

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>

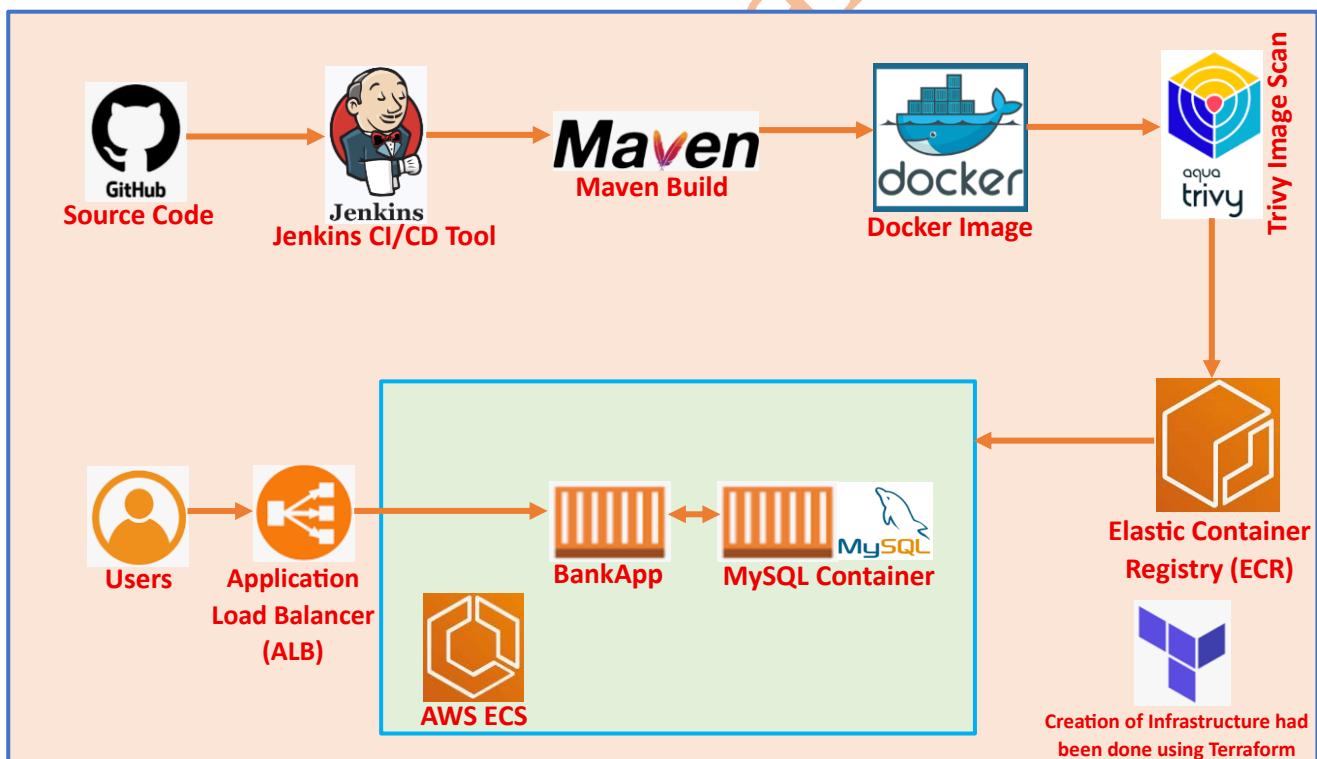


या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाङ्ग्यापहा ॥

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.

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=...
inghighsites85.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.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████████:loadbalancer/app/jenkins-ms-alb/daefc4af0115da6]
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████████:listener/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████████:listener/app/jenkins-ms-alb/████████]

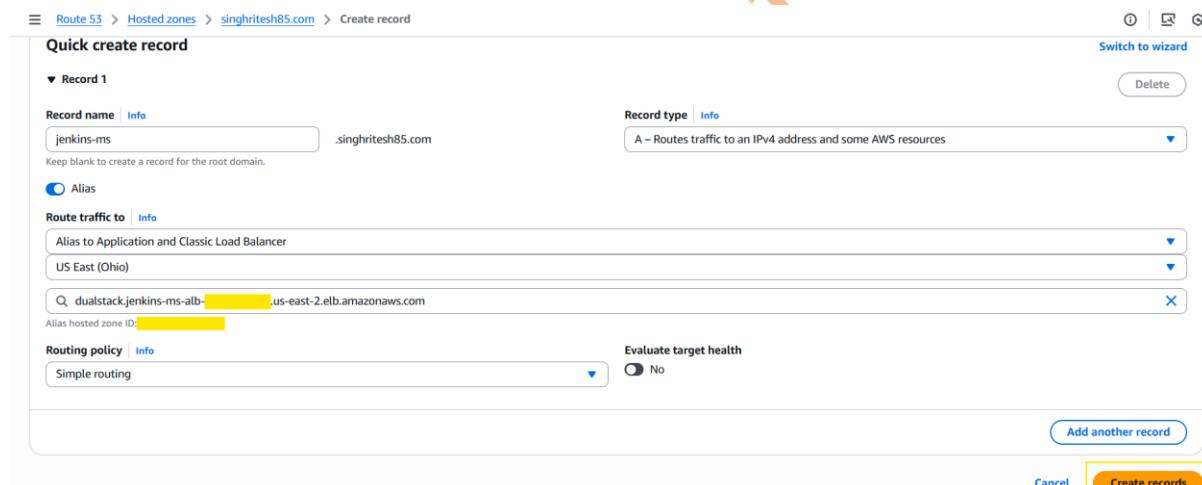
Apply complete! Resources: 40 added, 0 changed, 0 destroyed.

Outputs:

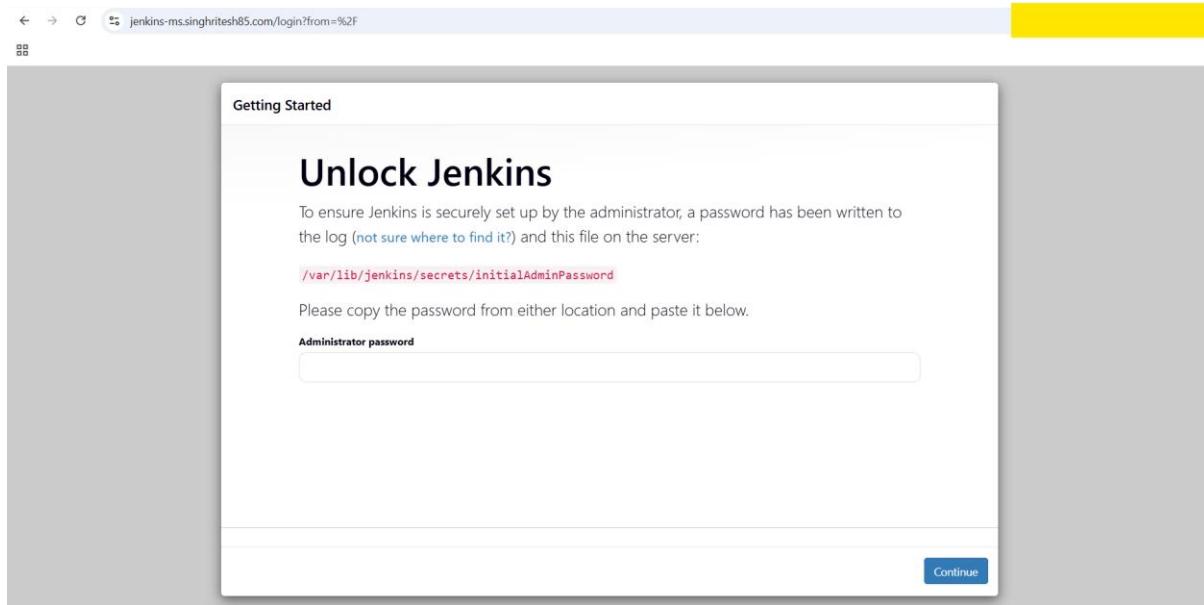
ECR_URI_jenkins_master_and_slave_ec2_private_ip_and_alb_dns_name = {
  "ecr_repository_uri" = "02████████.dkr.ecr.us-east-2.amazonaws.com/bankapp"
  "jenkins_alb_dns_name" = "jenkins-ms-alb-████████.us-east-2.elb.amazonaws.com"
  "jenkins_master_and_slave_private_ip" = [
    "172.16.4.████████",
   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

```

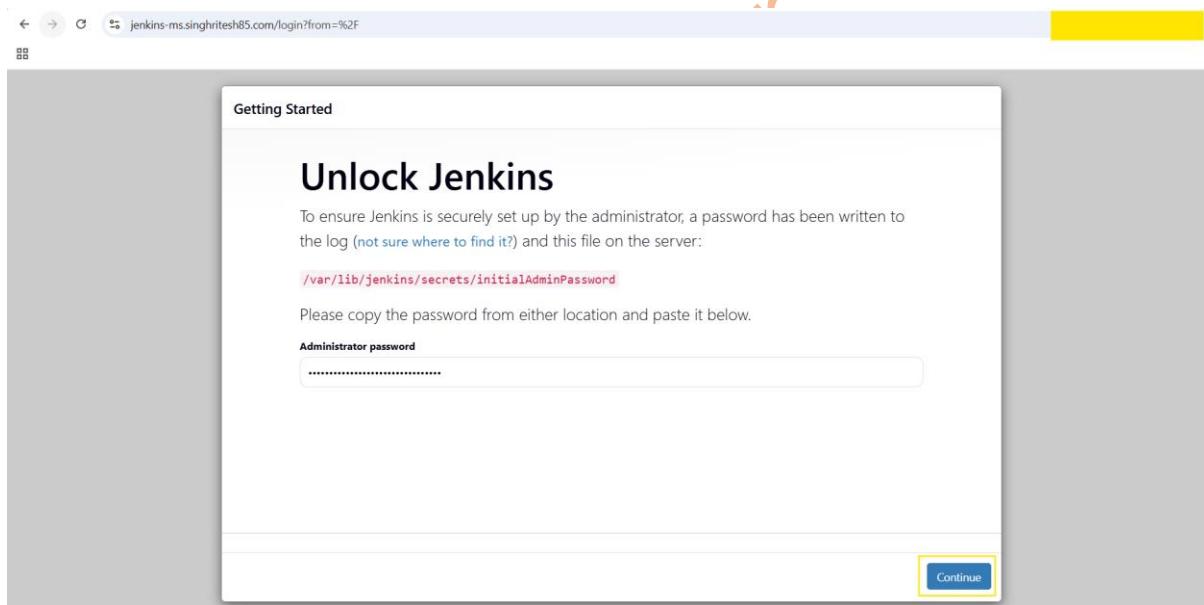
Do the entry for the generated ALB for Jenkins in Route 53 to create the Record Set of A-Type as shown in the screenshot attached below.



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



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



The image contains two screenshots of the Jenkins 'Getting Started' page. The top screenshot shows the 'Customize Jenkins' section with two options: 'Install suggested plugins' (selected) and 'Select plugins to install'. The bottom screenshot shows the 'Create First Admin User' form with fields for Username, Password, Confirm password, Full name, and E-mail address. The 'Save and Continue' button is highlighted.

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

Welcome to Jenkins!

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

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds ?

REST API Jenkins 2.528.3

Manage Jenkins

System Configuration

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

Security

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

Nodes

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	Built-In Node	Linux (amd64)	In sync	17.00 GiB	0 B	1.87 GiB	0ms
	Data obtained	1 min 8 sec	1 min 8 sec	1 min 8 sec	1 min 8 sec	1 min 8 sec	1 min 8 sec

Icon: S M L

Legend

REST API Jenkins 2.528.3

jenkins-ms.singhritesh85.com/computer/new

New node

Node name

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/createtitem

New node

Name

Description
Plain text Preview

Number of executors

Remote root directory

Labels

Save

jenkins-ms.singhritesh85.com/computer/createtitem

New node

Usage

Launch method

Host
Credentials + Add

Host Key Verification Strategy

Availability

Save

jenkins-ms.singhritesh85.com/computer/creatitem

Jenkins / Nodes

Non verifying Verification Strategy

Advanced

Availability ?

Keep this agent online as much as possible

Node Properties

- Disable deferred wipeout on this node ?
- Disk Space Monitoring Thresholds
- Environment variables
- Tool Locations

Save

REST API Jenkins 2.528.3

