

DevOps Project BankApp Deployment using GitLab Google-Cloud-Platform (GCP) and AWS



By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

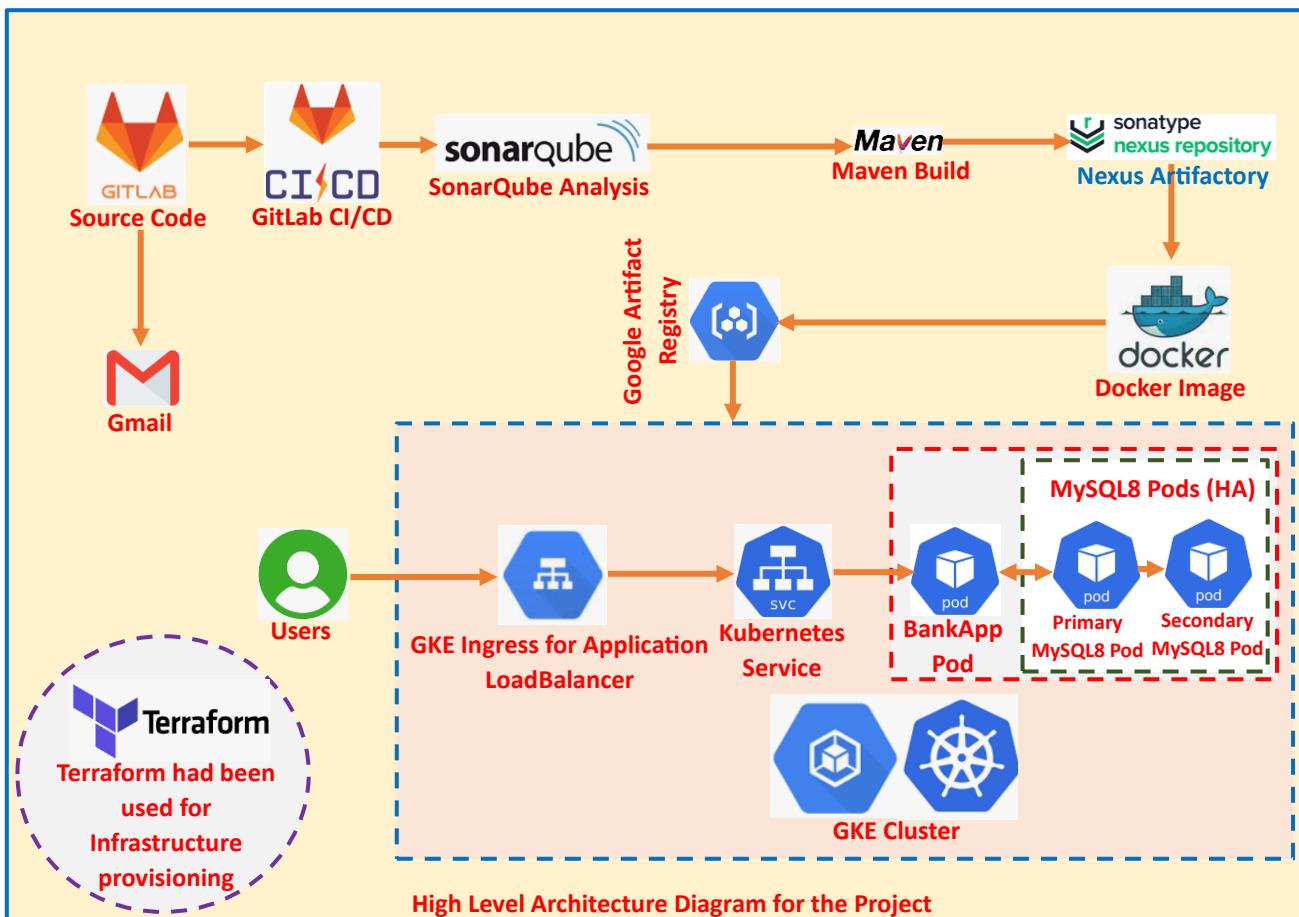
LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/>

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



या कुद्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाङ्घापहा ॥

DevOps Project BankApp Deployment using GitLab CI/CD GCP and AWS



High-level architecture diagram of the project is as shown in the screenshot attached above. The source code and helm charts for bankapp and bitnami mysql8 pods were present in the GitLab Private Repository. I had installed the GitLab on EC2 Instance using the Terraform script and for your reference it is available in the GitHub Repo <https://github.com/singhritesh85/terraform-gitlab-server.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 show how I had executed the terraform script.

The first resource which I created in this project was GCP Cloud DNS which I created using Terraform script present in the GitHub Repo <https://github.com/singhritesh85/terraform-google-cloud-platform.git> at the path **terraform-gcp-cloud-dns**. The state file for terraform I kept in the GCP bucket which automatically acquire state lock.

```
[root@terraform-server main]# 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 main]# 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.google_dns_managed_zone.cloud_logging_enabled_zone = {
  + 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([
    "192.168.1.1",
    "192.168.1.2",
    "192.168.1.3",
    "192.168.1.4"
  ])
  "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
Filter Filter record sets			
DNS name	Type	TTL (seconds)	Record data
singhritesh85.com.	SOA	21600	
singhritesh85.com.	NS	21600	

← 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	Status Issued			
ARN				
Type	Amazon Issued			
Domains (1)				
Create records in Route 53 Export to CSV				
Domain	Status	Renewal status	Type	CNAME name
*.singhritesh85.com	Success	-	CNAME	

For this project to create the GitLab Server I installed Terraform on GCP VM Instance and used Amazon S3 Bucket to store the terraform state file and to achieve the state lock. Below Screenshot shows how I ran the terraform script and created the Resources in AWS Cloud. For Authentication and Authorization, I installed **aws cli** on GCP VM instance and created an IAM user in AWS account and used its access key and secret key as shown in the screenshot attached below (This terraform script for GitLab Server I made handy for you so that if you want you can directly run it in your AWS environment, that is the reason I am not storing the terraform state file in GCP bucket rather than I used AWS S3 Bucket).

```
[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"
```

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 ~]# vim ~/.bashrc
[root@terraform-server ~]# logout
[ritesh@terraform-server ~]$ sudo -i
[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/2.27.50 Python/3.13.4 Linux/4.18.0-553.58.1.el8_10.x86_64 exe/x86_64.rocky.8
```

