

DevOps Project BankApp GCP ALB Instance-Group and AutoScaling



By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

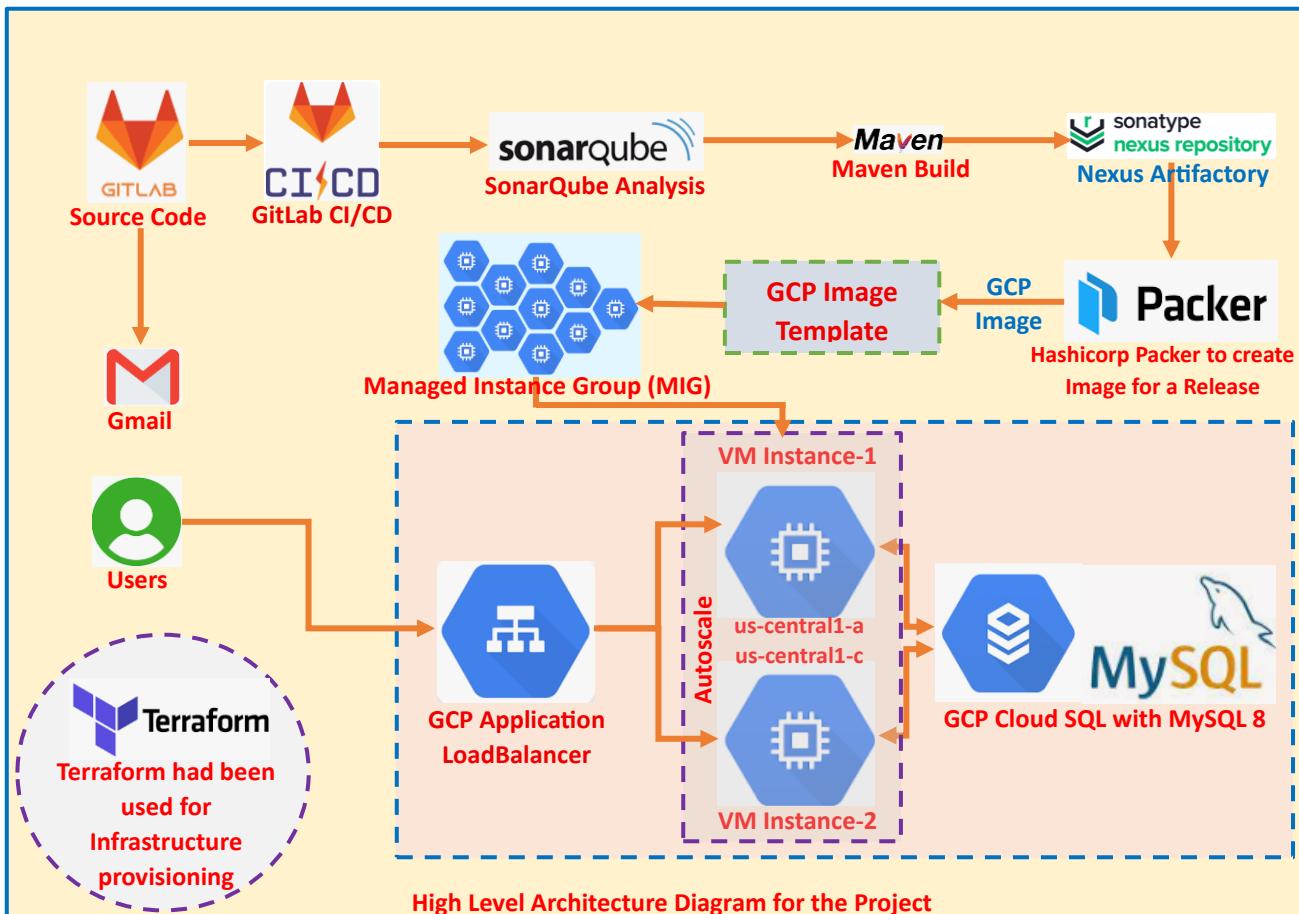
LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/>

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



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

DevOps Project BankApp GCP ALB Instance-Group and AutoScaling



This project deals with creation of Infrastructure using terraform and GitLab had been used as CI/CD tool. For DNS I used Google Cloud DNS. Creation of DNS zone and installation of GitLab Server do not happen daily (not included in day-to-day activity) but for your reference I had provided here the terraform script for both to create the DNS Zone and to install the GitLab Server. To make this handy I provided the terraform script to create the DNS Zone and to install the GitLab Server in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-GCP-ALB-Autoscaling.git> at the path **terraform-gcp-cloud-dns** and **terraform-gitlab-server** respectively. In this project SonarQube (installed on AWS EC2 Instance) had been used for code analysis, Maven used as build tool and Nexus (installed on AWS EC2 Instance) was used as an Artifactory. Hashicorp Packer was used to create the Image for a new Release. This image will be used to create a new GCP instance template and the newly created instance template will be updated with the existing instance-group (managed instance-group) and autoscaling. The Managed Instance-Group (MIG) will create VM-Instances which will autoscale by GCP Autoscaling. In this project for Autoscaling Signals I used CPU Utilization. Finally, the BankApp application starts running in the VM-Instances (which will be created using GCP Instances-Group). The user will access the BankApp application through Application LoadBalancer. The VM-Instances in which BankApp application was running acted as backend of this Application LoadBalancer. I created the GitLab-runner (installed on GCP VM-Instances), Instance-template, Instance-Group with Autoscaling and GCP Application LoadBalancer using the terraform script present at the path **terraform-gcp-global-autoscale-application-loadbalancer** in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-GCP-ALB-Autoscaling.git>. The High-Level architecture diagram of the project is as shown in the screenshot attached above.

Terraform I installed on GCP VM-Instance and state file will be kept in Cloud storage bucket (In GCP Cloud storage bucket automatically enables state lock for terraform, you do not need to explicitly enable a separate state lock feature on the GCS bucket). In this project the state file for Cloud DNS Zone and state file for GCP Instance-template, GCP Instance-Group with Autoscaling and GCP Application LoadBalancer was kept in GCP cloud storage bucket and state file for creation of GitLab Server was kept in S3 Bucket. As I already mentioned it earlier that GitLab Server installation task or Cloud DNS Zone creation tasks does not comes under daily activity but for your reference I provided terraform script in GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-GCP-ALB-Autoscaling.git>. As a part of disaster recovery, I had created the snapshots of EBS Volume attached to the GitLab Server using the Lifecycle Manager Policy and created the snapshots at 11:00 AM and 11:00 PM UTC (twice a day). The below steps and screenshot shown how I had executed the terraform script. Created AMI of SonarQube and Nexus Server twice a day (at 11:00 AM and 11:00 PM). The GitLab runner (installed on GCP VM-Instance) and VM-Instances on which BankApp Application was running, snapshots of its associated disk (boot disk or non-boot disk) were created twice a day using GCP snapshot schedules.

Initially when I created the Instance-template and Instance-Group with Autoscaling using terraform then I disabled the auto-healing in GCP Instance Group which I enabled it later at the time of GitLab CI/CD during the New-Release. Otherwise it will unnecessarily create new vm-instances and delete the old one as initially source-code was not deployed in the vm-instances.

```
# auto_healing_policies {    ### I disable it here otherwise old instances gets deleted as health check fails and new Instances will be created.
#   health_check      = google_compute_health_check.autohealing.id
#   initial_delay_sec = 300
# }

gcloud compute instance-groups managed update bankapp-instance-group --project=XXXX-XXXXXX-XXXX6 --health-check autohealing-health-check --initial-delay 300
--instance-redistribution-type="PROACTIVE" --region=us-central1
```

Before running any terraform script I installed aws-cli on GCP VM-Instance and created an AWS IAM user with appropriate permission to create resources in AWS and provided IAM User's **ACCESS_KEY** and **SECRET_ACCESS_KEY** using the command **aws configure** which is shown in the screenshot attached below.