```
[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
```

jenkins-ms.singhritesh85.com/computer/

Jenkins / Nodes

Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
█	Built-In Node	Linux (amd64)	In sync	16.50 GiB	512.00 MiB	▲ 1.87 GiB	0ms
█	Slave-1	Linux (amd64)	In sync	16.06 GiB	512.00 MiB	▲ 1.87 GiB	36ms
	Data obtained	2.9 sec	2.9 sec	2.9 sec	2.9 sec	2.9 sec	2.8 sec

Icon: S M L

+ New Node Configure Monitors

Legend

REST API Jenkins 2.528.3

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.

The image contains two screenshots of the Jenkins web interface. The top screenshot shows the main Jenkins dashboard with a 'Create a job' button highlighted. The bottom screenshot shows the 'New Item' creation dialog with the 'Pipeline' option selected and its description visible.

Main Jenkins Dashboard Screenshot:

- URL: jenkins-ms.singhritesh85.com
- Header: Jenkins
- Left sidebar:
 - + New Item
 - Build History
 - Project Relationship
 - Check File Fingerprint
- Middle section:
 - Build Queue: No builds in the queue.
 - Build Executor Status:

Node	Status
Built-in Node	0/2
Slave-1	0/2
 - Welcome to Jenkins! (Section)
 - 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 (Section)
 - Create a job (button)
 - Add description (button)
- Bottom right: REST API, Jenkins 2.528.3

New Item Creation Dialog Screenshot:

- URL: jenkins-ms.singhritesh85.com/new/job
- Header: Jenkins / New Item
- Title: New Item
- Form fields:
 - Enter an item name: terraform-create-infrastructure
 - Select an item type:
 - Pipeline** (Selected): Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
 - Freestyle project: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
 - Multi-configuration project: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
 - Folder: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Buttons: OK (highlighted), Cancel

The screenshot shows the Jenkins Pipeline configuration page. On the left, there's a sidebar with 'Configure' sections: General, Triggers, Pipeline (selected), and Advanced. The main area is titled 'Definition' and shows a 'Pipeline script' section with a code editor containing Groovy code for a Jenkins pipeline. The code defines a pipeline with an agent node labeled 'Slave-1' and a custom workspace, an environment block setting JAVA_HOME and PATH, and parameters for REPO_NAME and TAG_NUMBER. A 'try sample Pipeline...' button is at the top right of the code editor. Below the code editor is a checkbox for 'Use Groovy Sandbox'. At the bottom are 'Save' and 'Apply' buttons.

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.

The screenshot shows the 'New credentials' screen under 'Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. It's a 'Username with password' type credential. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains a redacted value, and the 'Password' field also contains a redacted value. The 'ID' field is set to 'GitHub-PAT'. A 'Create' button is at the bottom.

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

The screenshot shows two screenshots of a Jenkins pipeline job. The top screenshot shows the Jenkins interface with the pipeline configuration. It includes fields for REPO_NAME (027330342406.dkr.ecr.us-east-2.amazonaws.com/bankapp) and TAG_NUMBER (1.01). A green 'Build' button is highlighted with a yellow border. The bottom screenshot shows the Jenkins console output for the pipeline run, displaying the Terraform outputs and a 'Finished: SUCCESS' message.

```

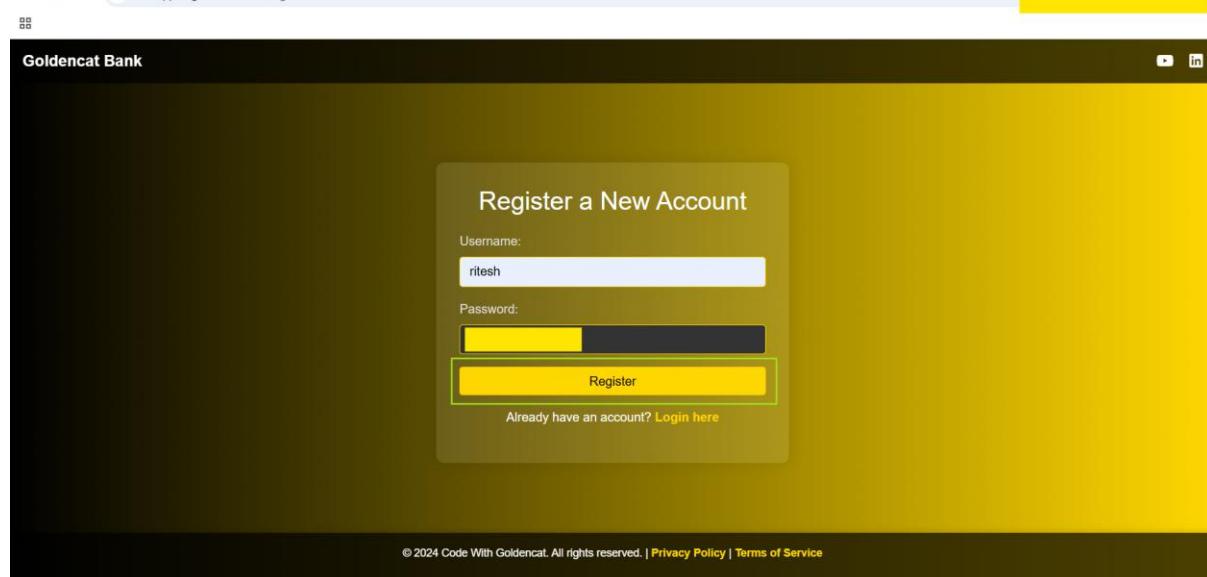
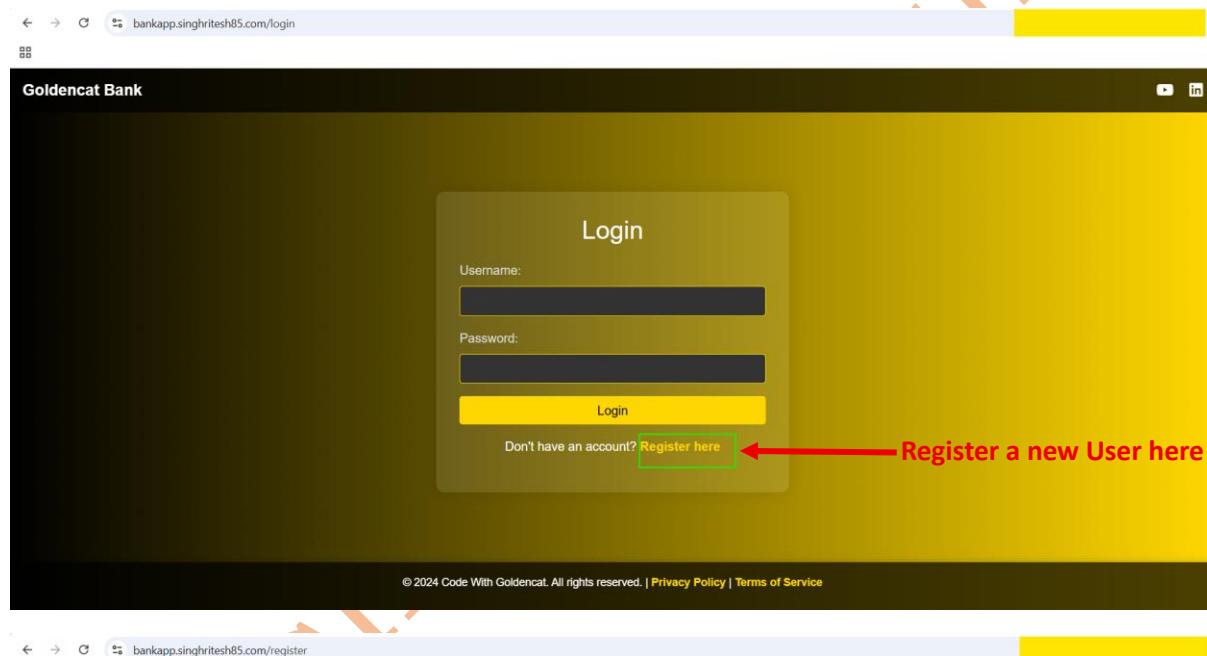
Outputs:
[0]ecs_details = {
  "ecs_cluster_name" = "ecs-ecs-cluster"
  "ecs_service_arn_bankapp" = "arn:aws:ecs:us-east-2:02[REDACTED]:service/ecs-ecs-cluster/ecs-service"
  "ecs_service_arn_mysql" = "arn:aws:ecs:us-east-2:02[REDACTED]:service/ecs-ecs-cluster/ecs-service-mysql2"
  "task_definition_arn_bankapp" = "arn:aws:ecs:us-east-2:02[REDACTED]:task-definition/ecs-task-definition:29"
  "task_definition_arn_mysql" = "arn:aws:ecs:us-east-2:02[REDACTED]:task-definition/ecs-task-definition-mysql:13"
  "task_definition_family_bankapp" = "ecs-task-definition"
  "task_definition_family_mysql" = "ecs-task-definition-mysql"
  "task_definition_revision_bankapp" = 29
  "task_definition_revision_mysql" = 13
}
[Pipeline]
[Pipeline] // dir
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // ws
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

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.