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.

```
[root@terraform-server main]# aws configure
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: us-east-2
Default output format [None]: text

module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-exporter: (pid 31149) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-kas: (pid 30462) 237s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: gitlab-workhorse: (pid 31124) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: logrotate: (pid 30089) 266s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: nginx: (pid 31132) 19s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: node-exporter: (pid 31143) 18s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgres-exporter: (pid 31191) 14s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: postgresql: (pid 30271) 243s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: prometheus: (pid 31167) 17s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: puma: (pid 30594) 141s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis: (pid 30136) 260s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: redis-exporter: (pid 31151) 17s
module.gitlab.null_resource.gitlab_server (remote-exec): ok: run: sidekiq: (pid 30622) 135s
module.gitlab.null_resource.gitlab_server (remote-exec): run: alertmanager: (pid 31183) 16s; run: log: (pid 30927) 88s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitaly: (pid 31158) 19s; run: log: (pid 30205) 255s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-exporter: (pid 31149) 20s; run: log: (pid 30823) 106s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-kas: (pid 30462) 239s; run: log: (pid 30472) 238s
module.gitlab.null_resource.gitlab_server (remote-exec): run: gitlab-workhorse: (pid 31124) 21s; run: log: (pid 30697) 128s
module.gitlab.null_resource.gitlab_server (remote-exec): run: logrotate: (pid 30089) 268s; run: log: (pid 30101) 266s
module.gitlab.null_resource.gitlab_server (remote-exec): run: nginx: (pid 31132) 21s; run: log: (pid 30727) 121s
module.gitlab.null_resource.gitlab_server (remote-exec): run: node-exporter: (pid 31143) 20s; run: log: (pid 30789) 112s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgres-exporter: (pid 31191) 16s; run: log: (pid 30966) 82s
module.gitlab.null_resource.gitlab_server (remote-exec): run: postgresql: (pid 30271) 245s; run: log: (pid 30295) 243s
module.gitlab.null_resource.gitlab_server (remote-exec): run: prometheus: (pid 31167) 19s; run: log: (pid 30893) 94s
module.gitlab.null_resource.gitlab_server (remote-exec): run: puma: (pid 30594) 143s; run: log: (pid 30605) 140s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis: (pid 30136) 262s; run: log: (pid 30145) 261s
module.gitlab.null_resource.gitlab_server (remote-exec): run: redis-exporter: (pid 31151) 19s; run: log: (pid 30858) 100s
module.gitlab.null_resource.gitlab_server (remote-exec): run: sidekiq: (pid 30622) 137s; run: log: (pid 30630) 135s
module.gitlab.null_resource.gitlab_server: Creation complete after 11m33s [REDACTED]

Apply complete! Resources: 39 added, 0 changed, 0 destroyed.

Outputs:

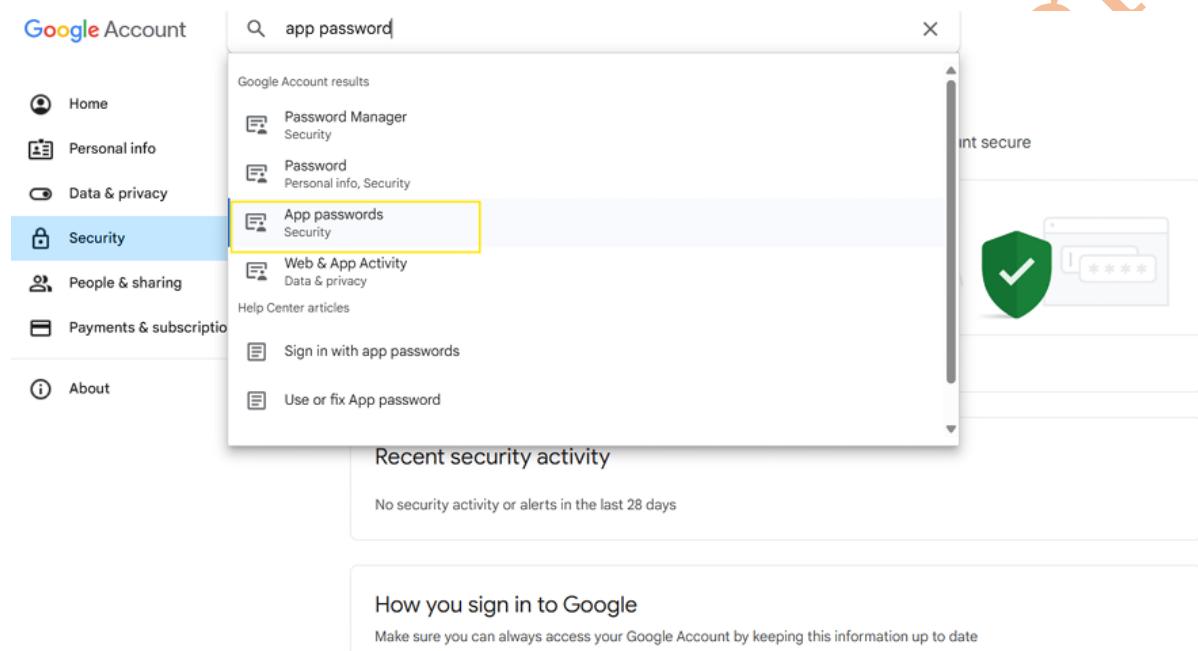
ec2_private_ip_alb_dns = {
  "EC2_Instance_GitLab_Server_Private_IP_Address" = "10. [REDACTED]"
  "GitLab_ALB_DNS_Name" = "gitlab-[REDACTED].us-east-2.elb.amazonaws.com"
}
```

