Nodes
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
17.00 GiB
0 B
1.87 GiB
0ms
Data obtained
1 min 8 sec
Icon: S M L
Legend
REST API
Jenkins 2.528.3


jenkins-ms.singhritesh85.com/computer/new
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 2.528.3

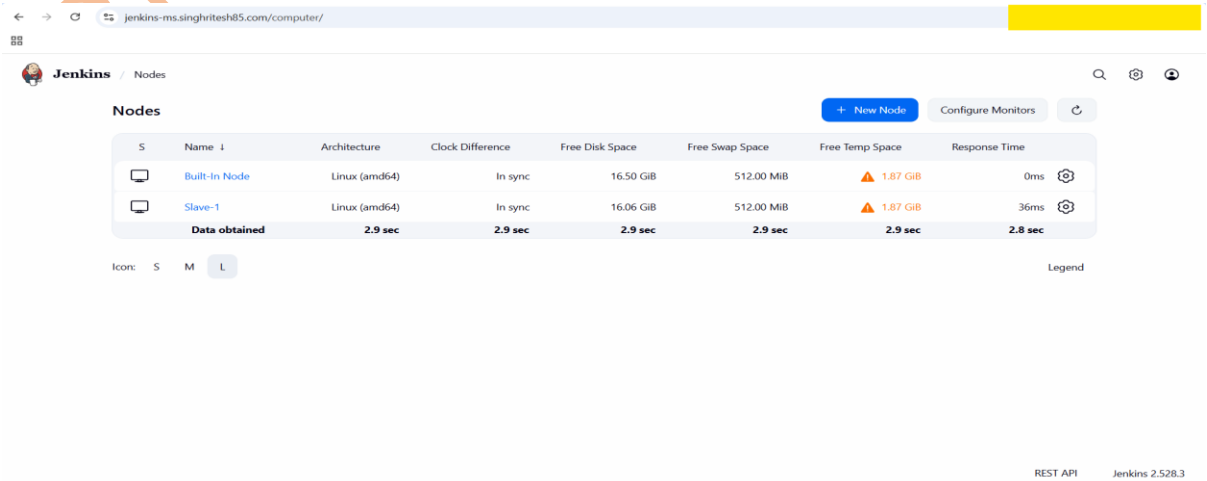

jenkins-ms.singhritesh85.com/computer/createItem
Jenkins / Nodes
Name
Slave-1
Description
This is a Slave Node.
Plain text Preview
Number of executors
2
Remote root directory
/home/jenkins
Labels
Slave-1
Save


jenkins-ms.singhritesh85.com/computer/createItem
Jenkins / Nodes
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.16.4
Credentials
jenkins/****** (jenkins-cred)
+ Add
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced
Availability
Save



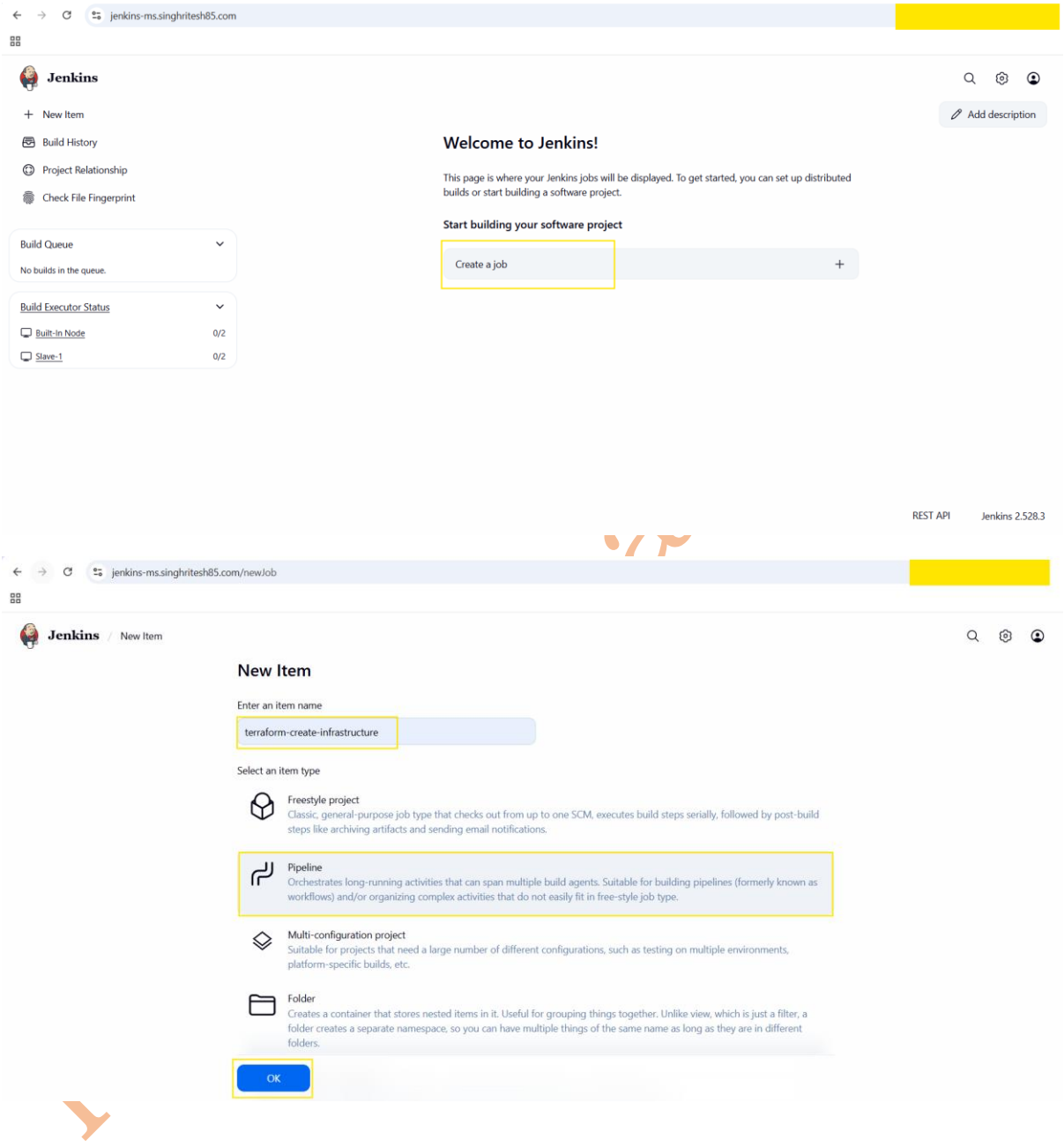

```
[root@jenkins-master ~]# fallocate -l 512M /swapfile
[root@jenkins-master ~]# chmod 600 /swapfile
[root@jenkins-master ~]# mkswap /swapfile
Setting up swapspace version 1, size = 512 MiB ( bytes)
no label, UUID=
[root@jenkins-master ~]# vim /etc/fstab
[root@jenkins-master ~]# cat /etc/fstab
#
UUID=                                   /           xfs    defaults,noatime  1   1
UUID=7BBE-6739        /boot/efi       vfat    defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2

/swapfile swap swap defaults 0 0
[root@jenkins-master ~]# swapon -a
[root@jenkins-master ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           3.7Gi       855Mi       1.3Gi       4.0Mi       1.6Gi       2.7Gi
Swap:          511Mi          0B       511Mi
```

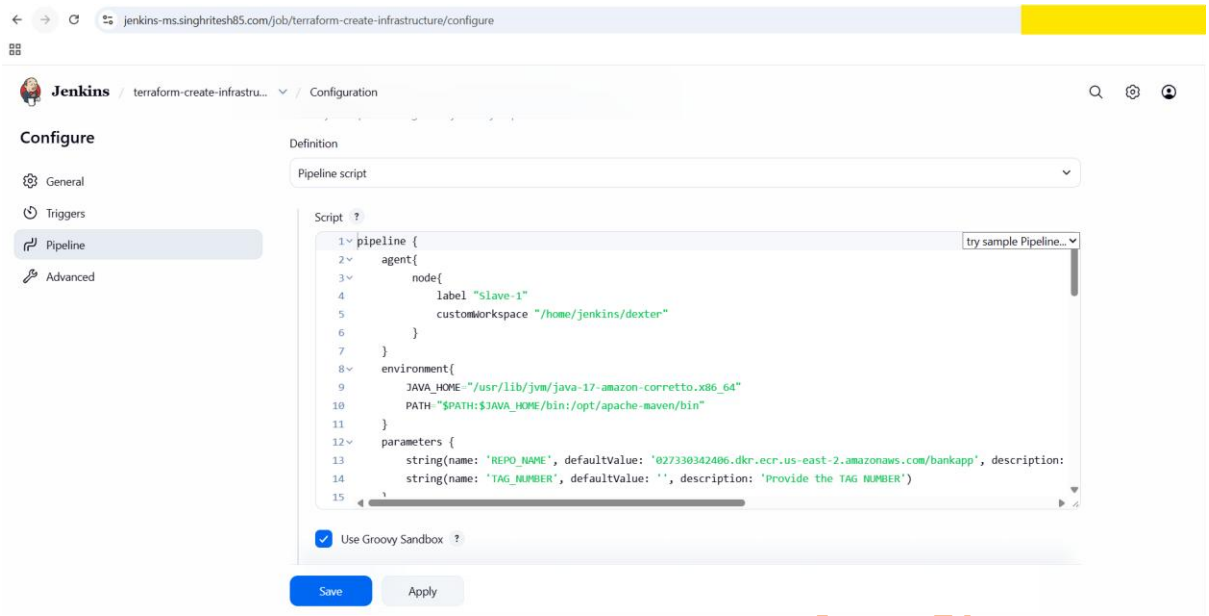
```
[root@jenkins-slave ~]# fallocate -l 512M /swapfile
[root@jenkins-slave ~]# chmod 600 /swapfile
[root@jenkins-slave ~]# mkswap /swapfile
Setting up swapspace version 1, size = 512 MiB ( bytes)
no label, UUID=
[root@jenkins-slave ~]# vim /etc/fstab
[root@jenkins-slave ~]# cat /etc/fstab
#
UUID=                                   /           xfs    defaults,noatime  1   1
UUID=           /boot/efi       vfat    defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2

/swapfile swap swap defaults 0 0
[root@jenkins-slave ~]# swapon -a
[root@jenkins-slave ~]# free -mh
               total        used        free      shared  buff/cache   available
Mem:           3.7Gi       379Mi       1.1Gi       0.0Ki       2.3Gi       3.1Gi
Swap:          511Mi          0B       511Mi
```



I had taken the instance type t3.medium for jenkins-master and jenkins-slave to refrain from higher cloud bills, you can select it as per your project requirement.

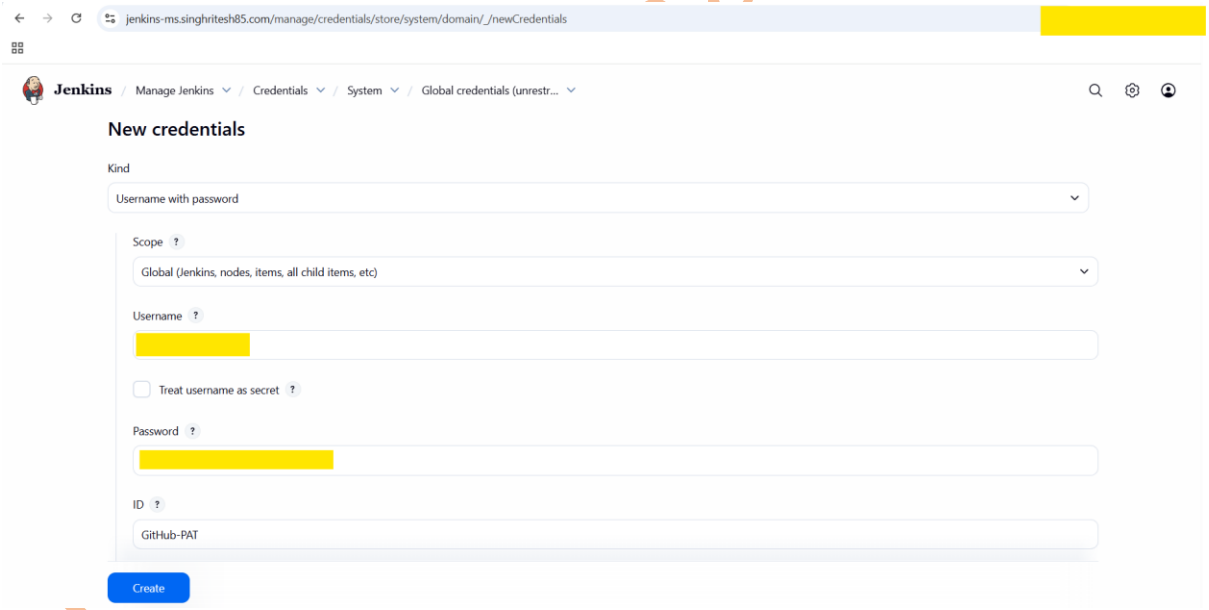
Now, create the Jenkins Job in declarative pipeline to deploy the ECS Cluster, Task definition and Service as shown in the screenshot attached below.



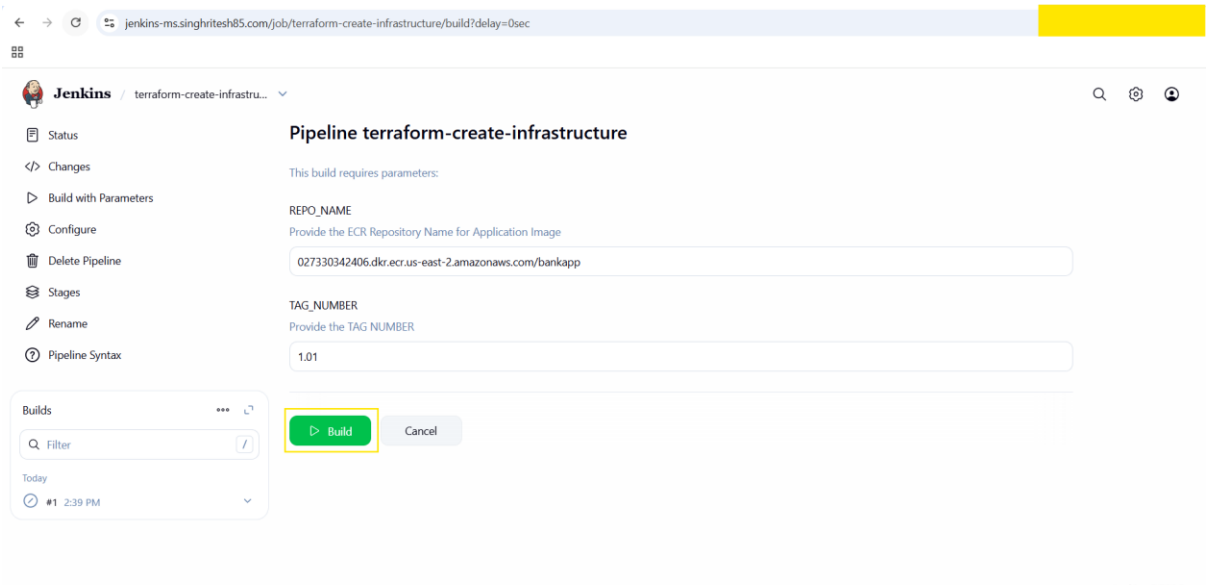


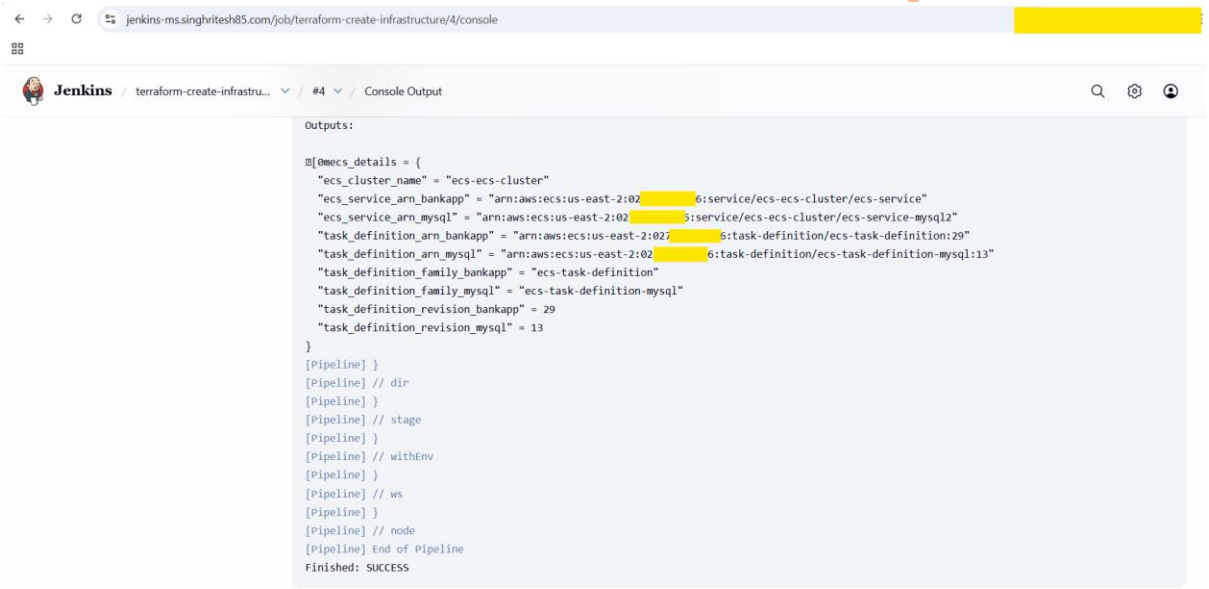

For your reference I kept the Jenkinsfile in the GitHub Repo https://github.com/singhritesh85/BankApp-Docker-the-Serverless-way.git.

Before Running the Jenkins Job make secure to create the Jenkins credentials to clone the source code or Terraform Script present in the GitHub as shown in the screenshot attached below.

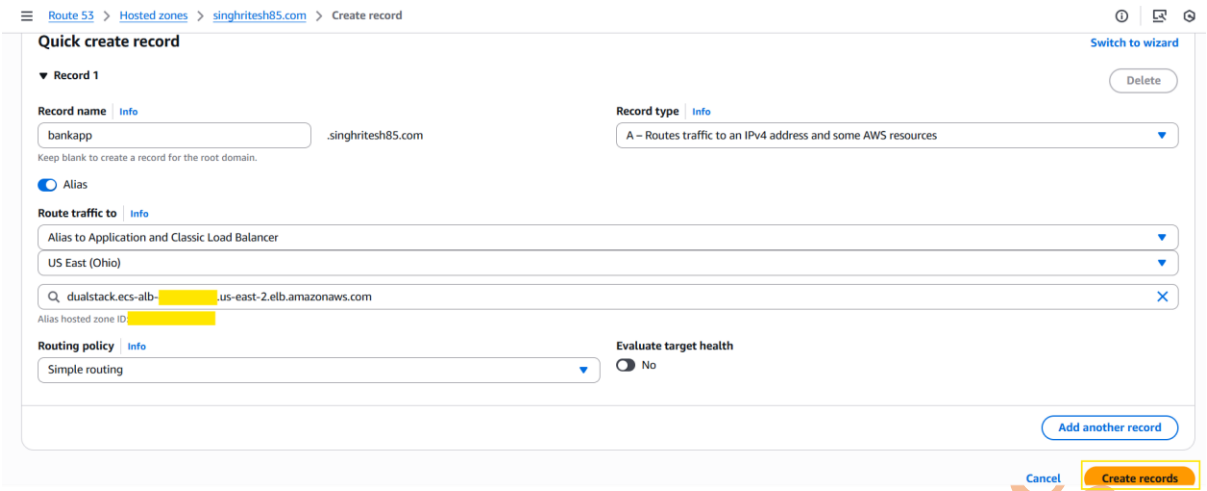



Now Run the Jenkins Job as shown in the screenshot attached below.

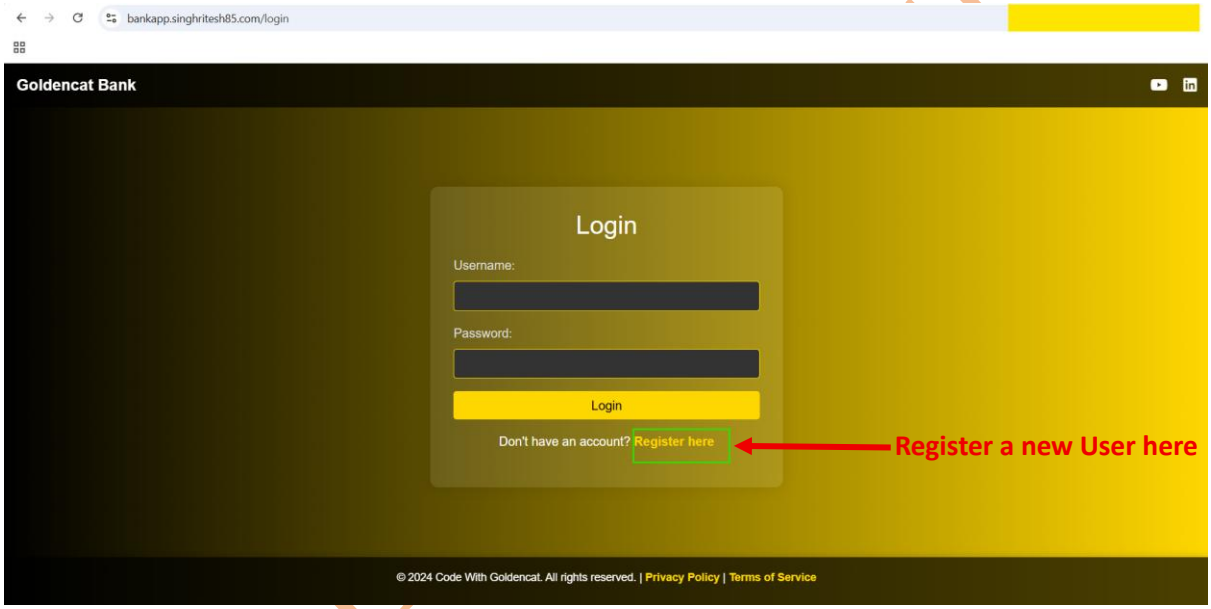


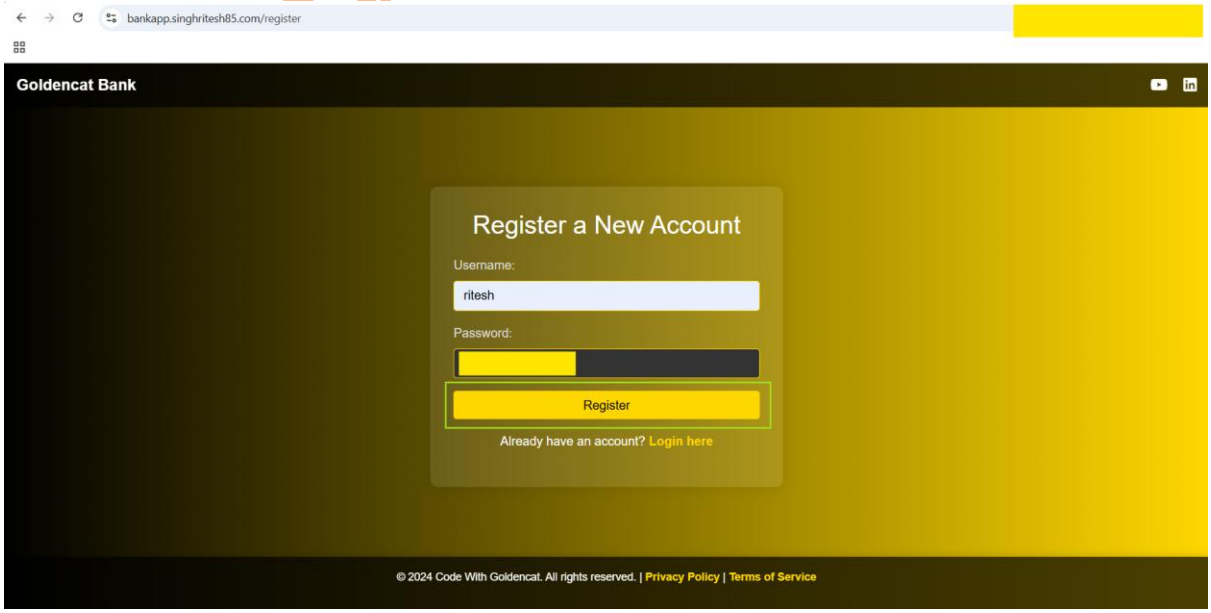



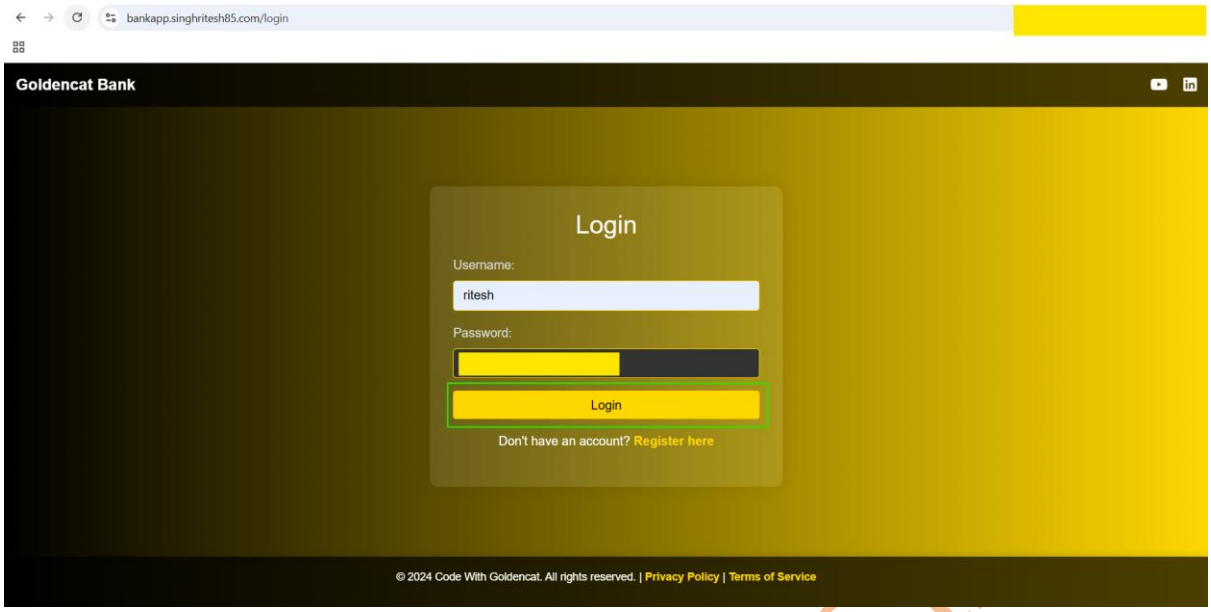

Jenkins Job was executed successfully as shown in the screenshot attached above. Finally, do the entry of created ALB DNS Name in the Route 53 to create the Record Set of A-Type as shown in the screenshot attached below.