To install aws cli I installed first python3.9 as shown in the screenshot attached below.

```
[root@terraform-server ~]# yum install -y python3.9
[root@terraform-server ~]# ln -s /usr/bin/python3.9 /usr/bin/python
[root@terraform-server ~]# python --version
Python 3.9.20
```

Then installed aws-cli using the commands as shown below.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

```
[root@terraform-server ~]# cat ~/.bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

PATH="$PATH:/usr/local/bin"
[root@terraform-server ~]# echo $PATH
/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/root/bin
[root@terraform-server ~]# aws --version
aws-cli/1.41.14 Python/3.9.20 Linux/4.18.0-553.58.1.el8_10.x86_64 botocore/1.39.14
```

```
[root@terraform-server ~]# aws configure
AWS Access Key ID [REDACTED]: [REDACTED]
AWS Secret Access Key [REDACTED]: [REDACTED]
Default region name [REDACTED]: [REDACTED]
Default output format [REDACTED]: [REDACTED]
```

The first resource which I created in this project was GCP Cloud DNS.

```
[root@terraform-server ~]# gcloud --version
Google Cloud SDK 528.0.0
alpha 2025.06.20
beta 2025.06.20
bq 2.1.18
bundled-python3-unix 3.12.9
core 2025.06.20
gcloud-crc32c 1.0.0
gsutil 5.34
```

```
[root@terraform-server ~]# cat /etc/os-release
NAME="Rocky Linux"
VERSION="8.10 (Green Obsidian)"
ID="rocky"
ID_LIKE="rhel centos fedora"
VERSION_ID="8.10"
PLATFORM_ID="platform:el8"
PRETTY_NAME="Rocky Linux 8.10 (Green Obsidian)"
ANSI_COLOR="0;32"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:rocky:rocky:8:GA"
HOME_URL="https://rockylinux.org/"
BUG_REPORT_URL="https://bugs.rockylinux.org/"
SUPPORT_END="2029-05-31"
ROCKY_SUPPORT_PRODUCT="Rocky-Linux-8"
ROCKY_SUPPORT_PRODUCT_VERSION="8.10"
REDHAT_SUPPORT_PRODUCT="Rocky Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="8.10"
```

For Google Cloud Authentication and Authorization had been using gcloud auth login as shown in the screenshot attached below.

```
[root@terraform-server ~]# 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:
[REDACTED]

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

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

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.

```
+ enable_logging = true
}

+ dnssec_config {
  + kind      = "dns#managedZoneDnsSecConfig"
  + non_existence = (known after apply)
  + state     = "on"

  + default_key_specs (known after apply)
}
}

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

Changes to Outputs:
+ gcp_cloud_dns_name_and_dns_name = {
  + dns_zone_nameservers = (known after apply)
  + zone_dns_name       = "singhritesh85.com."
  + zone_name            = "public-hosted-zone-logging-enabled"
}
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creating...
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creation complete after 1s [id=projects/[REDACTED]/managedZones/public-hosted-zone-logging-enabled]

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

Outputs:
gcp_cloud_dns_name_and_dns_name = {
  "dns_zone_nameservers" = tolist([
    "ns1.singhritesh85.com."
    "ns2.singhritesh85.com."
    "ns3.singhritesh85.com."
  ])
  "zone_dns_name" = "singhritesh85.com."
  "zone_name" = "public-hosted-zone-logging-enabled"
}
```

Then I updated the Domain Name Provider's nameserver after that In AWS Account I requested a Public SSL Certificate and got it validated using the GCP Cloud DNS record set of Type CNAME as shown in the screenshot attached below.

Record sets

Add standard **Add with routing policy** **Delete record sets** **Refresh**

Filter Filter record sets

DNS name ↑	Type	TTL (seconds)	Record data
singhritesh85.com.	SOA	21600	[REDACTED]
singhritesh85.com.	NS	21600	[REDACTED]

Create record set

DNS name .singhritesh85.com. **Resource record type** CNAME **TTL *** 5 **TTL unit** minutes

Canonical name **Canonical name 1 *** Example: server-1.example.com. **+ Add item**

Create **Cancel**

Finally, the SSL Certificate in AWS Certificate Manager got issued as shown in the screenshot attached below.

Certificate status

- Identifier [REDACTED]
- ARN [REDACTED]
- Type Amazon Issued
- Status Issued

Domains (1)

Domain	Status	Renewal status	Type	CNAME name
*.singhritesh85.com	Success	-	CNAME	[REDACTED]

Create records in Route 53 **Export to CSV**

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.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitaly: (pid 31711) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-exporter: (pid 31720) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-kas: (pid 31000) 256s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-workhorse: (pid 31685) 20s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: logrotate: (pid 30629) 285s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: nginx: (pid 31696) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: node-exporter: (pid 31704) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgres-exporter: (pid 31755) 15s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgresql: (pid 30809) 262s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: prometheus: (pid 31732) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: puma: (pid 31133) 144s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis: (pid 30675) 279s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis-exporter: (pid 31723) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: sidekiq: (pid 31160) 138s
module.gitlab.null_resource.gitlab_server (remote-exec): run: alertmanager: (pid 31747) 17s; run: log: (pid 31545) 89s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitaly: (pid 31711) 21s; run: log: (pid 30745) 272s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-exporter: (pid 31720) 20s; run: log: (pid 31369) 108s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-kas: (pid 31000) 258s; run: log: (pid 31011) 257s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-workhorse: (pid 31685) 22s; run: log: (pid 31240) 132s
module.gitlab.null_resource.gitlab_server (remote-exec): run: logrotate: (pid 30629) 287s; run: log: (pid 30637) 286s
module.gitlab.null_resource.gitlab_server (remote-exec): run: nginx: (pid 31696) 21s; run: log: (pid 31265) 125s
module.gitlab.null_resource.gitlab_server (remote-exec): run: node-exporter: (pid 31704) 21s; run: log: (pid 31333) 114s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgres-exporter: (pid 31755) 17s; run: log: (pid 31587) 83s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgresql: (pid 30809) 264s; run: log: (pid 30833) 261s
module.gitlab.null_resource.gitlab_server (remote-exec): run: prometheus: (pid 31732) 26s; run: log: (pid 31567) 96s
module.gitlab.null_resource.gitlab_server (remote-exec): run: puma: (pid 31133) 146s; run: log: (pid 31140) 145s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis: (pid 30675) 281s; run: log: (pid 30684) 280s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis-exporter: (pid 31723) 26s; run: log: (pid 31409) 102s
module.gitlab.null_resource.gitlab_server (remote-exec): run: sidekiq: (pid 31160) 140s; run: log: (pid 31168) 139s
module.gitlab.null_resource.gitlab_server: Creation complete after 12m6s [id=...]
Apply complete! Resources: 39 added, 0 changed, 0 destroyed.

Outputs:
ec2_private_ip_alb_dns = {
  "EC2_Instance_GitLab_Server_Private_IP_Address" = "10.10.10.10"
  "GitLab_ALB_DNS_Name" = "gitlab-10.10.10.10.us-east-2.elb.amazonaws.com"
}
```

In the GitLab server the Nginx was by default installed which you can check using the command as written below.

```
[root@gitlab-server ~]# gitlab-ctl status nginx
run: nginx: (pid [REDACTED]) [REDACTED]s; run: log: (pid [REDACTED]) [REDACTED]s
```

To configure Gmail to send notification to group Email ID I should have App Password for my Gmail account as shown in the screenshot attached below.