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]) 334s; run: log: (pid [REDACTED]) 434s
[root@gitlab-server ~]#
```

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.



← 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](#)

← App passwords

App passwords help you sign into your Google Account on older apps and services that don't support 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](#)

Your app password
Dexter

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

App name

Generated app password

Your app password for your device

[Done](#)

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

[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██████████] INFO: Running report handlers
Running handlers complete
[2025██████████] INFO: Report handlers complete
Infra Phase complete, 0/864 resources updated in 20 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.

Record sets			
+ Add standard + Add with routing policy Delete record sets Refresh			
<input type="checkbox"/> Filter Filter record sets			
	Type	TTL (seconds)	Record data
<input type="checkbox"/>	CNAME	300	██████████
<input type="checkbox"/>	SOA	21600	██████████
<input type="checkbox"/>	NS	21600	██████████

Create record set

DNS name ?

Resource record type ?

TTL * ?

TTL unit ?

Canonical name ?

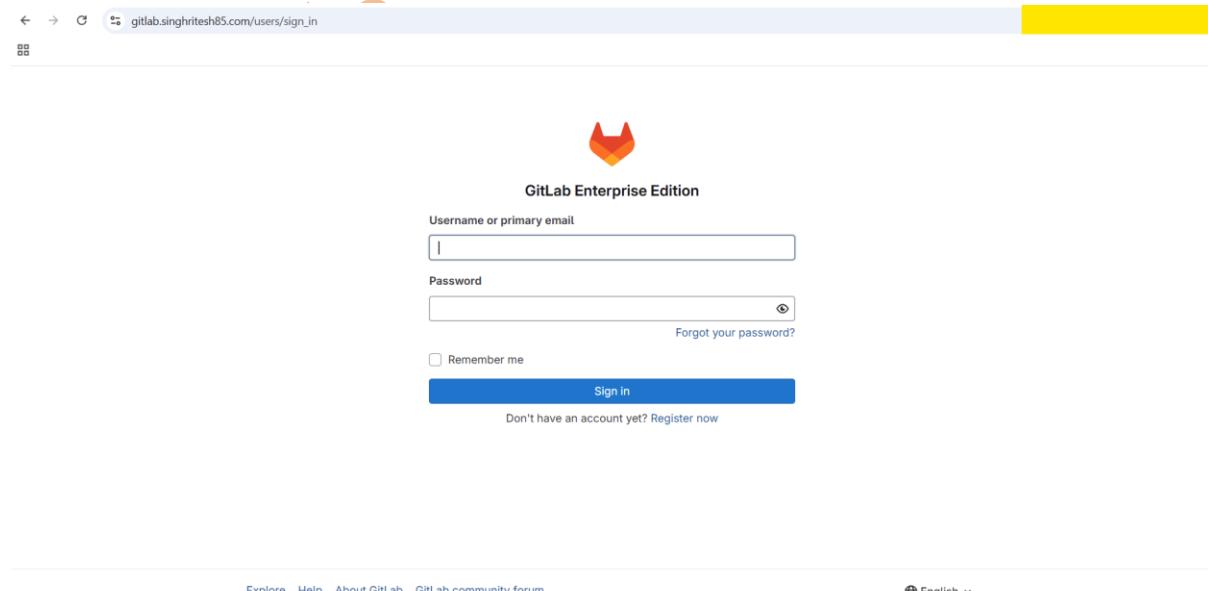
Canonical name 1 *

Example: server-1.example.com.

+ Add item

Create Cancel

Finally, I was able to access 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 image contains two screenshots of the GitLab web interface. The top screenshot shows the main dashboard ('Your work / Projects') with a sidebar on the left containing 'Projects', 'Groups', 'Issues', 'Merge requests', 'To-Do List', 'Milestones', 'Snippets', and 'Activity'. The top navigation bar has a yellow box around the 'Admin' button. The bottom screenshot shows the 'Admin area / Users' page, specifically the 'Users' section. It displays three boxes: 'Pending approval' (0), 'Administrators' (1), and 'Without two-factor authentication' (1). Below these is a table with columns 'Name', 'Projects', 'Groups', 'Created on', and 'Last activity'. A single user row is shown for 'Administrator' (gitlab_admin_0cc9e8@example.com), which includes a yellow box around the 'Edit' button.

gitlab.singhritesh85.com/admin/users/root/edit

Edit user: Administrator

Account

- Name: Administrator
- Username: dexter
- Email: [REDACTED]@gmail.com

Password

- Password: [REDACTED]
- Password confirmation: [REDACTED]

Buttons: Save changes, 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.

Password Changed Inbox x

R Dexter <[REDACTED]@gmail.com>
to me ▾

Hello, Administrator!

The password for your GitLab account on <http://gitlab.singhritesh85.com> has successfully been changed.

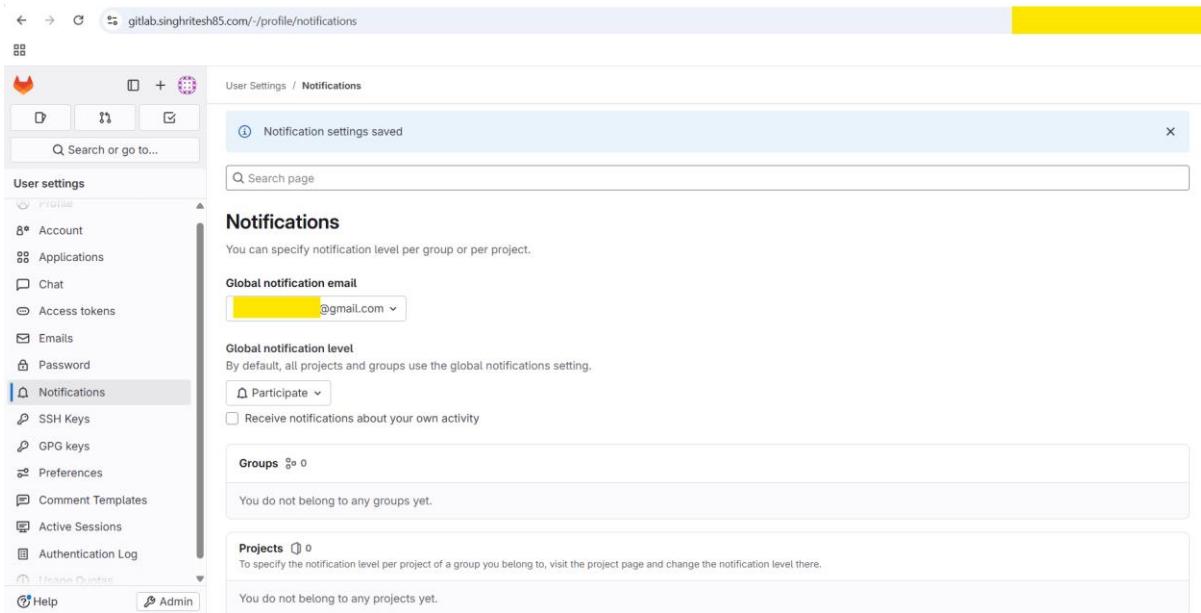
If you did not initiate this change, please contact your administrator immediately.

Now 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 welcome page for the user 'Administrator @dexter'. The sidebar on the left includes options like Set status, Edit profile (which is highlighted), Preferences, Sign out, Issues, Merge requests, To-Do List, Milestones, Snippets, and Activity. The main content area features a 'Welcome to GitLab, Administrator!' header, a brief introduction, and several call-to-action boxes: 'Unlock more features with GitLab Ultimate' (with a 'Start free trial' button), 'Create a project', 'Create a group', 'Add people', and 'Configure GitLab'. At the bottom of the sidebar, the 'Emails' option is also highlighted.

This screenshot shows the 'Email addresses' section of the user settings. It lists two linked emails: one at '@gmail.com' which is verified and set as the primary email, and another at 'gitlab_admin_0cc9e8@example.com' which is also verified. A red arrow points to the trash icon next to the second email address, with the text 'You can delete this email' written below it.

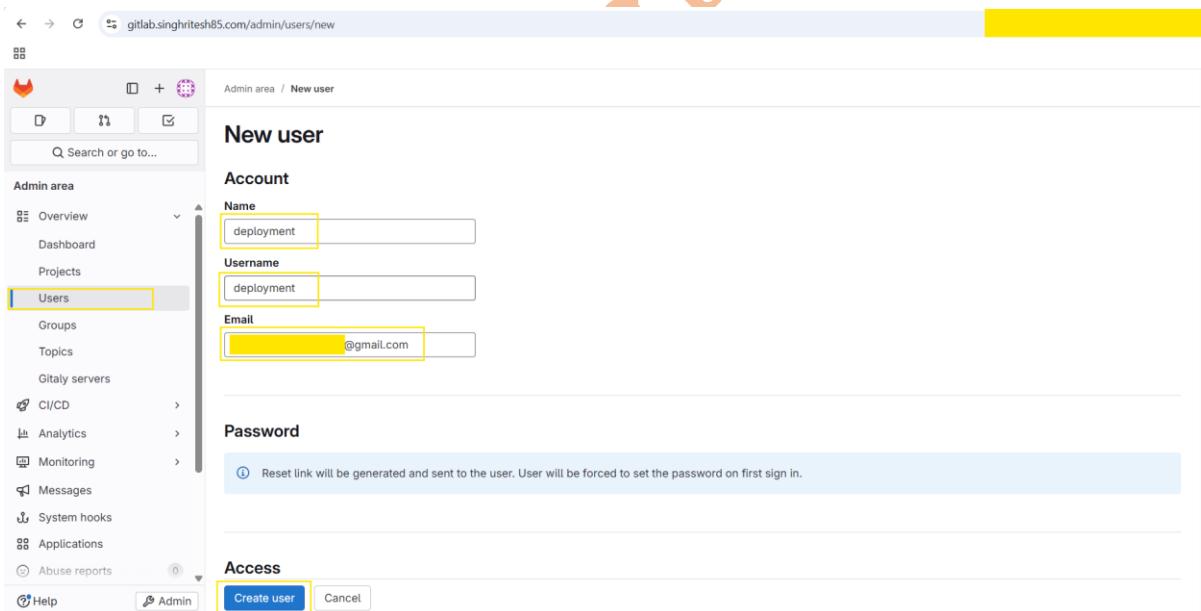
Now go to notification and update with your newly added email as shown in the screenshot attached below.



The screenshot shows the 'User Settings / Notifications' page on gitlab.singhritesh85.com. A success message 'Notification settings saved' is displayed. The 'Notifications' section is active, showing settings for 'Global notification email' (set to a redacted email address) and 'Global notification level' (set to 'Participate'). Below these are sections for 'Groups' (0) and 'Projects' (0), both indicating no members. The left sidebar shows various user settings like Account, Applications, Chat, etc., with 'Notifications' selected.

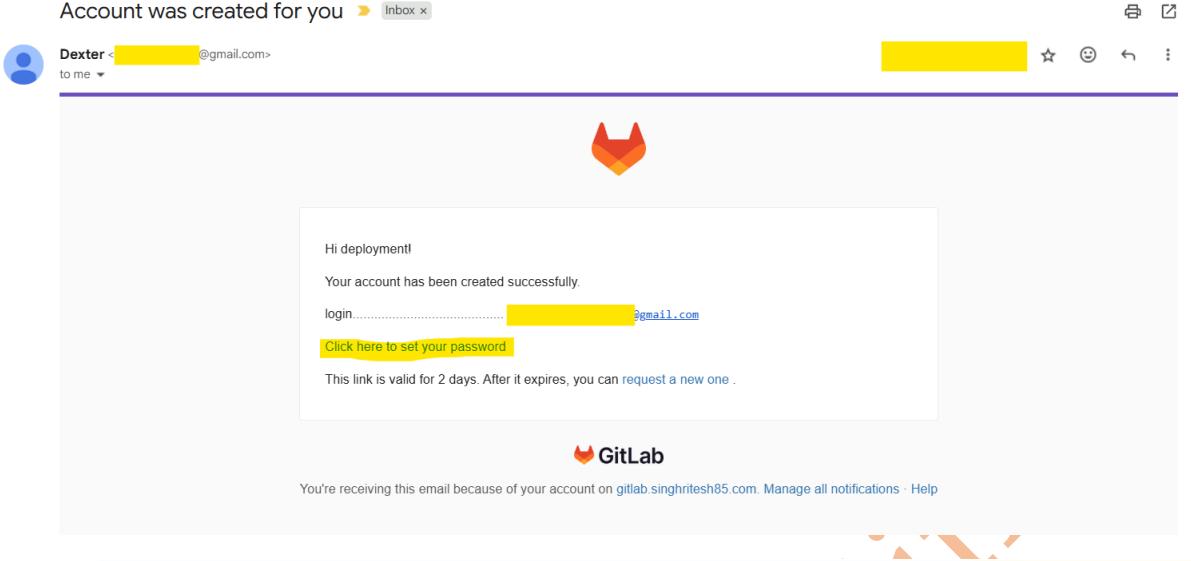
Now, the username, password and email of the Administrator was updated as shown in the screenshot attached below.

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.



The screenshot shows the 'Admin area / New user' page on gitlab.singhritesh85.com. The 'Users' option in the sidebar is selected. The 'New user' form is filled with the following data: Name: deployment, Username: deployment, and Email: deployment@gmail.com. The 'Create user' button is visible at the bottom. The left sidebar shows other admin areas like Overview, Dashboard, Projects, etc.

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



The screenshot shows an email from Dexter (dexter.singh85@gmail.com) in the inbox. The email subject is "Account was created for you". The body of the email contains a message from GitLab: "Hi deployment! Your account has been created successfully. login:@gmail.com Click here to set your password. This link is valid for 2 days. After it expires, you can request a new one." Below the email is a screenshot of a browser window showing the password reset page for gitlab.singhritesh85.com.

GitLab Enterprise Edition

Change your password

New password: Confirm new password:

Change your password

Didn't receive a confirmation email? Request a new one
Already have an account? Sign in

Explore Help About GitLab GitLab community forum

Now run the terraform script to create the GKE standard cluster (multi-zone cluster), gitlab-runner (on GCP VM Instance), SonarQube Server (on EC2 Instance) and Nexus Server (on EC2 Instance). For this terraform execution I kept the state file in GCP bucket. As a part of disaster recovery, I created the Lifecycle Policy to create the EBS backed AMI periodically 11:00 AM and 11:00 PM UTC (twice daily) for SonarQube and Nexus Server, snapshot schedule for gitlab-runner and GKE backup plan to take the backup daily twice a day (11:00 AM UTC and 11:00 PM UTC). For your reference I kept the Terraform script in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-BankApp-CICD-using-GitLab-GCP-and-AWS.git> at the path **terraform-gke-standard-cluster**.

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.gke.google_container_cluster.gke_cluster: Still creating... [09m50s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m00s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m10s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m20s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m30s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m40s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [10m50s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m00s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m10s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m20s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m30s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m40s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [11m50s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [12m00s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [12m10s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [12m20s elapsed]
module.gke.google_container_cluster.gke_cluster: Still creating... [12m30s elapsed]
module.gke.google_container_cluster.gke_cluster: Creation complete after 12m39s [id=projects/[REDACTED]/locations/us-central1/clusters/bankapp-gke-cluster]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Creating...
module.gke.google_backup_backup_plan.gke_backup: Creating...
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [00m10s elapsed]
module.gke.google_backup_backup_plan.gke_backup: Still creating... [00m10s elapsed]
module.gke.google_backup_backup_plan.gke_backup: Creation complete after 11s [id=projects/[REDACTED]/locations/us-central1/backupPlans/bankapp-gke-backup]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [00m20s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [00m30s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [00m40s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [00m50s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [01m00s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [01m10s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [01m20s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [01m30s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Still creating... [01m40s elapsed]
module.gke.google_container_node_pool.gke_linux_nodepool_1: Creation complete after 1m43s [id=projects/[REDACTED]/locations/us-central1/clusters/bankapp-gke-cluster/nodePools/bankapp-linux-nodepool-1]

Apply complete! Resources: 73 added, 0 changed, 0 destroyed.