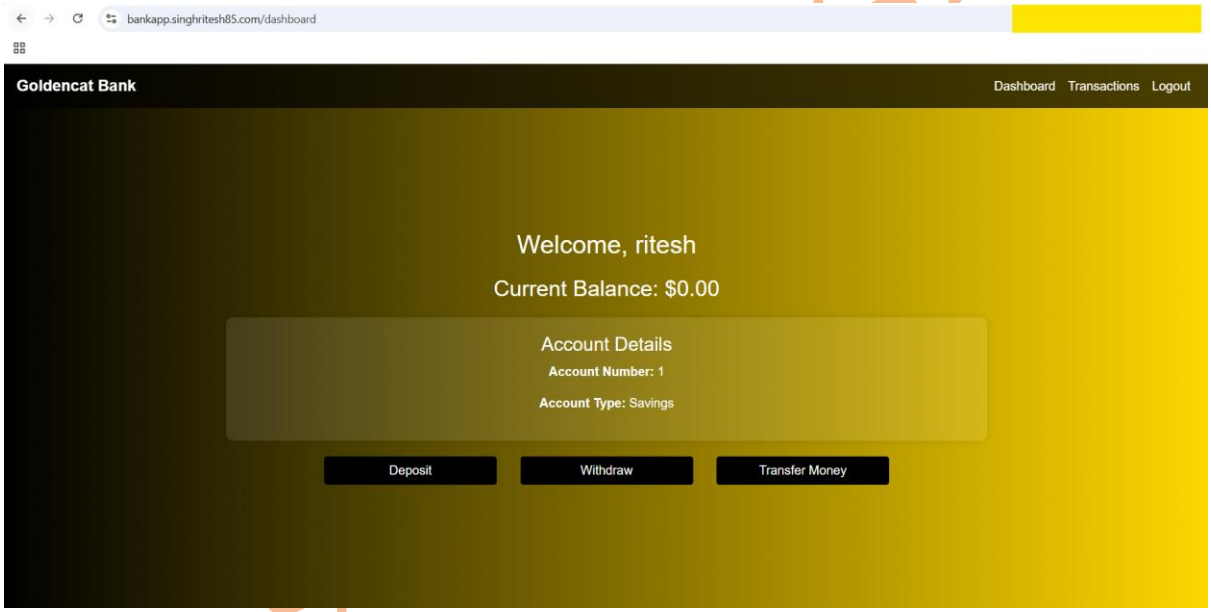



I was able to access the BankApp as shown in the screenshot attached below.






I connected with MySQL ECS Container and checked the entry for the same as shown below.

```
[root@jenkins-slave ~]# aws ecs execute-command --cluster ecs-ecs-cluster --task b                    0 --container mysql --interactive --command "/
bin/sh"

The Session Manager plugin was installed successfully. Use the AWS CLI to start a session.


Starting session with SessionId: ecs-execute-command-
# mysql -h localhost -u root --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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> show databases;
+--------------------+
| Database           |
+--------------------+
| bankappdb          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 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 | $2                                                         2 | ritesh   |
+----+---------+--------------------------------------------------------------+----------+
1 row in set (0.00 sec)
```

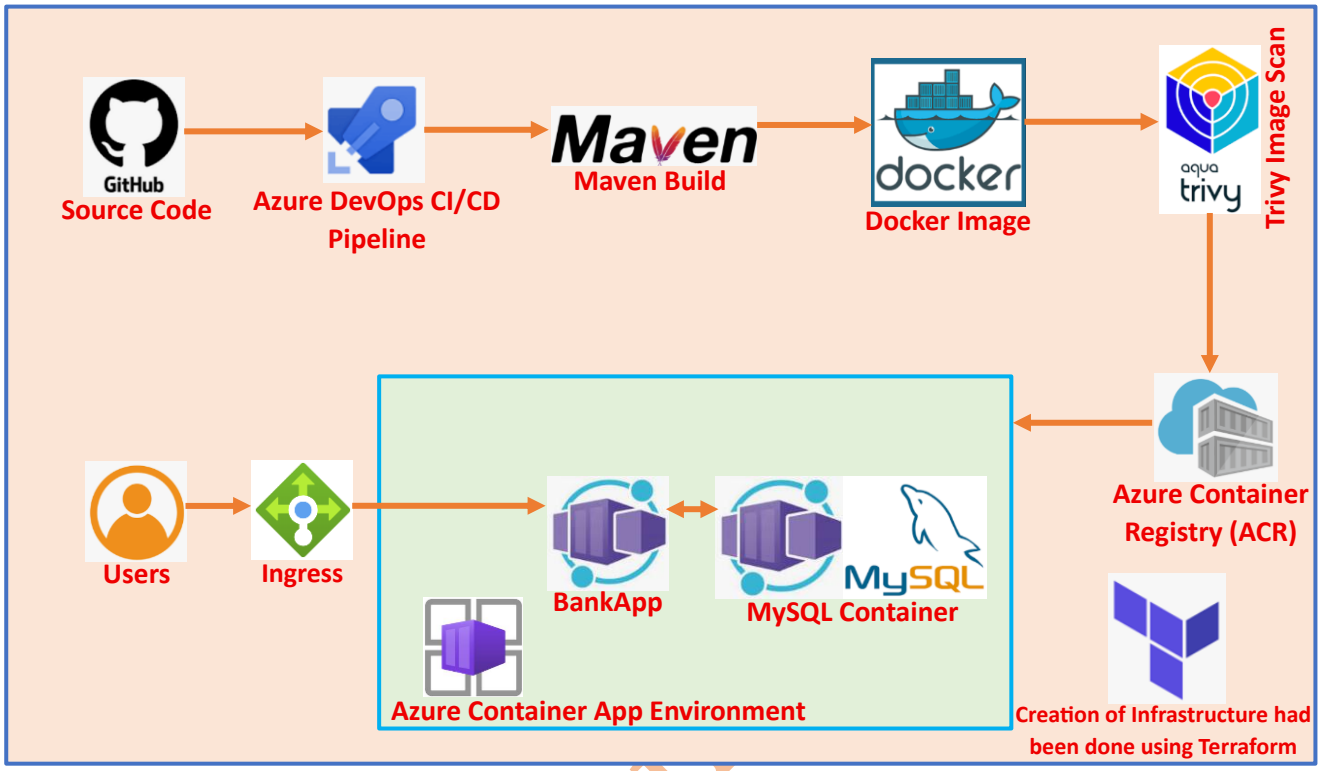
I had provided the Terraform script present in the GitHub Repo https://github.com/singhritesh85/terraform-docker-the-serverless-way.git at the path **terraform-ecs-task-definition-service-efs-ecr-alb** to autoscale the ECS Containers. In this project I had chosen MySQL Containers to be present in the ECS Containers however it is highly advisable to use RDS instead of MySQL Containers. In the Terraform script present at the path mentioned just above added with the functionality of Autoscaling the MySQL Containers with a common EFS as the data source (persistent shared storage) as well as autoscaling the BankApp, however I would suggest you to use RDS instead of MySQL Containers. It is highly recommended to use RDS (MySQL) for production environment.

**Source Code: -** https://github.com/singhritesh85/BankApp-Docker-the-Serverless-way.git

**Terraform Script: -** https://github.com/singhritesh85/terraform-docker-the-serverless-way.git

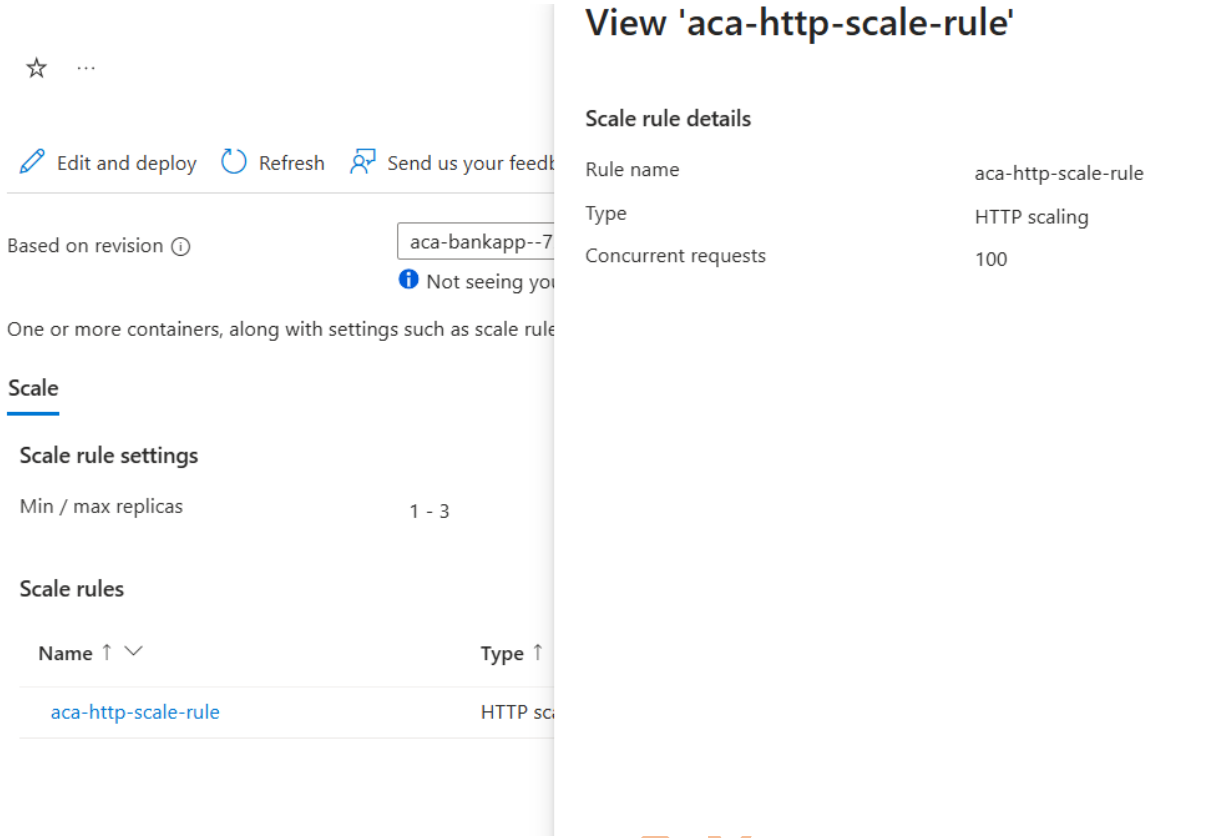
## Module -2

# Docker – The Serverless Way (Azure Container App)



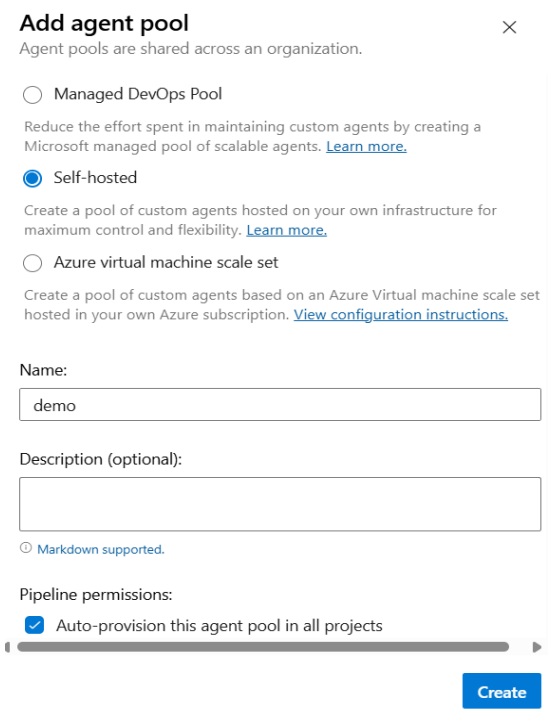

This DevOps Project deals with creation of Infrastructure using Terraform, Azure DevOps had been used as the CI/CD Tool and source code was present in the GitHub Repo https://github.com/singhritesh85/BankApp-Docker-ACA-the-Serverless-way.git. For every release when the Source code will be pushed to the GitHub Repo the Azure DevOps Pipeline will be executed which will build the Source Code using Maven and create the Docker Image. This Docker Image will be scanned by Trivy (a vulnerability scanner tool developed by Aqua security) and then push the Image to Azure Container Registry (ACR). At the Deployment stage for a fresh release Azure Container App Environment, Azure Container App (ACA) will be created, SSL Certificate will be uploaded and custom domain will be created. However, for an existed release if source code had been pushed to GitHub Repo after the update, then a new revision for Azure Container App will be created using the new Docker Image and hence will create a new replica of Azure Container App. For BankApp Azure Container App the ingress will route the 100% of the external traffic to the latest revision. For production environment it is suggested to use Azure Database for MySQL - Flexible Servers rather than MySQL based containers. For this project I had auto scaled both BankApp (HTTP based scale rule) and MySQL containers (TCP based scale rule). For HTTP based scale rule you can see in the Azure Console but TCP based scale rule you cannot see correctly in the Azure Console for that you need to check with the Azure CLI using the command as shown here **az containerapp show --name mysql-service --resource-group aca-rg**.

```
[demo@devopsagent-vm ~]$ az containerapp show --name mysql-service --resource-group aca-rg|grep -A4 "tcp"
            "name": "aca-tcp-scale-rule",
            "tcp": {
              "metadata": {
                "concurrentRequests": "10"
              }
            }
```




Start with creation of Azure DNS Zone and then Self hosted Azure DevOps Agent Server and Azure Container Registry. You can use the Terraform script present in the GitHub Repo https://github.com/singhritesh85/terraform-docker-aca-the-serverless-way.git at the path **terraform-azure-dns-zone** and **terraform-azure-vm-devops-agent-acr** respectively.

To create Azure DevOps Pipeline the first step was to create the Azure DevOps Self Hosted Agent Pool as shown in the screenshot attached below.



```
[root@devopsagent-vm ~]# cd /opt/
[root@devopsagent-vm opt]# mkdir myagent && cd myagent
```

```
[root@devopsagent-vm opt]# cd /opt/myagent/
```

```
[root@devopsagent-vm opt]# wget https://download.agent.dev.azure.com/agent/4.266.2/vsts-agent-linux-x64-4.266.2.tar.gz
```

```
[root@devopsagent-vm myagent]# ls
bin  config.sh  env.sh  externals  license.html  reauth.sh  run-docker.sh  run.sh  vsts-agent-linux-x64-4.266.2.tar.gz
[root@devopsagent-vm myagent]# rm -f vsts-agent-linux-x64-4.266.2.tar.gz
[root@devopsagent-vm myagent]# chown -R demo:demo /opt/myagent/
[root@devopsagent-vm myagent]# ls
bin  config.sh  env.sh  externals  license.html  reauth.sh  run-docker.sh  run.sh
[root@devopsagent-vm myagent]# su - demo
[demo@devopsagent-vm ~]$ cd /opt/myagent/
[demo@devopsagent-vm myagent]$ sudo ./bin/installdependencies.sh
```

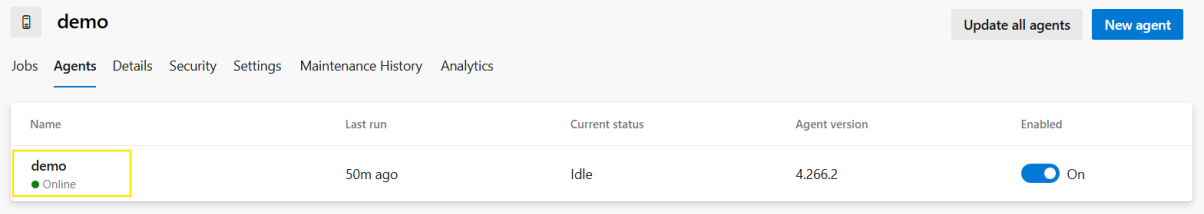
```
[demo@devopsagent-vm myagent]$ ./config.sh

  ___                    ______ _         _ _
 / _ \                   | ___ (_)       | (_)
/ /_\ \____  _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___  ___
|  _  |_  / | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
| | | |/ /| |_| | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                   | |
       agent v4.266.2              |_|         (commit 8[redacted]3)

>> 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 > [redacted]
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > demo
Enter agent name (press enter for devopsagent-vm) > demo
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2025-[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 Tue 2025-[redacted] UTC; 9ms ago
 Main PID: 17730 (runsvc.sh)
    Tasks: 2 (limit: 24538)
   Memory: 1.0M
   CGroup: /system.slice/vsts.agent.[redacted].demo.demo.service
           ├─17730 /bin/bash /opt/myagent/runsvc.sh
           └─17733 ./externals/node20_1/bin/node --version

Dec 23 11:00:04 devopsagent-vm systemd[1]: Started Azure Pipelines Agent ([redacted].demo.demo).
```

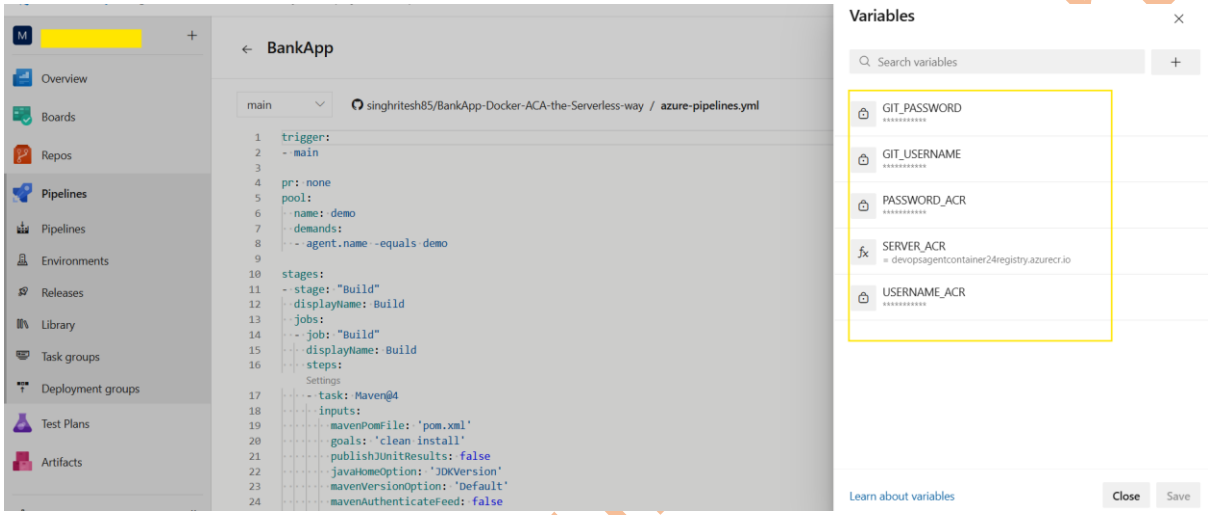
To authenticate from Azure account on Azure DevOps Self-hosted Agent I used the command **az login** and provide the generated URL in web browser and then the generated code.

```
[demo@devopsagent-vm main]$ az login
To sign in, use a web browser to open the page [redacted] and enter the code [redacted] to authenticate.
```

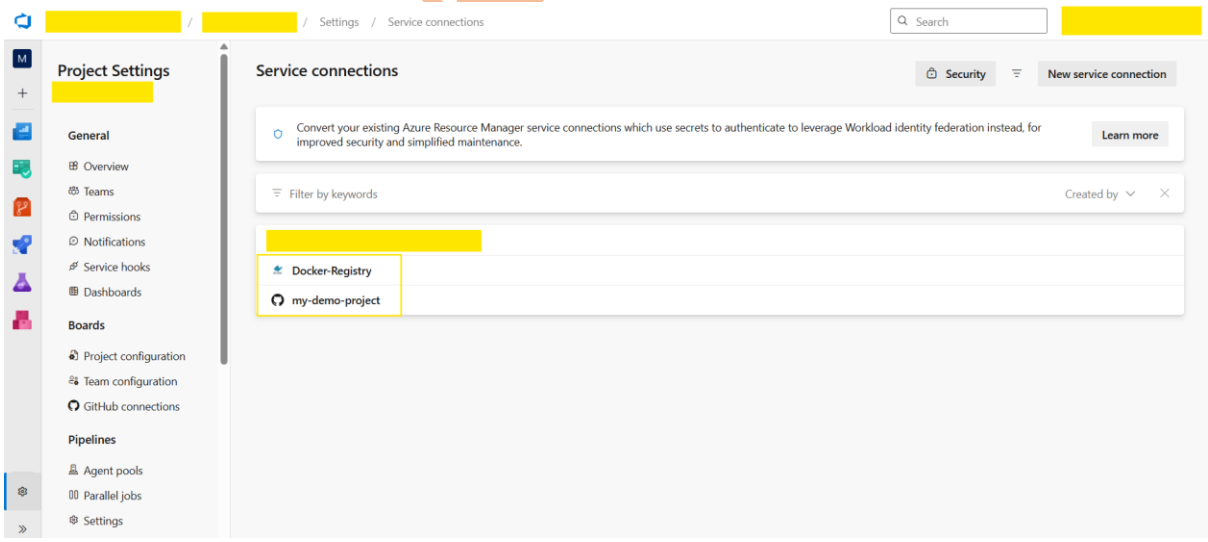
Now I checked the Agent Pool in Azure DevOps Console and found the Agent was in Online state.



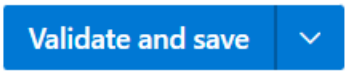

Now create the pipeline using the yaml file named as **azure-pipelines.yml** present in the GitHub Repo https://github.com/singhritesh85/BankApp-Docker-ACA-the-Serverless-way.git. Then create the variables as need for this pipeline to execute for this project I created five variables as shown below.



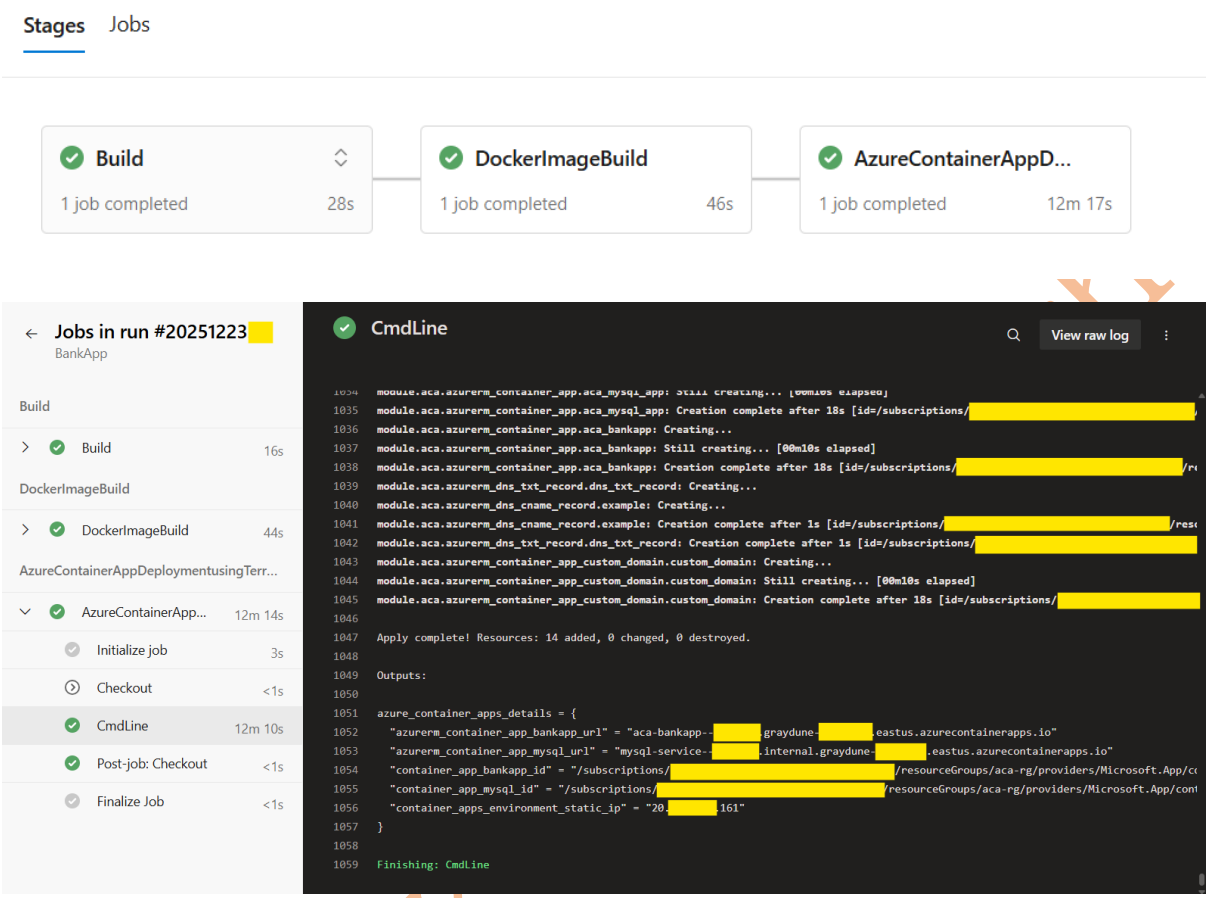

Before creation of the pipeline make sure you had created the service connection for your Docker Registry (Azure Container Registry) and to integrate the GitHub Account with Azure DevOps.



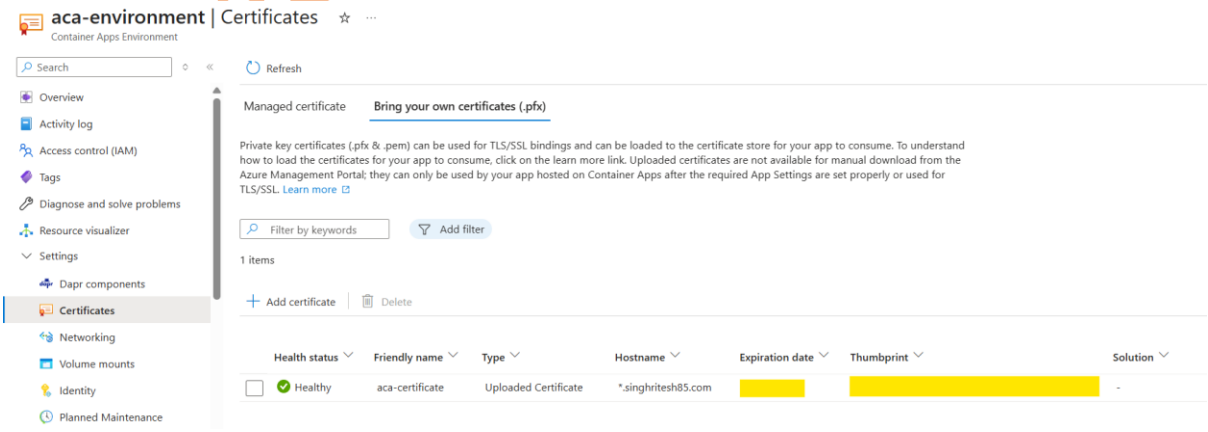

Then click on the option of Validate and Save as shown in the screenshot attached below.