Go to your **Gmail Account > Manage your Google Account > Security** and then search for **app password** and click on **App Passwords** as shown in the screenshot attached below.

The screenshot shows the Google Account interface under the 'Security' tab. A search bar at the top contains the text 'app password'. Below it, a sidebar lists various security-related options like Home, Personal info, Data & privacy, and Payments & subscriptions. The 'Security' tab is selected. A dropdown menu titled 'Google Account results' appears, listing items such as Password Manager, Password, App passwords, Web & App Activity, Help Center articles, Sign in with app passwords, and Use or fix App password. The 'App passwords' item is highlighted with a yellow box. To the right of the dropdown, there's a shield icon with a checkmark and a small diagram of a computer screen with a password field.

Recent security activity
No security activity or alerts in the last 28 days

How you sign in to Google
Make sure you can always access your Google Account by keeping this information up to date

← App passwords

App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

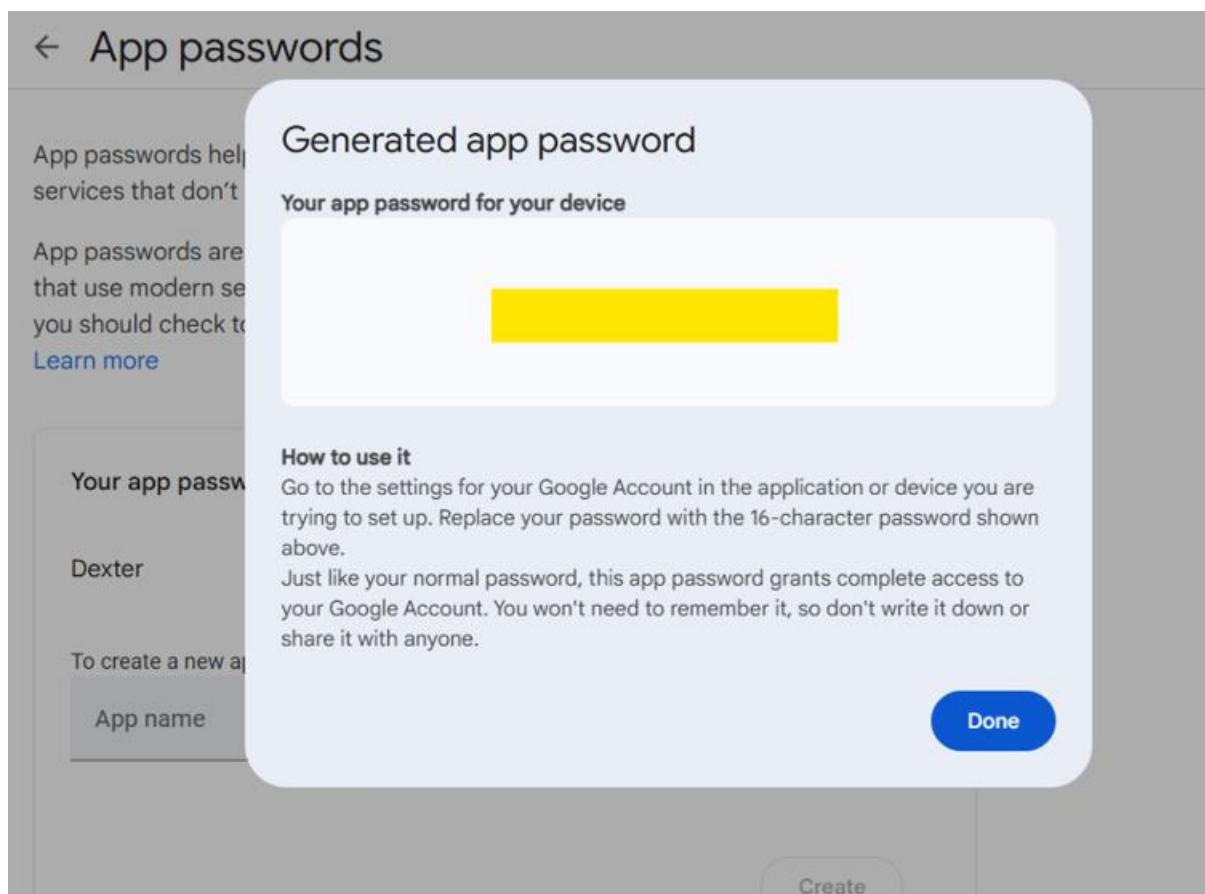
[Learn more](#)

You don't have any app passwords.

To create a new app specific password, type a name for it below...

App name
Dexter

Create



Then I updated the GitLab SMTP information in the file `/etc/gitlab/gitlab.rb` as shown in the screenshot attached below.

```
[root@gitlab-server ~]# vim /etc/gitlab/gitlab.rb
```

```
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.gmail.com"
gitlab_rails['smtp_port'] = 587
gitlab_rails['smtp_user_name'] = "████████████████████████████████"
gitlab_rails['smtp_password'] = "████████████████████████████████"
gitlab_rails['smtp_domain'] = "smtp.gmail.com"
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = true
gitlab_rails['smtp_tls'] = false

gitlab_rails['gitlab_email_from'] = '████████████████████████████████'
gitlab_rails['gitlab_email_display_name'] = 'Dexter'
gitlab_rails['gitlab_email_reply_to'] = '████████████████████████████████'
```

Finally, ran the command `gitlab-ctl reconfigure` as shown in the screenshot attached below.

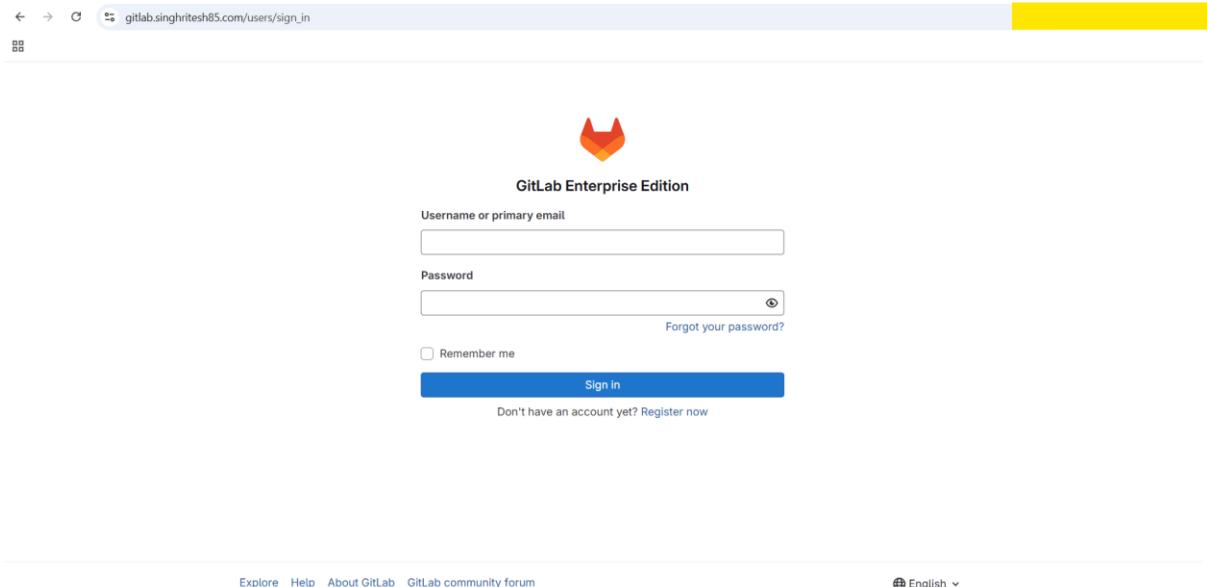
```
[root@gitlab-server ~]# gitlab-ctl reconfigure
```

```
Running handlers:
[2025-07-11T11:45:12.000Z] INFO: Running report handlers
Running handlers complete
[2025-07-11T11:45:12.000Z] INFO: Report handlers complete
Infra Phase complete, 0/871 resources updated in 19 seconds
gitlab Reconfigured!
```