The screenshot shows the 'Quick create record' interface in AWS Route 53. The 'Record name' is set to 'bankapp' and the 'Record type' is 'A'. The 'Route traffic to' section is configured with an alias to an Application Load Balancer in the 'US East (Ohio)' region, specifically targeting the dualstack.ecs-alb-...us-east-2.elb.amazonaws.com endpoint. The 'Evaluate target health' option is turned off ('No'). There is also a 'Simple routing' option available. At the bottom right, there are 'Cancel' and 'Create records' buttons.

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



The screenshots demonstrate a banking application interface. The top image shows the login screen with a yellow header and a central modal for entering credentials. The bottom image shows the user's dashboard after logging in, featuring a dark theme with a yellow header bar. The dashboard displays the user's name, current balance, account details, and various transaction options.

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[REDACTED]9 --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-[REDACTED]
# 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)
```

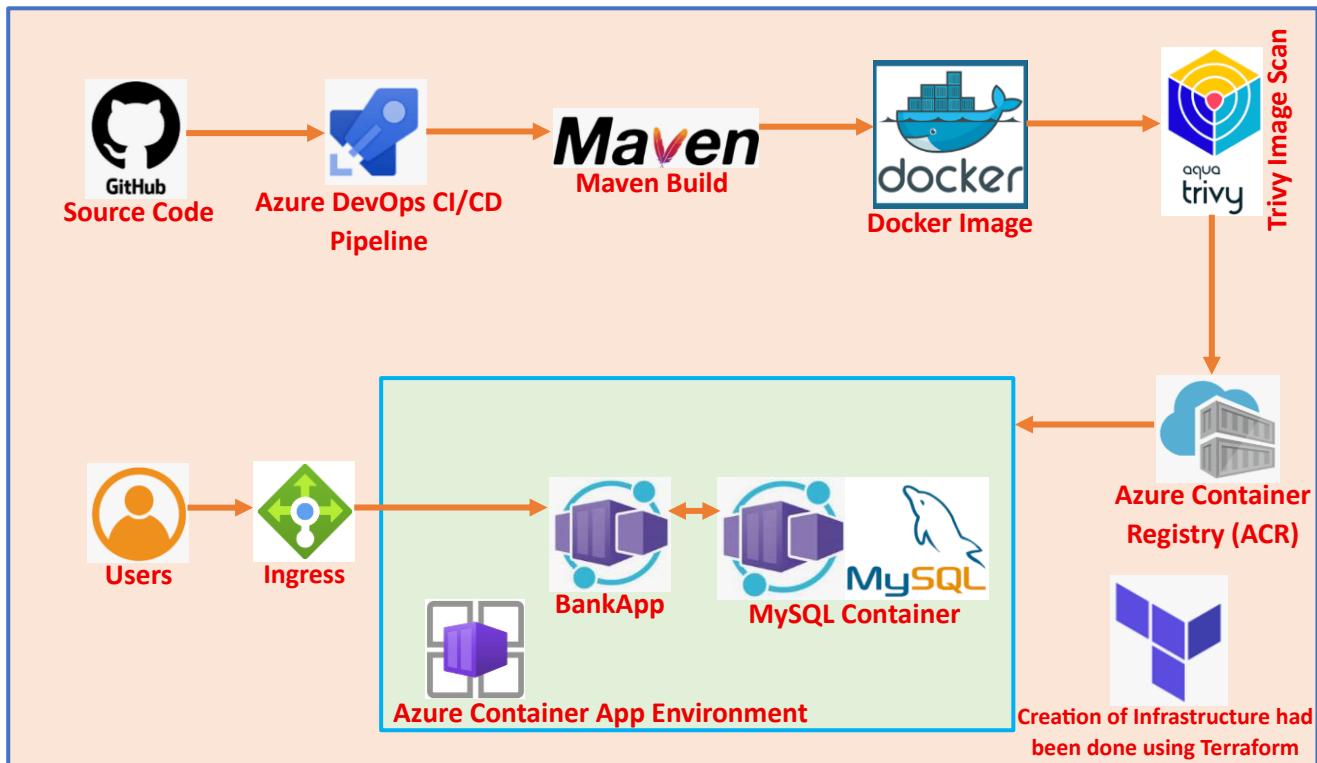
```
mysql> select * from account;
+----+----+----+----+----+----+
| id | balance | password | username |
+----+----+----+----+----+----+
| 1 | 0.00 | $2[REDACTED]2 | ritesh |
+----+----+----+----+----+
1 row in set (0.00 sec)
```

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>

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"
        }
      }
```

View 'aca-http-scale-rule'

Scale rule details

Rule name	aca-http-scale-rule
Type	HTTP scaling
Concurrent requests	100

One or more containers, along with settings such as scale rule

Scale

Scale rule settings

Min / max replicas	1 - 3
--------------------	-------

Scale rules

Name ↑	Type ↑
aca-http-scale-rule	HTTP sc

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/singhrithesh85/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.

Add agent pool

Agent pools are shared across an organization.

Managed DevOps Pool
Reduce the effort spent in maintaining custom agents by creating a Microsoft managed pool of scalable agents. [Learn more](#).

Self-hosted
Create a pool of custom agents hosted on your own infrastructure for maximum control and flexibility. [Learn more](#).

Azure virtual machine scale set
Create a pool of custom agents based on an Azure Virtual machine scale set hosted in your own Azure subscription. [View configuration instructions](#).

Name:

Description (optional):

① Markdown supported.

Pipeline permissions:

Auto-provision this agent pool in all projects

Create

```
[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
```

Azure Pipelines
agent v4.266.2 (commit 8...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 sources from Git repositories.
```

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

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

```
>> Connect:
```

```
Enter server URL > https://dev.azure.com/ [REDACTED]
Enter authentication type (press enter for PAT) >
Enter personal access token > [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
        CPU: 0.000 CPU(s)
       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.

The screenshot shows the 'Agents' tab in the Azure DevOps interface. It lists one agent named 'demo'. The agent is marked as 'Online' with a green dot. The 'Enabled' switch is turned on. Other tabs like 'Jobs', 'Details', 'Security', 'Settings', 'Maintenance History', and 'Analytics' are also visible.

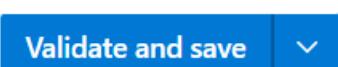
Now create the pipeline using the yaml file named as **azure-pipelines.yml** present in the GitHub Repo <https://github.com/singhritesh85/BankApp-Docker-ACA-the-Serverless-way.git>. Then create the variables as need for this pipeline to execute for this project I created five variables as shown below.

The screenshot shows the 'Pipelines' section of the Azure DevOps interface. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The 'Pipelines' option is selected. In the main area, it shows a pipeline named 'BankApp' with a single stage named 'Build'. To the right, there's a 'Variables' editor window. It contains five variables: GIT_PASSWORD, GIT_USERNAME, PASSWORD_ACR, SERVER_ACR, and USERNAME_ACR. The SERVER_ACR variable is highlighted with a yellow box. Below the variables, there are 'Learn about variables', 'Close', and 'Save' buttons.

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.

The screenshot shows the 'Project Settings' section of the Azure DevOps interface. Under 'Service connections', it lists two entries: 'Docker-Registry' and 'my-demo-project'. The 'Docker-Registry' entry is highlighted with a yellow box. The sidebar on the left includes sections for General, Pipelines, and Agent pools.

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.

The screenshot shows the Azure DevOps Pipeline interface. At the top, there are three stages: Build, DockerImageBuild, and AzureContainerAppDeployment. Each stage has a green checkmark indicating completion. The Build stage took 28 seconds and completed 1 job. The DockerImageBuild stage took 46 seconds and completed 1 job. The AzureContainerAppDeployment stage took 12m 17s and completed 1 job. Below the stages, there is a 'CmdLine' tab showing the logs for the AzureContainerAppDeployment stage. The logs include Terraform output for creating resources like MySQL and Container Apps.

```

1034 module.aca.azurerm_container_app.aca_mysql_app: Still creating... [00m08s elapsed]
1035 module.aca.azurerm_container_app.aca_mysql_app: Creation complete after 18s [id=/subscriptions//resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp]
1036 module.aca.azurerm_container_app.aca_bankapp: Creating...
1037 module.aca.azurerm_container_app.aca_bankapp: Still creating... [00m10s elapsed]
1038 module.aca.azurerm_container_app.aca_bankapp: Creation complete after 18s [id=/subscriptions//resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp]
1039 module.aca.azurerm_dns_txt_record.dns_txt_record: Creating...
1040 module.aca.azurerm_dns_cname_record.example: Creating...
1041 module.aca.azurerm_dns_cname_record.example: Creation complete after 1s [id=/subscriptions//resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp]
1042 module.aca.azurerm_dns_txt_record.dns_txt_record: Creation complete after 1s [id=/subscriptions//resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp]
1043 module.aca.azurerm_container_app_custom_domain.custom_domain: Creating...
1044 module.aca.azurerm_container_app_custom_domain.custom_domain: Still creating... [00m10s elapsed]
1045 module.aca.azurerm_container_app_custom_domain.custom_domain: Creation complete after 18s [id=/subscriptions//resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp]
1046
1047 Apply complete! Resources: 14 added, 0 changed, 0 destroyed.
1048
1049 Outputs:
1050
1051 azure_container_apps_details = {
1052   "azurerm_container_app_bankapp_url" = "aca-bankapp--.graydune--.eastus.azurecontainerapps.io"
1053   "azurerm_container_app_mysql_url" = "mysql-service--.internal.graydune--.eastus.azurecontainerapps.io"
1054   "container_app_bankapp_id" = "/subscriptions//resourceGroups/aca-rg/providers/Microsoft.App/containerApps/aca-bankapp"
1055   "container_app_mysql_id" = "/subscriptions//resourceGroups/aca-rg/providers/Microsoft.App/containerApps/mysql-service"
1056   "container_apps_environment_static_ip" = "20..161"
1057 }
1058
1059 Finishing: Cmdline

```

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.

The screenshot shows the Azure Container Apps Environment Certificates page. On the left, there is a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Settings, Certificates, Networking, Volume mounts, Identity, and Planned Maintenance. The 'Certificates' option is selected. In the main area, there is a table with one item:

Health status	Friendly name	Type	Hostname	Expiration date	Thumbprint	Solution
Healthy	aca-certificate	Uploaded Certificate	*.singhritesh85.com			-

Below the table, there is a link to 'Bring your own certificates (.pfx)'. The URL for the screenshot is https://portal.azure.com/#@singhritesh85.com/resourceGroups/aca-rg/providers/Microsoft.ContainerApp/containerApps/aca-bankapp/certificates?WT.mc_id=Portal-Header.