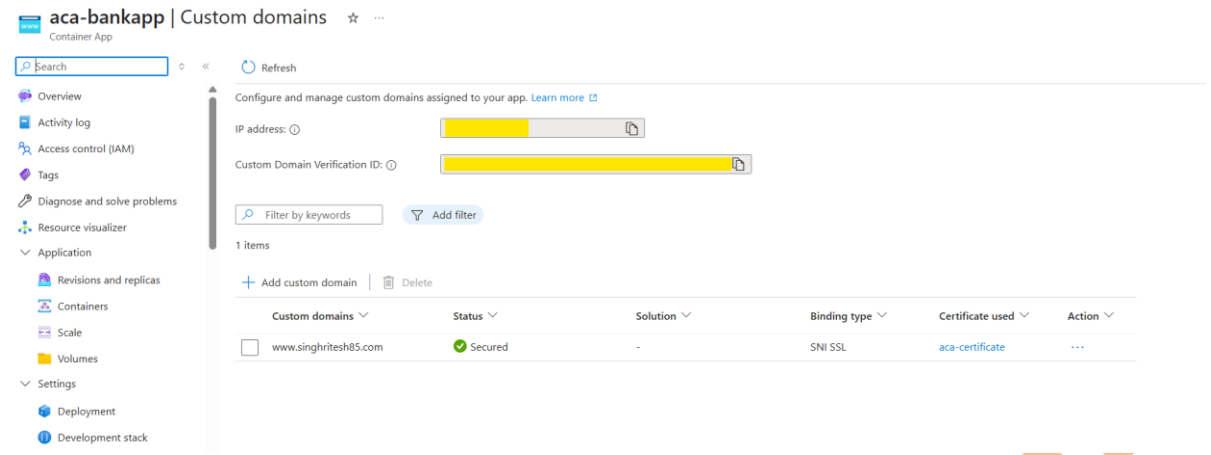


It will start executing the Azure DevOps Pipeline and after its successful execution the screenshot is as shown below.

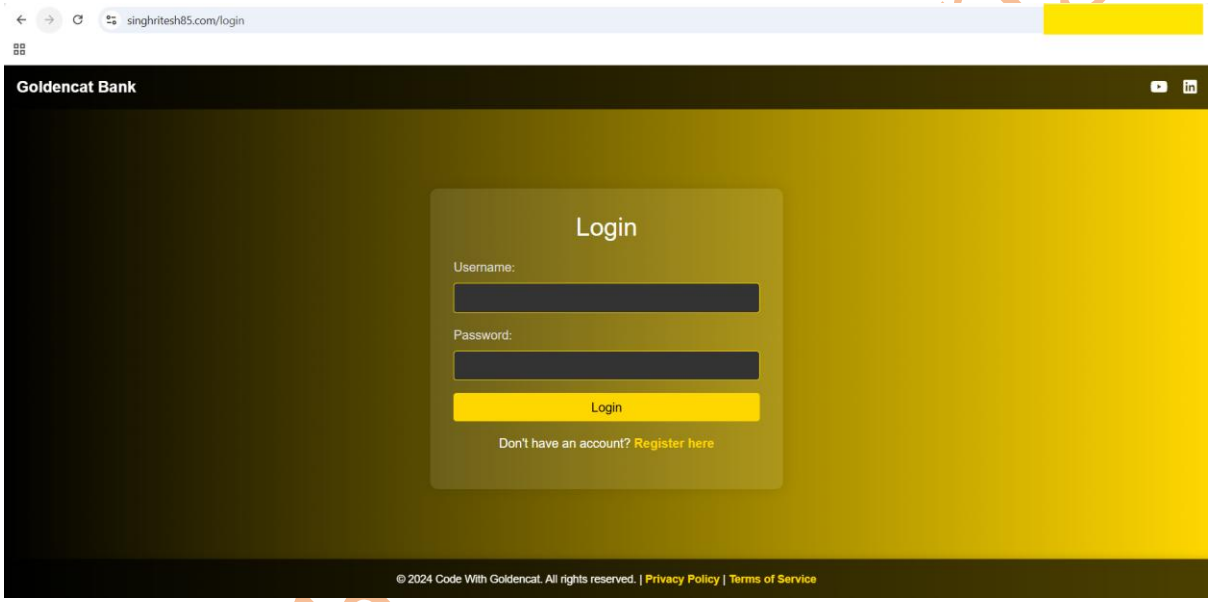


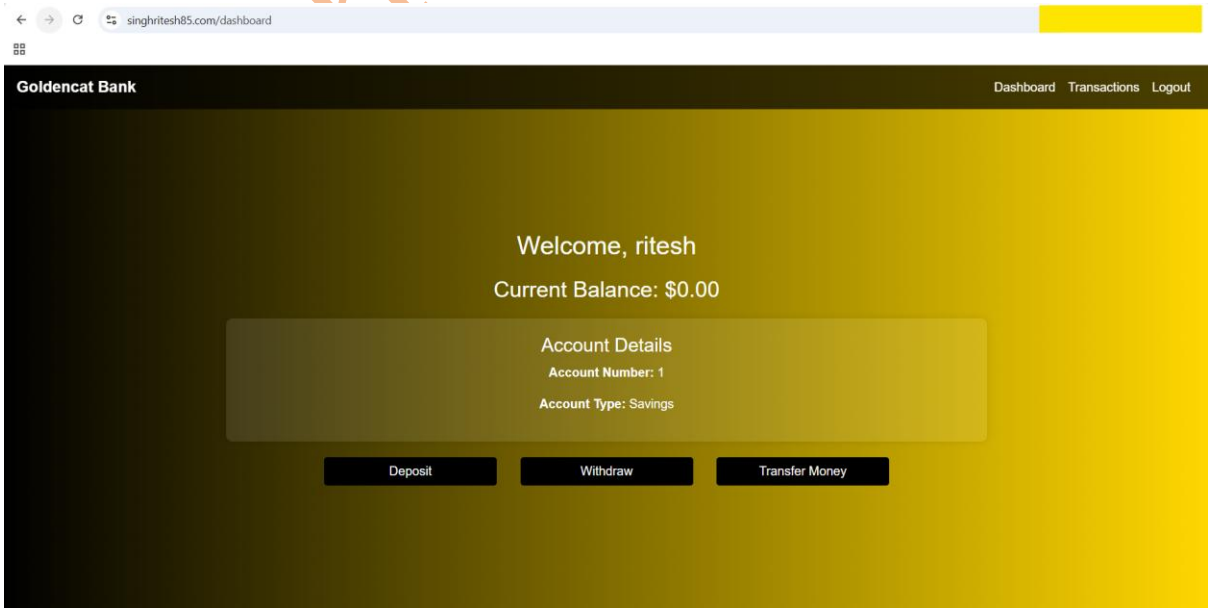

In the terraform script present in GitHub Repo https://github.com/singhritesh85/terraform-docker-aca-the-serverless-way.git at the path **terraform-azure-container-app-ingress-bankapp-mysql** you need to provide the **SSL Certificate .pfx** extension file. It will create the certificate in Azure Container App Environment which will be used to create the custom domain in BankApp Azure Container App as shown in the screenshot attached below.



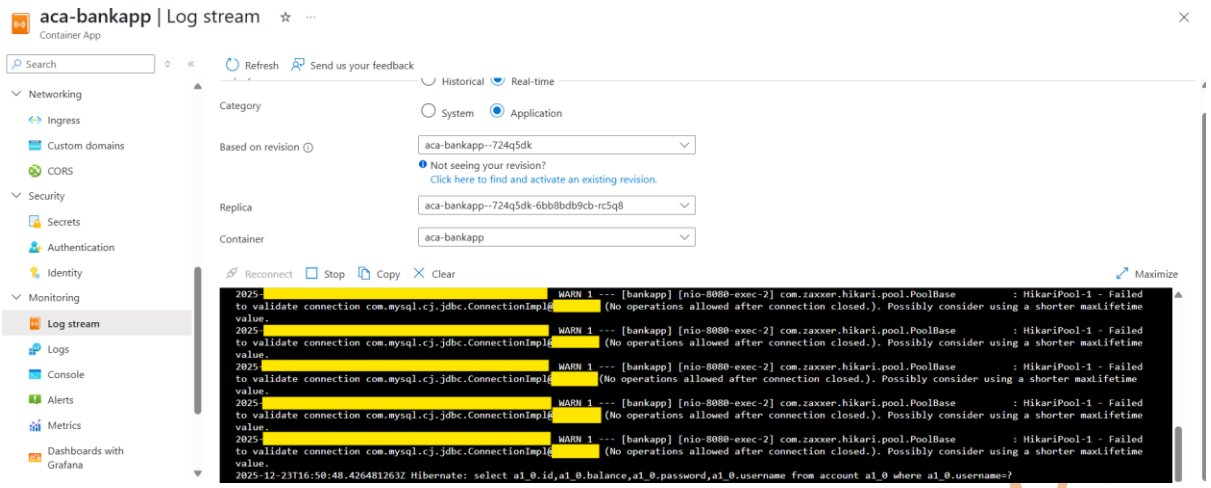


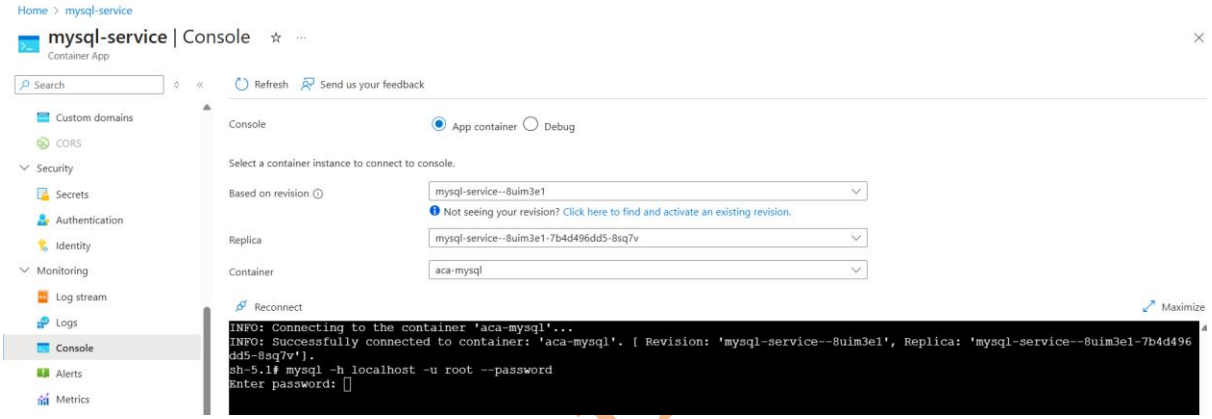

I opened the URL on web browser as shown in the screenshot attached below.

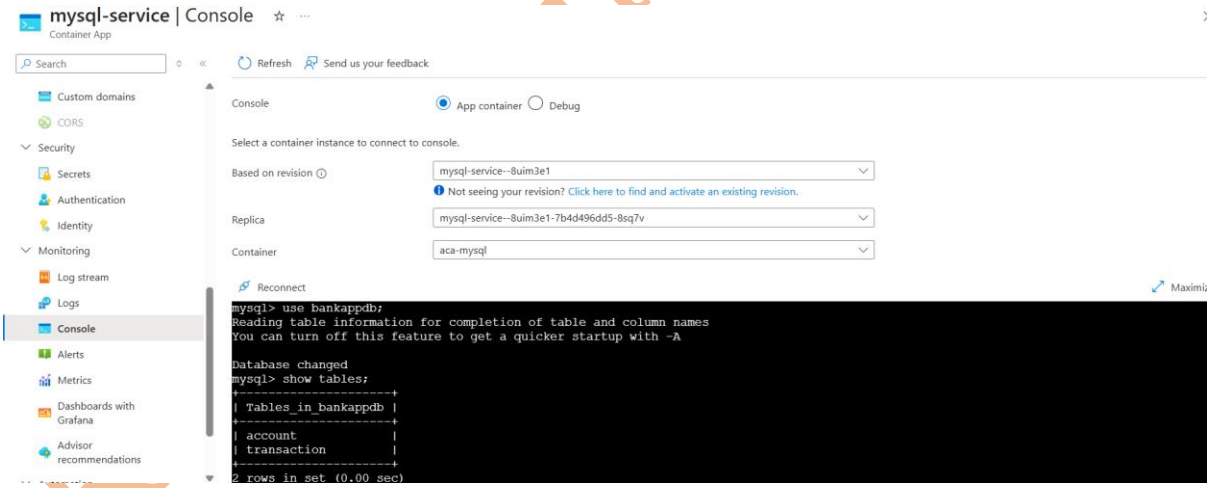


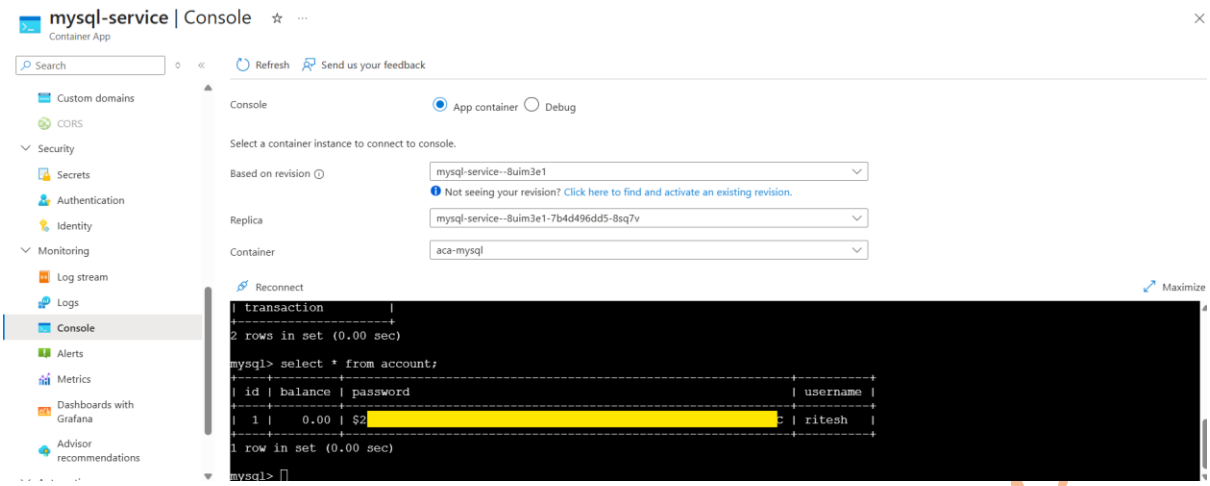




You can check the application Log and entry for bankappdb database as shown in the screenshot attached below.


aca-bankapp | Log stream
Container App
Search
Refresh
Send us your feedback
Networking
Ingress
Custom domains
CORS
Security
Secrets
Authentication
Identity
Monitoring
Log stream
Logs
Console
Alerts
Metrics
Dashboards with Grafana
Historical
Real-time
Category
System
Application
Based on revision
aca-bankapp--724q5dk
Not seeing your revision?
Click here to find and activate an existing revision.
Replica
aca-bankapp--724q5dk-6bb8bdb9cb-rc5q8
Container
aca-bankapp
Reconnect
Stop
Copy
Clear
Maximize
WARN 1 --- [bankapp] [nio-8080-exec-2] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Failed to validate connection com.mysql.cj.jdbc.ConnectionImpl@ (No operations allowed after connection closed.). Possibly consider using a shorter maxLifetime value.
2025-12-23T16:50:48.426481263Z Hibernate: select a1_0.id,a1_0.balance,a1_0.password,a1_0.username from account a1_0 where a1_0.username=?

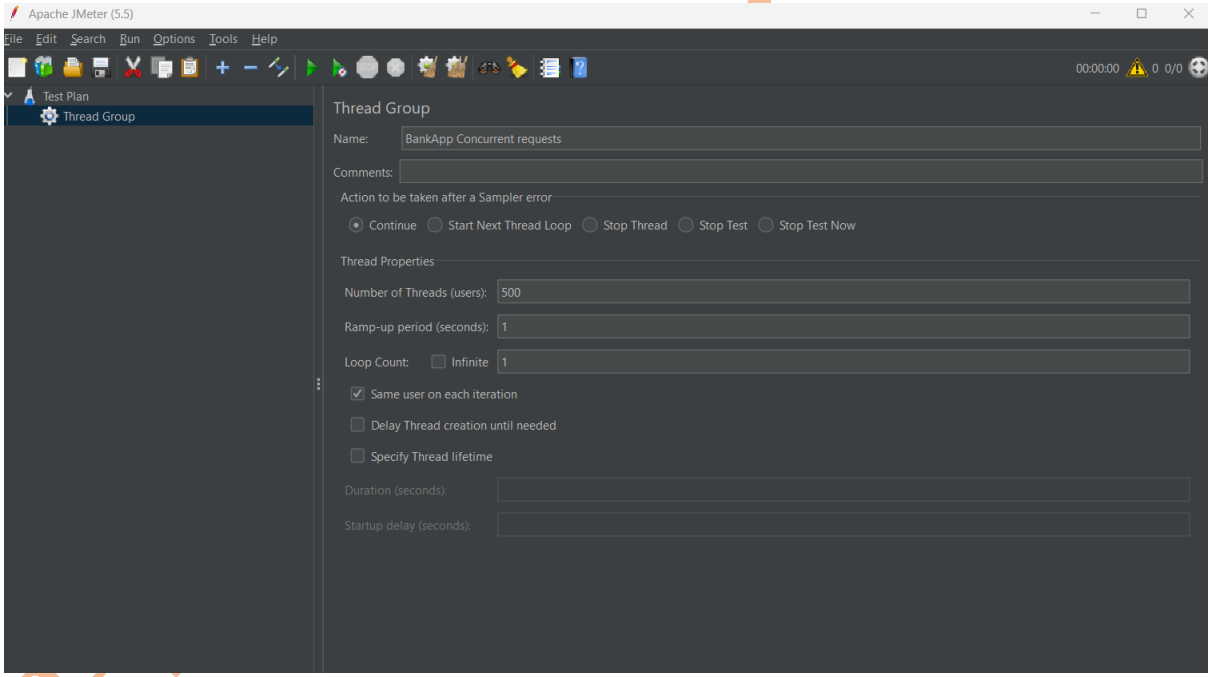

Home > mysql-service
mysql-service | Console
Container App
Search
Refresh
Send us your feedback
Custom domains
CORS
Security
Secrets
Authentication
Identity
Monitoring
Log stream
Logs
Console
Alerts
Metrics
Console
App container
Debug
Select a container instance to connect to console.
Based on revision
mysql-service--8uim3e1
Not seeing your revision? Click here to find and activate an existing revision.
Replica
mysql-service--8uim3e1-7b4d496dd5-8sq7v
Container
aca-mysql
Reconnect
Maximize
INFO: Connecting to the container 'aca-mysql'...
INFO: Successfully connected to container: 'aca-mysql'. [ Revision: 'mysql-service--8uim3e1', Replica: 'mysql-service--8uim3e1-7b4d496dd5-8sq7v'].
sh-5.1# mysql -h localhost -u root --password
Enter password:

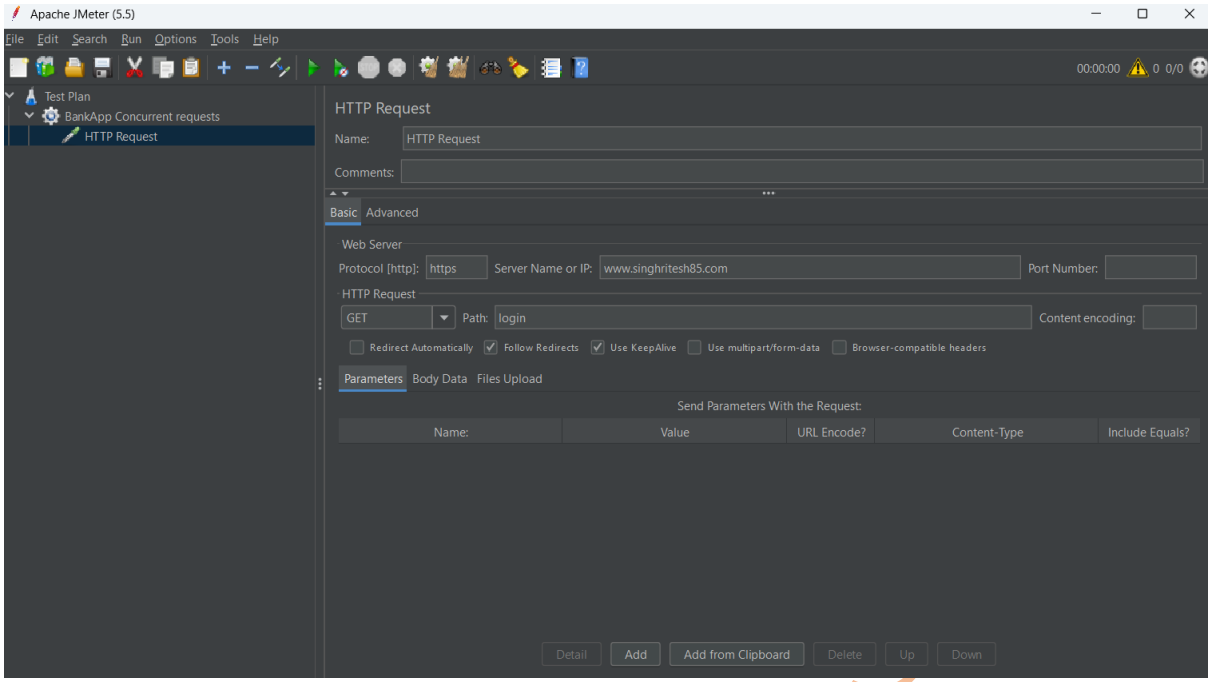

mysql-service | Console
Container App
Search
Refresh
Send us your feedback
Custom domains
CORS
Security
Secrets
Authentication
Identity
Monitoring
Log stream
Logs
Console
Alerts
Metrics
Dashboards with Grafana
Advisor recommendations
Console
App container
Debug
Select a container instance to connect to console.
Based on revision
mysql-service--8uim3e1
Not seeing your revision? Click here to find and activate an existing revision.
Replica
mysql-service--8uim3e1-7b4d496dd5-8sq7v
Container
aca-mysql
Reconnect
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)

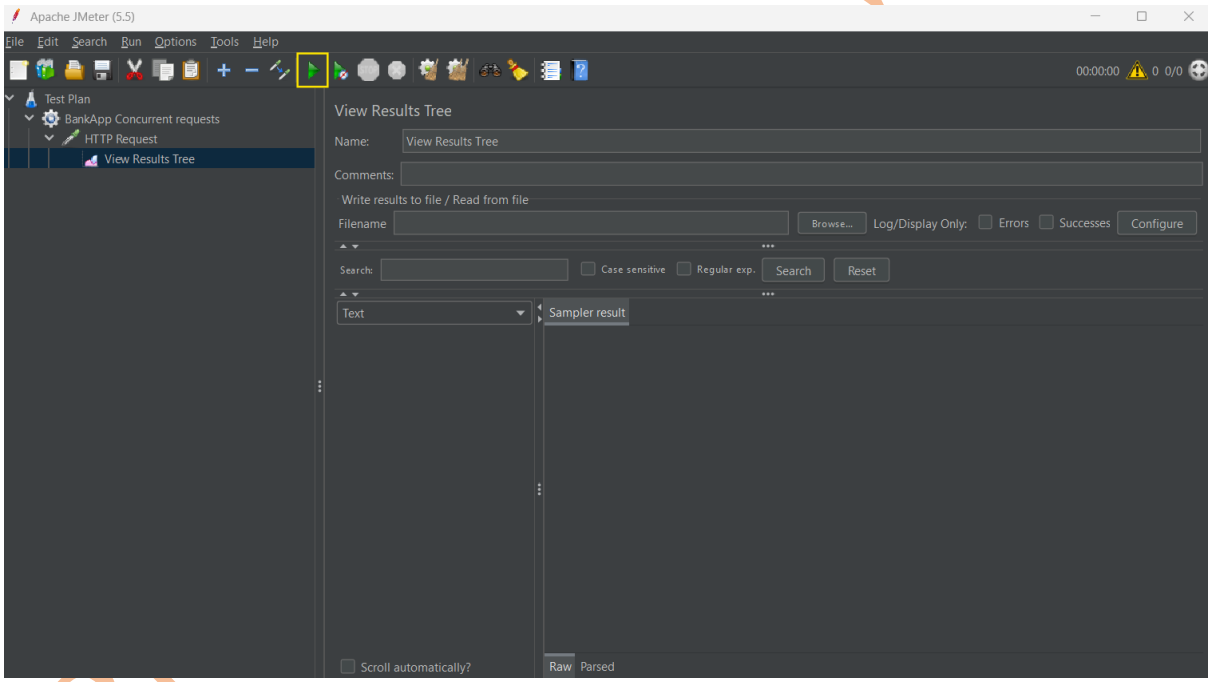


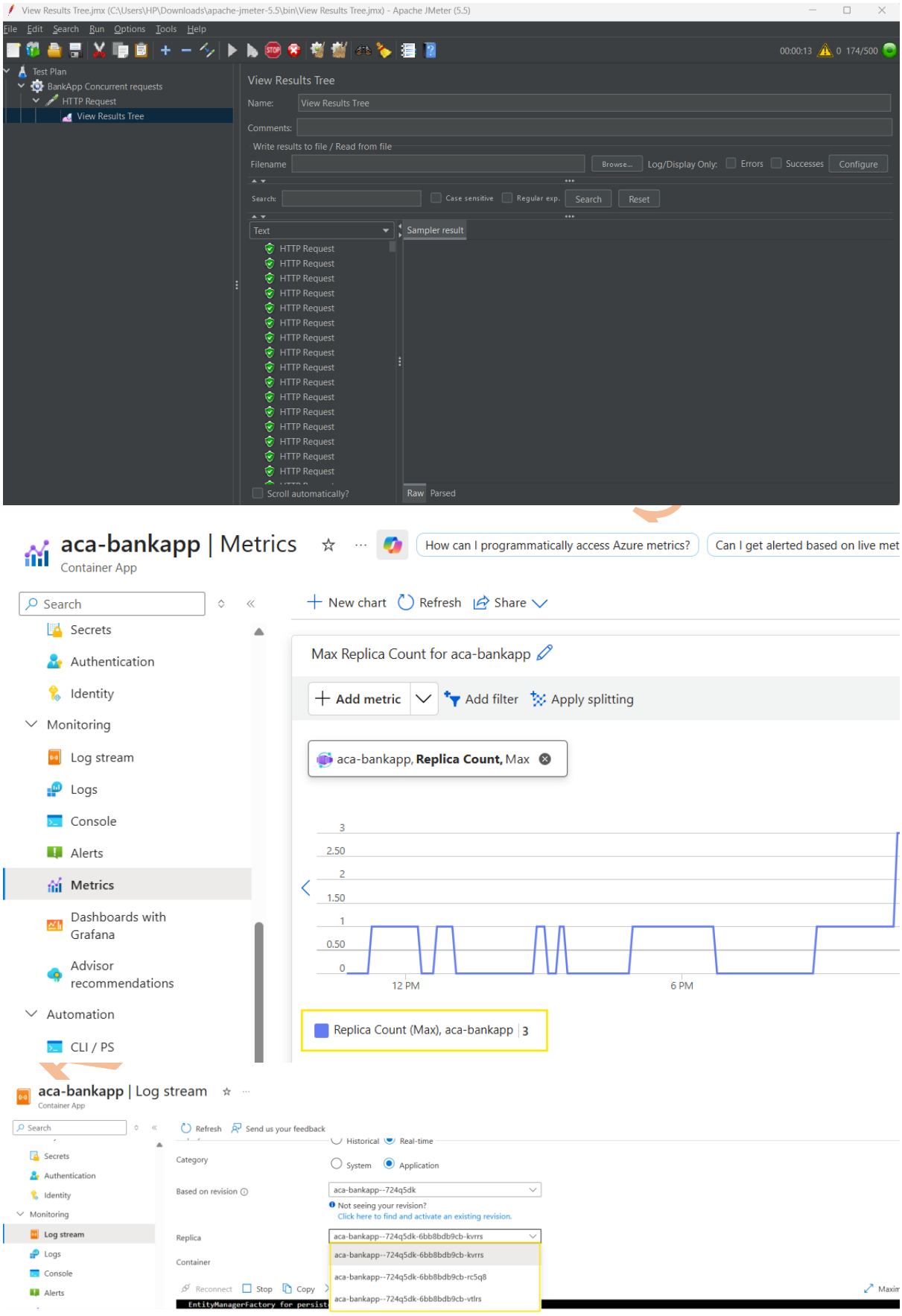

I found the entry for user ritesh in the database bankappdb as shown in the screenshot attached above.