Then I had created the Record Set of Type CNAME in GCP Cloud DNS using the DNS Name of the AWS Application LoadBalancer as shown in the screenshot attached below.

The screenshot shows the 'Create record set' form in the GCP Cloud DNS interface. The 'Resource record type' field is set to 'CNAME', which is highlighted with a yellow box. The 'DNS name' field contains 'gitlab .singhritesh85.com.' and is also highlighted with a yellow box. Other fields visible include 'TTL *' (set to 5), 'TTL unit' (set to minutes), and a 'Canonical name' section with a single item 'Canonical name 1 *' (highlighted with a yellow box) containing a redacted value. At the bottom, there are 'Add item' and 'Create' buttons, with 'Create' being highlighted with a yellow box.

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



Now you can Login into the GitLab using the Root user credentials, remember the time when you configured GitLab the root user password had been generated and present in the file **/etc/gitlab/initial_root_password** which will be valid for next 24 hours So utilised it within 24 hours after its installation.

After Login change the administrator username, password, and email for that go to the admin area and edit the Administrator user as shown in the screenshot attached below.

The screenshot shows two consecutive pages from the GitLab Admin area.

Page 1: User List

- The URL is gitlab.singhritesh85.com/admin/users.
- The sidebar on the left shows the "Users" section selected.
- The main content displays the "Users" section with three cards:
 - Pending approval: 0
 - Administrators: 1
 - Without two-factor authentication: 1
- A table lists the user "Administrator" with the following details:

Name	Projects	Groups	Created on	Last activity
Administrator gitlab_admin_0cc9e8@example.com <small>Admin It's you!</small>	0	0	2025	2025
- Buttons include "New user" and "Edit" for the administrator entry.

Page 2: Edit User Screen

- The URL is gitlab.singhritesh85.com/admin/users/root/edit.
- The sidebar on the left shows the "Users" section selected.
- The main content shows the "Edit user: Administrator" screen under the "Account" tab.
- Fields filled in:
 - Name: Administrator
 - Username: root
 - Email: [REDACTED]
- Under the "Password" tab, fields are shown:
 - Password: [REDACTED]
 - Password confirmation: [REDACTED]
- Buttons at the bottom include "Save changes" (highlighted with a yellow box) and "Cancel".

Then it will ask to sign-in again using the new credentials and a password change notification will come to your email Id as shown in the screenshot attached below.

The screenshot shows two consecutive pages from the GitLab Admin Area.

Top Page (gitlab.singhritesh85.com/admin/users/root):

- Left Sidebar:** Admin area, Overview, Dashboard, Projects, **Users** (selected), Groups, Topics, GitLab servers, CI/CD, Analytics, Monitoring, Messages, System hooks, Applications, What's new (4), Help, Admin.
- Header:** gitlab.singhritesh85.com/admin/users/root
- Content:** Administrator (Admin) - Account tab. Profile picture, Profile page: root. Account details: Name: Administrator, Username: root, Email: [REDACTED]@gmail.com (Verified). Profile: Member since Jul 28, 2025 5:52am.
- Buttons:** New identity, Edit.

Bottom Page (gitlab.singhritesh85.com/admin/users/root/edit):

- Left Sidebar:** Admin area, Overview, Dashboard, Projects, **Users** (selected), Groups, Topics, GitLab servers, CI/CD, Analytics, Monitoring, Messages, System hooks, Applications, What's new (4), Help, Admin.
- Header:** gitlab.singhritesh85.com/admin/users/Administrator/Edit
- Content:** Edit user: Administrator
- Account Section:**
 - Name: Administrator
 - Username:** dexter (highlighted with a yellow box)
 - Email: [REDACTED]@gmail.com
- Password Section:**
 - Password:
 - Password confirmation:
- Buttons:** Save changes (highlighted with a yellow box), Cancel.

A large orange watermark "Ritesh" is diagonally across the bottom left of the screenshots.

The screenshot shows the GitLab Admin area interface. The left sidebar is titled 'Admin area' and includes sections for Overview, Dashboard, Projects, Users (which is selected), Groups, Topics, GitLab servers, CI/CD, Analytics, Monitoring, Messages, System hooks, Applications, What's new (with 4 notifications), and Help. The main content area is titled 'Administrator' and shows the 'Account' tab selected. It displays the user's profile picture, name ('Administrator'), username ('dexter'), email ('[REDACTED]@gmail.com' with a 'Verified' badge), and other account details like email confirmation code ('never'). A success message at the top states 'User was successfully updated.' There are buttons for 'New identity' and 'Edit'.

Now if you sign-in again then you need to provide the new credentials. Go to the Administrator profile and check the same from in email and notification as shown in the screenshot attached below.

The screenshot shows the GitLab user profile page for the 'Administrator' user. The left sidebar includes options for Set status, Edit profile (which is highlighted in yellow), Preferences, Sign out, Issues, Merge requests, To-Do List, Milestones, Snippets, and Activity. The main content area features a 'Welcome to GitLab, Administrator!' message and a 'Ready to get started with GitLab? Follow these steps to get familiar with us:' section. This section contains four cards: 'Unlock more features with GitLab Ultimate' (with a 'Start free trial' button), 'Create a project' (describing projects as storage for code, issues, wiki, and other features), 'Create a group' (describing groups for organizing projects and people), and 'Add people' (describing how to add team members). A large orange 'K' watermark is overlaid on the left side of the screenshot.

User Settings / Emails

Email addresses

Control emails linked to your account

Linked emails 2

[\[REDACTED\]@gmail.com](#) Verified

- Primary email
Used for avatar detection. You can change it in your profile settings.
- Commit email
Used for web based operations, such as edits and merges.
- Default notification email
Used for account notifications if a group-specific email address is not set.

[gitlab_admin_0cc9e8@example.com](#) Verified

Add new email

You can delete this email

Now go to notification and update with your newly added email as shown in the screenshot attached below.

User Settings / Notifications

Notifications

You can specify notification level per group or per project.

Global notification email

[REDACTED]@gmail.com

Global notification level

By default, all projects and groups use the global notifications setting.

Participate

Receive notifications about your own activity

Groups 0

You do not belong to any groups yet.

Projects 0

To specify the notification level per project of a group you belong to, visit the project page and change the notification level there.