The screenshot shows the Azure portal interface for a 'Container App' named 'aca-bankapp'. On the left, a navigation sidebar lists various management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Application (with Revisions and replicas, Containers, Scale, Volumes), Settings (Deployment, Development stack), and Help. The main content area is titled 'Custom domains' and displays a table with one item: 'www.singhritesh85.com' which is 'Secured' with 'SNI SSL' and 'aca-certificate'.

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

The first screenshot shows a web browser with the URL 'singhritesh85.com/login'. The page title is 'Goldencat Bank' and features a 'Login' form with fields for 'Username' and 'Password', and a 'Login' button. Below the form is a link 'Don't have an account? Register here'. At the bottom, there is a footer with the text '© 2024 Code With Goldencat. All rights reserved. | Privacy Policy | Terms of Service'.

The second screenshot shows a web browser with the URL 'singhritesh85.com/dashboard'. The page title is 'Goldencat Bank' and displays a welcome message 'Welcome, ritesh', current balance '\$0.00', and account details ('Account Number: 1', 'Account Type: Savings'). It includes three buttons: 'Deposit', 'Withdraw', and 'Transfer Money'. At the top right, there are links for 'Dashboard', 'Transactions', and 'Logout'.

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

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

aca-bankapp | Log stream

Networking: Ingress, Custom domains, CORS

Security: Secrets, Authentication, Identity

Monitoring: Log stream, Logs, Console, Alerts, Metrics, Dashboards with Grafana

Category: Application

Based on revision: acabankapp--724q5dk

Replica: acabankapp--724q5dk-6bb8bdb9cb-rc5q8

Container: acabankapp

```

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=?
2025-12-23T16:50:48.426481263Z WARN 1 --- [bankapp] [nio-8080-exec-2] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Failed to validate connection com.mysql.cj.jdbc.ConnectionImpl@5d496dd5-8sq7v! (No operations allowed after connection closed.). Possibly consider using a shorter maxLifetime value.
2025-12-23T16:50:48.426481263Z WARN 1 --- [bankapp] [nio-8080-exec-2] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Failed to validate connection com.mysql.cj.jdbc.ConnectionImpl@5d496dd5-8sq7v! (No operations allowed after connection closed.). Possibly consider using a shorter maxLifetime value.
2025-12-23T16:50:48.426481263Z WARN 1 --- [bankapp] [nio-8080-exec-2] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Failed to validate connection com.mysql.cj.jdbc.ConnectionImpl@5d496dd5-8sq7v! (No operations allowed after connection closed.). Possibly consider using a shorter maxLifetime value.
2025-12-23T16:50:48.426481263Z WARN 1 --- [bankapp] [nio-8080-exec-2] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Failed to validate connection com.mysql.cj.jdbc.ConnectionImpl@5d496dd5-8sq7v! (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=?

```

mysql-service | Console

Custom domains, CORS

Security: Secrets, Authentication, Identity

Monitoring: Log stream, Logs, Console, Alerts, Metrics

Console: App container

Based on revision: mysql-service--8uim3e1

Replica: mysql-service--8uim3e1-7b4d496dd5-8sq7v

Container: acamysql

```

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

Custom domains, CORS

Security: Secrets, Authentication, Identity

Monitoring: Log stream, Logs, Console, Metrics, Dashboards with Grafana, Advisor recommendations

Console: App container

Based on revision: mysql-service--8uim3e1

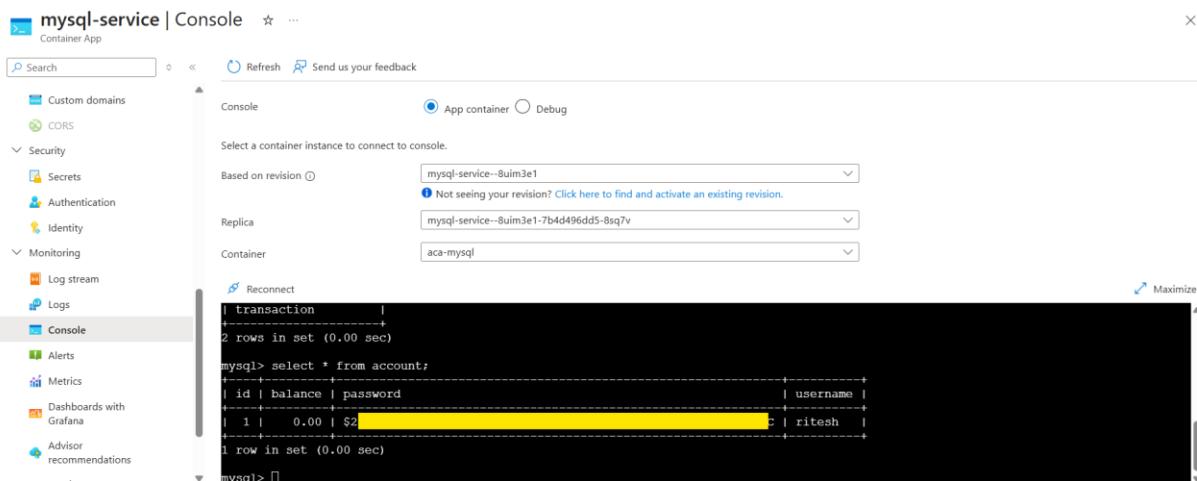
Replica: mysql-service--8uim3e1-7b4d496dd5-8sq7v

Container: acamysql

```

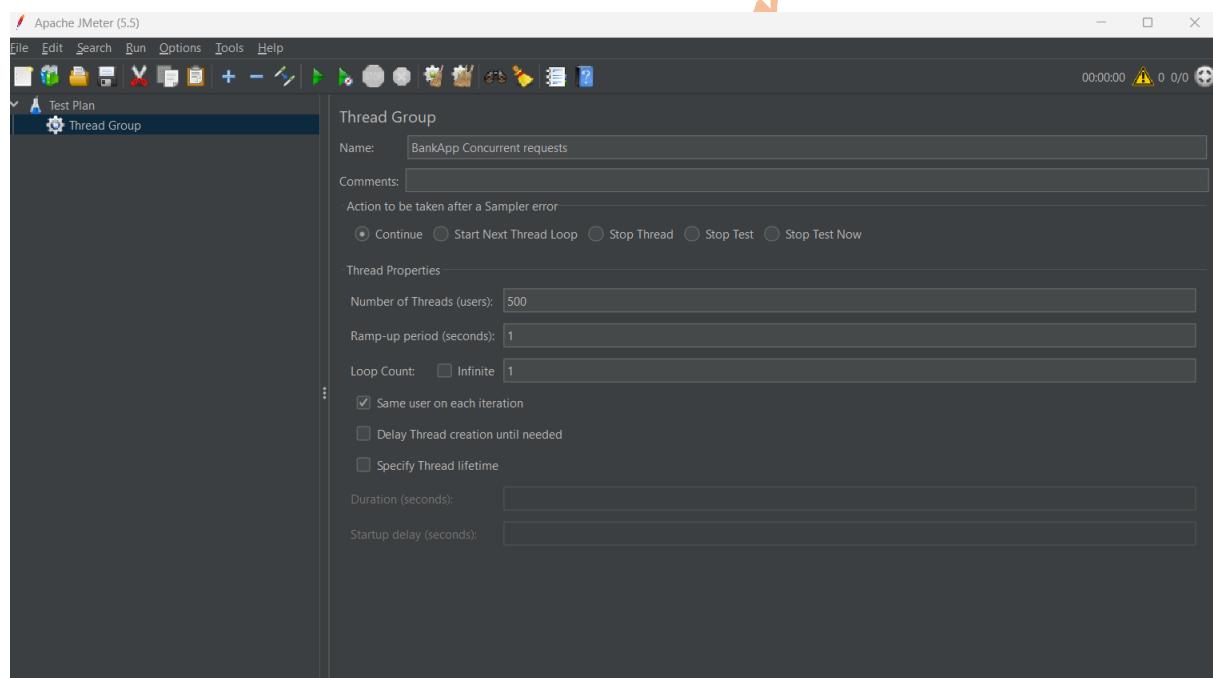
mysql> use bankappdb;
Reading table information for completion of table and column names
you can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+-----+
| Tables_in_bankappdb |
+-----+
| account             |
| transaction         |
+-----+
2 rows in set (0.00 sec)