```

Now, create the kubeconfig file on gitlab-runner using the command as shown in the screenshot attached below.



```

[gitlab-runner@bankapp-gitlab-runner ~]$ gcloud auth login
You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

You can run:
$ gcloud config set account `ACCOUNT`

to switch accounts if necessary.

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

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

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

```



```

[gitlab-runner@bankapp-gitlab-runner ~]$ gcloud container clusters get-credentials bankapp-gke-cluster --region us-central1 --project [REDACTED]
Fetching cluster endpoint and auth data.
kubeconfig entry generated for bankapp-gke-cluster.

```

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@bankapp-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.
```

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.

The screenshot shows the GCP Cloud DNS interface. At the top, there's a navigation bar with 'Record sets', 'Add standard', 'Add with routing policy', 'Delete record sets', and 'Refresh'. Below it is a table of existing DNS records:

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

Below the table is a 'Create record set' dialog:

- DNS name:** sonarqube.singhritesh85.com. (highlighted with a yellow box)
- Resource record type:** CNAME (highlighted with a yellow box)
- TTL *:** 5
- TTL unit:** minutes
- Canonical name:** (highlighted with a yellow box)
 - Canonical name 1 ***: (highlighted with a yellow box)
 - Example: server-1.example.com.
- Buttons:** '+ Add item' (disabled), 'Create' (highlighted with a yellow box), and 'Cancel'

Create record set

DNS name ?

Resource record type ▼ ?

TTL * ?

TTL unit ▼ ?

Canonical name ?

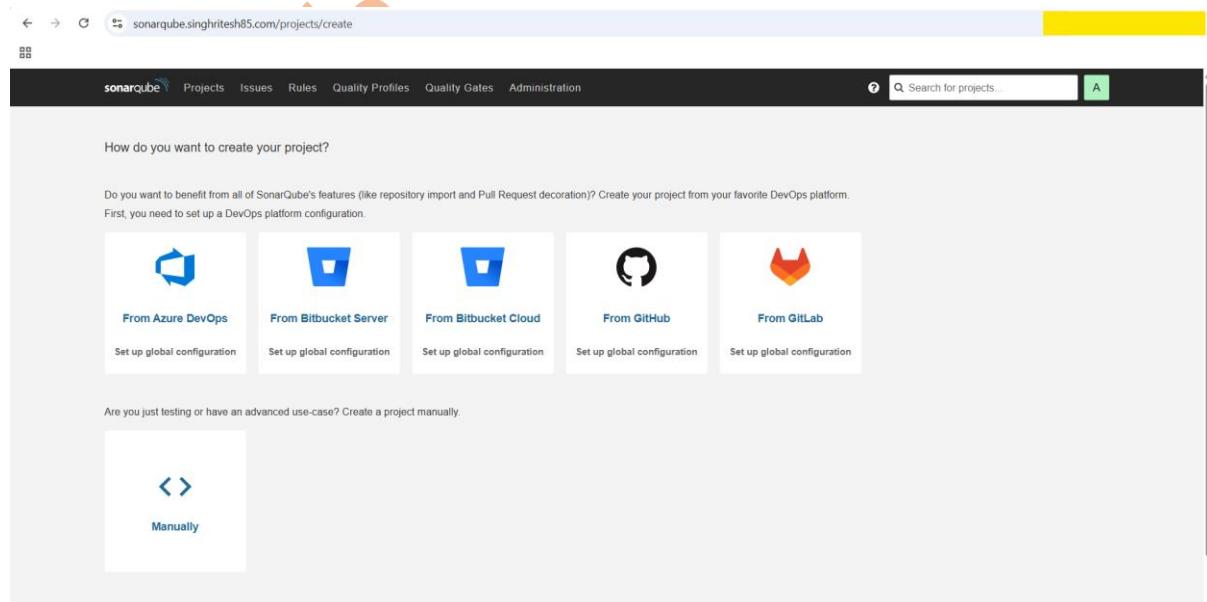
Canonical name 1 *

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.



The screenshot shows the Sonatype Nexus Repository OSS 3.68.1-02 administration interface. The left sidebar is titled 'Administration' and contains sections for 'Repository' (Repositories, Blob Stores, Proprietary Repositories, Content Selectors, Cleanup Policies, Routing Rules) and 'Security' (Privileges, Roles, Users, Anonymous Access, LDAP, Realms). The main content area is titled 'Repository' and includes tabs for Blob Stores, Cleanup Policies, Proprietary Repositories, Repositories, and Routing Rules.

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.

The screenshot shows the SonarQube account security page. It displays a 'Tokens' section where a new token named 'SonarQube' has been generated. The token details are: Name: SonarQube, Type: User, Project: None, Last use: Never, Created: July 11, 2025, and Expiration: Never. A 'Copy' button is visible next to the token ID.

Name	Type	Project	Last use	Created	Expiration
SonarQube	User		Never	July 11, 2025	Never

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.

The screenshot shows the GitLab Admin area dashboard. On the left, there's a sidebar with 'Admin area' navigation. Under 'CI/CD', 'Runners' is highlighted. The main content area is titled 'Instance overview' and includes sections for 'Projects', 'Total Users', 'Groups', 'Statistics', 'Features', and 'Components'. The 'Components' section shows various GitLab services with their status and versions.

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=18.1.1
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: kubernetes, docker-autoscaler, instance, custom, shell, virtualbox, docker+machine, ssh, parallels, docker, docker-windows: 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=18.1.1
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.

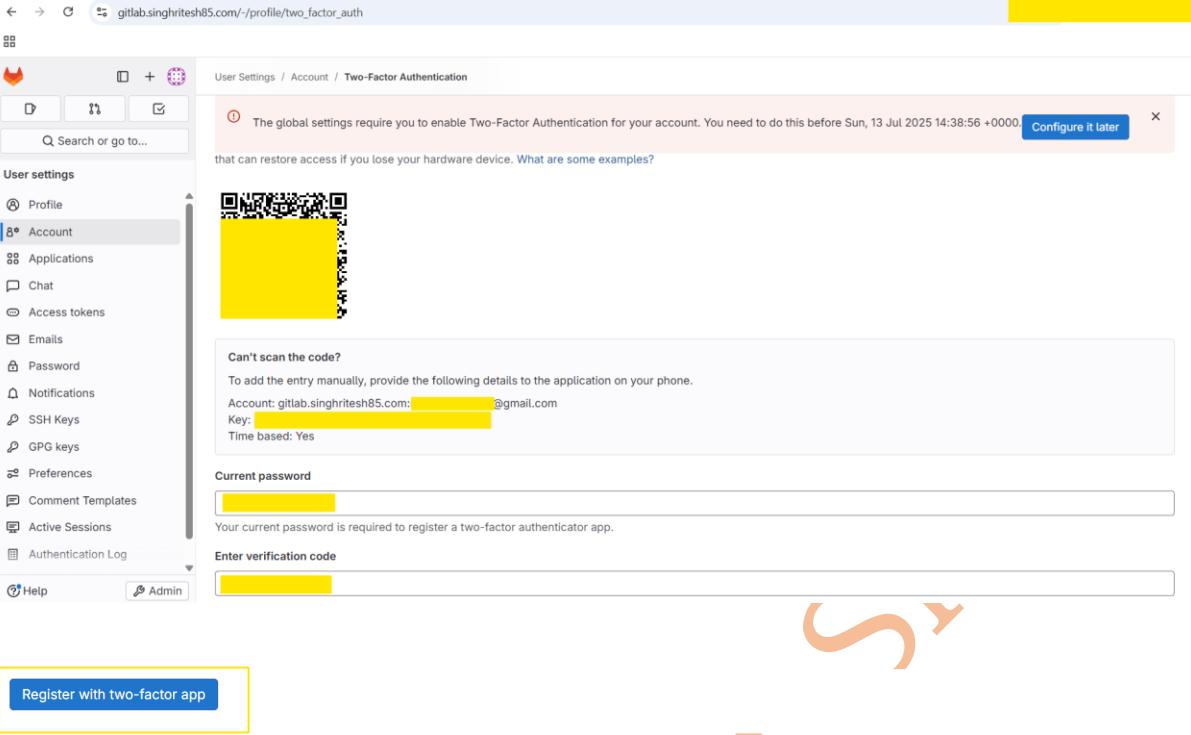
The screenshot shows the 'Runners' configuration page. At the top, it says 'All 1' and has filters for 'Instance', 'Group', and 'Project'. Below is a search bar and a toolbar with 'Create instance runner' and other options. The main table lists one runner: '#1 [REDACTED] Instance' (Status: Online, Idle), owned by 'Administrator'. There are edit, pause, and delete icons for this runner.

Enable two-factor Authentication in GitLab for All the users

I had enabled **Two-Factor authentication** for All the users as shown in the screenshot attached below. This configuration will forcefully enable 2FA (two-factor authentication for all the users). If you need to enable 2FA for Administrator then enable the option **Enforce Two-Factor authentication for administrator users** Go to Admin area > Settings > General > Sign-in restrictions check the two options and then save changes.

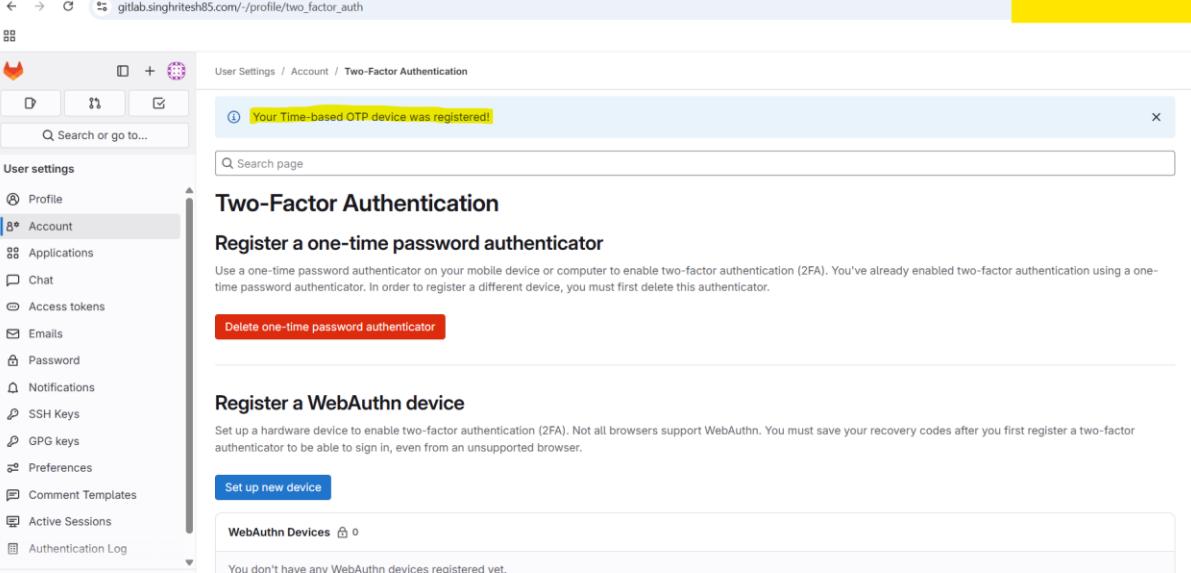
Now it will ask you to configure 2FA/MFA as shown in the screenshot attached below.

Now you can configure 2FA/MFA from your device, in the current scenario I used my mobile phone and installed octa for 2FA and scan the image from octa App installed in my mobile and provided the generated 6 digits number from the octa App and the current password of the user then clicked on register with two-factor app as shown in the screenshot attached below finally 2FA/MFA had been configured as shown in the screenshot attached below.



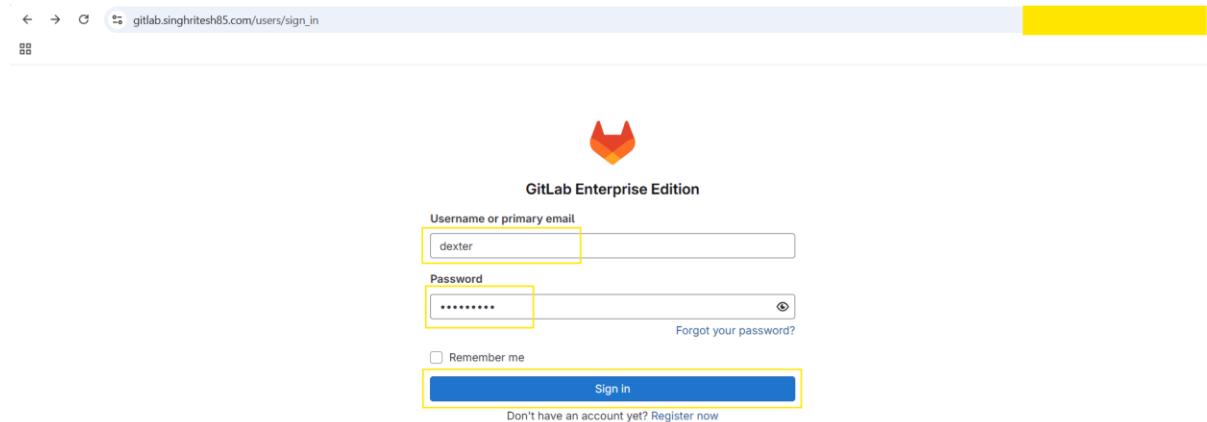
The screenshot shows the 'Two-Factor Authentication' page in GitLab. A QR code is displayed for scanning. Below it, there's a section for manual entry of details if scanning fails, including fields for Account (gitlab.singhritesh85.com: [REDACTED]@gmail.com), Key, and Time-based (Yes). There are also fields for 'Current password' and 'Enter verification code'. A large orange arrow points from the bottom left towards the QR code.

Register with two-factor app



The screenshot shows the confirmation page after registering a device. It displays a message: 'Your Time-based OTP device was registered!'. Below this, there's a section for 'Two-Factor Authentication' with a heading 'Register a one-time password authenticator'. It includes a red button 'Delete one-time password authenticator'. Further down, there's a section for 'Register a WebAuthn device' with a blue button 'Set up new device'. A small orange arrow points from the bottom left towards the registration message.

Next time when the user will login then user need to provide the username, password and then the OTP generated in octa App installed on users mobile/device.