You do not belong to any projects yet.

Now, the username, password and email of the Administrator was updated as shown in the screenshot attached below.

Then I had created a user named as **deployment** (regular user) which I only use in CI/CD operation. To achieve the same, I went to Admin area > Users > New User as shown in the screenshot attached below.

New user

Account

Name: deployment

Username: deployment

Email: deployment@gmail.com

Password

Reset link will be generated and sent to the user. User will be forced to set the password on first sign in.

Access

Create user Cancel

Now newly created user will login into his/her email ID and will reset the password as shown the screenshot attached below

Account was created for you

Dexter <deployment@gmail.com>

Hi deployment!

Your account has been created successfully.

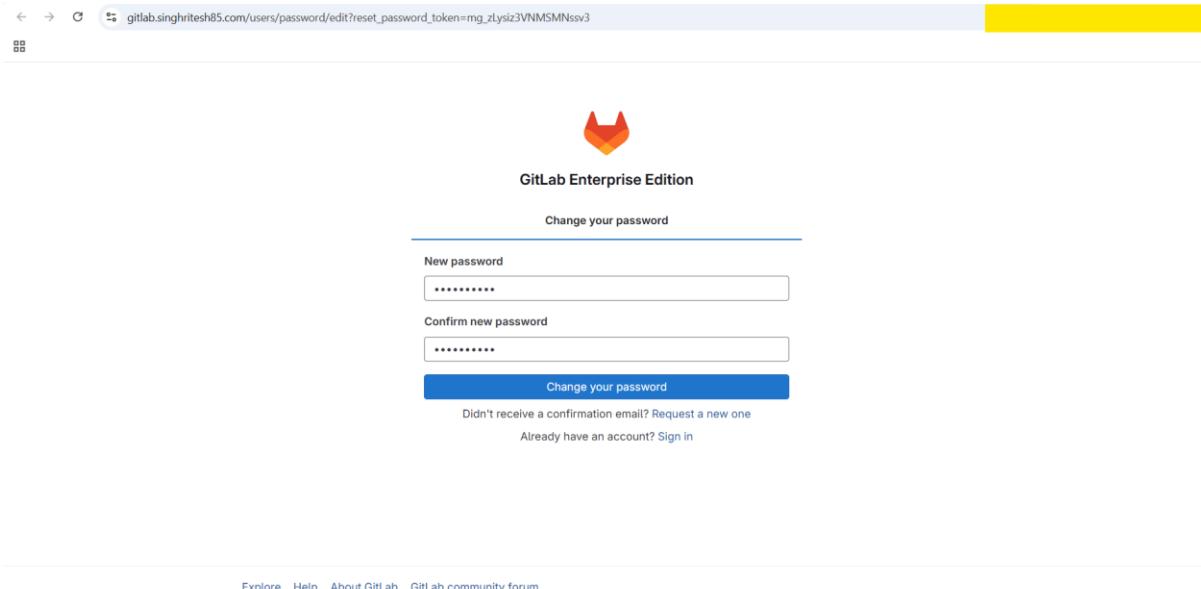
login.....deployment@gmail.com

[Click here to set your password](#)

This link is valid for 2 days. After it expires, you can [request a new one](#).

GitLab

You're receiving this email because of your account on [gitlab.singhritesh85.com](#). [Manage all notifications](#) · [Help](#)



Now run the terraform script to create the GCP GitLab-Runner (on GCP VM-Instance), Instance-template, Instance-Group with Autoscaling, SonarQube Server (on EC2 Instance) and Nexus Server (on EC2 Instance).

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.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m08s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m10s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m20s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m30s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m40s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [19m50s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m00s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m10s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m20s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m30s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m40s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [20m50s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [21m00s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [21m10s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Still creating... [21m20s elapsed]
module.autoscale_alb.google_sql_database_instance.db_instance: Creation complete after 21m31s [id=bankapp-private-dbinstance-[REDACTED]]
module.autoscale_alb.google_sql_database.dexter_dbschema: Creating...
module.autoscale_alb.google_sql_user.db_users: Creating...
module.autoscale_alb.google_sql_database.dexter_dbschema: Creation complete after @s [id=projects/[REDACTED]/instances/bankapp-private-dbinstance-[REDACTED]]
module.autoscale_alb.google_sql_database.dexter_dbschema: Creation complete after @s [id=projects/[REDACTED]/instances/bankapp-private-dbinstance-[REDACTED]]
[REDACTED]databases/bankappdb]
module.autoscale_alb.google_sql_user.db_users: Creation complete after 1s [id=dbadmin/%/bankapp-private-dbinstance-[REDACTED]]
```

Apply complete! Resources: 96 added, 0 changed, 0 destroyed.

Outputs:

```
gcp_instance_template_vm_instance_autoscale_alb_mysql_gitlab_vm_instance_private_and_static_ip_gcp_alb_static_ip = {
  "db_connection_name" = "[REDACTED];us-central1:bankapp-private-dbinstance-[REDACTED]"
  "db_instance_name" = "[REDACTED]-us-central1:bankapp-private-dbinstance-[REDACTED]"
  "db_instance_private_ip_address" = "10.[REDACTED]"
  "gcp_alb_static_ip" = "34.[REDACTED]"
  "gitlab_runner_vm_instance_private_ip_address" = "172.[REDACTED]"
  "gitlab_runner_vm_instance_public_ip_address" = "34.[REDACTED]"
  "instance_template_id" = "projects/[REDACTED]/global/instanceTemplates/bankapp-template"
  "instance_template_name" = "bankapp-template"
  "instance_template_self_link" = "https://www.googleapis.com/compute/v1/projects/[REDACTED]/global/instanceTemplates/bankapp-template"
}
```

After creation of GCP and AWS Resources you need to validate the SSL Certificate in GCP Cloud DNS, to do so I created a Record-Set of Type CNAME as shown in the screenshot attached below.

Certificates / Certificate: bankapp-global-cert	
	Edit Delete
Create time	2025, 5:21:14 PM
Update time	2025, 5:21:15 PM
Status	pending
Expiry time	N/A
Region	global
Key distribution scope	DEFAULT ?
Hostnames	*.singhrithesh85.com
Certificate type	Google-managed
Labels	goog-terra...:true
In use by	None

[Equivalent REST](#)

Certificate's DNS Authz resources

Filter Enter property name or value ?					
Status	Name ↑	Domain	DNS Record Name	DNS Record Type	DNS Record Data
pending	bankapp-dns-auth	singhrithesh85.com	.singhrithesh85.com.	CNAME	



[Create record set](#)

DNS name

.singhrithesh85.com. [?](#)

Resource record type

[?](#)

TTL *

? [?](#)

TTL unit

[?](#)

Canonical name [?](#)

Canonical name 1 *

Example: server-1.example.com.

[+ Add item](#)

[Create](#)

[Cancel](#)

Check after 10-15 minutes the GCP SSL certificate gets validated.

 Add Certificate	 Refresh	 Delete							
<input type="text"/> Filter Enter property name or value									?
Status	Certificate name	Region	Hostnames	Type	Create time	Expire time	Key distribution scope	?	Labels
<input checked="" type="checkbox"/> active	bankapp-global-cert	global	*.singhritesh85.com	Google-managed	Jul 28, 2025	Oct 26, 2025	Default		goog-terra.

I create the URL for SonarQube and Nexus Server by doing the entry of DNS Name of the LoadBalancer sonarqube and nexus in the GCP Cloud DNS to create the Record Set of Type CNAME as shown in the screenshot attached below.

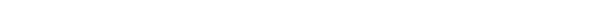
Record sets				
+ Add standard		+ Add with routing policy	Delete record sets	Refresh
Filter record sets				
<input type="checkbox"/>	DNS name ↑	Type	TTL (seconds)	Record data
<input type="checkbox"/>	[REDACTED]	CNAME	300	[REDACTED] ▼ ✎
<input type="checkbox"/>	[REDACTED]	CNAME	300	[REDACTED] ▼ ✎
<input type="checkbox"/>	gitlab.singhritesh85.com.	CNAME	300	gitlab-[REDACTED].us-east-2.elb.amazonaws.com. ✎
<input type="checkbox"/>	singhritesh85.com.	SOA	21600	ns-cloud-b1.googledomains.com.cloud-dns-[REDACTED] ▼ ✎
<input type="checkbox"/>	singhritesh85.com.	NS	21600	ns-cloud-b1.googledomains.com. ▼ ✎

 Create record set

DNS name	sonarqube .singhritesh85.com.	?
Resource record type	CNAME	?
TTL *	5	?
TTL unit	minutes	?

Canonical name [?](#)

Canonical name 1 * _____



Example: server-1.example.com.

[+ Add item](#)

Cancel

Create record set

DNS name [?](#)

Resource record type [?](#)

TTL * [?](#)

TTL unit [?](#)

Canonical name [?](#)

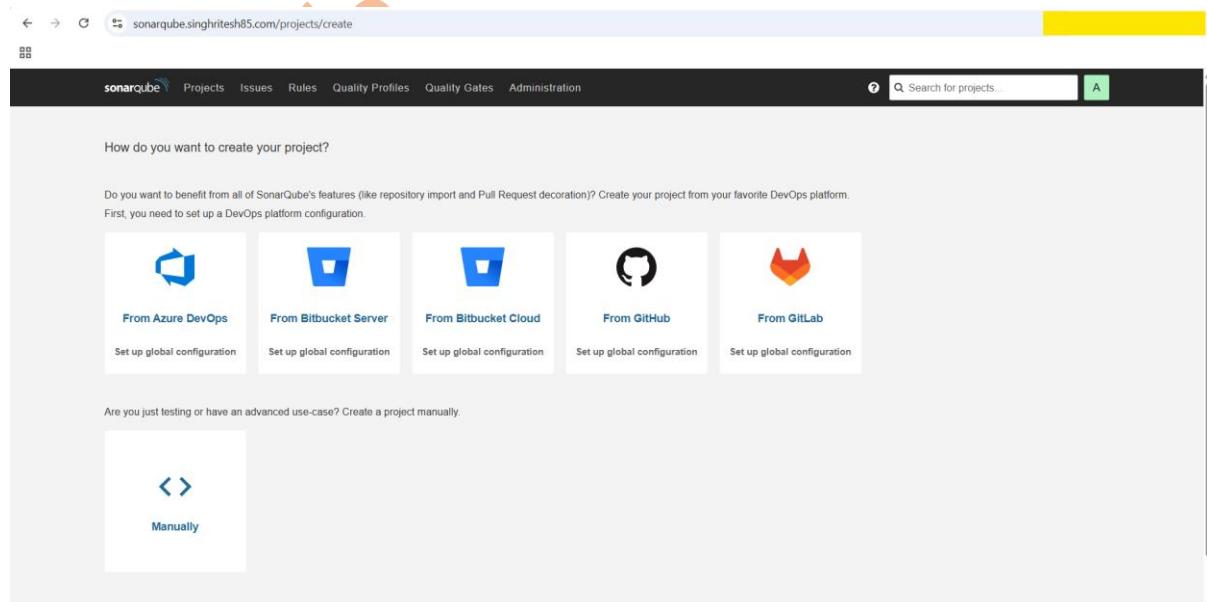
Canonical name 1 *

Example: server-1.example.com.

[+ Add item](#)

[Create](#) [Cancel](#)

Finally, I was able to access SonarQube and Nexus as shown in the screenshot attached below.



In Nexus I created two Repositories with the name of **maven-snapshot** and **maven-release** and in SonarQube I generated the Token as shown in the screenshot attached below.

Name	Type	Project	Last use	Created	Expiration
SonarQube	Global	-	Never	July 28, 2025	-

I added the GitLab Runner to the GitLab Server as shown in the screenshot attached below. Go to Admin area > CI/CD > Runners > Create Instance runner as shown in the screenshot attached below.

In the Tags provide **gitlab-runner-1** as the tag name and then click on create runner and select Operating System as Linux then go to your gitlab-runner instance and run the command **gitlab-runner register** to register the gitlab-runner with the gitlab server and subsequently provide the GitLab URL and Runner Authentication Token as shown in the screenshot attached below. Then you can check the status of gitlab-runner using the command **gitlab-runner status**.

```
[root@bankapp-gitlab-runner ~]# gitlab-runner register
Runtime platform          arch=amd64 os=linux pid=[REDACTED] revision=[REDACTED] version=18.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.singhritesh85.com/
Enter the registration token:
[REDACTED]
Verifying runner... is valid correlation_id=[REDACTED] runner=[REDACTED]
Enter a name for the runner. This is stored only in the local config.toml file:
[bankapp-gitlab-runner]: gitlab-runner-1
Enter an executor: shell, ssh, parallels, docker, docker-windows, docker+machine, docker-autoscaler, custom, virtualbox, 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"

[root@bankapp-gitlab-runner ~]# gitlab-runner status
Runtime platform          arch=amd64 os=linux pid=[REDACTED] revision=[REDACTED] version=18.2.0
gitlab-runner: Service is running
```

Then click on view Runner and you will find Runner is in Online state as shown in the screenshot attached below.

You can enable two-factor authentication (2FA) for all users to login into GitLab for that please refer the project present in GitHub Repo https://github.com/singhritesh85/DevOps-Project-BankApp-CI_CD-using-GitLab-GCP-and-AWS.git.

Before proceeding further make sure clone_url should be added on GitLab-Runner in the file `/etc/gitlab-runner/config.toml` as shown in the screenshot attached below.

```
[root@bankapp-gitlab-runner ~]# vim /etc/gitlab-runner/config.toml

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 = 1
    token = "[REDACTED]" ]
    token_obtained_at = 2025 [REDACTED]
    token_expires_at = 0001-01-01T00:00:00Z
    executor = "shell"
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

Then I created two Repositories in Sonatype Nexus named as **maven-snapshot** and **maven-release**.

 maven-snapshot	hosted	maven2	default	Online				
 maven-release	hosted	maven2	default	Online				

For this project the source Code was present in the GitLab Repo as shown in the screenshot attached below. But for your reference I kept it along with `.gitlab-ci.yml` file in GitHub Repo <https://github.com/singhritesh85/Bank-App-GCP-ALB-Autoscaling.git>.

Bank-App-GCP-ALB-Autoscaling

Project information

- > 1 Commit
- > 1 Branch
- 0 Tags
- 28 KIB Project Storage
- Auto DevOps enabled
- + Add README
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Add Kubernetes cluster
- + Add Wiki
- + Configure Integrations

Created on
July 28, 2025

Before creating the GitLab CI/CD pipeline, I created variables which will be used during execution of CI/CD pipeline as shown in the screenshot attached below.