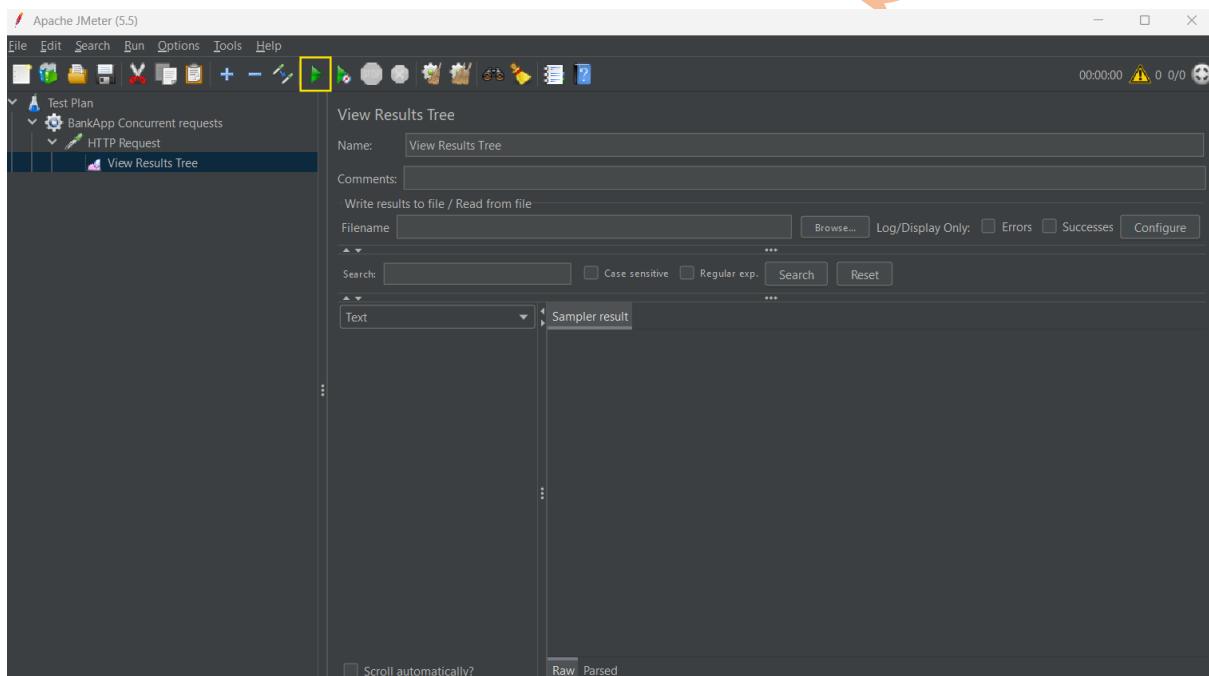
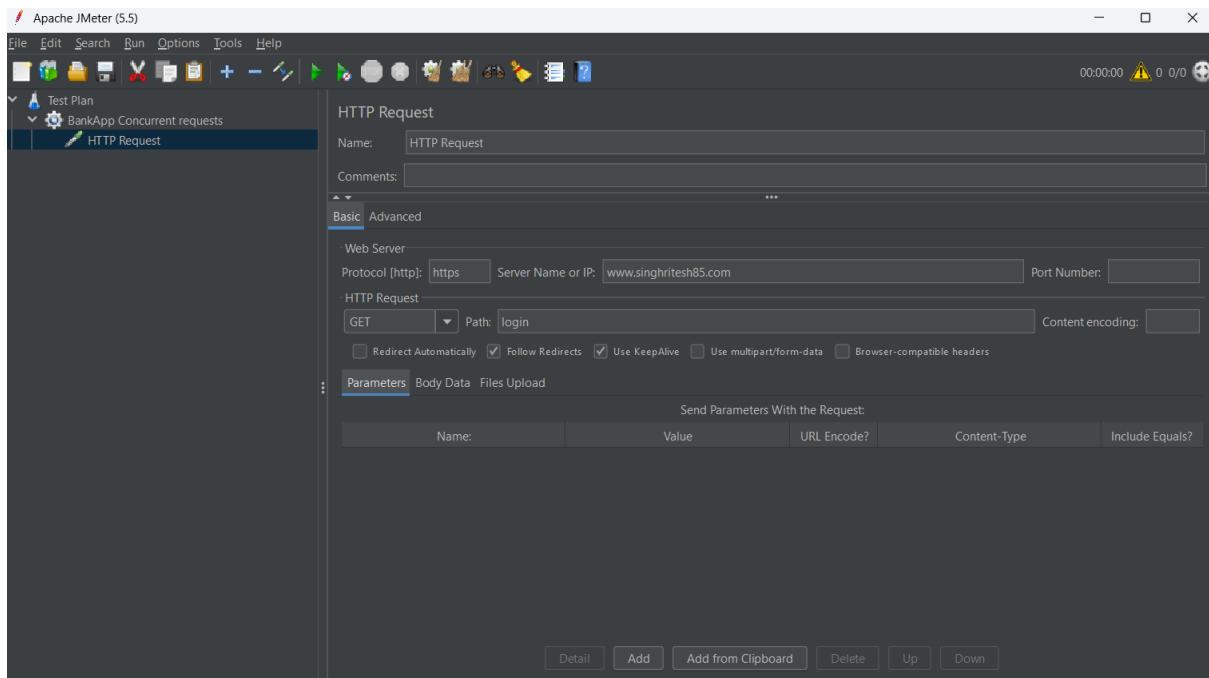
```

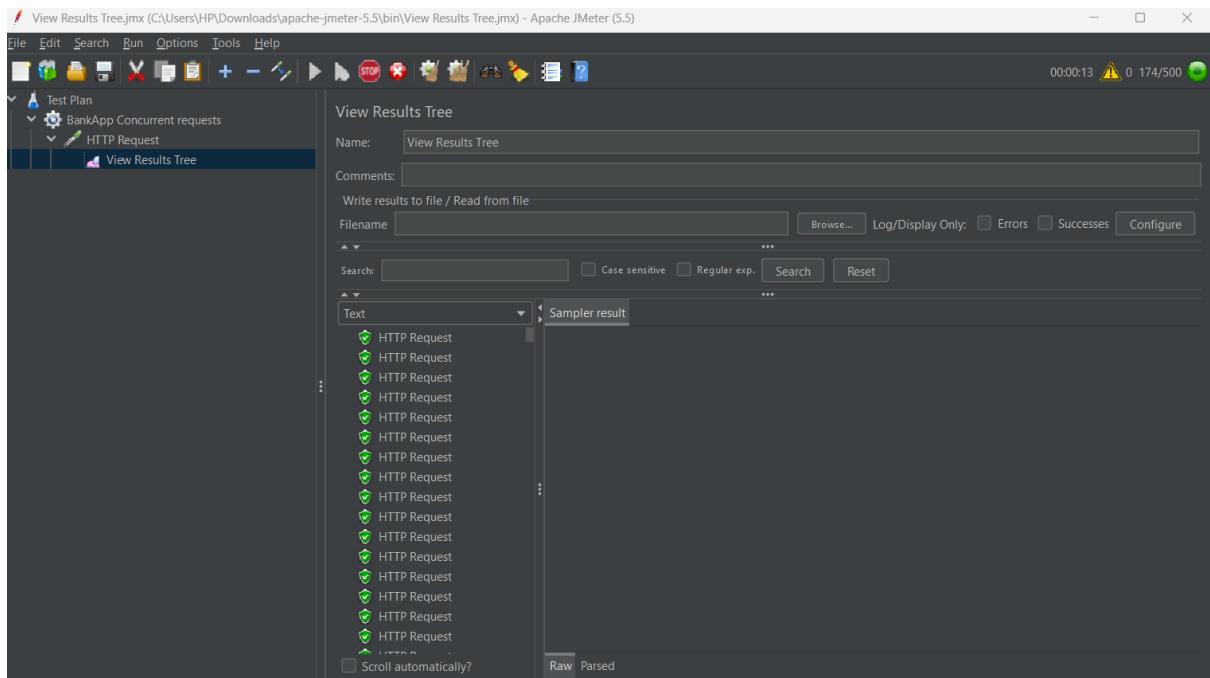


I found the entry for user ritesh in the database bankappdb as shown in the screenshot attached above.

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.







aca-bankapp | Metrics Container App

Search New chart Refresh Share

Max Replica Count for aca-bankapp

+ Add metric Add filter Apply splitting

aca-bankapp, Replica Count, Max

Replica Count (Max), aca-bankapp | 3

3
2.50
2
1.50
1
0.50
0

12 PM 6 PM

aca-bankapp | Log stream Container App

Search Refresh Send us your feedback

Historical Real-time

System Application

aca-bankapp--724q5dk

Not seeing your revision? Click here to find and activate an existing revision.

Reconnect Stop Copy EntityManagerFactory for persist

Maxir

aca-bankapp--724q5dk-6bb8bdb9cb-kvrrs
aca-bankapp--724q5dk-6bb8bdb9cb-kvrrs
aca-bankapp--724q5dk-6bb8bdb9cb-rc5q8
aca-bankapp--724q5dk-6bb8bdb9cb-vttrs

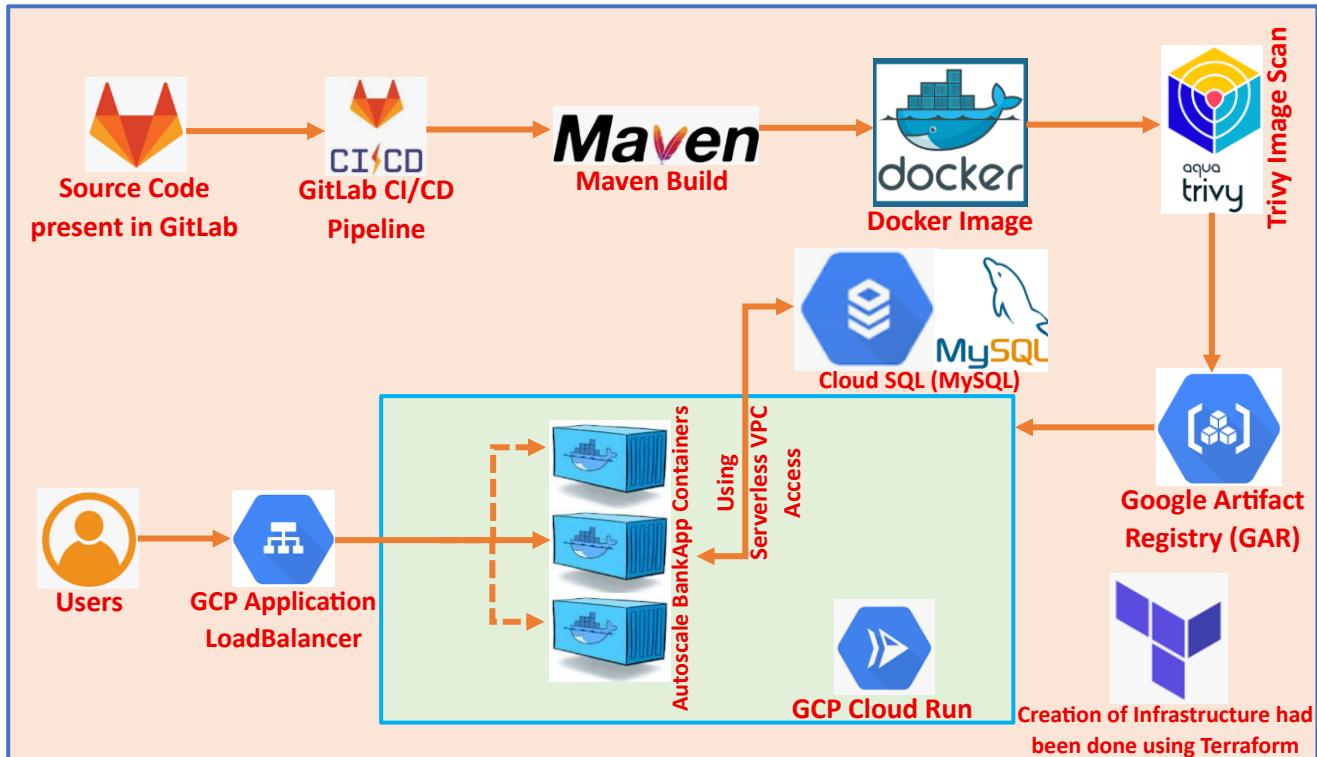
From above screenshot it has been confirmed that replica count increased from 1 to 3 which confirmed Autoscaling.

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>

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). 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, GAR (GCP Artifact Registry), Cloud SQL (MySQL) and Serverless VPC access. However, for your project you can also use the concept that to create the Cloud SQL (MySQL), GAR (GCP Artifact Registry) and GCP VPC using the separate terraform script and then pass the VPC related values as a parameter in the terraform script to create the GCP Cloud Run Service, with this concept you will have the advantage that whenever a new release comes your entire terraform script will not be going to expose. But as it is well known fact that by running the terraform apply will not affect the existing resources until you will not change the parameters. 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>. For the very first time I had created the infrastructure using terraform apply command from Terraform-Server then you can run the GitLab CI/CD Pipeline, however for the very first time this Terraform script will not be executed successfully as this time Cloud SQL (MySQL) and GAR (GCP Artifact Registry) will be created and no Docker Image was present in it, as I explained earlier you can create the GAR(GCP Artifact Registry), Cloud SQL(MySQL) and GCP VPC separately. Terraform state file is present in the GCP Bucket.