The screenshot shows the GitLab Enterprise Edition login page. At the top, there is a navigation bar with icons for back, forward, and search, followed by the URL "gitlab.singhritesh85.com/users/sign_in". Below the URL is a yellow rectangular box. The main content area features the GitLab logo (a red and orange fox head) and the text "GitLab Enterprise Edition". It has two input fields: "Username or primary email" containing "dexter" and "Password" containing "*****". To the right of the password field is a "Forgot your password?" link. Below the fields is a "Remember me" checkbox and a blue "Sign in" button. At the bottom, there is a link "Don't have an account yet? Register now".



The screenshot shows the GitLab Enterprise Edition two-factor authentication page. It features the GitLab logo and the text "GitLab Enterprise Edition". There is a single input field labeled "Enter verification code" with a yellow background. Below the field is a note: "Enter the code from your two-factor authenticator app. If you've lost your device, you can enter one of your recovery codes." At the bottom is a blue "Verify code" button.

Finally, user was logged-in into their GitLab Account as shown in the screenshot attached below.

A large, semi-transparent watermark reading "Ritesh" diagonally across the page.

The screenshot shows the GitLab homepage at gitlab.singhritesh85.com. The left sidebar is titled "Your work" and includes links for Projects, Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, and Activity. The main content area is titled "Welcome to GitLab, Administrator!" and contains several introductory cards: "Unlock more features with GitLab Ultimate" (with a "Start free trial" button), "Create a project" (describing projects as storage for code, issues, and wikis), "Create a group" (describing groups for organizing projects and people), and "Configure GitLab" (describing instance configuration). At the bottom of the sidebar are "Help" and "Admin" buttons.

In the same way when you login with deployment user it will ask you to configure 2FA/MFA and you can configure the 2FA in the same as discussed above. The final screenshot is as shown below.

The screenshot shows the "User Settings / Account" page at gitlab.singhritesh85.com/-/profile/two_factor_auth. The left sidebar under "User settings" has "Account" selected. The main content area is titled "Two-factor Authentication Recovery codes" and contains instructions: "Please copy, download, or print your recovery codes before proceeding." Below this is a grid of 20 recovery codes, each consisting of a letter and a number (e.g., a1, b2, c3, d4, e5, f6, g7, h8, i9, j0). At the bottom are buttons for "Copy codes", "Download codes", "Print codes", and "Proceed".

Using the **deployment** user, I had created the three GitLab Private Repos named as **helm-repo-for-bitnami-GKE**, **helm-repo-for-GKE** and **Bank-App** as shown in the screenshot shot attached below.



gitlab.singhritesh85.com/projects/new#blank_project

Your work / Projects / New project / Create blank project

Project name
helm-repo-for-bitnami-GKE
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
http://gitlab.singhritesh85.com/deployment/

Project slug
helm-repo-for-bitnami-gke

Visibility Level Private
 Internal
 Public

Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
 Internal
The project can be accessed by any logged in user except external users.
 Public
The project can be accessed without any authentication.

Project Configuration

- Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
- Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. Learn more.
- Enable Secret Detection
Scan your code for secrets and credentials to prevent unauthorized access. Learn more.

Create project Cancel

gitlab.singhritesh85.com/projects/new#blank_project

Your work / Projects / New project / Create blank project

Project name
helm-repo-for-GKE
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
http://gitlab.singhritesh85.com/deployment/

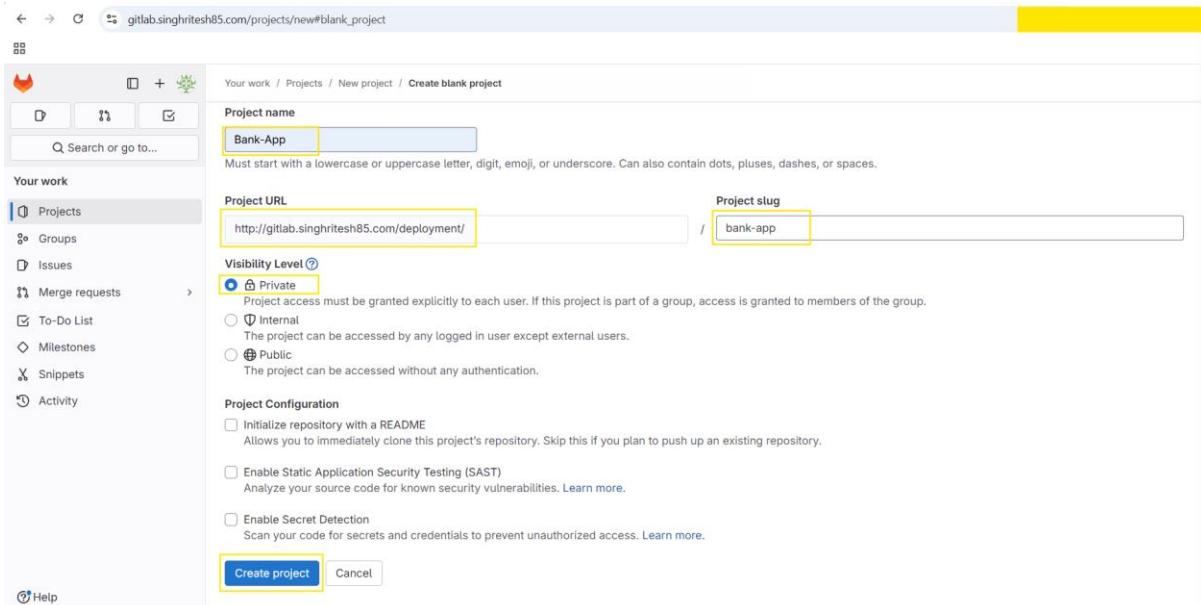
Project slug
helm-repo-for-gke

Visibility Level Private
 Internal
The project can be accessed by any logged in user except external users.
 Public
The project can be accessed without any authentication.

Project Configuration

- Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
- Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. Learn more.
- Enable Secret Detection
Scan your code for secrets and credentials to prevent unauthorized access. Learn more.

Create project Cancel



Using the **deployment** user, I created the access token in each of the GitLab Repo as shown in the screenshot attached below. Go to the GitLab Repo > Settings > Access Tokens > Add new token as shown in the screenshot attached below.

Project access tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API. You can also use project access tokens with Git to authenticate over HTTP(S). Learn more.

Token name	Description	Scopes	Created	Last Used	Last Used IPs	Expires	Role	Action
This project has no active access tokens.								

Token name	Description	Scopes	Created	Last Used	Expired	Role
An error occurred while fetching the tokens.						

Active project access tokens 0

Add a project access token

Token name (highlighted with a yellow box)

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Token description

Expiration date (highlighted with a yellow box)

An administrator has set the maximum expiration date to 2026-07-11. Learn more.

Select a role (highlighted with a yellow box)

Project access tokens

Maintainer

Select scopes

- api
- read_api
- create_runner
- manage_runner
- k8s_proxy
- self_rotate
- read_repository
- write_repository
- ai.features

Create project access token Cancel

In the above screenshot I created the access token for the project **Bank-App** and for the other two projects **helm-repo-for-bitnami-GKE** and **helm-repo-for-GKE** you can create in the similar manner.

Your new project access token has been created.

Project access tokens

Your new project access token

thimpu1

Active project access tokens (1)

Token name	Description	Scopes	Created	Last Used	Last Used IPs	Expires	Role	Action
thimpu1		self_rotate, read_repository, write_repository	Jul 11, 2025	Never	-	in 11 months	Maintainer	

Inactive project access tokens (0)

I cloned the source code and helm charts from GitHub Public Repo (<https://github.com/singhritesh85/Bank-App.git>, <https://github.com/singhritesh85/helm-repo-for-GKE.git> and <https://github.com/singhritesh85/helm-repo-for-bitnami-GKE.git>) and then pushed the code to the GitLab private Repo as shown in the screenshot attached below.