I created the concurrent http request and checked how auto scaling behaves for BankApp if it reaches and crosses the threshold 100 and recorded the response as shown in the screenshot attached below.




Apache JMeter (5.5)
File Edit Search Run Options Tools Help
00:00:00 0 0/0
Test Plan
BankApp Concurrent requests
HTTP Request
HTTP Request
Name: HTTP Request
Comments:
Basic Advanced
Web Server
Protocol [http]: https
Server Name or IP: www.singhritesh85.com
Port Number:
HTTP Request
GET
Path: login
Content encoding:
Redirect Automatically
Follow Redirects
Use KeepAlive
Use multipart/form-data
Browser-compatible headers
Parameters Body Data Files Upload
Send Parameters With the Request:
Name:
Value
URL Encode?
Content-Type
Include Equals?
Detail
Add
Add from Clipboard
Delete
Up
Down


Apache JMeter (5.5)
File Edit Search Run Options Tools Help
00:00:00 0 0/0
Test Plan
BankApp Concurrent requests
HTTP Request
View Results Tree
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Search:
Case sensitive
Regular exp.
Search
Reset
Text
Sampler result
Scroll automatically?
Raw
Parsed




From above screenshot it has been confirmed that replica count increased from 1 to 3 which confirmed Autoscaling.

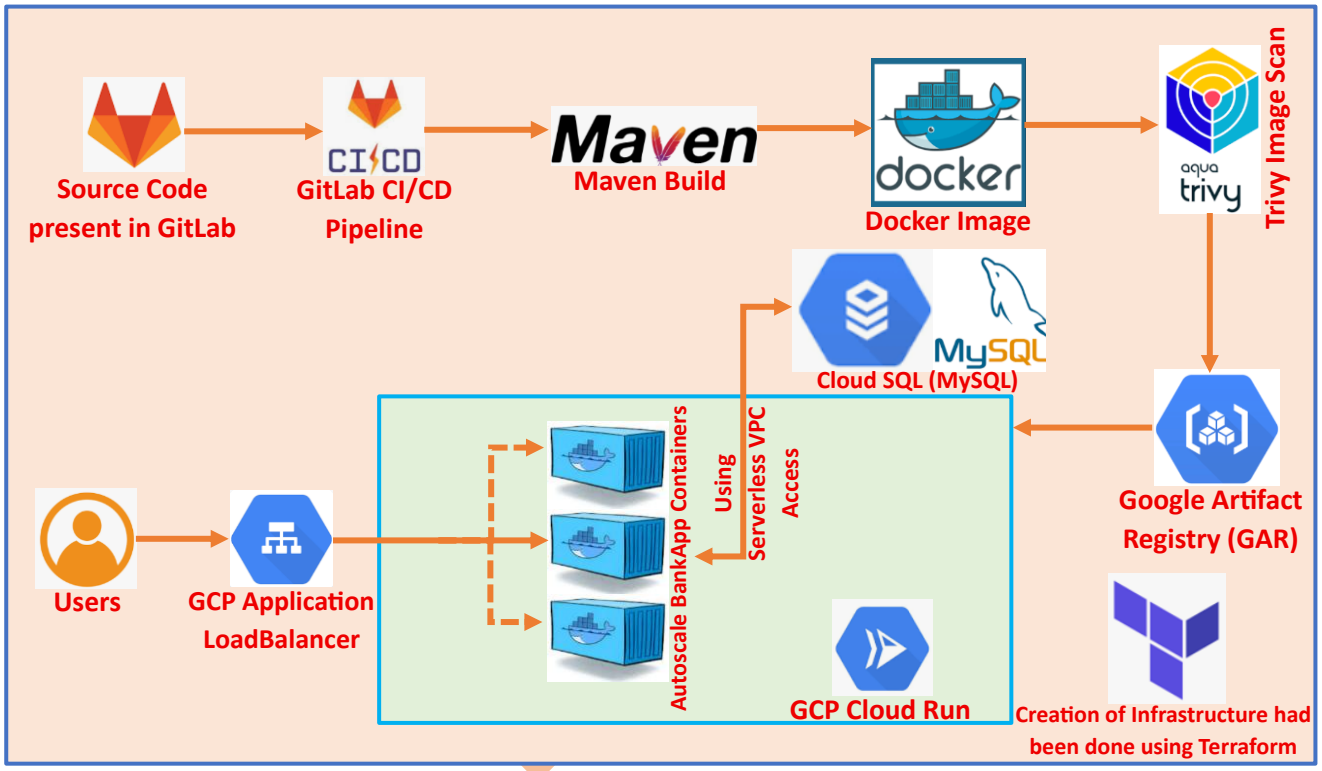
**Ritesh Kumar Singh || Email Address: -** riteshkumarsingh9559@gmail.com **|| LinkedIn: -** https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/ **|| GitHub: -** https://github.com/singhritesh85

GitHub Repo: - https://github.com/singhritesh85/BankApp-Docker-ACA-the-Serverless-way.git

Terraform Script: - https://github.com/singhritesh85/terraform-docker-aca-the-serverless-way.git

## Module-3

# Docker – The Serverless Way (GCP Cloud Run)




This DevOps Project deals with creation of infrastructure using Terraform, GitLab has been used to keep the source code and GitLab CI/CD was used as the CI/CD Tool. Terraform script to install the GitLab present in the GitHub Repo https://github.com/singhritesh85/terraform-gitlab-server-gcp.git for your reference. I kept the Terraform script to install the GitLab in a separate GitHub Repo as mentioned above for your reference because in industry you will rarely get the chance to install GitLab and when you need to install it you can use this Terraform script (you should select the machine type depending on the project requirement). Terraform script to create the GAR (GCP Artifact Registry) is present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git at the path **terraform-gcp-artifact-registry**. The Terraform script to create the infrastructure is present in the GitHub Repo https://github.com/singhritesh85/terraform-docker-cloudrun-the-serverless-way.git for your reference. This Terraform script will create Cloud Run Service for BankApp and Cloud SQL (MySQL). The BankApp Containers created using Cloud Run Service will communicate with the GCP Cloud SQL (MySQL) through **Serverless VPC access**. For this project the terraform script will create the entire infrastructure including GCP VPC, Cloud Run Service, Cloud SQL (MySQL) and Serverless VPC access. For a new release I only change the Docker Image Tag Name and hence a new revision will be created and this will recreate the containers using the new Docker Image and **Application loadbalancer will route 100% traffic to the newly created containers**.

```
  traffic {
    type = "TRAFFIC_TARGET_ALLOCATION_TYPE_LATEST"
    percent = 100
  }
```

Here I had used the standard SSL certificate, you can provide your SSL Certificate in the file certificate.crt and key in the file mykey.key at the path **main** in the terraform script present in the GitHub Repo https://github.com/singhritesh85/terraform-docker-cloudrun-the-serverless-way.git. Terraform state files will be presented in the GCP Bucket.

First of all, I will create the GitLab set-up using the terraform script present in the GitHub Repo https://github.com/singhritesh85/terraform-gitlab-server-gcp.git. Using GCP Application Load Balancer GitLab will be accessed and here I had used GCP Certificate Manager based SSL Certificate which will be auto renewed after 90 days. After running the terraform apply command you will see that GCP DNS Zone will be created and do the entry for this DNS Zone Nameserver in your Domain Name provider Nameserver (for this activity I introduced a time delay of 150 seconds). Then to validate the GCP certificate manager SSL Certificate a record set will be created in the GCP Cloud DNS Zone, to validate GCP Certificate manager SSL Certificate will take 8-10 minutes and this entire terraform script execution will take around 20 minutes.

Commands as written below had been used to create infrastructure using Terraform.

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

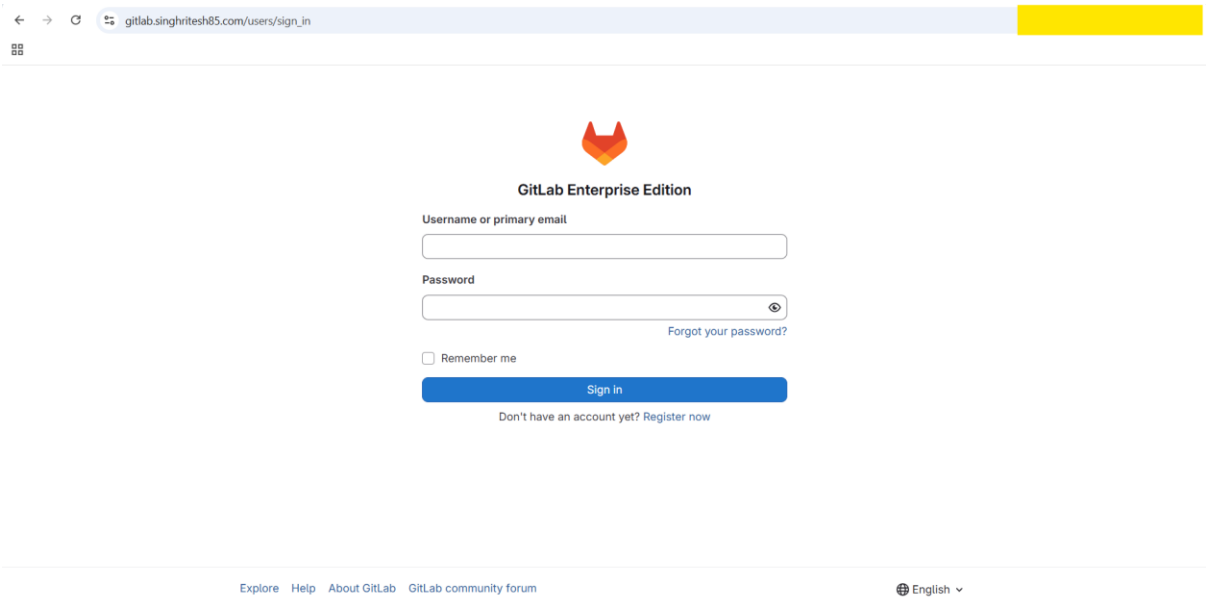
**terraform validate** --------> verify that terraform configuration file is correct or not

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

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

```
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: postgres-exporter: (pid 91843) 60s; run: log: (pid 91653) 158s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: postgresql: (pid 90866) 414s; run: log: (pid 90883) 413s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: prometheus: (pid 91820) 64s; run: log: (pid 91577) 169s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: puma: (pid 91232) 235s; run: log: (pid 91239) 234s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: redis: (pid 90729) 431s; run: log: (pid 90739) 430s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: redis-exporter: (pid 91811) 65s; run: log: (pid 91532) 180s
module.gcp_cloud_dns_zone.null_resource.gitlab_server (remote-exec): run: sidekiq: (pid 91269) 229s; run: log: (pid 91277) 228s
module.gcp_cloud_dns_zone.null_resource.gitlab_server: Creation complete after 19m11s [id=7[redacted]4]
module.gcp_cloud_dns_zone.google_dns_record_set.a_record: Creating...
module.gcp_cloud_dns_zone.google_dns_record_set.a_record: Creation complete after 2s [id=projects/[redacted]6/managedZones/gitlab-public-zone/rrsets/gitlab.singhritesh85.com./A]

Apply complete! Resources: 33 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_dns_zone_vm_instances_and_gcp_alb_details = {
  "cloud_dns_zone_id" = "projects/[redacted]5/managedZones/gitlab-public-zone"
  "gcp_alb_public_ip" = "[redacted]"
  "gcp_cloud_dns_zone_name_servers" = tolist([
    "[redacted]",
    "[redacted]",
    "[redacted]",
    "[redacted]",
  ])
  "gcp_vm_instances_name" = [
    "gitlab-server",
    "gitlab-gitlab-runner",
  ]
  "gcp_vm_private_ip_address" = [
    "172.20.[redacted]",
    "172.20.[redacted]",
  ]
  "gcp_vm_public_ip_address" = [
    "[redacted]",
    "[redacted]",
  ]
}
```

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




You can Login with root user and the root user password which is present at the path **/etc/gitlab/initial_root_password** on GitLab-Server.

```
[root@gitlab-server ~]# cat /etc/gitlab/initial_root_password
# WARNING: This password is only valid if ALL of the following are true:
#          • You set it manually via the GITLAB_ROOT_PASSWORD environment variable
#            OR the gitlab_rails['initial_root_password'] setting in /etc/gitlab/gitlab.rb
#          • You set it BEFORE the initial database setup (typically during first installation)
#          • You have NOT changed the password since then (via web UI or command line)
#
#          If this password doesn't work, reset the admin password using:
#          https://docs.gitlab.com/security/reset_user_password/#reset-the-root-password

Password: 5                                                    =

# NOTE: This file is automatically deleted after 24 hours on the next reconfigure run.
```

After first Login I changed the root user's username and password and then re-login with new username and password. To know more about GitLab you can refer the project present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-CICD-using-GitLab-GCP-and-AWS.git.

Then created GCP Artifact Registry (GAR) using terraform as shown in the screenshot attached below.

Commands as written below had been used to create infrastructure using Terraform.

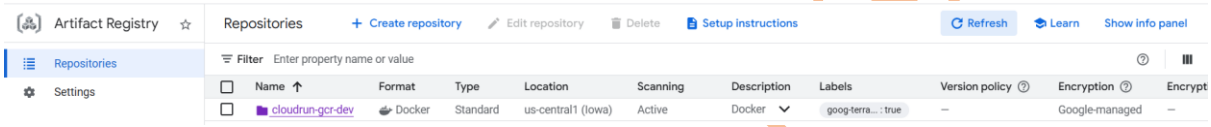
**terraform init** -------> initializes a working directory containing configuration files and installs plugins for required providers.

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

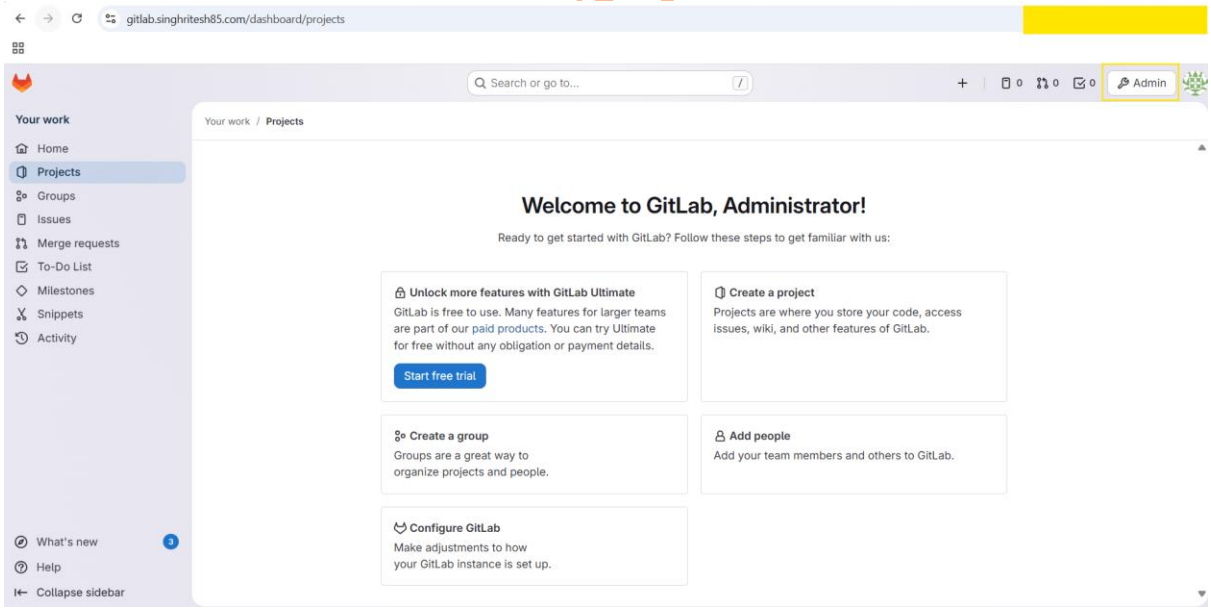
**terraform plan** ---------> Check which resources are going to be created.

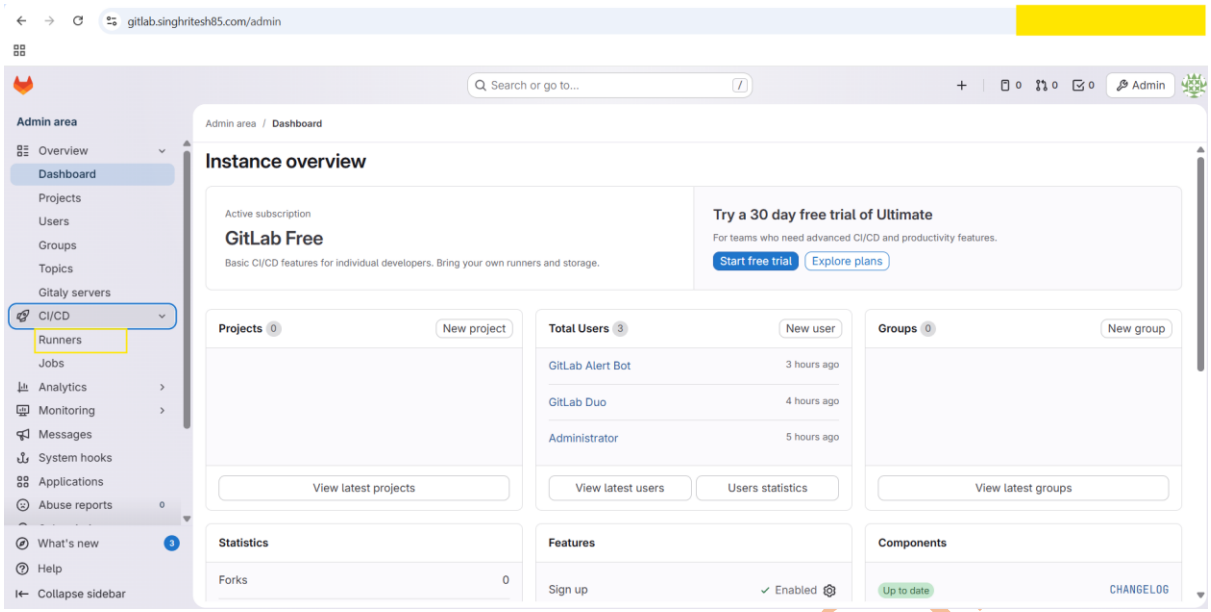
Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

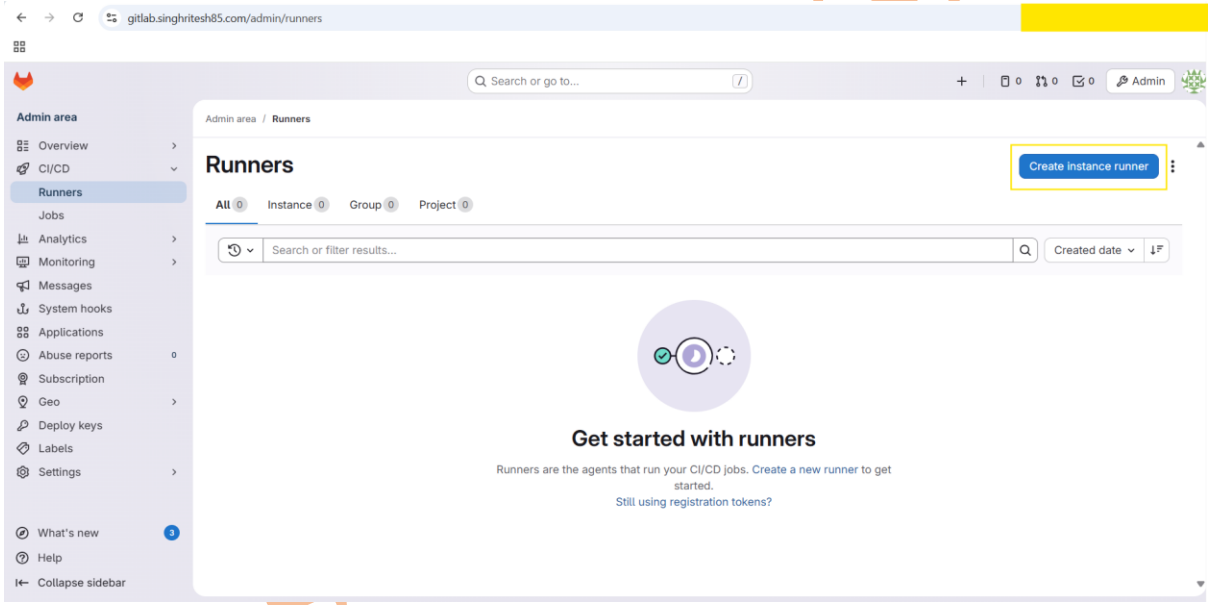
```
      + location         = "us-central1"
      + mode             = "STANDARD_REPOSITORY"
      + name             = (known after apply)
      + project          = "w                  6"
      + registry_uri     = (known after apply)
      + repository_id    = "cloudrun-gcr-dev"
      + terraform_labels = {
          + "goog-terraform-provisioned" = "true"
        }
      + update_time      = (known after apply)

      + vulnerability_scanning_config {
          + enablement_config       = "INHERITED"
          + enablement_state        = (known after apply)
          + enablement_state_reason = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + gcp_artifact_registry_detials = {
      + repository_endpoint = (known after apply)
      + repository_url      = (known after apply)
    }
module.artifact_registry.google_artifact_registry_repository.google_container_registry: Creating...
module.artifact_registry.google_artifact_registry_repository.google_container_registry: Still creating... [00m10s elapsed]
module.artifact_registry.google_artifact_registry_repository.google_container_registry: Creation complete after 10s [id=projects/w                  6/locations
/us-central1/repositories/cloudrun-gcr-dev]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

gcp_artifact_registry_detials = {
  "repository_endpoint" = "us-central1-docker.pkg.dev/w                  6/cloudrun-gcr-dev"
  "repository_url" = "projects/w                  6/locations/us-central1/repositories/cloudrun-gcr-dev"
}
```

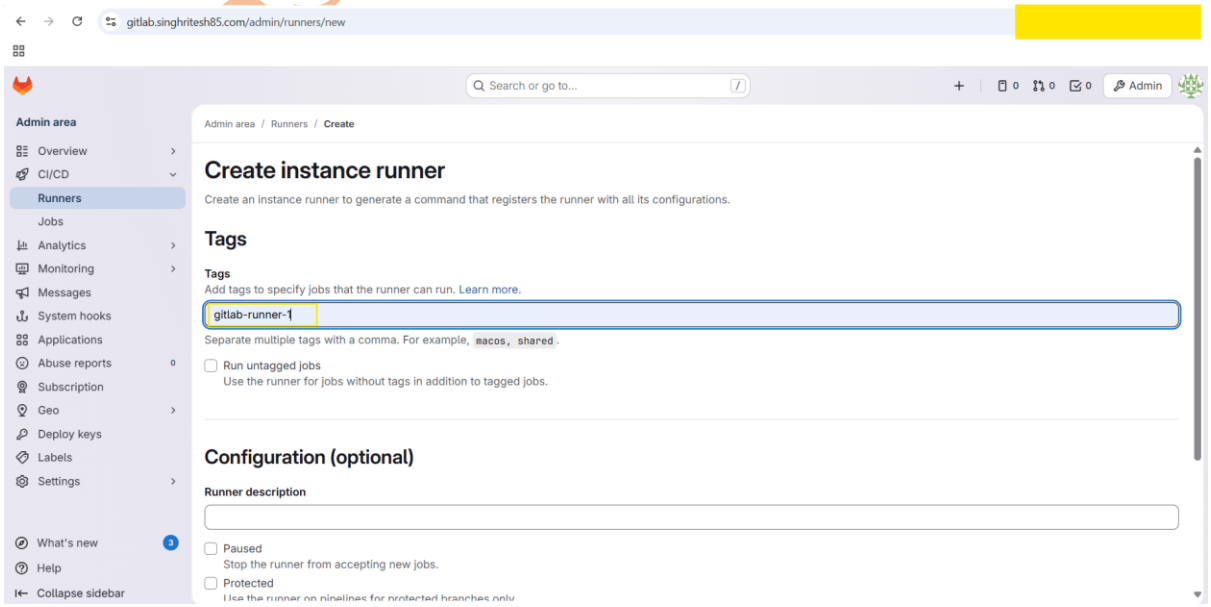


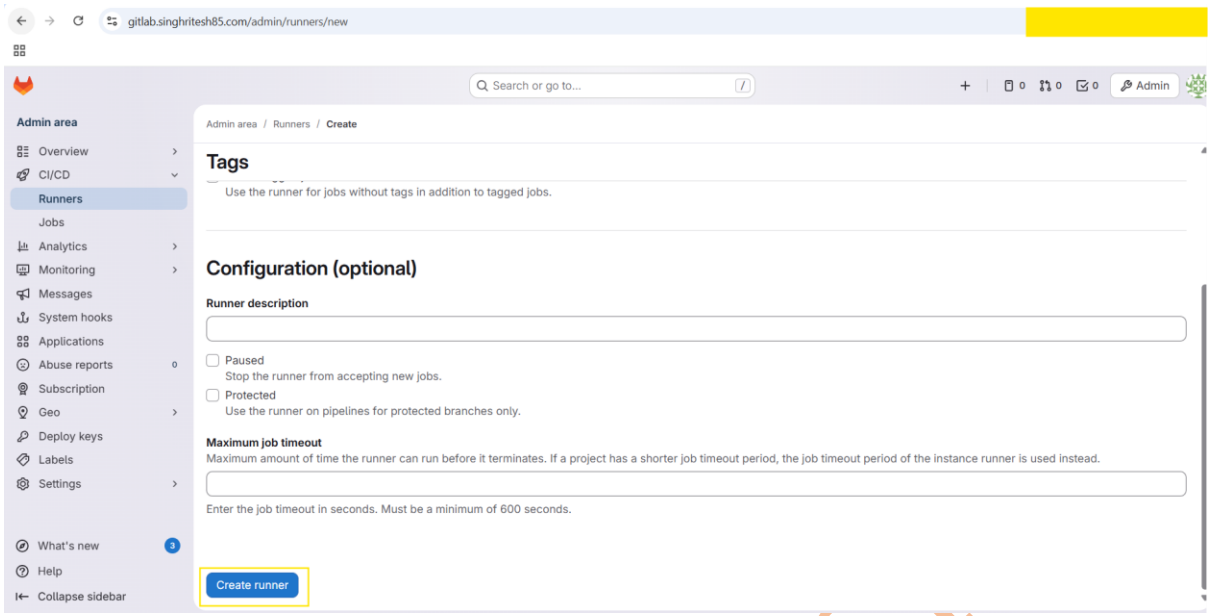

Then added the GitLab Runner as shown in the screenshot attached below. This GitLab Runner server was created along with GitLab Server using terraform script. Go to Admin area > CI/CD > Runners as shown in the screenshot attached below.