```
[root@terraform-server main]# terraform init
```

```
[root@terraform-server main]# terraform apply -auto-approve
```

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 91828) 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.google_dns_record_set_a_record: Creation complete after 19m1s [id=7] [path=...]
module.gcp_cloud_dns_zone.google_dns_record_set_a_record: Creation complete after 2s [id=projects/...-2/.../managedZones/gitlab-public-zone/rrsets/g
itlab.singhritesh85.com/A]

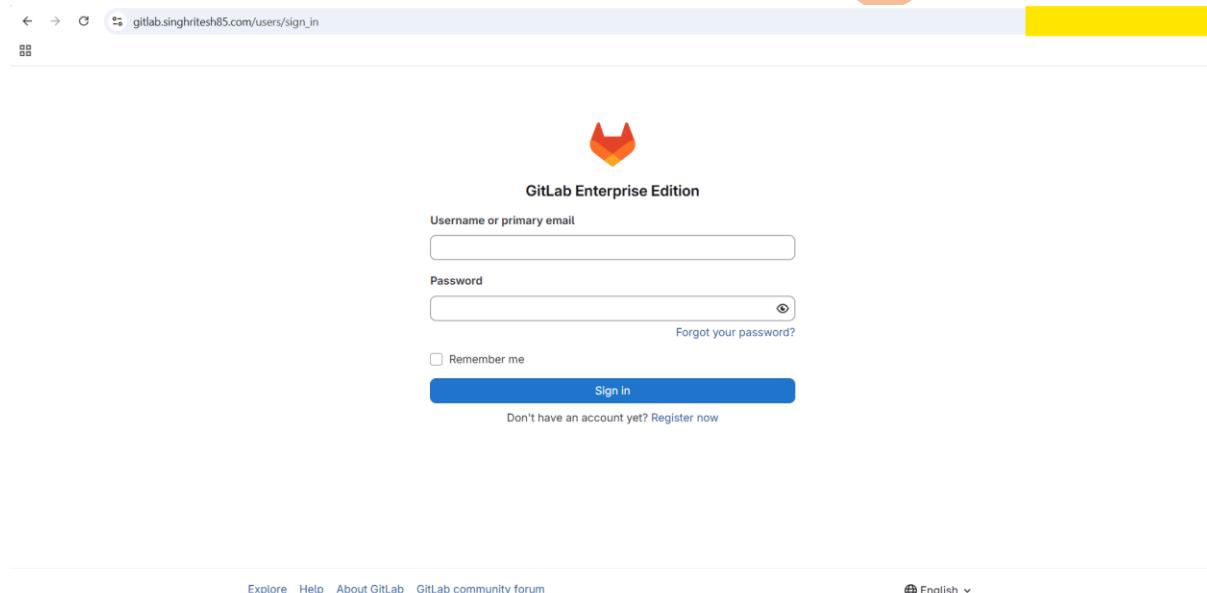
Apply complete! Resources: 33 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_dns_zone_vm_instances_and_gcp_alb_details = {
  "cloud_dns_zone_id" = "projects/...-2/.../managedZones/gitlab-public-zone"
  "gcp_alb_public_ip" = "..."
  "gcp_cloud_dns_zone_name_servers" = tolist([
    ...
    ...
    ...
    ...
  ])
  "gcp_vm_instances_name" = [
    "gitlab-server",
    "gitlab-gitlab-runner",
  ]
  "gcp_vm_private_ip_address" = [
    "172.20.1.1",
    "172.20.1.2",
  ]
  "gcp_vm_public_ip_address" = [
    "...",
    "...",
  ]
}

```

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[REDACTED] = 

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

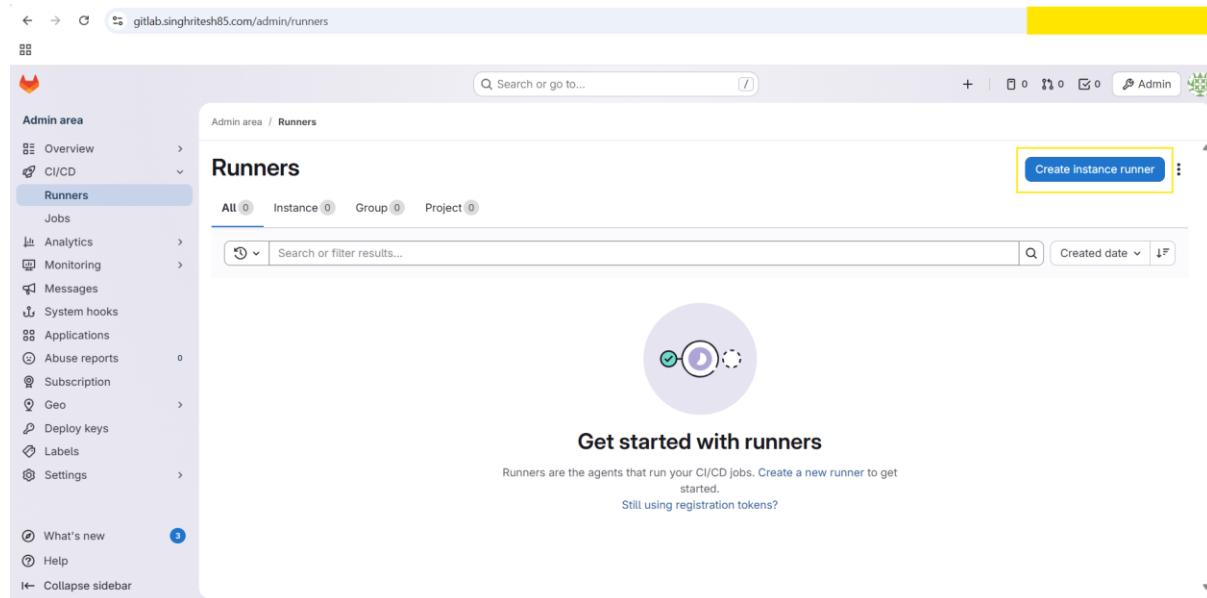
```

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-CI/CD-using-GitLab-GCP-and-AWS.git>.

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.

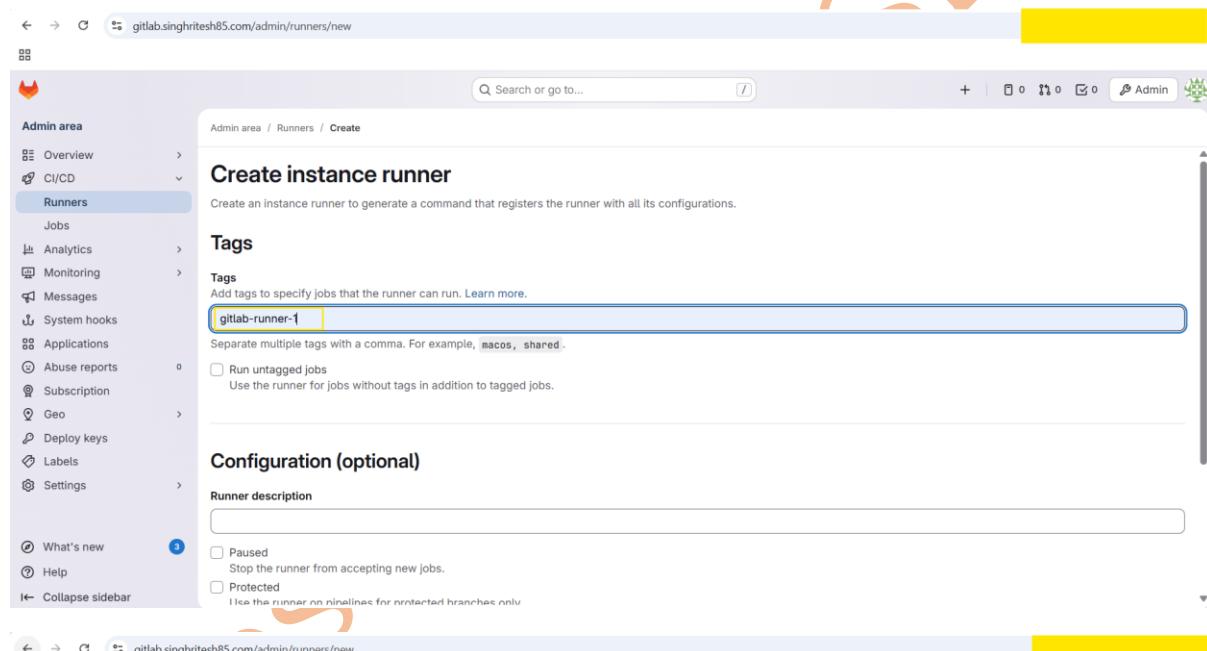
The image contains two screenshots of the GitLab web interface. The top screenshot shows the main user dashboard with a 'Welcome to GitLab, Administrator!' message and several project-related cards. The bottom screenshot shows the 'Admin area / Dashboard' with a focus on the 'CI/CD' section, specifically the 'Runners' sub-section, which is highlighted with a yellow box.



The screenshot shows the 'Runners' page in the GitLab Admin area. The sidebar on the left is collapsed. The main content area has a heading 'Runners' and tabs for 'All', 'Instance', 'Group', and 'Project'. A large button labeled 'Create instance runner' is visible. Below it is a 'Get started with runners' section featuring a circular icon with a green checkmark and a purple outline.

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?



The screenshot shows the 'Create instance runner' form. The 'Tags' section is highlighted with a yellow box. It contains a text input field with 'gitlab-runner-1' and a note: 'Separate multiple tags with a comma. For example, `macos, shared`'. There are two checkboxes: 'Run untagged jobs' (unchecked) and 'Protected' (unchecked).

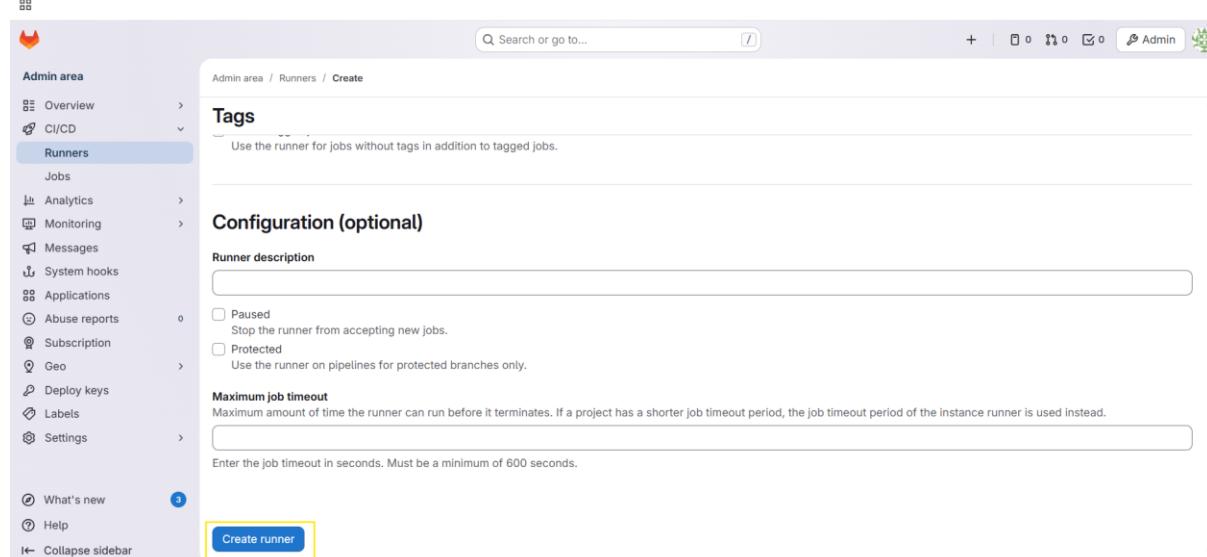
Tags

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.



The screenshot shows the 'Create instance runner' form with the 'Configuration (optional)' section highlighted with a yellow box. It includes a 'Runner description' text input field, a 'Paused' checkbox (unchecked), a 'Protected' checkbox (unchecked), and a note: 'Use the runner on pipelines for protected branches only'. At the bottom is a 'Create runner' button.

Configuration (optional)

Runner description

Paused
Stop the runner from accepting new jobs.

Protected
Use the runner on pipelines for protected branches only.

Create runner

<gitlab.singhritesh85.com/admin/runners/1/register>

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

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?](#)

<gitlab.singhritesh85.com/admin/runners/1/register>

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

Step 1

Copy and paste the following command into your command line to register the runner.