Go to **GitLab > GitLab Project/Repo > Settings > CI/CD > Variables** and Add new variables as shown in the screenshot attached below.

Bank-App-GCP-ALB-Autoscaling

Project information

- > 1 Commit
- > 1 Branch
- 0 Tags
- 28 KIB Project Storage
- Auto DevOps enabled
- + Add README
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Add Kubernetes cluster
- + Add Wiki
- + Configure Integrations

Created on
July 28, 2025

Variables

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more](#).

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.
- Owner
- Maintainer
- Developer

Save changes

Access protected resources in merge request pipelines

Make protected CI/CD variables and runners available in merge request pipelines. Protected resources will only be available in merge request pipelines if both the source and target branches of the merge request are protected. [Learn more](#).

Allow merge request pipelines to access protected variables and runners

Save changes

Project variables

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

CI/CD Variables < 0 **Add variable**

Add variable

Type: Variable (default)

Environments: All (default)

Visibility: Masked
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.

Flags: Protect variable Export variable to pipelines running on protected branches and 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.

The screenshot shows two views of the GitLab CI/CD Settings page for a project named "Bank-App-GCP-ALB-Autoscaling".

Top View: Shows the "Variables" section under the "CI/CD" tab. It includes settings for the minimum role to use pipeline variables (Developer selected), a note about protected resources in merge request pipelines (checkbox checked), and a "Project variables" section. A modal window titled "Add variable" is open, showing fields for "Key" (SONARQUBE_TOKEN), "Value" (redacted), and "Flags" (Protect variable checked). The "Add variable" button is highlighted with a yellow box.

Bottom View: Shows the same "Variables" section with the "Add variable" modal still open. The "Add variable" button is again highlighted with a yellow box. The "Project variables" section is also visible.

The screenshot shows two instances of the GitLab CI/CD Settings page for a project named "Bank-App-GCP-ALB-Autoscaling".

Top Window (Variable Creation):

- Project Variables:** A table lists variables: SONARQUBE_TOKEN (Protected, Masked, Expanded) and NEXUS_USER (Protected, Masked, Expanded).
- Flags:** Options include Protect variable (checked), Expand variable reference (checked), and Description (optional) field.
- Add Variable Form:** Fields include Key (NEXUS_USER), Value (admin), and Environment (All (default)). Buttons: Add variable (highlighted), Cancel.

Bottom Window (Variable Configuration):

- Project Variables:** A table lists variables: NEXUS_USER (Protected, Expanded) and SONARQUBE_TOKEN (Protected, Masked, Expanded).
- Flags:** Options include Protect variable (checked), Expand variable reference (checked), and Description (optional) field.
- Add Variable Form:** Fields include Key (NEXUS_USER), Value (admin), Environment (All (default)), and Visibility (Masked). Buttons: Add variable (highlighted), Cancel.

The screenshot shows two screenshots of the GitLab CI/CD Settings page for a project named "Bank-App-GCP-ALB-Autoscaling".

Screenshot 1 (Top): Adding a new CI/CD Variable

- Project Variables:** A table lists three variables:
 - NEXUS_USER**: Value is masked (****), Environment is All (default), Flags include Protected (checked) and Expanded.
 - SONARQUBE_TOKEN**: Value is masked (****), Environment is All (default), Flags include Protected (checked), Masked (checked), and Expanded.
 - NEXUS_PASSWORD**: Key is highlighted, Value is masked (****), Environment is All (default), and a tooltip explains the use of variables across environments.
- Flags:** Options include Protect variable (checked) and Expand variable reference (checked).
- Description (optional):** A text input field with placeholder "The description of the variable's value or usage."
- Key:** A text input field containing "NEXUS_PASSWORD".
- Value:** A text input field containing "password".
- Add variable:** A blue button with a plus sign.

Screenshot 2 (Bottom): List of CI/CD Variables

- CI/CD Variables:** A table lists the same three variables:
 - NEXUS_PASSWORD**: Value is masked (****), Environment is All (default), Actions column has edit and delete icons.
 - NEXUS_USER**: Value is masked (****), Environment is All (default), Actions column has edit and delete icons.
 - SONARQUBE_TOKEN**: Value is masked (****), Environment is All (default), Actions column has edit and delete icons.
- Pipeline trigger tokens:** A section with a brief description.
- Deploy freezes:** A section with a brief description.

Now create the GitLab CI/CD Pipeline, go to your GitLab Project/GitLab Repo > Build > Pipeline editor and provide/write the **.gitlab-ci.yml** file as shown in the screenshot

The screenshot shows two views of the GitLab Pipeline editor for the project "Bank-App-GCP-ALB-Autoscaling".

Main Dashboard:

- Header:** Shows the URL `gitlab.singhrites85.com/dexter/bank-app-gcp-alb-autoscaling/-/ci/editor?branch_name=main`.
- Project Sidebar:** Includes options like Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor (which is selected), Pipeline schedules, Artifacts, Secure, Deploy, Operate, What's new (4 notifications), Help, and Admin.
- Pipeline Status:** Shows a green circle with a blue play button icon and a circular progress bar.
- Section Header:** "Optimize your workflow with CI/CD Pipelines".
- Text:** "Create a new `.gitlab-ci.yml` file at the root of the repository to get started."
- Button:** "Configure pipeline" (highlighted with a yellow box).

Configuration Editor:

- Header:** Shows the URL `gitlab.singhrites85.com/dexter/bank-app-gcp-alb-autoscaling/-/ci/editor?branch_name=main`.
- Project Sidebar:** Same as the main dashboard.
- Validation Message:** "✓ Pipeline syntax is correct. Learn more"
- Toolbar:** Edit, Visualize, Validate (NEW), Full configuration.
- Code View:** Displays the `.gitlab-ci.yml` configuration file with syntax highlighting for YAML and shell commands.

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

stages:
  - build_and_sonarqube_analysis
  - nexus_artifact_upload
  - deployment

build_job:
  stage: build_and_sonarqube_analysis
  before_script:
    - export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.16.0.8-2.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 sonar:sonar --sonar.qualitygate.wait=true --sonar.qualitygate.timeout=600 --sonar.projectKey=bankapp --sonar.pr...
```

Using this GitLab-CI/CD Pipeline (through .gitlab-ci.yml file) what I am trying to achieve is as mentioned below:-

There are three stages in this CI/CD Pipeline.

1. In **build_and_sonarqube_analysis** stage performed SonarQube Analysis and Build the code using Maven.
 2. In **nexus_artifact_upload** stage Upload the Artifacts to Nexus Artifactory.
 3. In **deployment** stage I created the GCP Image using Hashicorp Packer in this Image I copied the **.jar** file using file provisioner then using this Image, GCP Image-template had been created then with this new image-template the existed Managed Image-Group (MIG) was updated and I also enabled the auto-healing in GCP MIG. In the newly created VM-Instances I deployed the **.jar** file using the systemctl command and started it from boot-time as shown in the screenshot attached below. The Linux service to deploy the **.jar** file had been created using bootstrap script **startup-image-template.sh** present in the GitHub Repo
<https://github.com/singhritesh85/Bank-App-GCP-ALB-Autoscaling.git>.

Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/> || GitHub: - <https://github.com/singhriteshs85>