```
[root@... ~]# mkdir banku
[root@... ~]# cd banku/
[root@... banku]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /root/banku/.git/
[root@... banku]# git remote add origin https://deployment:...@gitlab.singhritesh85.com/deployment/bank-app.git
[root@... banku]# git checkout -b main
Switched to a new branch 'main'
```

```
[root@yellow banku]# git config --local user.name "deployment"
[root@yellow banku]# git config --local user.email "yellow@yellow>.com"
[root@yellow banku]# git add .
[root@yellow banku]# git commit -m "dexter"
```

```
[root@yellow banku]# git push origin main
Enumerating objects: 42, done.
Counting objects: 100% (42/42), done.
Compressing objects: 100% (29/29), done.
Writing objects: 100% (42/42), 18.11 KiB | 3.02 MiB/s, done.
Total 42 (delta 2), reused 0 (delta 0), pack-reused 0
To https://gitlab.singhritesh85.com/deployment/bank-app.git
 * [new branch]      main -> main
[root@yellow banku]# git remote -v
origin  https://deployment:yellow">@gitlab.singhritesh85.com/deployment/bank-app.git (fetch)
origin  https://deployment:yellow">@gitlab.singhritesh85.com/deployment/bank-app.git (push)
```

[root@**yellow** ~]# mkdir bolo

```
[root@yellow ~]# cd bolo/
[root@yellow bolo]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /root/bolo/.git/
[root@yellow bolo]# git remote add origin https://deployment:yellow">@gitlab.singhritesh85.com/deployment/helm-repo-for-bitnami-gke.git
[root@yellow bolo]# cp -r ~/helm-repo-for-bitnami/* .
origin https://deployment:yellow">@gitlab.singhritesh85.com/deployment/helm-repo-for-bitnami-gke.git (fetch)
origin https://deployment:yellow">@gitlab.singhritesh85.com/deployment/helm-repo-for-bitnami-gke.git (push)
[root@bankapp-gitlab-runner bolo]# git checkout -b main
Switched to a new branch 'main'
[root@bankapp-gitlab-runner bolo]# git config --local user.name "deployment"
[root@bankapp-gitlab-runner bolo]# git config --local user.email "yellow">@yellow>.com"
[root@bankapp-gitlab-runner bolo]# git add .
[root@bankapp-gitlab-runner bolo]# git commit -m "medera"
```

[root@**yellow** bolo]# git push origin main

```
Enumerating objects: 138, done.
Counting objects: 100% (138/138), done.
Compressing objects: 100% (136/136), done.
Writing objects: 100% (138/138), 518.56 KiB | 7.30 MiB/s, done.
Total 138 (delta 68), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (68/68), done.
To https://gitlab.singhritesh85.com/deployment/helm-repo-for-bitnami-gke.git
 * [new branch]      main -> main
```

[root@**yellow** ~]# mkdir mangus

[root@**yellow** ~]# cp -r helm-repo-for-GKE/* mangus/

```
[root@yellow mangus]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /root/mangus/.git/
```

```
[root@yellow mangus]# git config --local user.name "deployment"
[root@yellow mangus]# git config --local user.email "yellow">@yellow>.com"
```

```
[root@mangus ~]# git remote add origin https://deployment:@gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git
[mangus]# git remote -v
origin https://deployment:@gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git (fetch)
origin https://deployment:@gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git (push)

[mangus]# git add .
[mangus]# git commit -m "therema"

[mangus]# git push origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Compressing objects: 100% (15/15), done.
Writing objects: 100% (17/17), 9.15 KiB | 4.58 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git
 * [new branch]      main -> main
```

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.