```
$ gitlab-runner register
--url http://gitlab.singhritesh85.com
--token [REDACTED]
```

ⓘ The runner authentication token [REDACTED] displays here for a short time only. After you register the runner, this token is stored in the config.toml and cannot be accessed again from the UI.

Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. Not sure which one to select? [?](#)

Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a system or user service [?](#)

View runners

```
[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[REDACTED]3
Verifying runner... is valid correlation_id=0[REDACTED]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"
```

The screenshot shows the GitLab Admin area with the 'Runners' section selected. The interface includes a sidebar with various admin options like Overview, CI/CD, and Analytics. The main panel shows a summary of runner status: Online (1), Offline (0), and Stale (0). A detailed view of the single online runner ('gitlab-runner-1') is shown, including its status as 'Online (Idle)', last contact time (52 seconds ago), and creation details (Created by Administrator 6 minutes ago).

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

The screenshot shows the GitLab dashboard with the 'Projects' section selected. The main area features a 'Welcome to GitLab, Administrator!' message and a list of initial setup steps: 'Unlock more features with GitLab Ultimate', 'Create a project', 'Create a group', 'Add people', and 'Configure GitLab'. On the right, a sidebar offers options to 'New project/repository', 'New group', and 'New snippet'. A large watermark 'Ritesh' is visible across the bottom left of the screen.

The image consists of three vertically stacked screenshots of the GitLab web interface.

Screenshot 1: Create new project

- The URL is gitlab.singhritesh85.com/projects/new.
- The sidebar shows "Your work" with "Projects" selected.
- The main area has three options: "Create blank project", "Create from template", and "Import project".
- The "Create blank project" section is highlighted with a yellow box.

Screenshot 2: Create blank project

- The URL is gitlab.singhritesh85.com/projects/new#blank_project.
- The sidebar shows "Your work" with "Projects" selected.
- The main area shows project configuration fields:
 - Project name:** Bank-App-GCP-Cloudrun
 - Project URL:** http://gitlab.singhritesh85.com/ dexter
 - Project slug:** bank-app-gcp-cloudrun
 - Visibility Level:** Private (selected)
 - Project Configuration:**
 - Initialize repository with a README
 - Enable Static Application Security Testing (SAST)
 - Enable Secret Detection
- The "Create project" button is highlighted with a yellow box.

Screenshot 3: Project Overview

- The URL is gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun.
- The sidebar shows "Your work" with "Projects" selected.
- The main area shows the repository overview for "Bank-App-GCP-Cloudrun":
 - Project information:**
 - 2 Commits
 - 1 Branch
 - 0 Tags
 - 32 KIB Project Storage
 - Code:** README, Auto DevOps enabled, Add LICENSE, Add CHANGELOG, Add CONTRIBUTING, Add Kubernetes cluster, Add Wiki, Configure Integrations.
 - Files:** main, Dockerfile-Project-1, README.md, deployment.sh, docker-compose.yaml, mvnw, mvnw.cmd, pom.xml.

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 = "XXXXXXXXXXXXXX"
  token_obtained_at = 2025-01-01T12:00:00Z
  token_expires_at = 2025-01-01T12:00:00Z
  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.

Key ↑	Value	Enviro
REPO_NAME	XXXXXXXXXXXXXX	

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

Add variable

Type: Variable (default)

Environments: All (default)

Visibility: Visible (Can be seen in job logs.)

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: IMAGE_NAME

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

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

CI/CD Variables

Key ↑	Value	Enviro
IMAGE_NAME	All (de)
REPO_NAME	All (de)
TAG_NUMBER	1.01	All (de)

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

CI/CD Variables

Key ↑	Value	Enviro
IMAGE_NAME	All (de)
REPO_NAME	All (de)
TAG_NUMBER	All (de)

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 bankappdb
Variable value will be evaluated as raw string.

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

General pipelines
Customize your pipeline configuration.

Auto DevOps
Automate building, testing, and deploying your applications based on your continuous integration and delivery configuration.

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 more about variables.

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 Pipeline variables cannot be used.

Add variable Cancel

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

The screenshot shows the GitLab CI/CD Variables settings page for a project named 'Bank-App-GCP-Cloudrun'. The sidebar on the left is expanded, showing various project settings like Secure, Deploy, Operate, Monitor, Analyze, Settings, General, Integrations, Webhooks, Access tokens, Repository, Merge requests, and CI/CD. The CI/CD section is currently selected.

The main content area displays a table of CI/CD Variables:

Key ↑	Value	Environments	Actions
IMAGE_NAME	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
MYSQL_DATABASE	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
MYSQL_ROOT_PASSWORD	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
REPO_NAME	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
TAG_NUMBER	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

A modal dialog titled 'Add variable' is open on the right side of the screen. It contains fields for 'Type' (set to 'Variable (default)'), 'Environments' (set to 'All (default)'), 'Visibility' (set to 'Masked'), and 'Flags' (checkboxes for 'Protect variable' and 'Expand variable reference' are checked). A description field contains 'GITLAB_TOKEN'. The 'Value' field is a large text input box containing a raw string value.

Pipeline trigger tokens

Triggers a pipeline for a branch or tag by appending a trigger token and using it with an API call. The token is prepended to the pipeline name.

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

The screenshot shows the same CI/CD Variables table, but now it includes an additional row for 'GITLAB_TOKEN' at the bottom:

Key ↑	Value	Environments	Actions
GITLAB_TOKEN	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
IMAGE_NAME	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
MYSQL_DATABASE	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
MYSQL_ROOT_PASSWORD	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
REPO_NAME	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
TAG_NUMBER	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Pipeline trigger tokens

Triggers a pipeline for a branch or tag by appending a trigger token and using it with an API call. The token is prepended to the pipeline name.

gitlab.singhritesh85.com/dexter/bank-app-gcp-cloudrun/-/settings/ci_cd#js-cicd-variables-settings

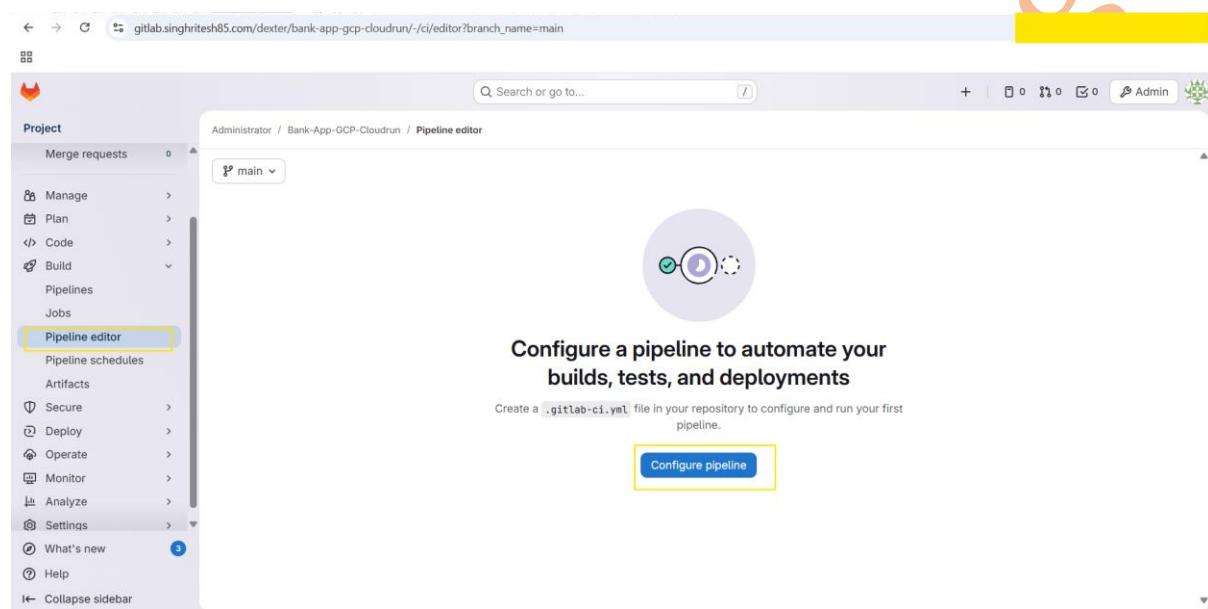
To push Docker Images from gitlab-runner to GCP Artifact Registry (earlier known as GCP Container Registry), you need to authenticate it first using the command as written below.

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 .
- 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.

Update .gitlab-ci.yml file

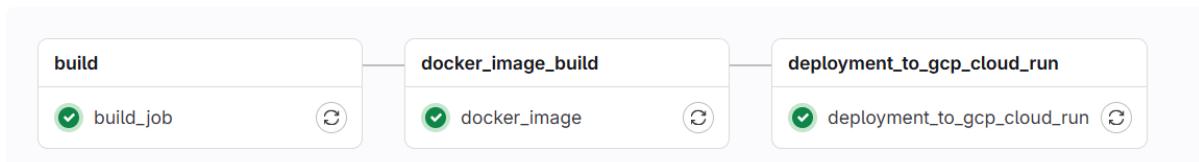
✓ Passed Created 7 minutes ago by Administrator, finished 3 minutes ago

For commit XXXXXXXXXX ↻

In main

latest branch ↻ 3 jobs ⌚ 4 minutes 14 seconds, queued for 1 seconds

Pipeline Jobs (3) Tests (0)