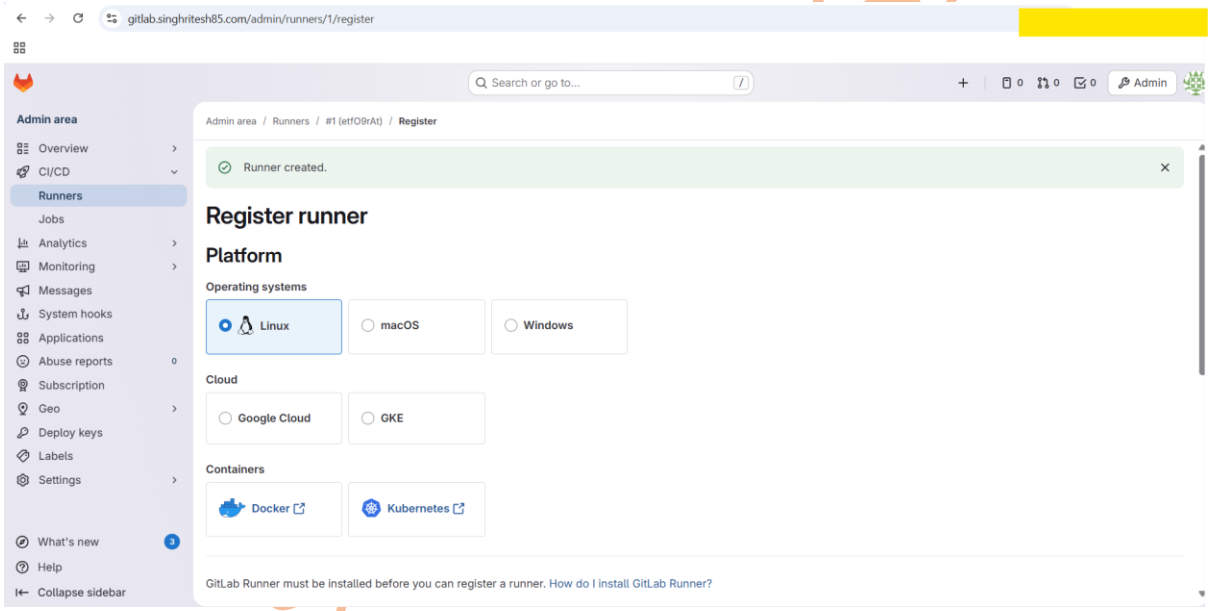


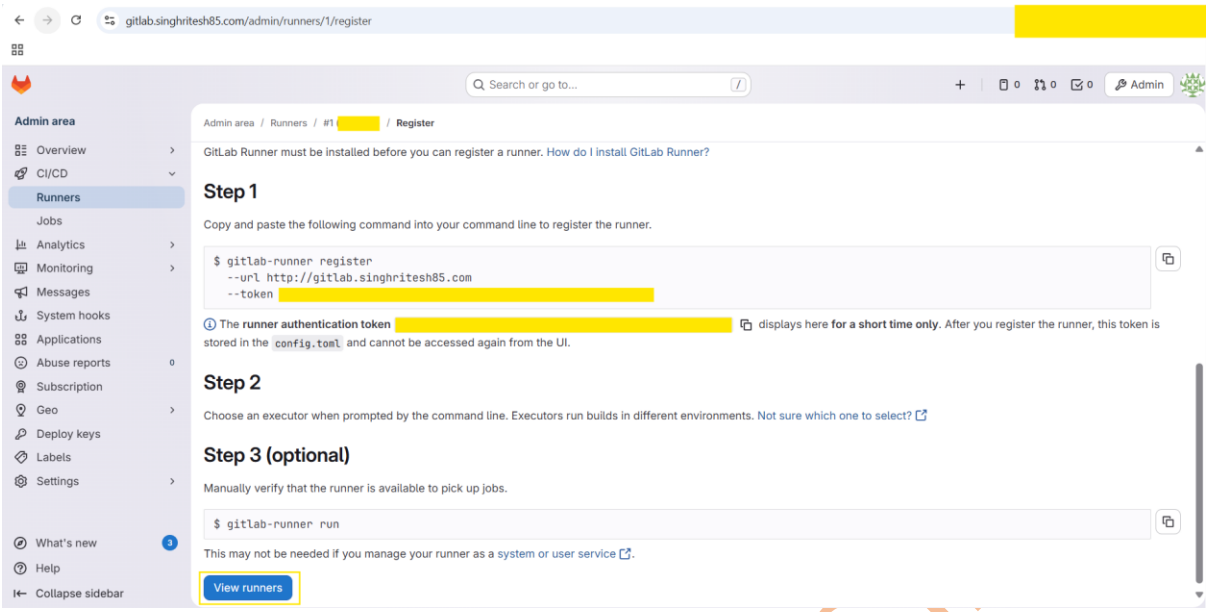

gitlab.singhritesh85.com/admin
Search or go to...
Admin
Admin area
Overview
Dashboard
Projects
Users
Groups
Topics
Gitaly servers
CI/CD
Runners
Jobs
Analytics
Monitoring
Messages
System hooks
Applications
Abuse reports
What's new
Help
Collapse sidebar
Admin area / Dashboard
Instance overview
Active subscription
GitLab Free
Basic CI/CD features for individual developers. Bring your own runners and storage.
Try a 30 day free trial of Ultimate
For teams who need advanced CI/CD and productivity features.
Start free trial
Explore plans
Projects 0
New project
Total Users 3
New user
Groups 0
New group
GitLab Alert Bot
3 hours ago
GitLab Duo
4 hours ago
Administrator
5 hours ago
View latest projects
View latest users
Users statistics
View latest groups
Statistics
Forks 0
Features
Sign up
Enabled
Components
Up to date
CHANGELOG


gitlab.singhritesh85.com/admin/runners
Search or go to...
Admin
Admin area
Overview
CI/CD
Runners
Jobs
Analytics
Monitoring
Messages
System hooks
Applications
Abuse reports
Subscription
Geo
Deploy keys
Labels
Settings
What's new
Help
Collapse sidebar
Admin area / Runners
Runners
Create instance runner
All 0
Instance 0
Group 0
Project 0
Search or filter results...
Created date
Get started with runners
Runners are the agents that run your CI/CD jobs. Create a new runner to get started.
Still using registration tokens?

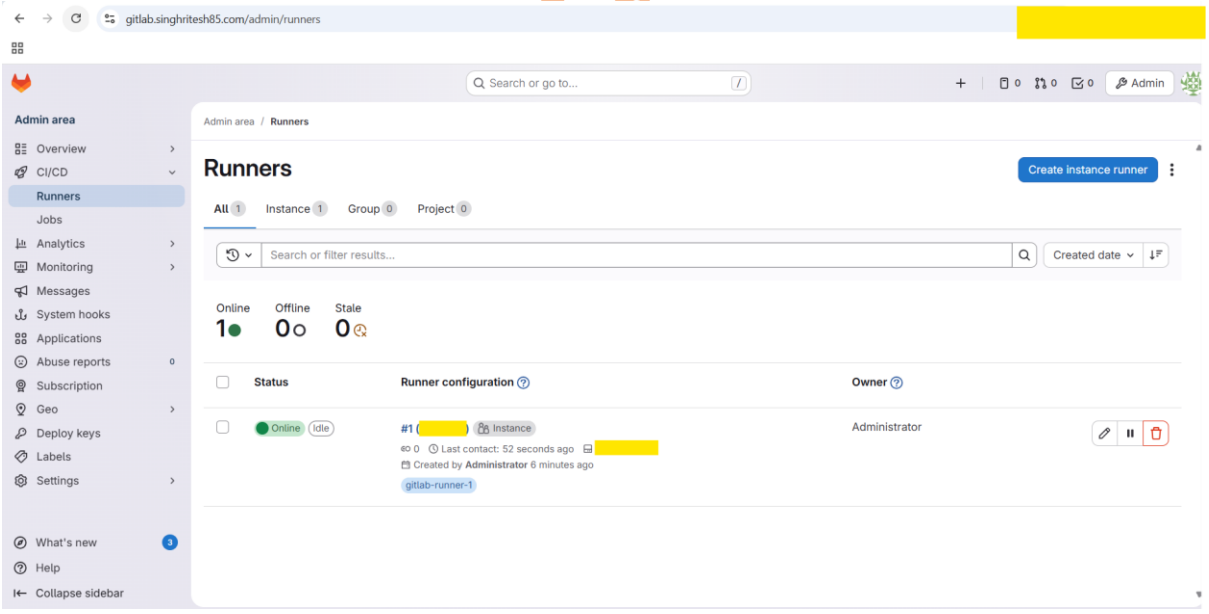

gitlab.singhritesh85.com/admin/runners/new
Search or go to...
Admin
Admin area
Overview
CI/CD
Runners
Jobs
Analytics
Monitoring
Messages
System hooks
Applications
Abuse reports
Subscription
Geo
Deploy keys
Labels
Settings
What's new
Help
Collapse sidebar
Admin area / Runners / Create
Create instance runner
Create an instance runner to generate a command that registers the runner with all its configurations.
Tags
Tags
Add tags to specify jobs that the runner can run. Learn more.
gitlab-runner-1
Separate multiple tags with a comma. For example, macos, shared.
Run untagged jobs
Use the runner for jobs without tags in addition to tagged jobs.
Configuration (optional)
Runner description
Paused
Stop the runner from accepting new jobs.
Protected


gitlab.singhritesh85.com/admin/runners/new
Search or go to...
Admin
Admin area
Overview
CI/CD
Runners
Jobs
Analytics
Monitoring
Messages
System hooks
Applications
Abuse reports
Subscription
Geo
Deploy keys
Labels
Settings
What's new
Help
Collapse sidebar
Admin area / Runners / Create
Tags
Use the runner for jobs without tags in addition to tagged jobs.
Configuration (optional)
Runner description
Paused
Stop the runner from accepting new jobs.
Protected
Use the runner on pipelines for protected branches only.
Maximum job timeout
Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.
Enter the job timeout in seconds. Must be a minimum of 600 seconds.
Create runner

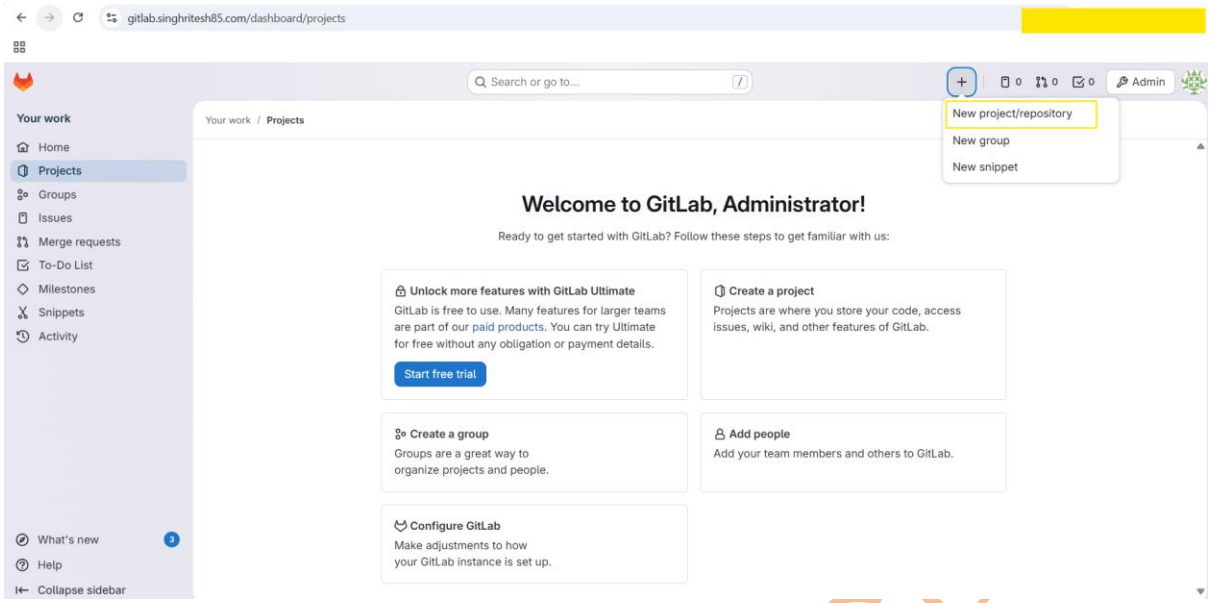

gitlab.singhritesh85.com/admin/runners/1/register
Search or go to...
Admin
Admin area
Overview
CI/CD
Runners
Jobs
Analytics
Monitoring
Messages
System hooks
Applications
Abuse reports
Subscription
Geo
Deploy keys
Labels
Settings
What's new
Help
Collapse sidebar
Admin area / Runners / #1 (etfO9rAt) / Register
Runner created.
Register runner
Platform
Operating systems
Linux
macOS
Windows
Cloud
Google Cloud
GKE
Containers
Docker
Kubernetes
GitLab Runner must be installed before you can register a runner. How do I install GitLab Runner?