```
[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 = 2025-01-01T00:00:00Z
  token_expires_at = 2001-01-01T00:00:00Z
  executor = "shell"
  [runners.cache]
    MaxUploadedArchiveSize = 0
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

Now create the GitLab CI/CD Pipeline. Go to the file GitLab Repo **helm-repo-for-bitnami-GKE** > Build > Pipeline editor > Configure Pipeline and create the .gitlab-ci.yml file as shown in the screenshot attached below.

```

default:
tags:
- gitlab-runner-1

stages:
- deployment_of_mysql_pods_to_gke_cluster

deployment_to_gke:
stage: deployment_of_mysql_pods_to_gke_cluster
script:
- helm upgrade --install mysql bitnami/mysql/ -f bitnami/mysql/values.yaml --create-namespace --namespace mysql --set secondary.replicaCount=1
- kubectl get pods -n mysql
when: manual
# only:
# - main # Or your desired branch

```

Now go to the Pipelines and run it manually as shown in the screenshot attached below.

Status	Pipeline	Created by	Stages	Actions
Skipped	Update .gitlab-ci.yml file #2 # main			<button>deployment_to_gke</button>

After successful execution of the pipeline pods were created as shown in the screenshot attached below.

```

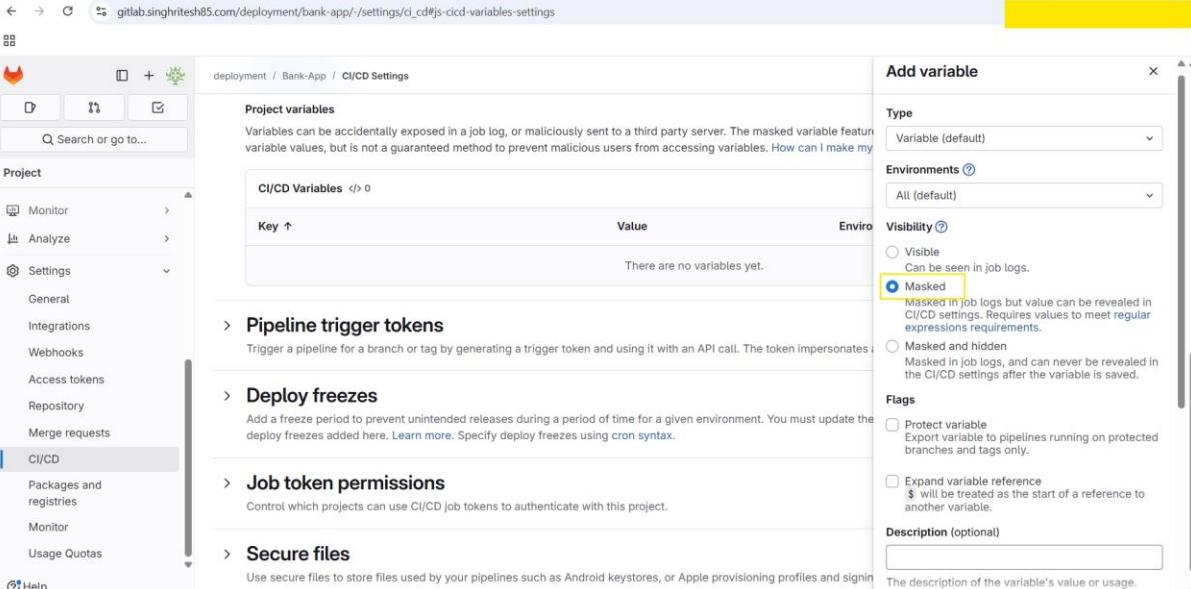
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n mysql --watch
NAME        READY   STATUS    RESTARTS   AGE
mysql-primary-0 1/1     Running   0          83s
mysql-secondary-0 1/1     Running   0          83s
^C[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pvc -n mysql --watch
NAME           STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
data-mysql-primary-0 Bound    pvc-[REDACTED]                                1Gi        RWO          standard-rwo   <unset>          92s
data-mysql-secondary-0 Bound    pvc-[REDACTED]                                1Gi        RWO          standard-rwo   <unset>          92s

```

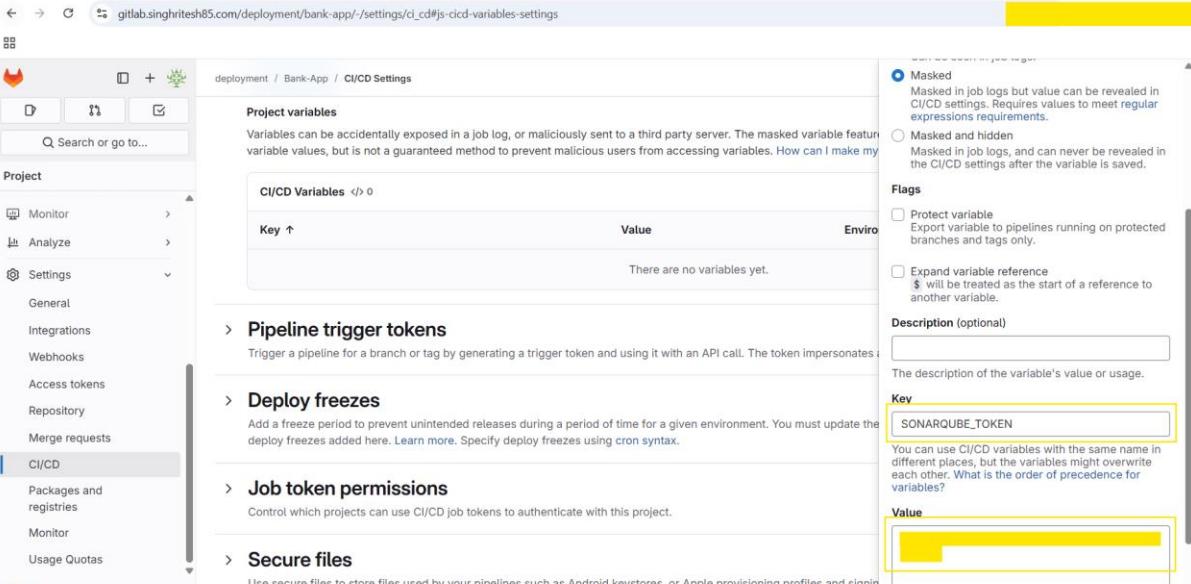
Now to the GitLab Repo Bank-App > Build > Pipeline editor > Configure pipeline to create the GitLab CI/CD Pipeline as shown in the screenshot attached below. For your reference these two .gitlab-ci.yml file I had provided in the GitHub Repo <https://github.com/singhritesh85/Bank-App-GKE.git> and <https://github.com/singhritesh85/helm-repo-for-bitnami-GKE.git>.

Hardcoding username and password in gitlab ci/cd pipeline are not a good idea so I added the pipeline variables in each of the GitLab Project which I utilized during the CI/CD Pipeline as shown in the screenshot attached below. Go to each of the GitLab Projects/Repository > Settings > CI/CD > Variables and start adding the pipeline variables.

One point I want to clarify here that deployment user is the owner of the GitLab Project/Repository as that user had created the project otherwise you can make the user as maintainer and proceed to the deployment.



The screenshot shows the GitLab interface for managing CI/CD variables. The left sidebar is visible with various project management options like Monitor, Analyze, Settings, and CI/CD. The main area displays the 'CI/CD Settings' page for a project named 'Bank-App'. On the right, a modal window titled 'Add variable' is open, allowing the creation of a new CI/CD variable. The 'Type' dropdown is set to 'Variable (default)'. Under the 'Environments' section, 'All (default)' is selected. The 'Visibility' section has two options: 'Visible' (unchecked) and 'Masked' (checked). The 'Masked' option is described as being visible in job logs but requiring regular expression requirements. Other visibility options include 'Masked and hidden', which is described as being masked in job logs and never revealed in CI/CD settings. The 'Flags' section contains three checkboxes: 'Protect variable', 'Expand variable reference', and 'Description (optional)'. A text input field for 'Description (optional)' is present, and a note below it states 'The description of the variable's value or usage.'