```

    727 19:38:03 module.cloudrun.google_compute_forwarding_rule.cloudrun_forwarding_rule_443: Still creating... [0m
    728 19:38:03 module.cloudrun.google_compute_forwarding_rule.cloudrun_forwarding_rule_443: Still creating... [0m
    729 19:38:05 module.cloudrun.google_compute_forwarding_rule.cloudrun_forwarding_rule_80: Creation complete after
    11s [id:projects/[REDACTED]/regions/us-central1/forwardingRules/cloudrun-forwarding-rule-8
    730 19:38:13 module.cloudrun.google_compute_forwarding_rule.cloudrun_forwarding_rule_443: Still creating... [0m
    731 19:38:14 module.cloudrun.google_compute_forwarding_rule.cloudrun_forwarding_rule_443: Creation complete after
    21s [id:projects/[REDACTED]/regions/us-central1/forwardingRules/cloudrun-forwarding-rule-443]
    732 19:38:14 Apply complete! Resources: 10 added, 1 changed, 1 destroyed.
    733 19:38:14 Outputs:
    734 19:38:14 Details_for_cloud_run_bankapp_alb_public_ip_and_cloud_sql = {
    735 19:38:14   "db_connection_name" = "bankapp-us-central1:cloudrun-private-dbinstance-[REDACTED]"
    736 19:38:14   "db_instance_private_ip_address" = "10.[REDACTED]"
    737 19:38:14   "gcp_alb_public_ip" = "projects/[REDACTED]/regions/us-central1/addresses/cloudrun-cloud
    un-lb-ip"
    738 19:38:14   "repository_endpoint" = "[MASKED]"
    739 19:38:14   "repository_id" = "projects/[REDACTED]/locations/us-central1/repositories/cloudrun-gcr-d
    ev"
    740 19:38:14 }
    741 19:38:14 Cleaning up project directory and file based variables
    742 19:38:14 Job succeeded
  
```

Duration: 3 minutes 34 seconds
Finished: 16 minutes ago
Queued: 2 seconds
Timeout: 1h (from project) ?
Runner: #2 (hqrSYTUm)
Source: Push
Tags: gitlab-runner-1

Commit 8686718a Diff
Update .gitlab-ci.yml file

Pipeline #16 Passed for main View
deployment_to_gcp_cloud_run View

Related jobs
→ Passed deployment_to_gcp_clo...

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.

Forwarding rule name	Frontend type	Scope	Address	IP version	Protocol	Network tier	Load balancer	Labels
cloudrun-forwarding-rule-443	Application	Regional (us-central1)	34.[REDACTED] 443	IPv4	HTTPS	Premium	cloudrun-url-map	
cloudrun-forwarding-rule-80	Application	Regional (us-central1)	34.[REDACTED] 80	IPv4	HTTP	Premium	cloudrun-url-map	
gitlab-lb-frontend-http	Application	Global	35.[REDACTED] 80	IPv4	HTTP	Premium	gitlab-http-redirect	
gitlab-lb-frontend-https	Application	Global	35.[REDACTED] 443	IPv4	HTTPS	Premium	gitlab-urlmap	

Load balancing

Cloud DNS

Cloud CDN

Cloud NAT

Cloud Service Mesh (Traffic Manager)

Service Directory

Cloud Domains

Private Service Connect

SSL policies

Service Extensions

Load balancing

Cloud DNS

Cloud CDN

Cloud NAT

Cloud Service Mesh (Traffic Manager)

Service Directory

Cloud Domains

Private Service Connect

SSL policies

Service Extensions

Marketplace

Release Notes

Edit record set

DNS name: www.singhritesh85.com. ?

Resource record type: A ?

TTL *: 5 ?

TTL unit: minutes ?

IPv4 Address ?

IPv4 Address 1 *: 34.[REDACTED] Select

Example: 192.0.2.91

+ Add item

Save Cancel

Equivalent command line Equivalent REST

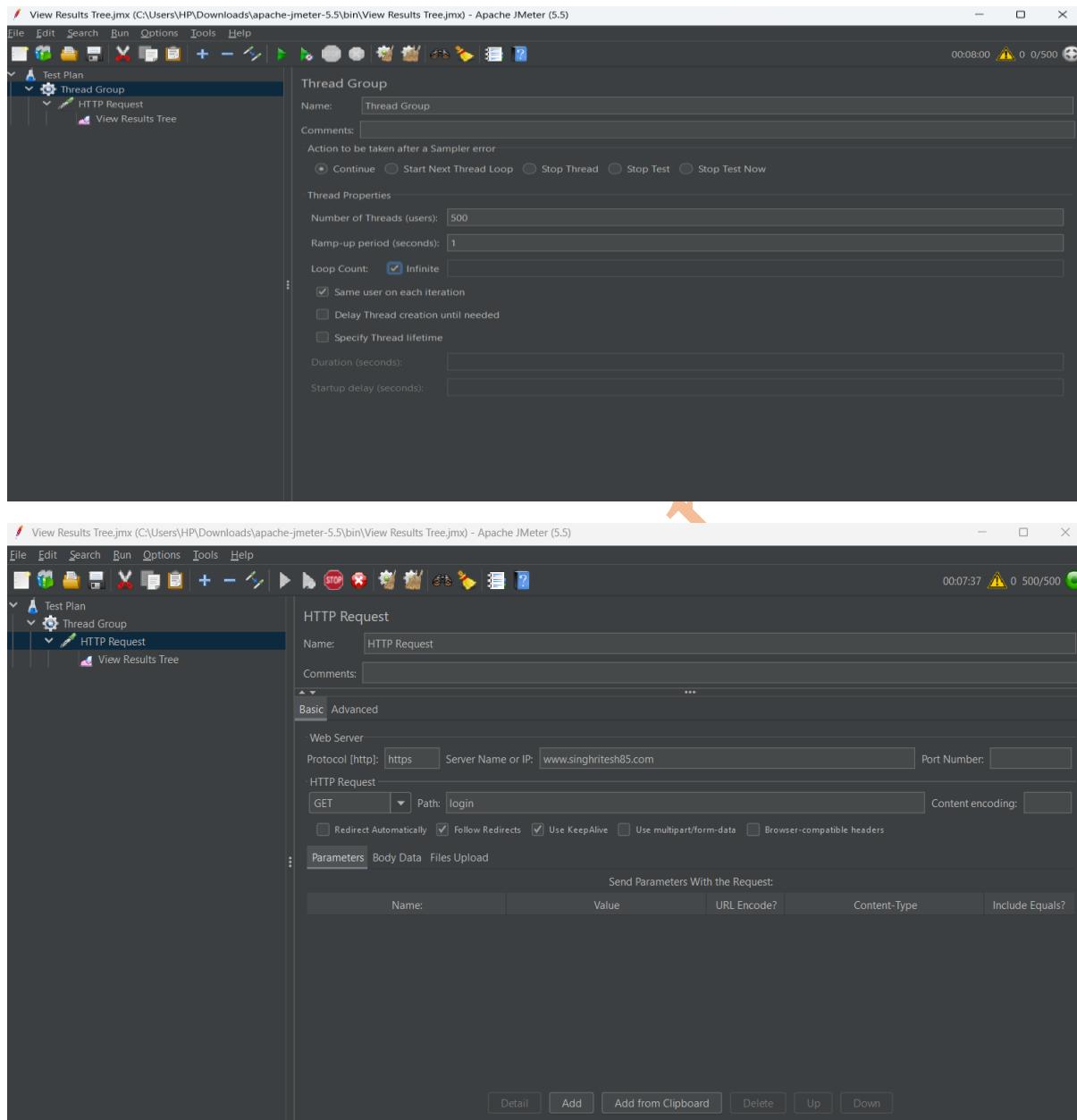
Finally, I was able to access the BankApp using the URL as shown in the screenshot attached below.

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.

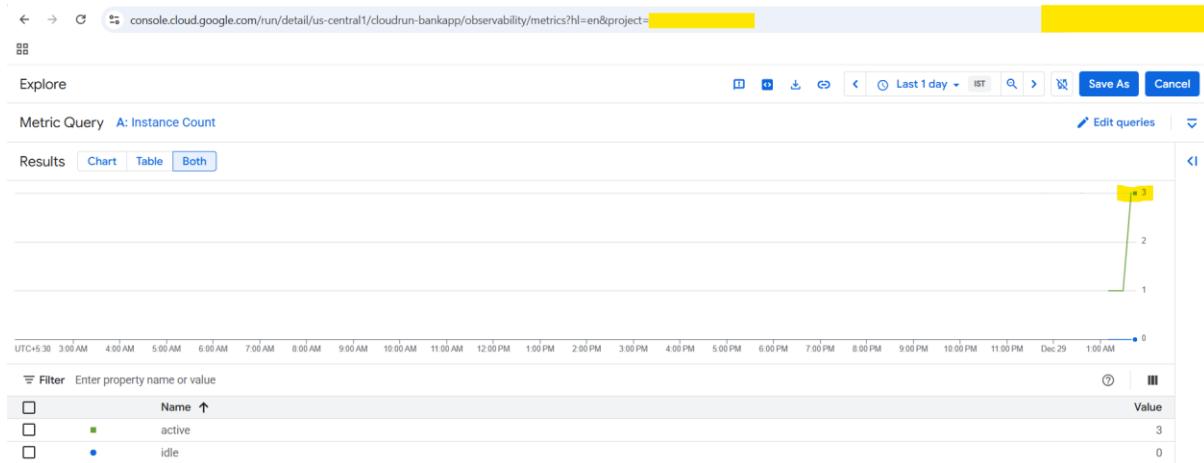
```
template {
  health_check_disabled = false
  max_instance_request_concurrency = 80
  timeout = "300s"

  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>