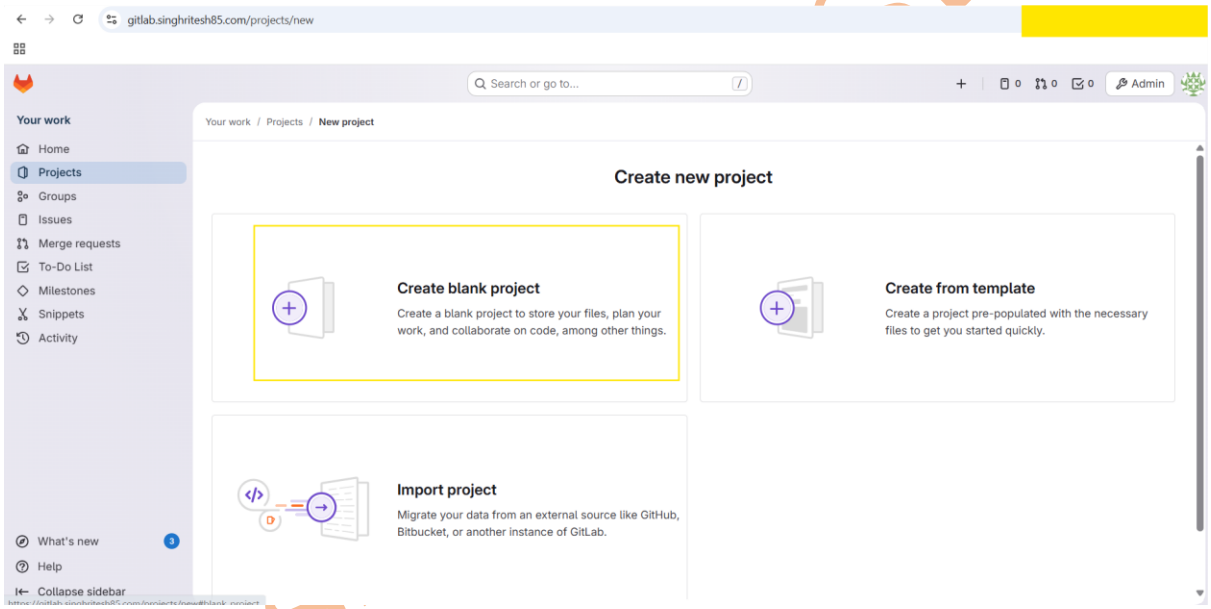


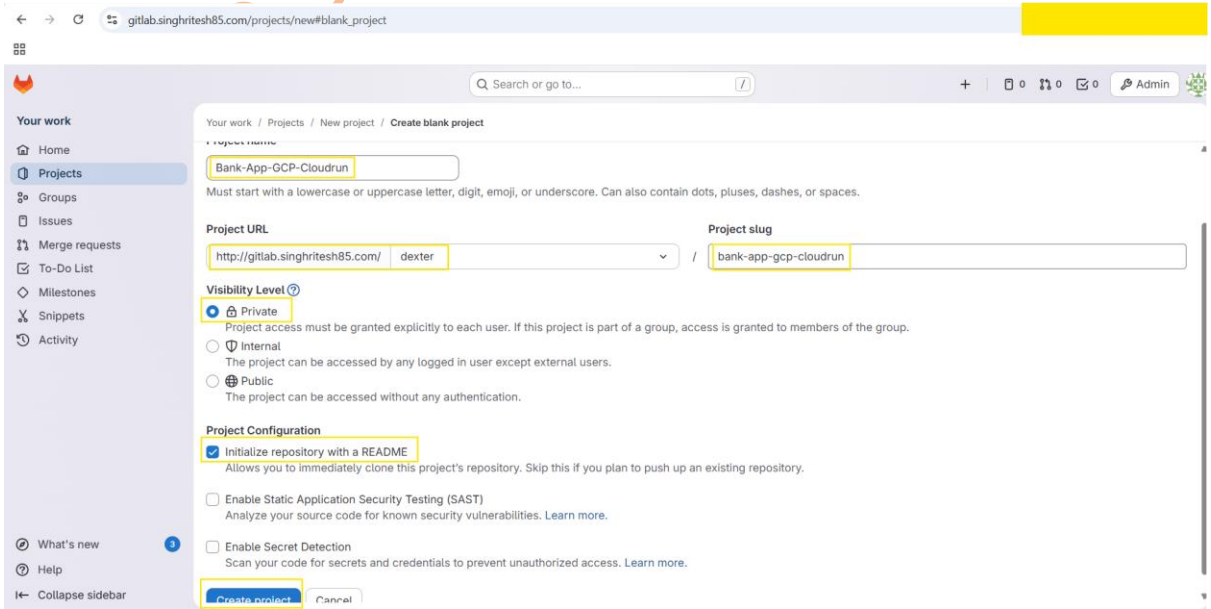

```
[root@gitlab-gitlab-runner ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=97575 revision=cc7f9277 version=18.7.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.singhritesh85.com
Enter the registration token:
g                                                   3
Verifying runner... is valid                        correlation_id=0                                    a
Enter a name for the runner. This is stored only in the local config.toml file:
[gitlab-gitlab-runner]: gitlab-runner-1
Enter an executor: custom, shell, ssh, virtualbox, docker, docker-windows, docker+machine, docker-autoscaler, parallels, kubernetes, instance:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

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



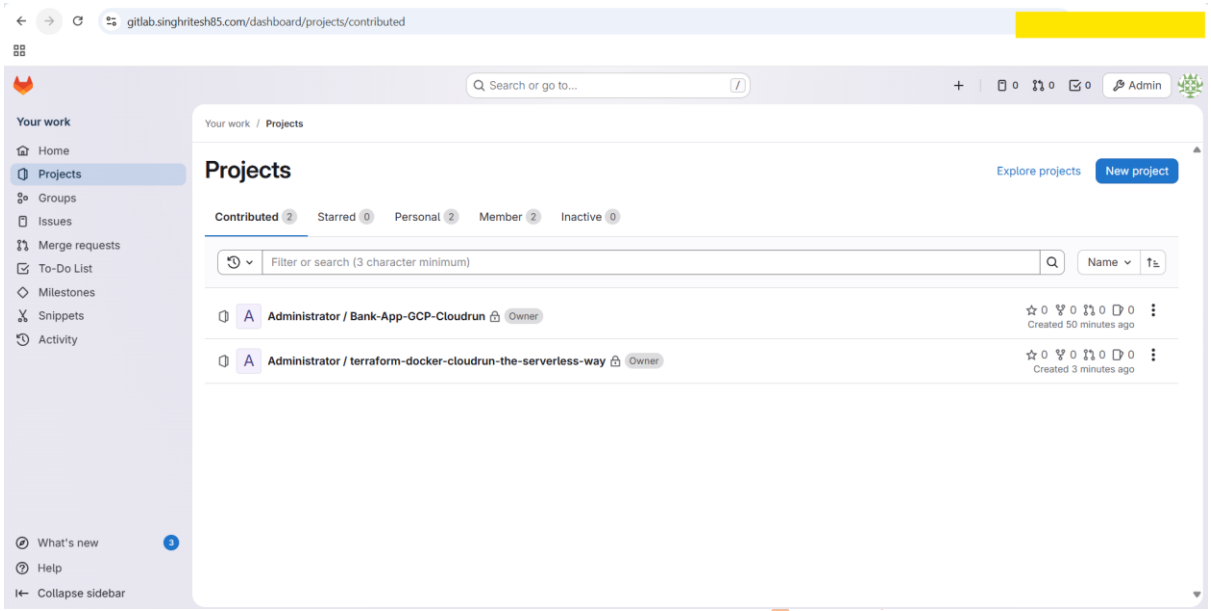

Created the first repository in GitLab as shown in the screenshot attached below.

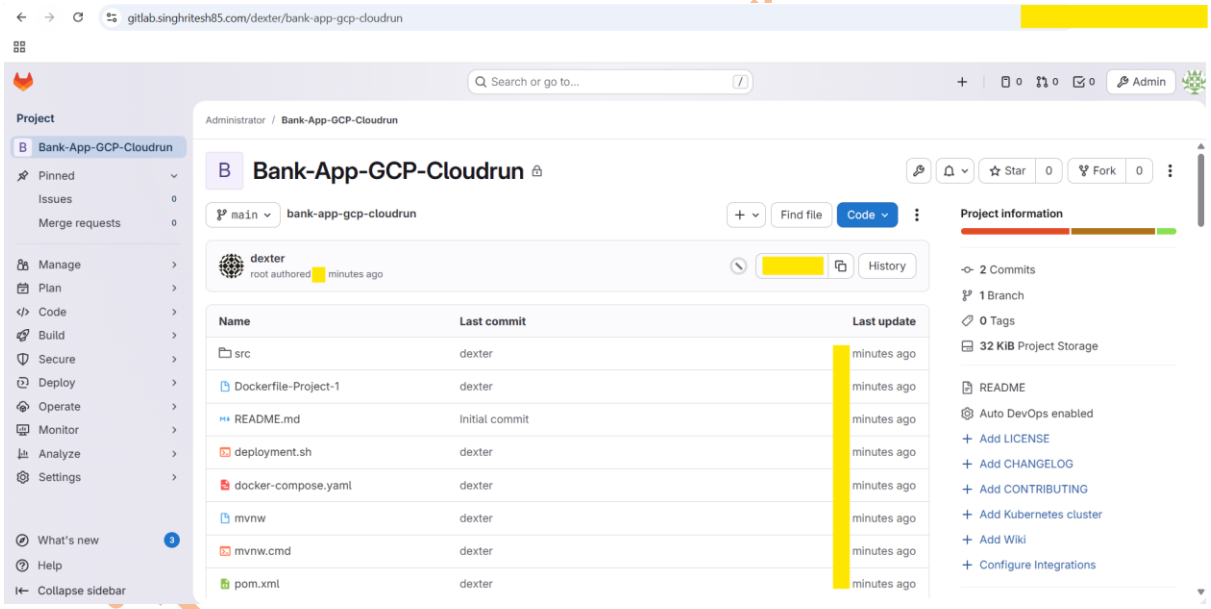

gitlab.singhritesh85.com/dashboard/projects
Search or go to...
Admin
New project/repository
New group
New snippet
Your work
Home
Projects
Groups
Issues
Merge requests
To-Do List
Milestones
Snippets
Activity
What's new
Help
Collapse sidebar
Your work / Projects
Welcome to GitLab, Administrator!
Ready to get started with GitLab? Follow these steps to get familiar with us:
Unlock more features with GitLab Ultimate
GitLab is free to use. Many features for larger teams are part of our paid products. You can try Ultimate for free without any obligation or payment details.
Start free trial
Create a project
Projects are where you store your code, access issues, wiki, and other features of GitLab.
Create a group
Groups are a great way to organize projects and people.
Add people
Add your team members and others to GitLab.
Configure GitLab
Make adjustments to how your GitLab instance is set up.


gitlab.singhritesh85.com/projects/new
Search or go to...
Admin
Your work
Home
Projects
Groups
Issues
Merge requests
To-Do List
Milestones
Snippets
Activity
What's new
Help
Collapse sidebar
Your work / Projects / New project
Create new project
Create blank project
Create a blank project to store your files, plan your work, and collaborate on code, among other things.
Create from template
Create a project pre-populated with the necessary files to get you started quickly.
Import project
Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.


gitlab.singhritesh85.com/projects/new#blank_project
Search or go to...
Admin
Your work
Home
Projects
Groups
Issues
Merge requests
To-Do List
Milestones
Snippets
Activity
What's new
Help
Collapse sidebar
Your work / Projects / New project / Create blank project
Bank-App-GCP-Cloudrun
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.
Project URL
http://gitlab.singhritesh85.com/
dexter
Project slug
bank-app-gcp-cloudrun
Visibility Level
Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
Internal
The project can be accessed by any logged in user except external users.
Public
The project can be accessed without any authentication.
Project Configuration
Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. Learn more.
Enable Secret Detection
Scan your code for secrets and credentials to prevent unauthorized access. Learn more.
Create project
Cancel

For this project I had created two repository one to keep the source code and another to keep the terraform script.

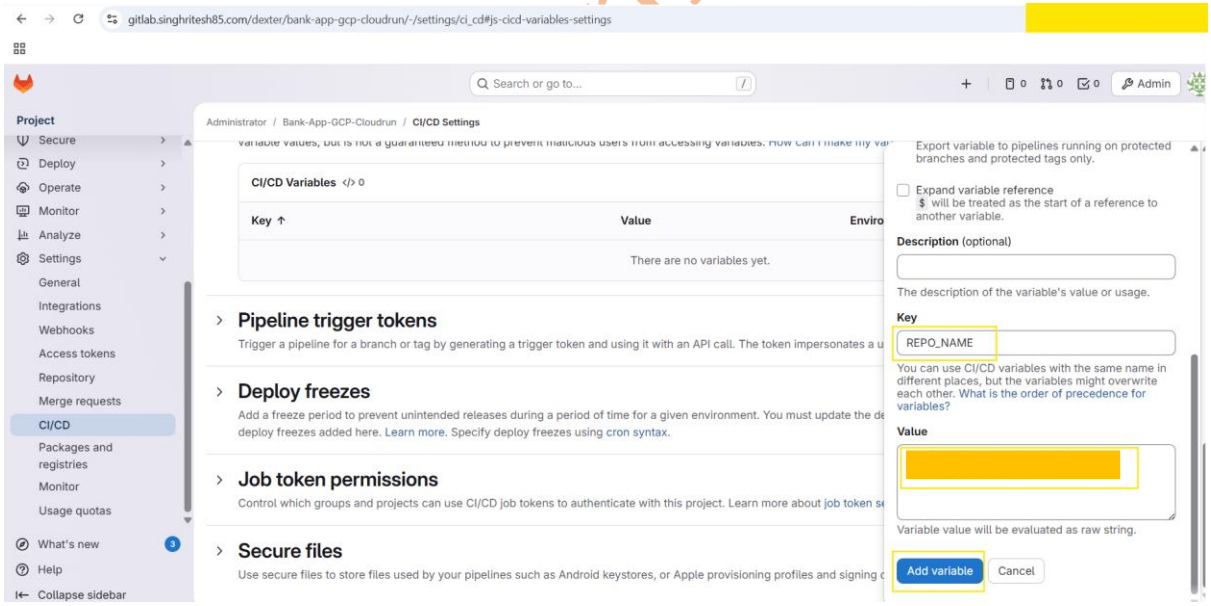


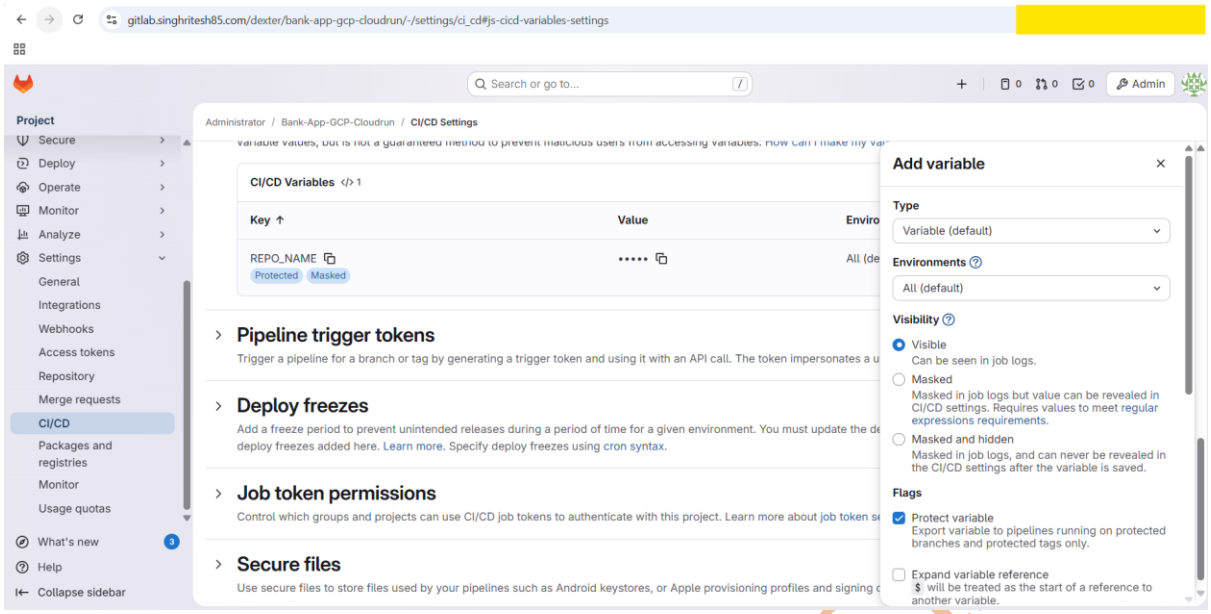



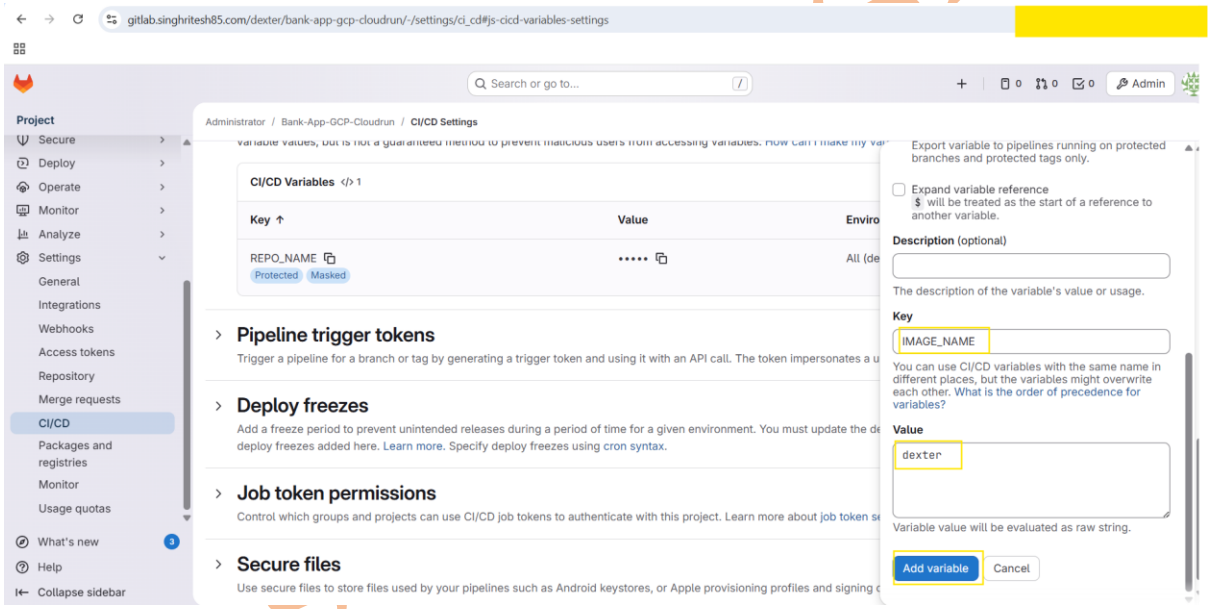

Now go to gitlab-runner instance and update the file **/etc/gitlab-runner/config.toml** with the line **clone_url** as shown in the screenshot attached below.

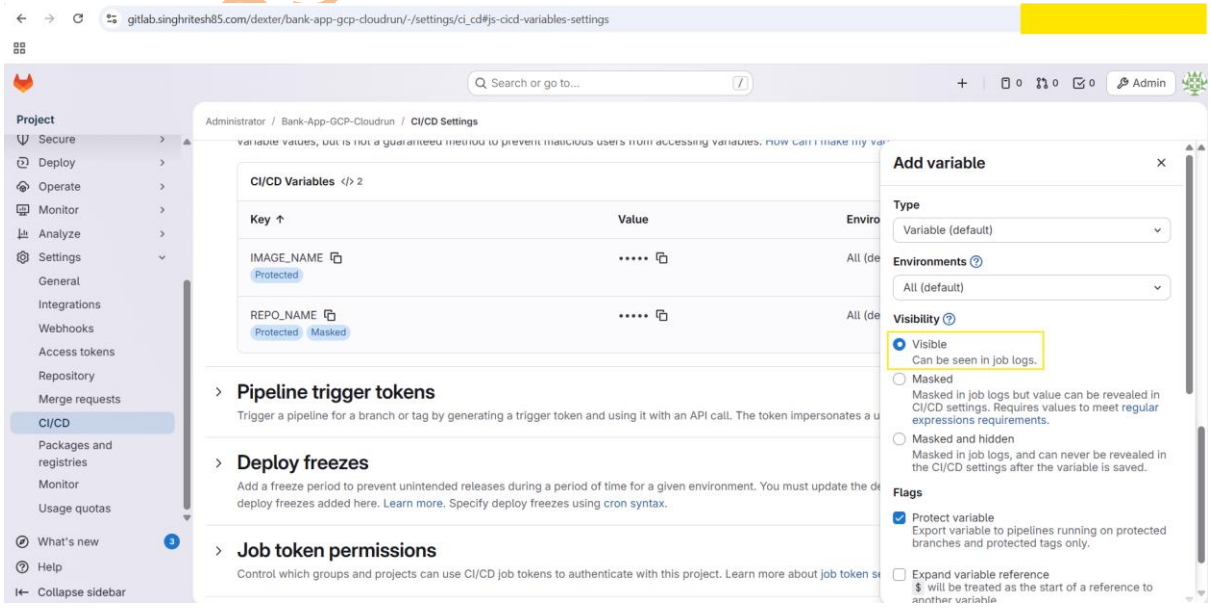
```
concurrent = 1
check_interval = 0
connection_max_age = "15m0s"
shutdown_timeout = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "gitlab-runner-1"
  url = "https://gitlab.singhritesh85.com"
  clone_url = "https://gitlab.singhritesh85.com"
  id = 2
  token = "                                                                    "
  token_obtained_at = 2025-
  token_expires_at = 0001-01-01
  executor = "shell"
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

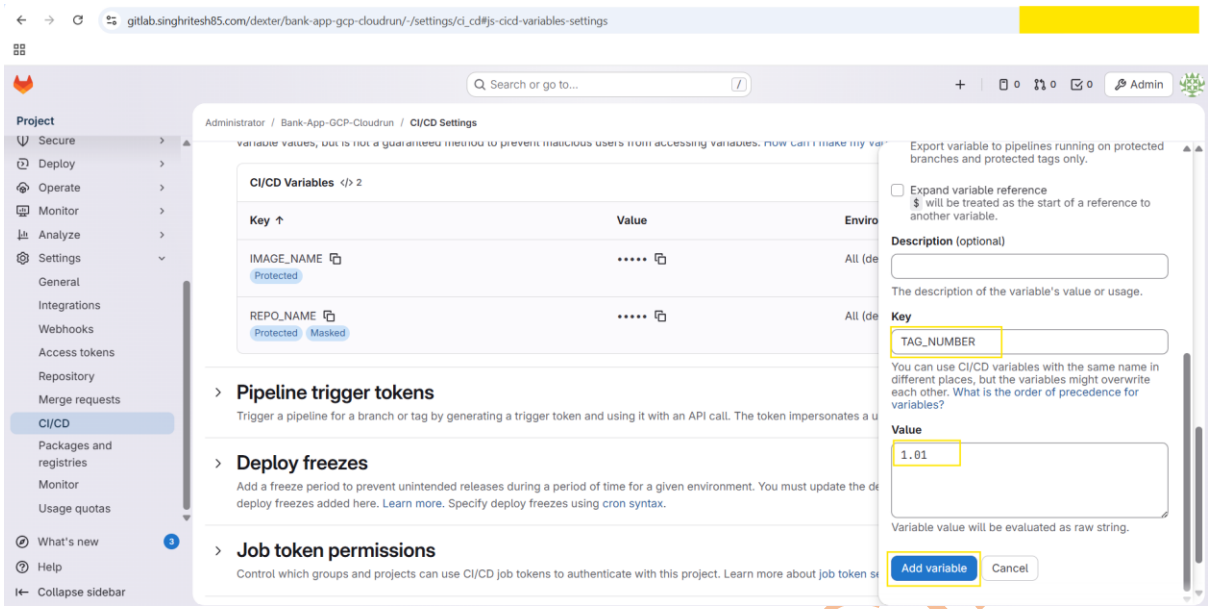
You can set the variables in the GitLab Repository as shown in the screenshot attached below. **Go to GitLab Repo > Settings > CI/CD > Variables**. Then create variables (Masked) as shown below.

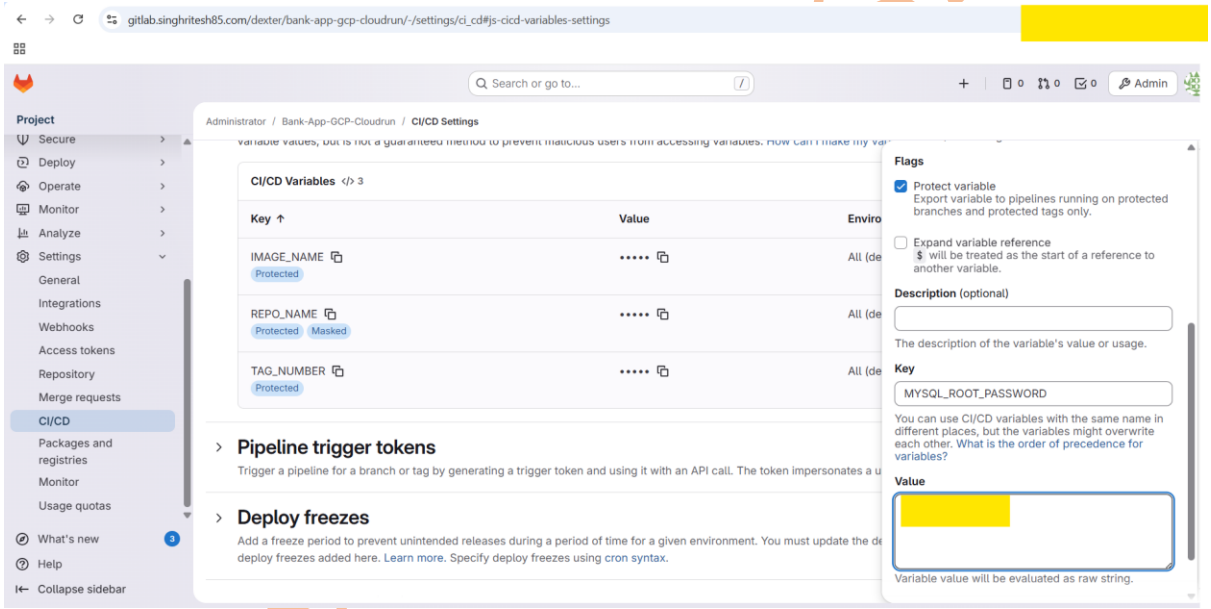


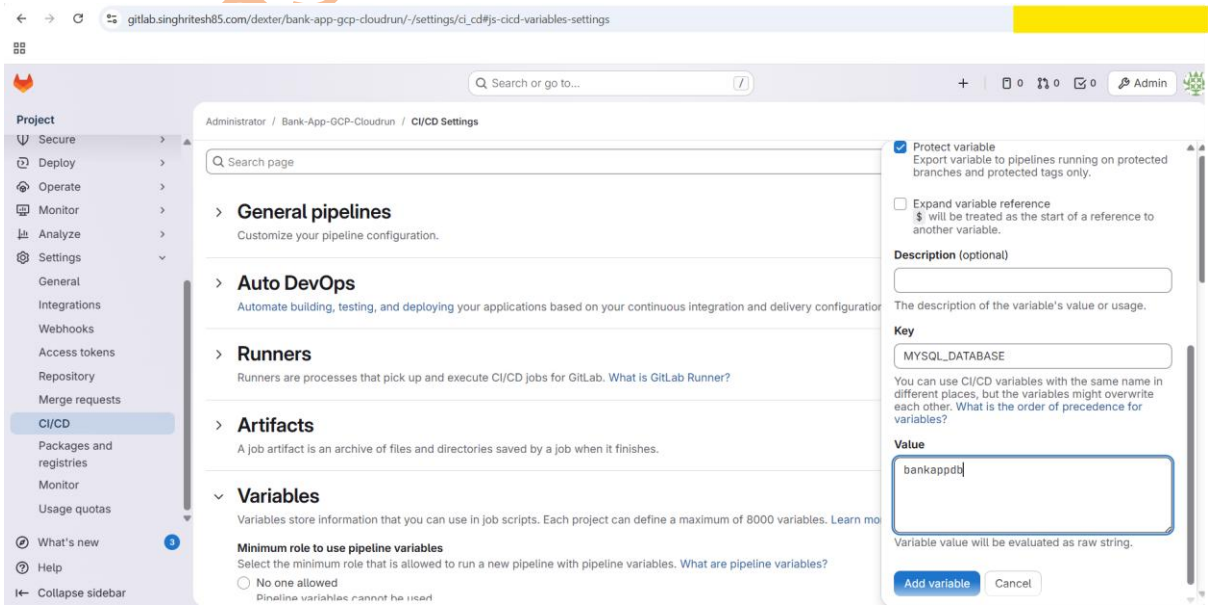

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Secure
Deploy
Operate
Monitor
Analyze
Settings
General
Integrations
Webhooks
Access tokens
Repository
Merge requests
CI/CD
Packages and registries
Monitor
Usage quotas
What's new
Help
Collapse sidebar
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
CI/CD Variables 1
Key
Value
REPO_NAME
Protected
Masked
Pipeline trigger tokens
Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a u
Deploy freezes
Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the de
deploy freezes added here. Learn more. Specify deploy freezes using cron syntax.
Job token permissions
Control which groups and projects can use CI/CD job tokens to authenticate with this project. Learn more about job token s
Secure files
Use secure files to store files used by your pipelines such as Android keystores, or Apple provisioning profiles and signing
Add variable
Type
Variable (default)
Environments
All (default)
Visibility
Visible
Can be seen in job logs.
Masked
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.
Masked and hidden
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.
Flags
Protect variable
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.

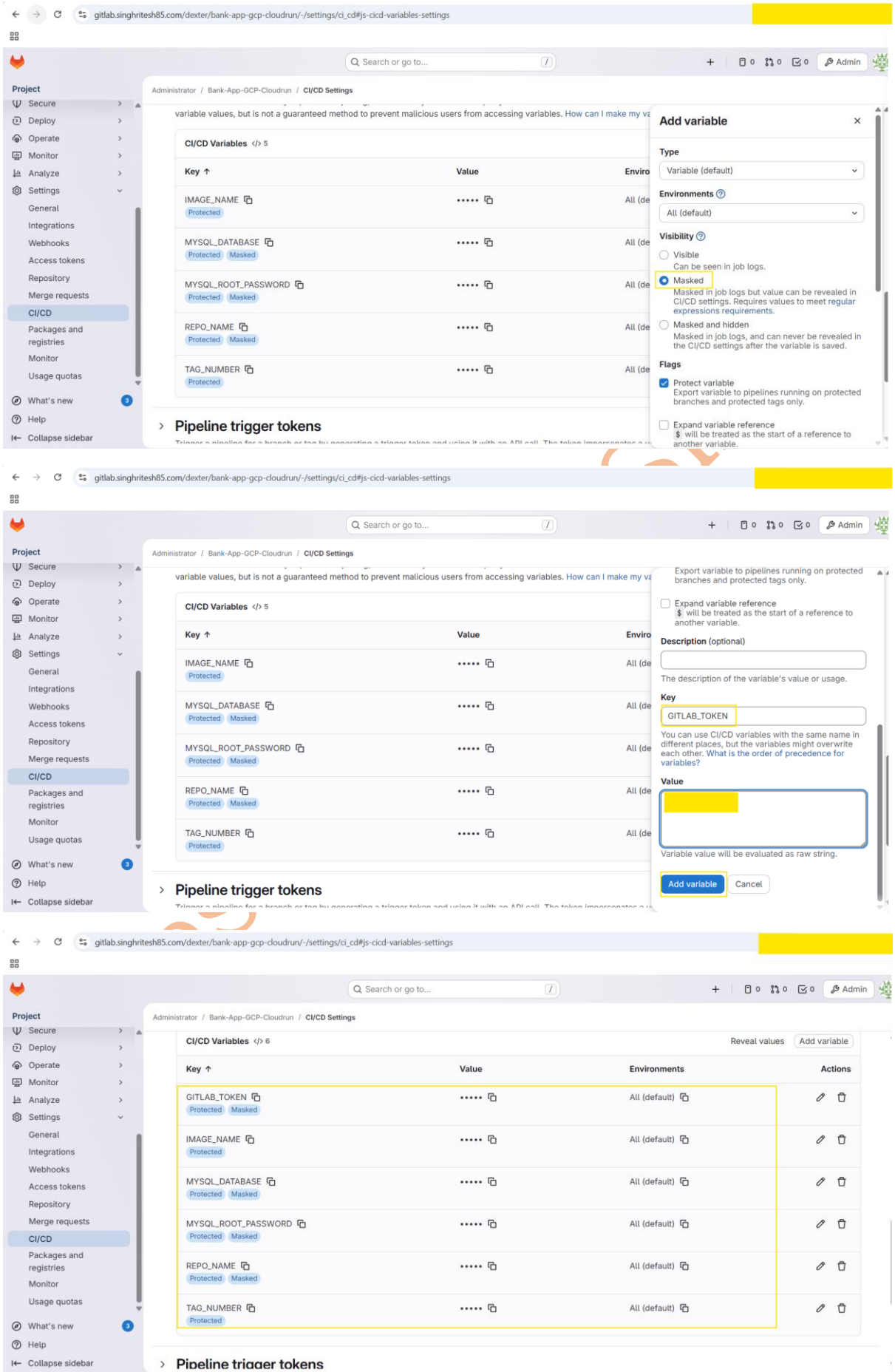

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
CI/CD Variables 1
Key
Value
REPO_NAME
Protected
Masked
Pipeline trigger tokens
Deploy freezes
Job token permissions
Secure files
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.
Description (optional)
The description of the variable's value or usage.
Key
IMAGE_NAME
You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?
Value
dexter
Variable value will be evaluated as raw string.
Add variable
Cancel


gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
CI/CD Variables 2
Key
Value
IMAGE_NAME
Protected
REPO_NAME
Protected
Masked
Pipeline trigger tokens
Deploy freezes
Job token permissions
Add variable
Type
Variable (default)
Environments
All (default)
Visibility
Visible
Can be seen in job logs.
Masked
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.
Masked and hidden
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.
Flags
Protect variable
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.


gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
Secure
Deploy
Operate
Monitor
Analyze
Settings
General
Integrations
Webhooks
Access tokens
Repository
Merge requests
CI/CD
Packages and registries
Monitor
Usage quotas
What's new
Help
Collapse sidebar
CI/CD Variables 2
Key
Value
IMAGE_NAME
Protected
REPO_NAME
Protected
Masked
Pipeline trigger tokens
Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a u
Deploy freezes
Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the de
deploy freezes added here. Learn more. Specify deploy freezes using cron syntax.
Job token permissions
Control which groups and projects can use CI/CD job tokens to authenticate with this project. Learn more about job token s
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.
Description (optional)
The description of the variable's value or usage.
Key
TAG_NUMBER
You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?
Value
1.01
Variable value will be evaluated as raw string.
Add variable
Cancel


gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
CI/CD Variables 3
Key
Value
IMAGE_NAME
Protected
REPO_NAME
Protected
Masked
TAG_NUMBER
Protected
Pipeline trigger tokens
Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a u
Deploy freezes
Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the de
deploy freezes added here. Learn more. Specify deploy freezes using cron syntax.
Flags
Protect variable
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.
Description (optional)
The description of the variable's value or usage.
Key
MYSQL_ROOT_PASSWORD
You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?
Value
Variable value will be evaluated as raw string.


gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings
Search or go to...
Admin
Project
Administrator / Bank-App-GCP-Cloudrun / CI/CD Settings
Search page
General pipelines
Customize your pipeline configuration.
Auto DevOps
Automate building, testing, and deploying your applications based on your continuous integration and delivery configuratio
Runners
Runners are processes that pick up and execute CI/CD jobs for GitLab. What is GitLab Runner?
Artifacts
A job artifact is an archive of files and directories saved by a job when it finishes.
Variables
Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. Learn mo
Minimum role to use pipeline variables
Select the minimum role that is allowed to run a new pipeline with pipeline variables. What are pipeline variables?
No one allowed
Protect variable
Export variable to pipelines running on protected branches and protected tags only.
Expand variable reference
$ will be treated as the start of a reference to another variable.
Description (optional)
The description of the variable's value or usage.
Key
MYSQL_DATABASE
You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?
Value
bankappdb
Variable value will be evaluated as raw string.
Add variable
Cancel




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

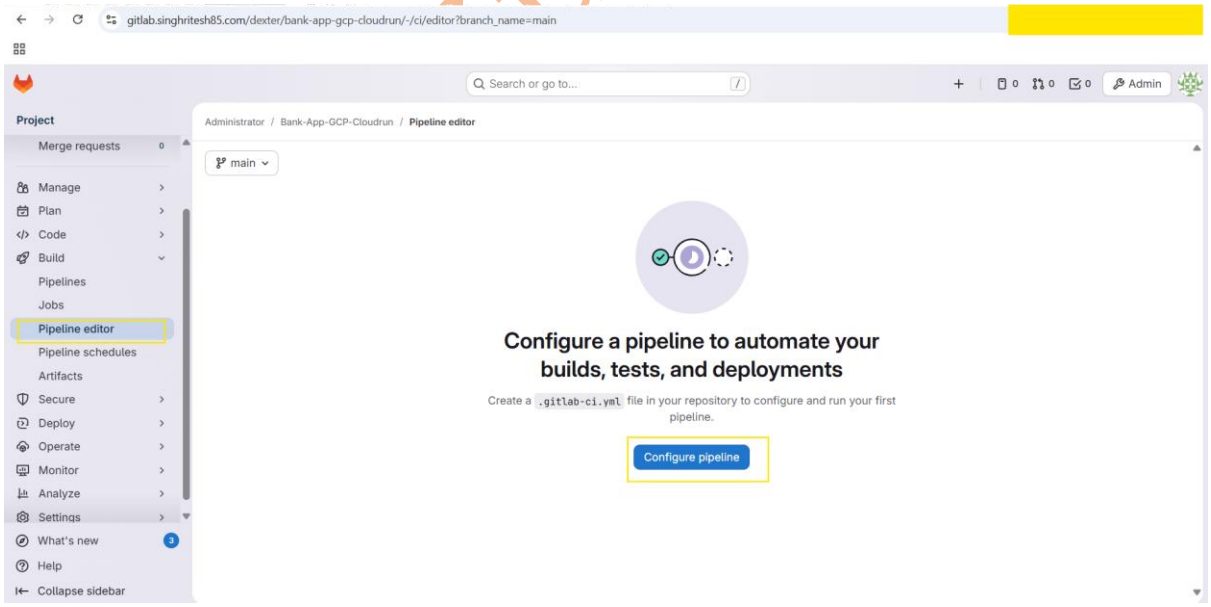
To push Docker Images from gitlab-runner to GCP Artifact Registry (earlier known as GCP Container Registry), you need to authenticate it first using the command as written below.

```
[gitlab-runner@gitlab-gitlab-runner ~]$ gcloud auth login

You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

You can run:

  $ gcloud config set account `ACCOUNT`

to switch accounts if necessary.

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

Do you want to continue (Y/n)?  Y

Go to the following link in your browser, and complete the sign-in prompts:

    https://

Once finished, enter the verification code provided in your browser:

You are now logged in as [                ].
Your current project is [                ].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```

**gcloud auth configure-docker us-central1-docker.pkg.dev**

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

Do you want to continue (Y/n)?  Y

Docker configuration file updated.
```

Created GitLab CI/CD pipeline as shown in the screenshot attached below.