```

cat > /opt/bankapp-shell-script.sh <<BANKAPP
#!/bin/bash

BANKAPP=`ps -ef|grep "java -jar /opt/bankapp/bankapp.jar"|grep -v "color=auto"|awk '{print $2}'`>
echo $BANKAPP
kill -9 $BANKAPP
nohup java -jar /opt/bankapp/bankapp.jar >/dev/null 2>&1 &
BANKAPP

chmod +x /opt/bankapp-shell-script.sh
cp /opt/bankapp-shell-script.sh /usr/local/bin/
```



```

cat > /etc/systemd/system/bankapp.service <<END_FOR_SCRIPT
[Unit]
Description=bankapp service
After=network.target

[Service]
Type=forking
Environment="PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/lib/jvm/java-17-openjdk-17.0.16.0.8-2.el8.x86_64/bin"
ExecStart=/usr/local/bin/bankapp-shell-script.sh start
User=root
Restart=on-failure

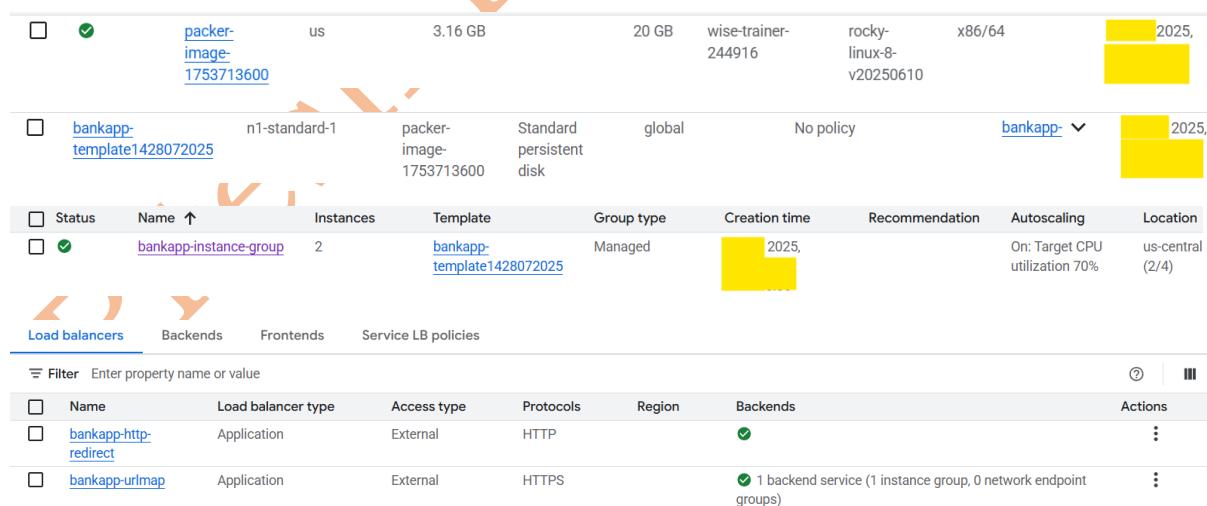
[Install]
WantedBy=multi-user.target
END_FOR_SCRIPT

systemctl daemon-reload
systemctl enable bankapp
systemctl start bankapp
systemctl status bankapp

```



Below screenshots shows GCP Image, Instance-Template, Instance-Group and LoadBalancer.



The screenshot displays the GCP interface for creating infrastructure components:

- Image:** A table showing details for the 'packer-image-1753713600' image, including Region (us), Disk Type (3.16 GB), and Labels (wise-trainer-244916, rocky-linux-8-v20250610).
- Instance Template:** A table showing the 'bankapp-template1428072025' template, which uses the 'n1-standard-1' machine type, the 'packer-image-1753713600' image, and a Standard persistent disk.
- Instance Group:** A table showing the 'bankapp-instance-group' managed instance group, which contains 2 instances based on the 'bankapp-template1428072025' template.
- Load Balancer:** A table showing the 'bankapp-http-redirect' application load balancer, which has one backend service (1 instance group, 0 network endpoint groups) and is configured with HTTP protocol.

I had created a Record-Set of A-Type in GCP Cloud DNS with GCP Application LoadBalancer Static IP as shown in the screenshot attached below.

Load balancers	Backends	Frontends	Service LB policies					
<input type="text"/> Filter Enter property name or value								?
<input type="checkbox"/>	Forwarding rule name ↑	Frontend type	Scope	Address	IP version	Protocol	Network tier	Load balancer
<input type="checkbox"/>	bankapp-lb-frontend-http	Application	Global	34 [REDACTED]:80	IPv4	HTTP	Premium	bankapp-http-redirect
<input type="checkbox"/>	bankapp-lb-frontend-https	Application	Global	34 [REDACTED]:443	IPv4	HTTPS	Premium	bankapp-urlmap

← Create record set

DNS name ?

Resource record type ▼ ?

TTL * ?

TTL unit ▼ ?

IPv4 Address ?

IPv4 Address 1 * Select

Example: 192.0.2.91

+ Add item

Create

Cancel

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

The image displays two screenshots of a web-based banking application, likely a clone of a real banking system.

Login Page (Top Screenshot):

- URL: bankapp.singhritesh85.com/login
- Header: Goldencat Bank
- Form Fields:
 - Username: [Redacted]
 - Password: [Redacted]
- Buttons:
 - Yellow 'Login' button
 - Text link: Don't have an account? [Register here](#)
- Footer: © 2024 Code With Goldencat. All rights reserved. | [Privacy Policy](#) | [Terms of Service](#)

Dashboard (Bottom Screenshot):

- URL: bankapp.singhritesh85.com/dashboard
- Header: Goldencat Bank
- Navigation: Dashboard, Transactions, Logout
- Welcome Message: Welcome, ritesh
- Current Balance: \$0.00
- Account Details:
 - Account Number: 1
 - Account Type: Savings
- Action Buttons:
 - Deposit
 - Withdraw
 - Transfer Money

```
[root@10.0.0.10 ~]# mysql -h 10.10.0.0.10 -u dbadmin --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5444
Server version: 8.0.41-google (Google)

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 (0.02 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 (0.01 sec)

mysql> select * from account;
+----+-----+-----+-----+
| id | balance | password | username |
+----+-----+-----+-----+
| 1 |    0.00 | [REDACTED] | ritesh   |
+----+-----+-----+-----+
1 row in set (0.01 sec)
```

The entry for Cloud DNS Zone for Record Sets is as shown in the screenshot attached below.

Record sets

+ Add standard + Add with routing policy Delete record sets Refresh

Filter Filter record sets

<input type="checkbox"/> DNS name ↑	Type	TTL (seconds)	Record data		
<input type="checkbox"/> .singhritesh85.com.	CNAME	300			
<input type="checkbox"/> singhritesh85.com.	CNAME	300			
<input type="checkbox"/> bankapp.singhritesh85.com.	A	300			
<input type="checkbox"/> gitlab.singhritesh85.com.	CNAME	300			
<input type="checkbox"/> nexus.singhritesh85.com.	CNAME	300			
<input type="checkbox"/> singhritesh85.com.	SOA	21600			
<input type="checkbox"/> singhritesh85.com.	NS	21600			
<input type="checkbox"/> sonarqube.singhritesh85.com.	CNAME	300			

Source Code: <https://github.com/singhritesh85/Bank-App-GCP-ALB-Autoscaling.git>

GitHub Repo: <https://github.com/singhritesh85/DevOps-Project-BankApp-GCP-ALB-Autoscaling.git>

Terraform Script: <https://github.com/singhritesh85/DevOps-Project-BankApp-GCP-ALB-Autoscaling.git>

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