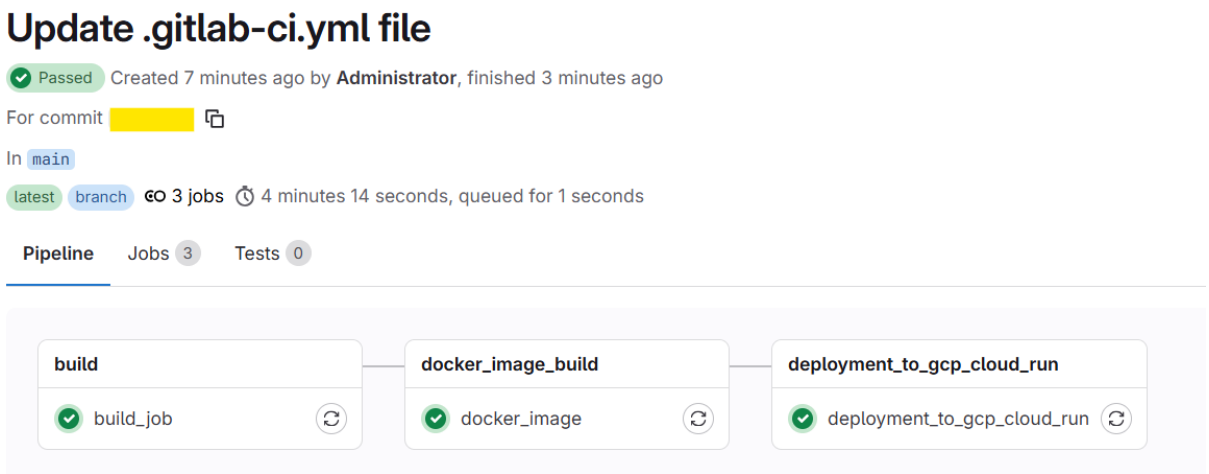


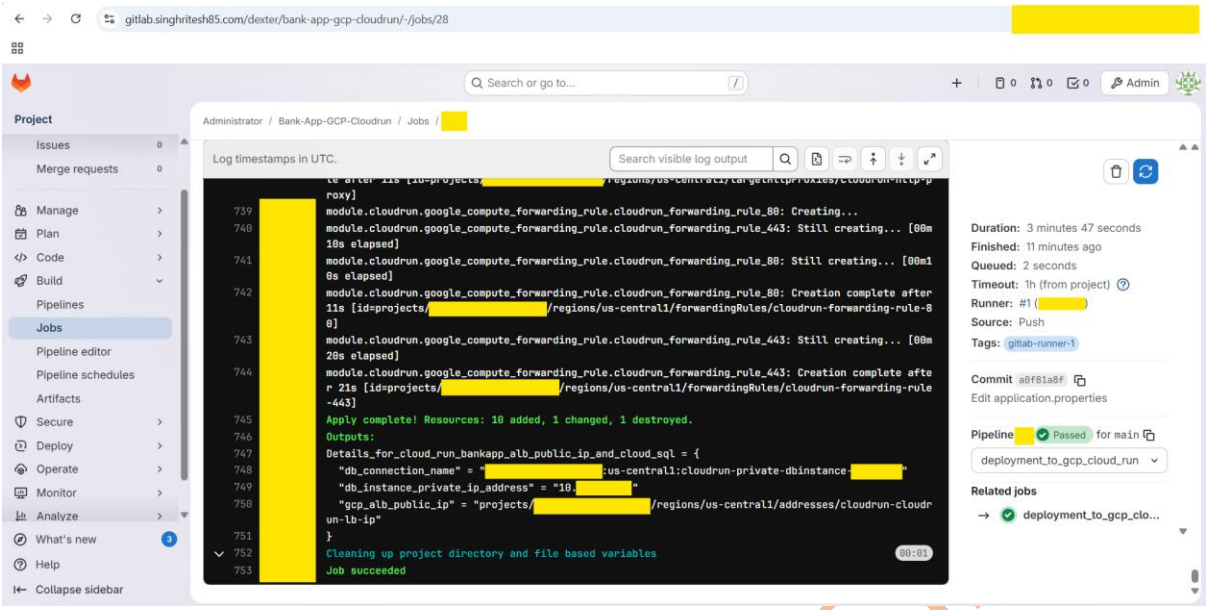

Copy and paste the **.gitlab-ci.yml** file present in the GitHub Repo https://github.com/singhritesh85/Bank-App-GCP-Cloudrun.git.

```
default:
  tags:
  - gitlab-runner-1

stages:
  - build
  - docker_image_build
  - deployment_to_gcp_cloud_run

build_job:
  stage: build
  before_script:
    - export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.17.0.10-1.el8.x86_64"
    - export PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
  script:
    - echo "Building artifact..."
    - mvn clean install
  artifacts:
    paths:
      - target/*.jar

docker_image:
  stage: docker_image_build
  before_script:
    - export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.17.0.10-1.el8.x86_64"
    - export PATH="$PATH:$JAVA_HOME/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/usr/local/bin"
    - mvn clean install
  script:
    - docker build -t $REPO_NAME/$IMAGE_NAME:$TAG_NUMBER -f Dockerfile-Project-1 .
    - trivy image --exit-code 0 --severity MEDIUM,HIGH $REPO_NAME/$IMAGE_NAME:$TAG_NUMBER
#    - trivy image --exit-code 1 --severity CRITICAL $REPO_NAME/$IMAGE_NAME:$TAG_NUMBER
    - docker push $REPO_NAME/$IMAGE_NAME:$TAG_NUMBER
  dependencies:
```

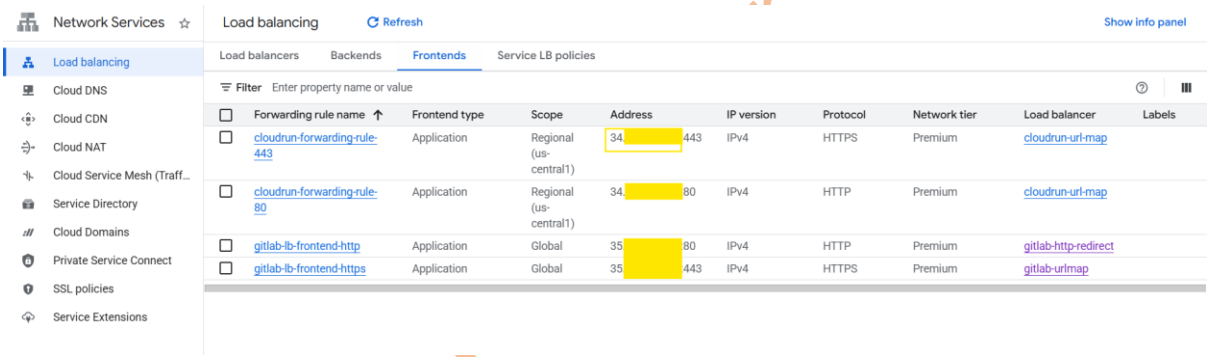
```
  - build_job


deployment_to_gcp_cloud_run:

 stage: deployment_to_gcp_cloud_run

 script:

  - git clone https://dexter:$GITLAB_TOKEN@gitlab.singhritesh85.com/dexter/terraform-docker-cloudrun-the-serverless-way.git

  - cd terraform-docker-cloudrun-the-serverless-way/main

  - terraform init

  - terraform plan -var REPO_NAME=$REPO_NAME -var IMAGE_NAME=$IMAGE_NAME -var TAG_NUMBER=$TAG_NUMBER -var MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD -var MYSQL_DATABASE=$MYSQL_DATABASE

  - terraform apply -auto-approve -var REPO_NAME=$REPO_NAME -var IMAGE_NAME=$IMAGE_NAME -var TAG_NUMBER=$TAG_NUMBER -var MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD -var MYSQL_DATABASE=$MYSQL_DATABASE

 dependencies:

  - docker_image


#  only:

#    - main # Or your desired branch
```

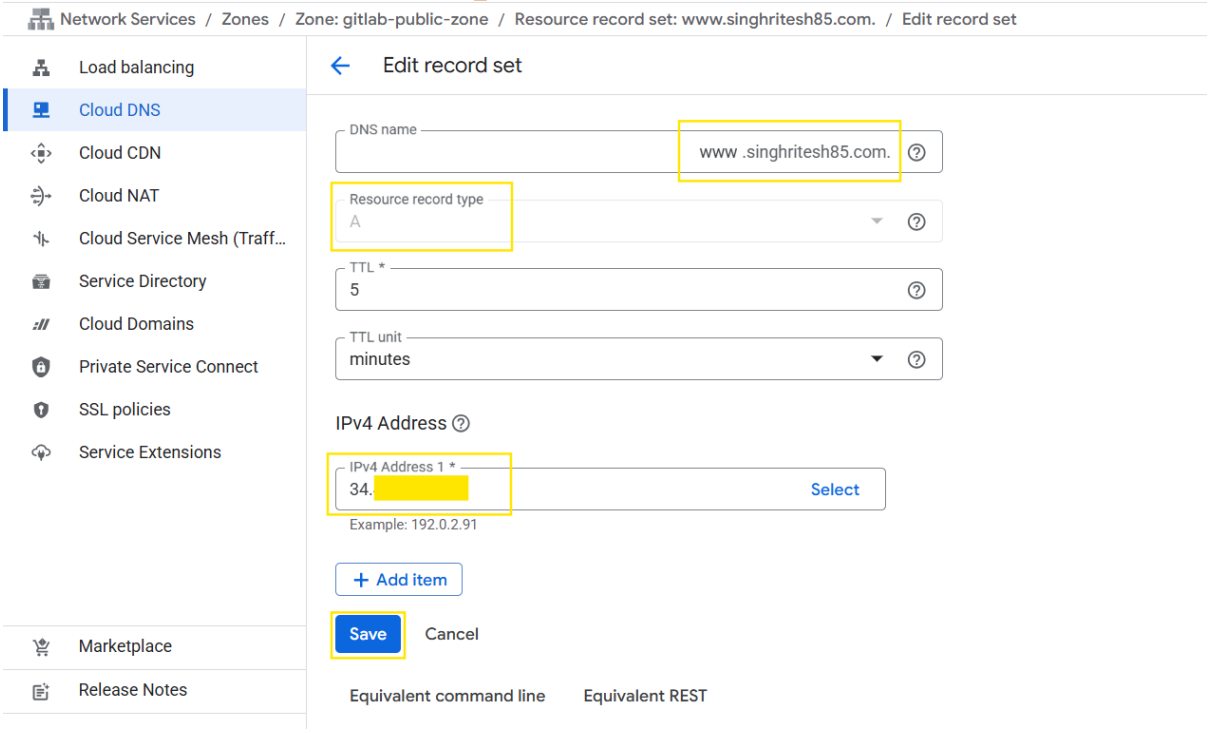
The GitLab CI/CD Pipeline executed successfully as shown in the screenshot attached below.



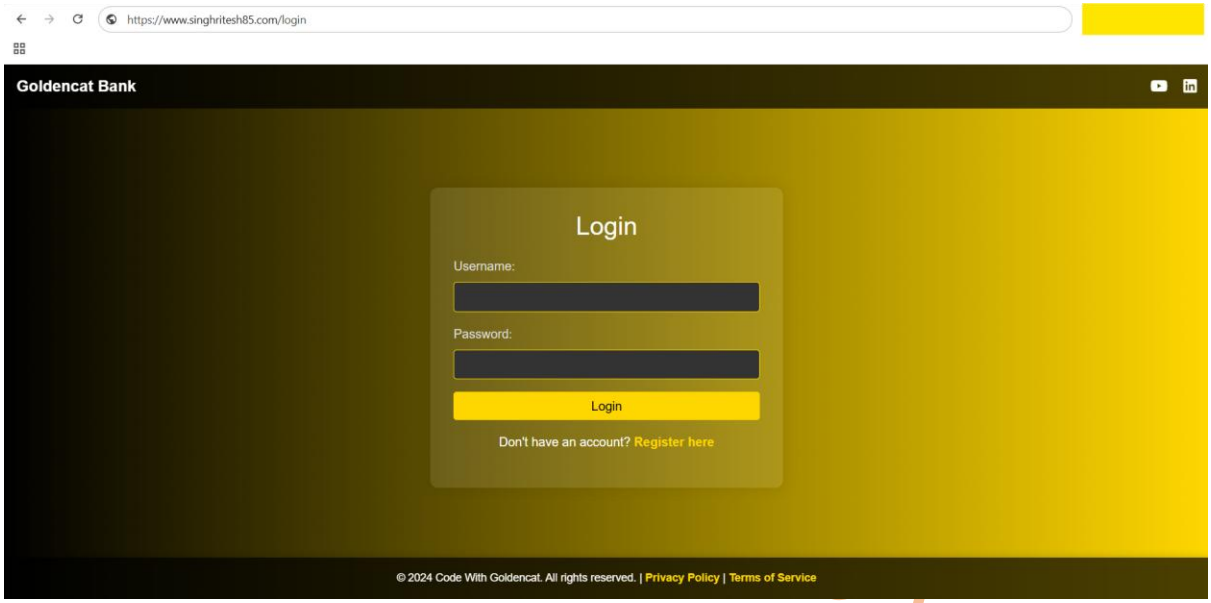


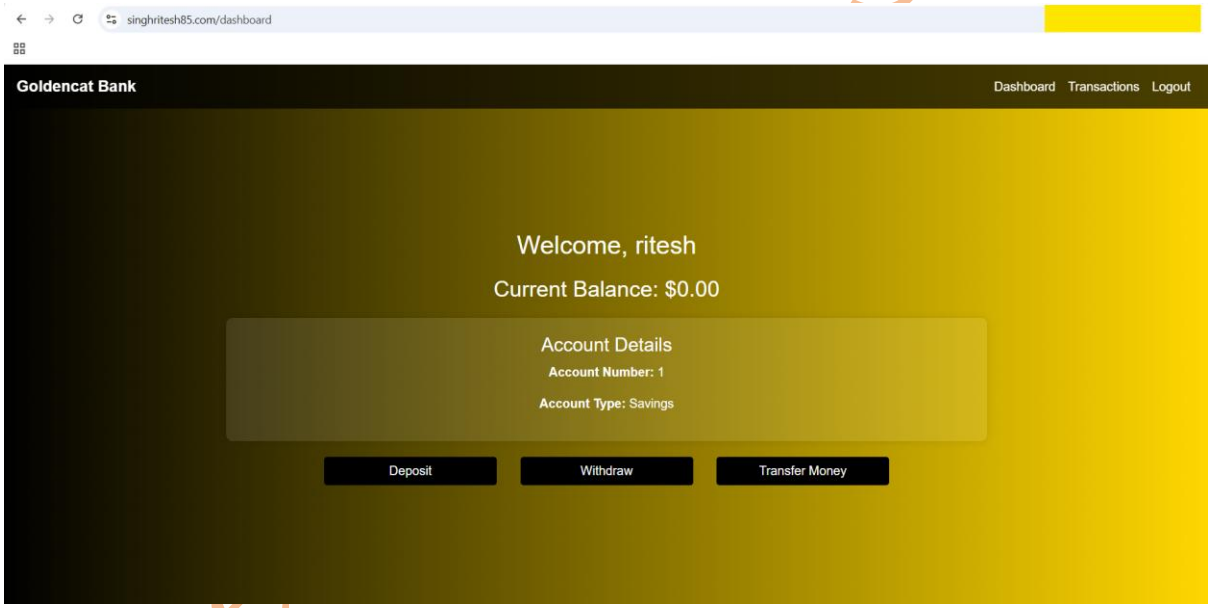

I did the entry for Public IP Address for GCP LoadBalancer in the GCP DNS Zone to create the Record Set as shown in the screenshot attached below.



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



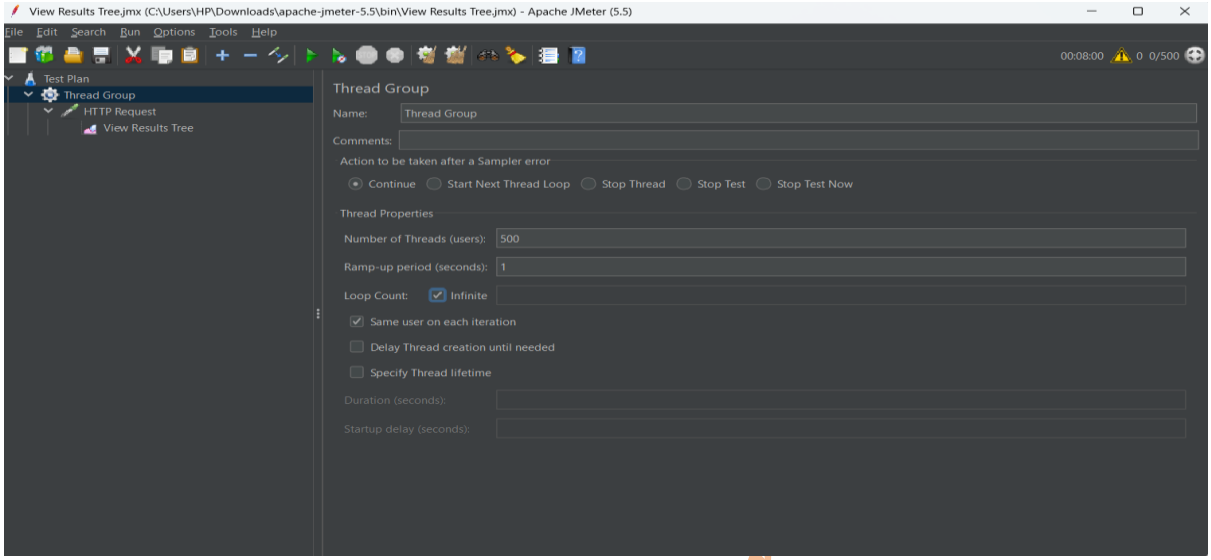



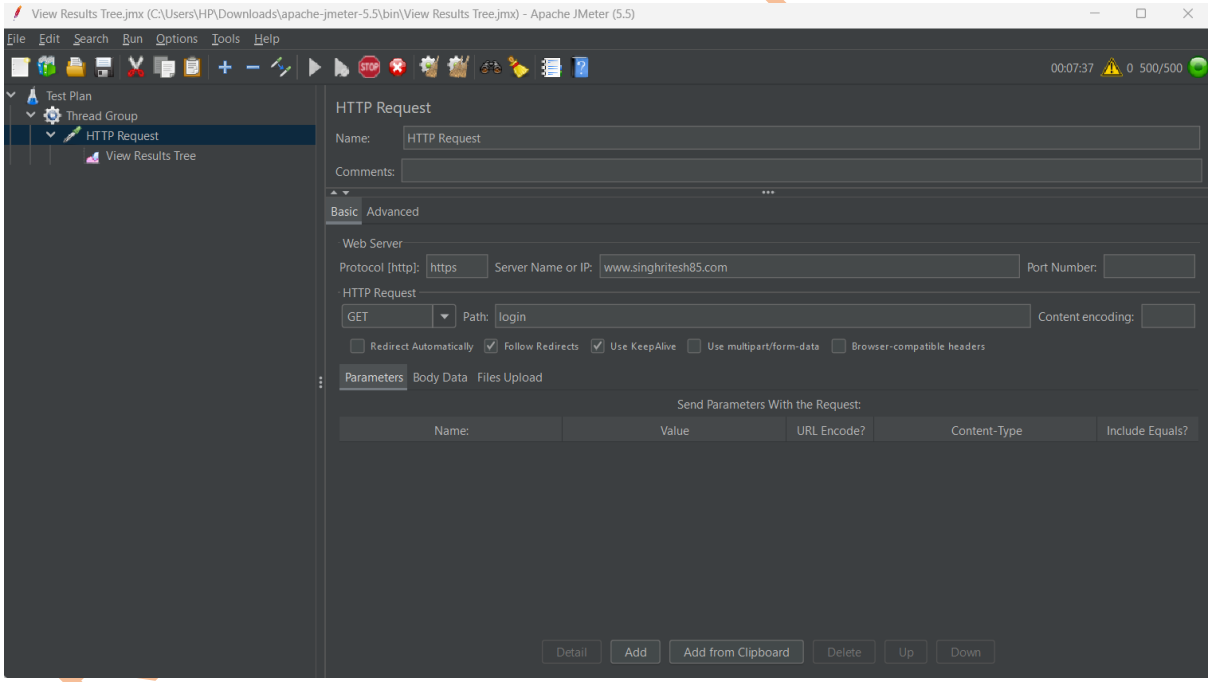

Initially one pod will be created for BankApp but if HTTP request on the cloud run instance will be more than 80 than maximum container count will be increased up to 3 (using autoscaling) as shown in the screenshot attached below.

```
template {
  health_check_disabled = false
  max_instance_request_concurrency = 80
  timeout = "300s"
```

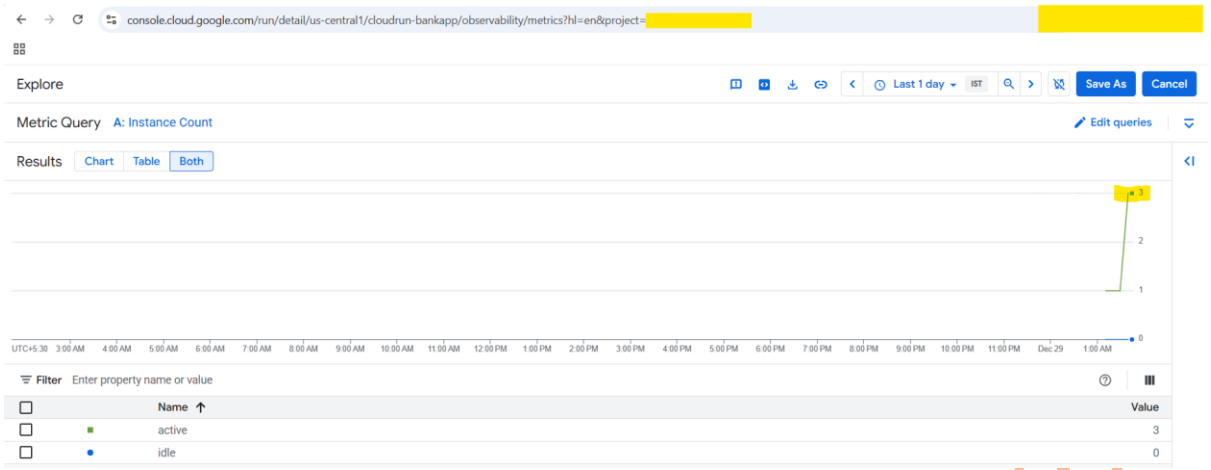
```
scaling {
  min_instance_count = 1
  max_instance_count = 3
}
```

I created the concurrent http request using JMeter and checked how auto scaling behaves for BankApp if it reaches and crosses the threshold 80 and recorded the response as shown in the screenshot attached below. I did the testing using JMeter in infinite loop with number of concurrent request equal to 500.




I found that number of containers had been increased to 3 (due to autoscaling) as shown in the screenshot attached below. To check the number of containers, go to **Cloud Run > Observability > Metrics > Instance Count**.




For this project to refrain myself from higher cloud bills I had not taken the Backup of Cloud SQL (MySQL). For your project I would highly recommend to take the Backup of the Cloud SQL (MySQL).

**Source Code**: - https://github.com/singhritesh85/Bank-App-GCP-Cloudrun.git

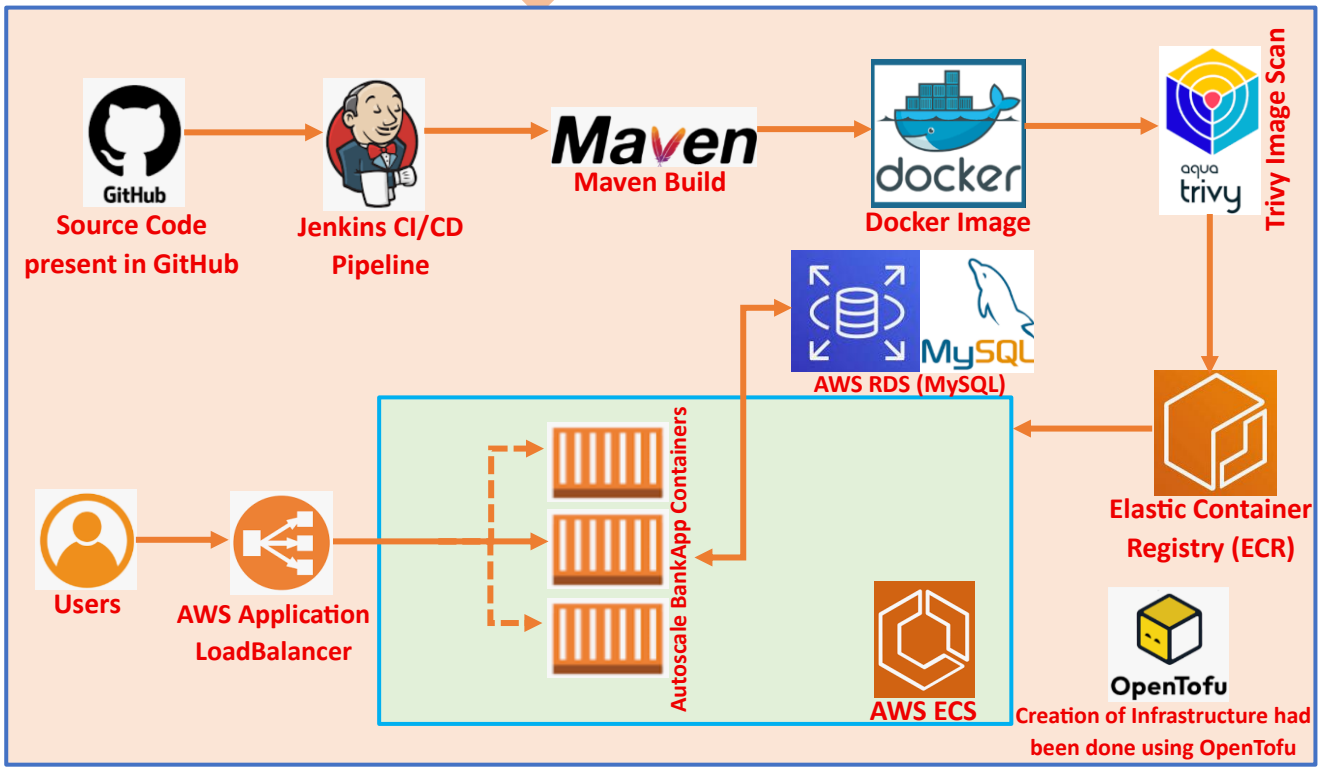
**Terraform Script**: - https://github.com/singhritesh85/terraform-docker-cloudrun-the-serverless-way.git

https://github.com/singhritesh85/terraform-gitlab-server-gcp.git

**GitHub Repo**: - https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git

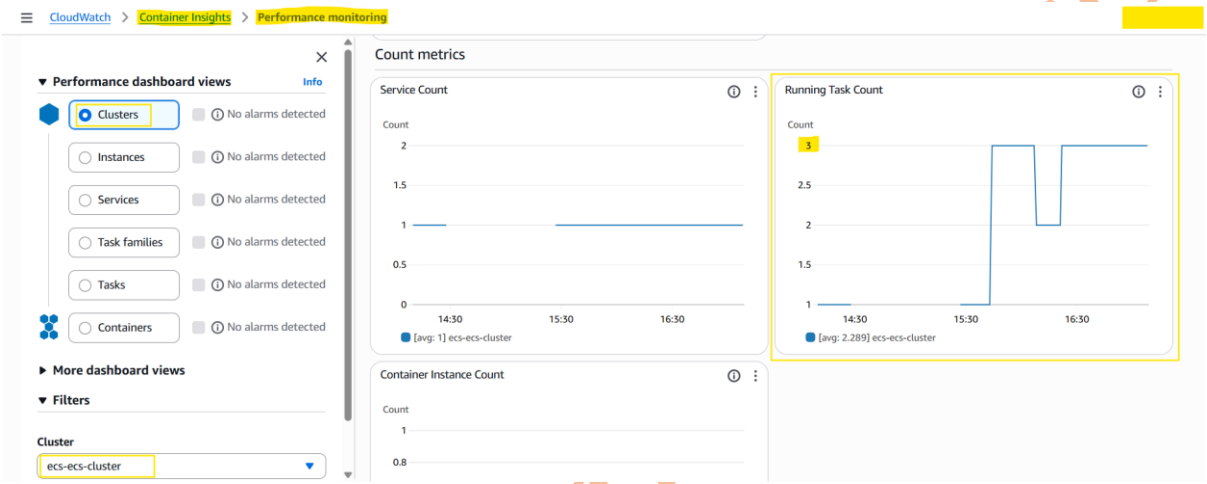
## Module-4

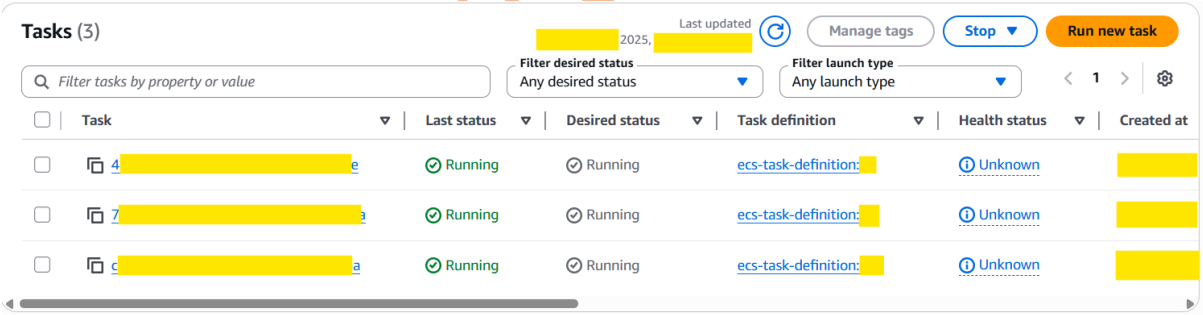
## Docker – The Serverless Way (AWS ECS)-In Production



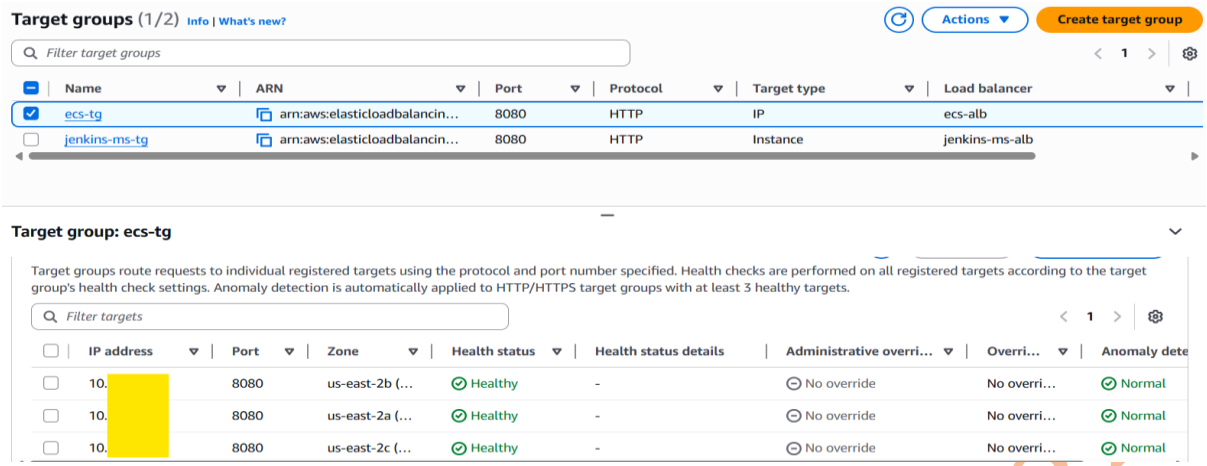
Like I explained in Module-1, here the BankApp is running in AWS ECS container and MySQL Database exist in AWS RDS. The AWS ECS Containers will be Auto Scaled based on three conditions 1. CPU Utilization greater than 65% 2. Memory Utilization greater than 75% and 3. HTTP Concurrent Request greater than 100. ECS Containers will be Auto Scaled between min_count=1 and max_count=3. You can set these values depending on your project requirement. This approach can be used in production environment. Application Logs are stored in AWS CloudWatch Log Group and Container Insight had been enabled. I used OpenTofu to create the infrastructure.

Using Apache JMeter I created concurrent request more than 100 then decreased, again increased, and found that containers count had been increased, decreased, and again increased accordingly as shown in the screenshot attached below.




Using Application LoadBalancer I was able to access the BankApp. You need to create the Record Set of A-Type with Alias in Route 53 Hosted Zone using the DNS Name of the Application LoadBalancer for ECS to access the BankApp using the URL. The Target Group attached with the LoadBalancer is as shown in the screenshot attached below. As you can see in the screenshot attached below there are three containers present in the Target Group after the replica count had been increased due to autoscaling.




For this project OpenTofu had been used to provision the infrastructure, first I had created Route53 Hosted Zone and AWS Certificate Manager based SSL Certificate. After running this OpenTofu script I had done the entry of Created Route53 Hosted Zone Nameserver in the Domain Name Provider's Nameserver. You need to use this created AWS SSL Certificate ARN in the OpenTofu Script for Jenkins Master Slave and ECR and OpenTofu Script for ECS with LoadBalancer. These OpenTofu scripts are present at the path **opentofu-route53-hosted-zone-with-acm-certificate**, **opentofu-jenkins-master-slave-ecr** and **opentofu-ecs-task-definition-service-rds-alb-autoscale** respectively in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git. I would like to mention here that ECS Service and RDS presented in the same VPC. If it is in different VPC then you need to establish VPC Peering between those two VPCs. If you want to know more about OpenTofu then you can refer the Project present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git. The Jenkinsfile to create the Jenkins CI/CD Pipeline is present at the root path with the file name **Jenkinsfile** in the GitHub Repo https://github.com/singhritesh85/Bank-App-ECS-Prod.git.

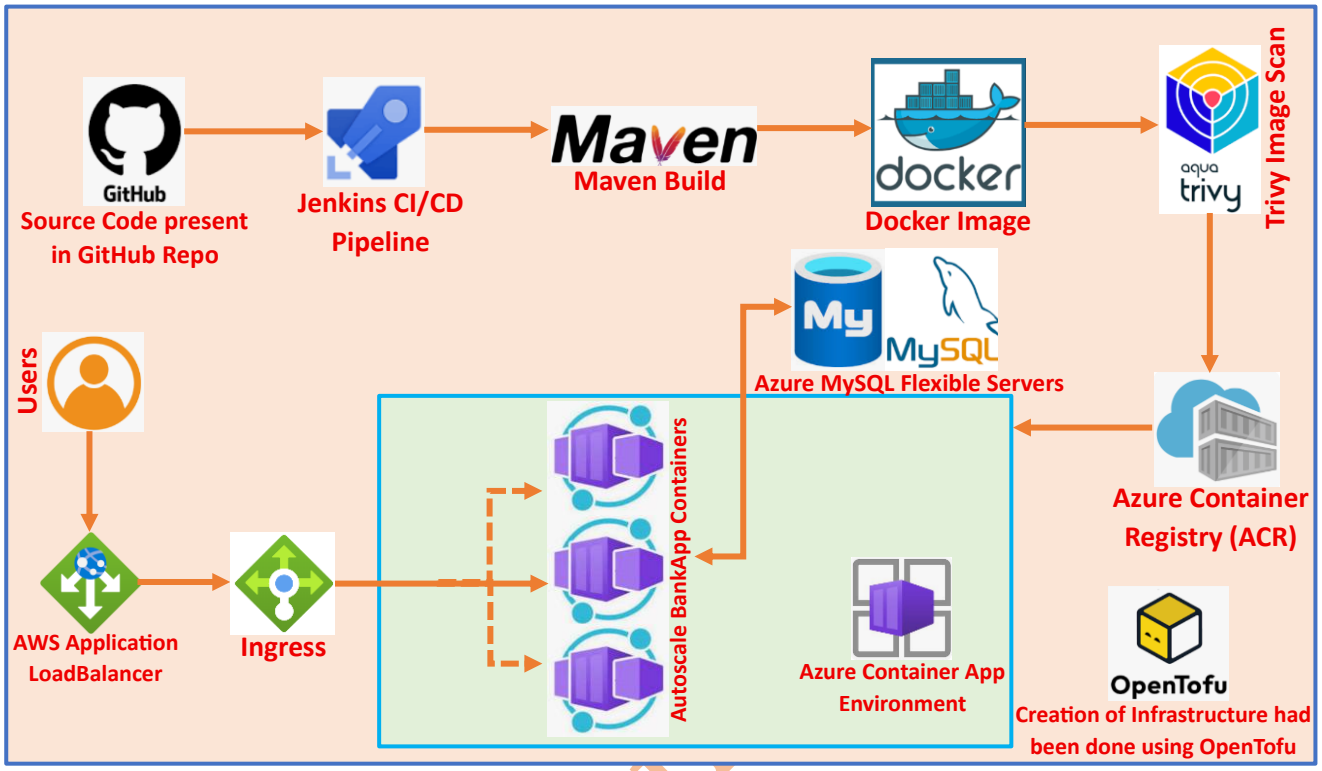
**Source Code: -** **https://github.com/singhritesh85/Bank-App-ECS-Prod.git**

**OpenTofu Script: -** **https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git**

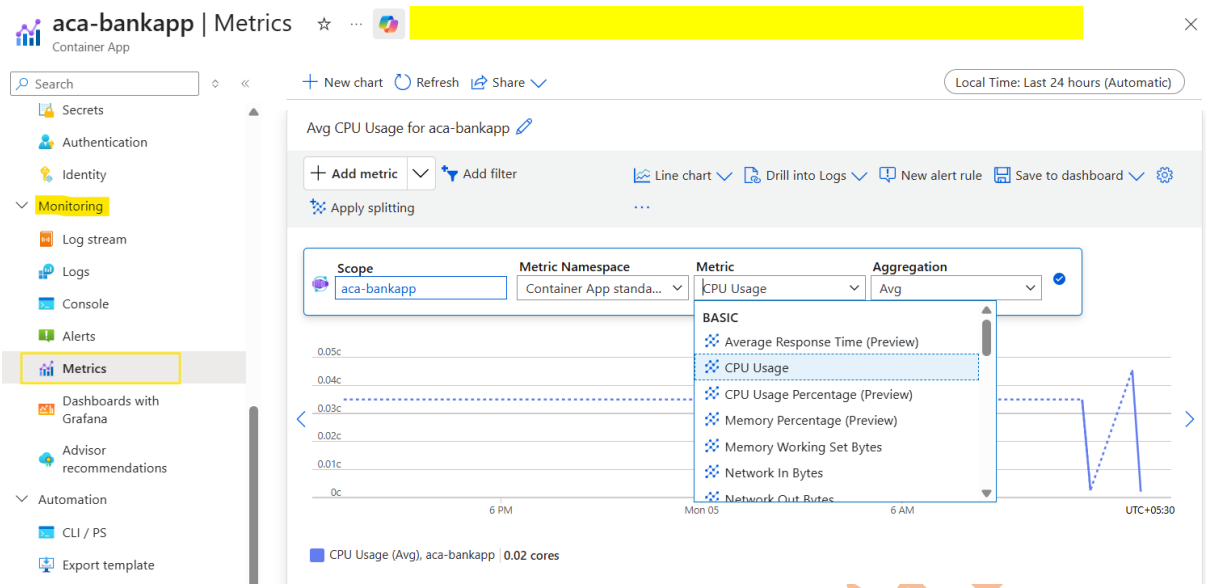
**GitHub Repo: -** **https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git**

## Module-5

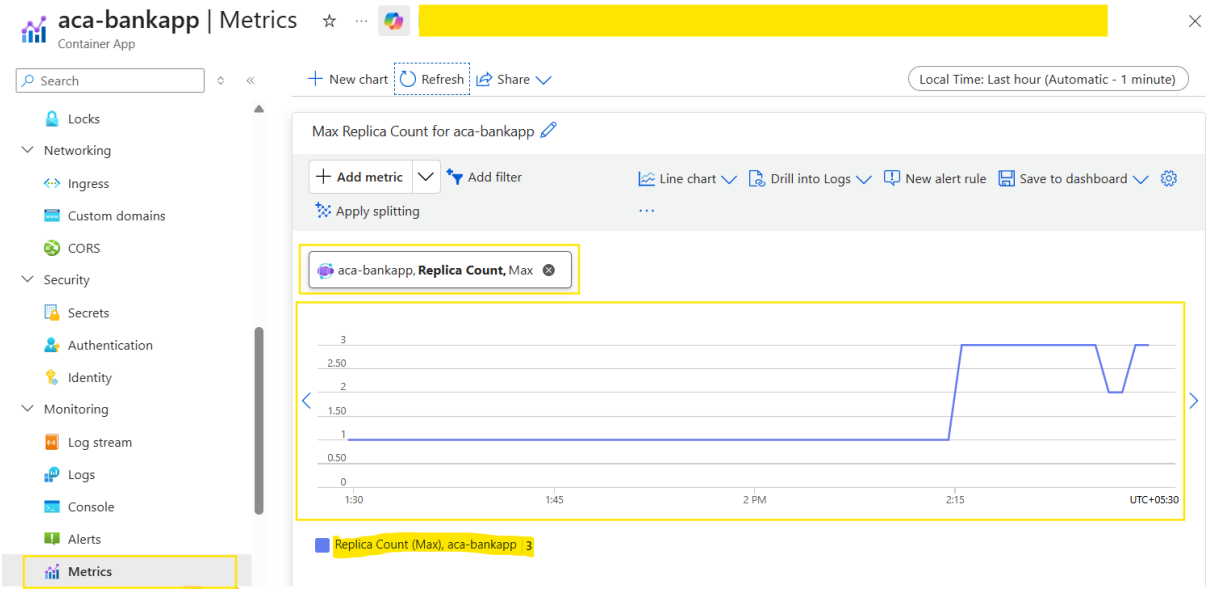
# Docker – The Serverless Way (Azure ACA)-In Production

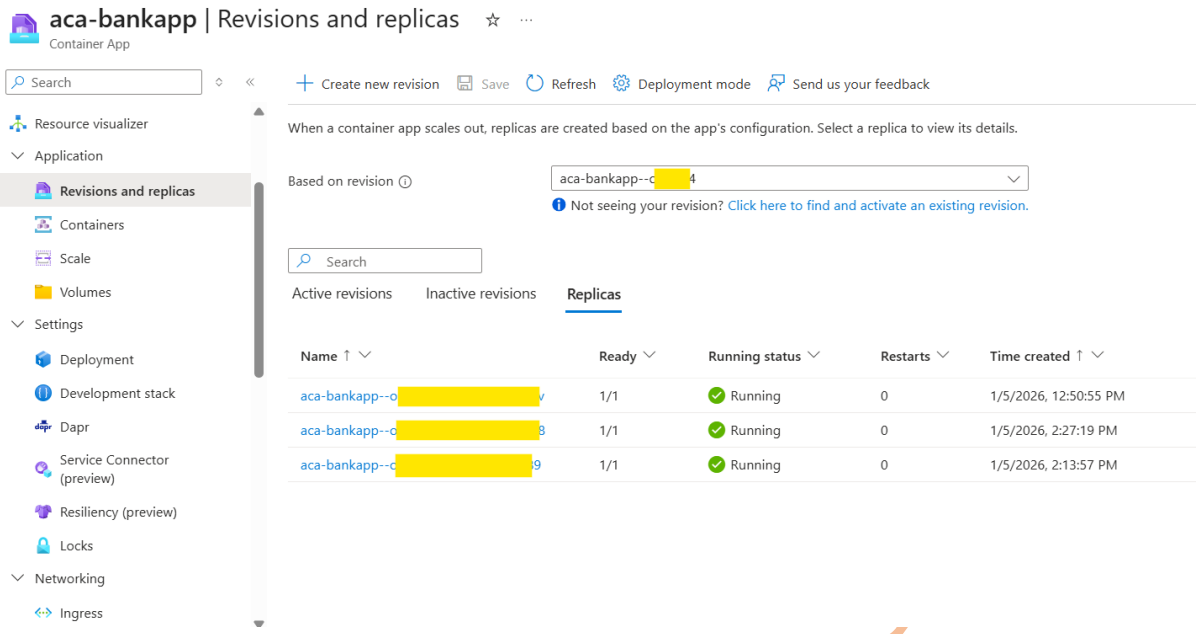



Like I explained in Module-2, here the BankApp is running in Azure Container App Container and MySQL Database exist in Azure Database for MySQL Flexible Servers. The Azure Container App Containers will be Auto Scaled based on three conditions 1. CPU Utilization greater than 70% 2. Memory Utilization greater than 80% and 3. HTTP Concurrent Request greater than 100. Azure Container App Containers will be Auto Scaled between min_count=1 and max_count=3. You can set these values depending on your project requirement. This approach can be used in production environment. Application Logs are stored in Azure Monitor Log Analytics workspace and can check the metrics from left side column **Metrics Option** under **Monitoring** as shown in the screenshot attached below. I used OpenTofu to create the infrastructure.

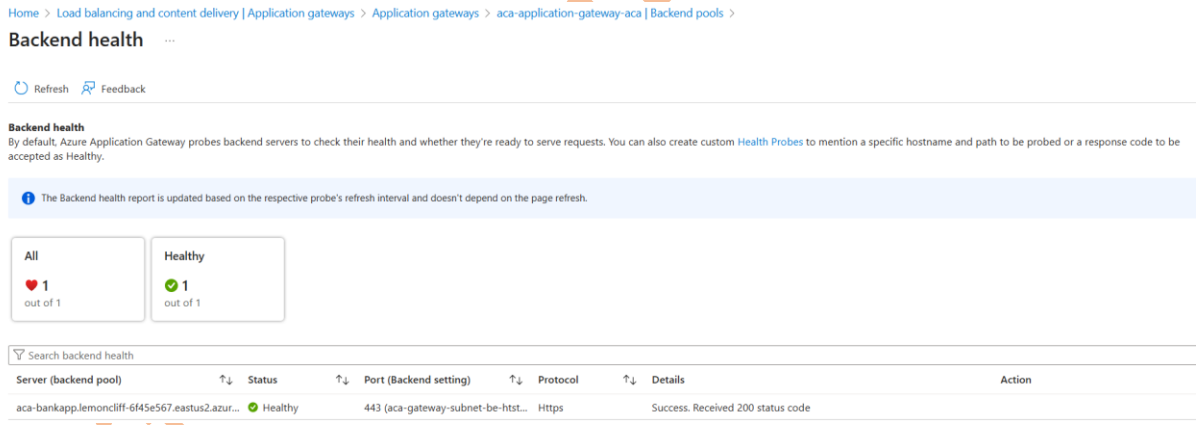



Using Apache JMeter I created concurrent request more than 100 then decreased, again increased, and found that containers count had been increased, decreased, and again increased accordingly as shown in the screenshot attached below.





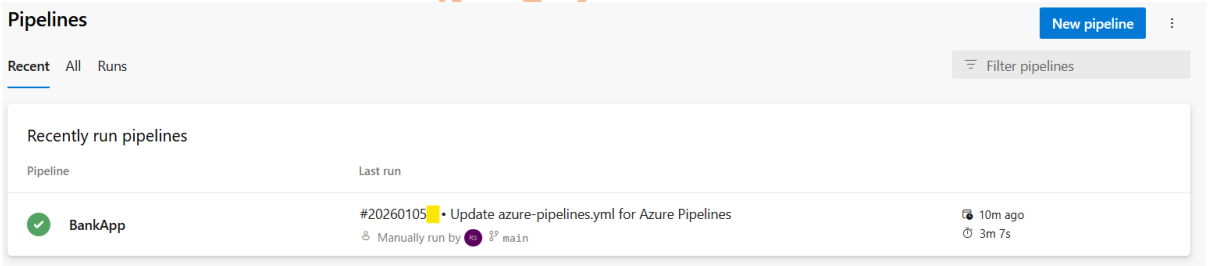

Using Application LoadBalancer I was able to access the BankApp. You need to create the Record Set of A-Type in Azure DNS Hosted Zone using the Public IP Address of the Azure Application Gateway for Azure Container App to access the BankApp using the URL. The Health Check for Azure Application Gateway is as shown in the screenshot attached below, which was healthy as shown in the screenshot attached below.



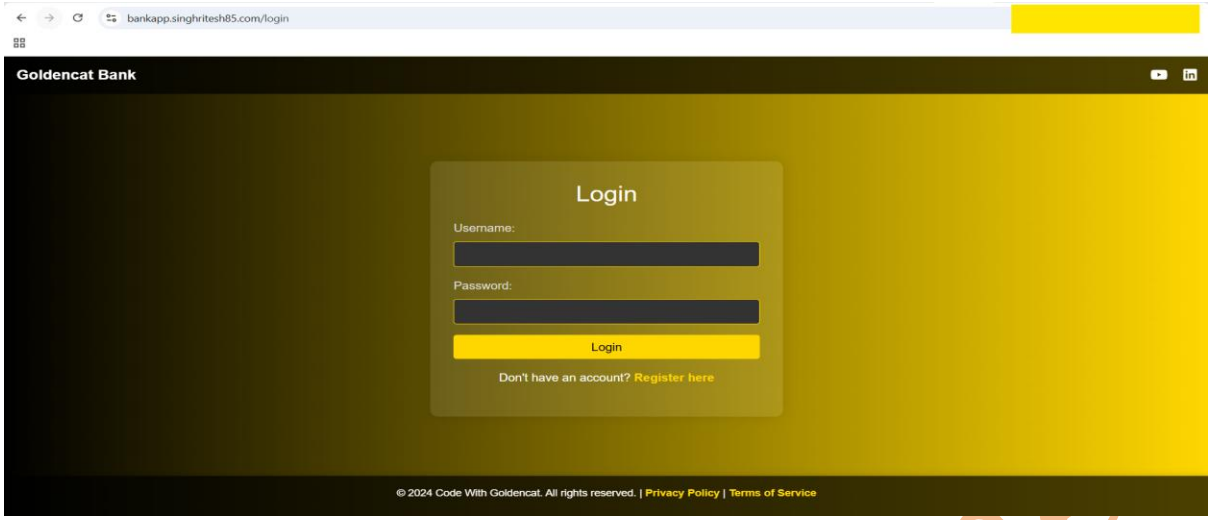

For this project OpenTofu had been used to provision the infrastructure, first I had created Azure DNS Hosted Zone. After running this OpenTofu script I had done the entry of Created Azure DNS Hosted Zone Nameserver in the Domain Name Provider's Nameserver. For Azure Artifact Registry and Azure DevOps Agent Server then Azure Application Gateway and Azure Container App Environment Certificate Wild Card .pfx suffix SSL Certificate (also known as PKCS#12 format) had been used. These OpenTofu scripts are present at the path **opentofu-azure-dns-zone**, **opentofu-azure-vm-devops-agent-acr** and **opentofu-azure-container-app-appgtw-bankapp-mysql-flexible-servers** respectively in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git. I would like to mention here that Private Azure Container App Environment and Azure Database for MySQL Flexible Servers presented in the same VNet. If it is in different VNet then you need to establish VNet Peering between those two VNet. The Azure Application Gateway routes the traffic to the Azure Container App through the Ingress. As the Azure Container App Environment is Private and hence you can access the Azure Container App using the Ingress URL directly with the help of

browser. To achieve this Application Gateway Private Link Configuration and Private DNS Zone had been used (for default domain of Azure Container App Environment). If you want to know more about OpenTofu then you can refer the Project present in the GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-terraform-upgrade-to-opentofu-secure-kubernetes-clusters.git. The **azure-pipelines.yml** to create the Azure DevOps CI/CD Pipeline is present at the root path with the file name **azure-pipelines.yml** in the GitHub Repo https://github.com/singhritesh85/BankApp-Docker-ACA-Prod.git.

The Azure DevOps Pipeline after its successful execution is as shown in the screenshot attached below.


Finally I was able to access the BankApp with the help of URL as shown in the screenshot attcahed below.




**Source Code**: - https://github.com/singhritesh85/BankApp-Docker-ACA-Prod.git

**OpenTofu Script**: - https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git

**GitHub Repo**: - https://github.com/singhritesh85/DevOps-Project-Docker-the-Serverless-way.git