This screenshot shows the same GitLab interface and CI/CD Settings page as the first one. The 'Add variable' dialog is still open, but now the 'Key' field contains the value 'SONARQUBE_TOKEN'. The 'Value' field is also populated with some placeholder text. The rest of the dialog fields and descriptions remain the same as in the first screenshot.

gitlab.singhritesh85.com/deployment/bank-app/-/settings/ci_cd#js-cicd-variables-settings

Project variables

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature allows you to mask sensitive values, but is not a guaranteed method to prevent malicious users from accessing variables. How can I make my variables more secure?

Key ↑	Value	Enviro
SONARQUBE_TOKEN	*****	All (default)
	(Masked)	

Pipeline trigger tokens

Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates the user who triggered the pipeline.

Deploy freezes

Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the deploy freezes added here. Learn more. Specify deploy freezes using cron syntax.

Job token permissions

Control which projects can use CI/CD job tokens to authenticate with this project.

Secure files

Use secure files to store files used by your pipelines such as Android keystores, or Apple provisioning profiles and signing certificates.

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

Masked and hidden
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

Flags

Protect variable
Export variable to pipelines running on protected branches and 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

NEXUS_USER

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

[REDACTED]

Variable value will be evaluated as raw string.

Add variable

Type: Variable (default)

Environments: All (default)

Visibility: Visible
Can be seen in job logs.

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

Masked and hidden
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

Flags:

Protect variable
Export variable to pipelines running on protected branches and 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 separate instances of the GitLab CI/CD Variables Settings page for a project named "Bank-App".

Top Instance:

- Variables:** NEXUS_USER (Value: masked), SONARQUBE_TOKEN (Value: masked). Both are marked as "Masked".
- Flags:** "Visible" is selected, while "Masked" is highlighted with a yellow box.
- Description (optional):** "The description of the variable's value or usage."

Bottom Instance:

- Variables:** NEXUS_USER (Value: masked), SONARQUBE_TOKEN (Value: masked). Both are marked as "Masked".
- Flags:** "Visible" is selected, while "Masked" is highlighted with a yellow box.
- Description (optional):** "The description of the variable's value or usage."
- Key:** "NEXUS_PASSWORD" is entered in the Key field.
- Value:** A yellow box highlights the Value field, which contains a placeholder value.
- Buttons:** "Add variable" and "Cancel".

The screenshot shows the GitLab CI/CD Variables settings page for a project named "Bank-App". The page displays three variables:

Key	Value	Environments	Actions
NEXUS_PASSWORD	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
NEXUS_USER	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
SONARQUBE_TOKEN	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Below the table, there are sections for "Pipeline trigger tokens" and "Deploy freezes". A modal window titled "Add variable" is open on the right side, showing the configuration for a new variable. The "Type" dropdown is set to "Variable (default)". The "Visibility" section has a radio button selected for "Masked", which is highlighted with a yellow box. Other options include "Visible" and "Masked and hidden". The "Flags" section contains checkboxes for "Protect variable" and "Expand variable reference", both of which are unselected. A "Description (optional)" input field is present at the bottom of the modal.

The screenshot shows two consecutive screenshots of the GitLab CI/CD Variables settings page for a project named 'Bank-App'.

Screenshot 1 (Top): A modal window is open for adding a new CI/CD Variable. The variable key is 'HELM_REPO_FOR_GKE_ACCESS_TOKEN' and the value is 'HELM_REPO_FOR_GKE_ACCESS_TOKEN'. The 'Add variable' button is highlighted in yellow.

Screenshot 2 (Bottom): The modal has been closed, and the variable is now listed in the main table. The table shows four variables:

Key	Value	Environments	Actions
NEXUS_PASSWORD	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
NEXUS_USER	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
SONARQUBE_TOKEN	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
HELM_REPO_FOR_GKE_ACCESS_TOKEN	*****	All (default)	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Project Variables Section:

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?

Pipeline trigger tokens and Deploy freezes sections:

Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a user's project access and permissions. Learn more. Specify deploy freezes using cron syntax.

The screenshot shows two overlapping dialog boxes from the GitLab interface, both titled "Add variable".

Top Dialog (Visible):

- Type:** Variable (default)
- Environments:** All (default)
- Visibility:** Masked (selected)
- Description (optional):** The description of the variable's value or usage.

Bottom Dialog (Visible):

- 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.
- Key:** A yellow box highlights the input field for the variable key.
- Value:** A yellow box highlights the input field for the variable value.

Common UI Elements:

- Project Sidebar:** Includes options like Monitor, Analyze, Settings, CI/CD, Packages and registries, Monitor, Usage Quotas, and Help.
- Page Header:** gitlab.singhritesh85.com/deployment/helm-repo-for-bitnami-gke/-/settings/ci_cd#js-cicd-variables-settings
- Form Fields:** Minimum role to use pipeline variables, Access protected resources in merge request pipelines, Project variables, and a table for CI/CD Variables.
- Buttons:** Save changes, Add variable, and Cancel.

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 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 protects variable values, but is not a guaranteed method to prevent malicious users from accessing variables. How can I make my variables more secure?

Key ↑	Value	Environments	Actions
MYSQL_DATABASE	*****	All (default)	
MYSQL_ROOT_PASSWORD	*****	All (default)	
	MASKED		

CI/CD Variables

Reveal values [Add variable](#)

Pipeline trigger tokens

Trigger a pipeline for a branch or tag by generating a trigger token and using it with an API call. The token impersonates a user's project access and permissions. [Learn more](#).

Deploy freezes

Add a freeze period to prevent unintended releases during a period of time for a given environment. You must update the deployment jobs in `.gitlab-ci.yml` according to the deploy freezes added here. [Learn more](#). Specify deploy freezes using cron syntax.

Job token permissions

Control which projects can use CI/CD job tokens to authenticate with this project.

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n bankapp --watch
NAME           READY   STATUS    RESTARTS   AGE
bankapp-folo-  1/1     Running   0          12m
^C[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get svc -n bankapp
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
bankapp-folo  ClusterIP  172.16.1.100 <none>        80/TCP       12m
```

After successful execution of gitlab ci/cd pipeline the bankapp pod had been created as shown in the screenshot attached below.

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get pods -n bankapp --watch
NAME           READY   STATUS    RESTARTS   AGE
bankapp-folo-  1/1     Running   0          12m
^C[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl get svc -n bankapp
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
bankapp-folo  ClusterIP  172.16.1.100 <none>        80/TCP       12m
```

Then I had created the Kubernetes Secrets and Ingress Rule as shown in the screenshot attached below.

kubectl create secret tls ingress-tls --key server.key --cert 2XXXXXXXXXXXXXX.crt --namespace bankapp

Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/> || GitHub: - <https://github.com/singhrithesh85>

```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl create secret tls ingress-tls --key server.key --cert 2d972e3ffe62a0f8.crt --namespace bankapp  
secret/ingress-tls created  
  
[gitlab-runner@bankapp-gitlab-runner ~]$ cat ingress-rule.yaml  
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
  name: dexter-ingress  
  namespace: bankapp  
  annotations:  
    kubernetes.io/ingress.class: "gce" # Specify the Ingress class  
    kubernetes.io/ingress.allow-http: "false"  
  #   cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'  
spec:  
  ingressClassName: "gce"  
  tls:  
    - hosts:  
      - bankapp.singhritesh85.com  
      secretName: ingress-tls  
    rules:  
      - host: "bankapp.singhritesh85.com"  
        http:  
          paths:  
            - path: /  
              pathType: Prefix  
              backend:  
                service:  
                  name: bankapp-folo  
                  port:  
                    number: 80
```

Ritesh Kumar Singh

```

cat ingress-rule.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dexter-ingress
  namespace: bankapp
annotations:
  kubernetes.io/ingress.class: "gce" # Specify the Ingress class
  kubernetes.io/ingress.allow-http: "false"
#  cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'
spec:
  ingressClassName: "gce"
  tls:
    - hosts:
        - bankapp.singhritesh85.com
      secretName: ingress-tls
  rules:
    - host: "bankapp.singhritesh85.com"
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: bankapp-folo
              port:
                number: 80

```

NAMESPACE	NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
bankapp	dexter-ingress	gce	bankapp.singhritesh85.com	34.12.12.12	80, 443	3m53s

I did the entry for this IP Address with HOSTS in GCP Cloud DNS to create the Record Set of Type-A as shown in the screenshot attached below.

[Create record set](#)

DNS name bankapp .singhritesh85.com. [?](#)

Resource record type A [?](#)

TTL * 5 [?](#)

TTL unit minutes [?](#)

IPv4 Address [?](#)

IPv4 Address 1 * 34. [Select](#)

Example: 192.0.2.91

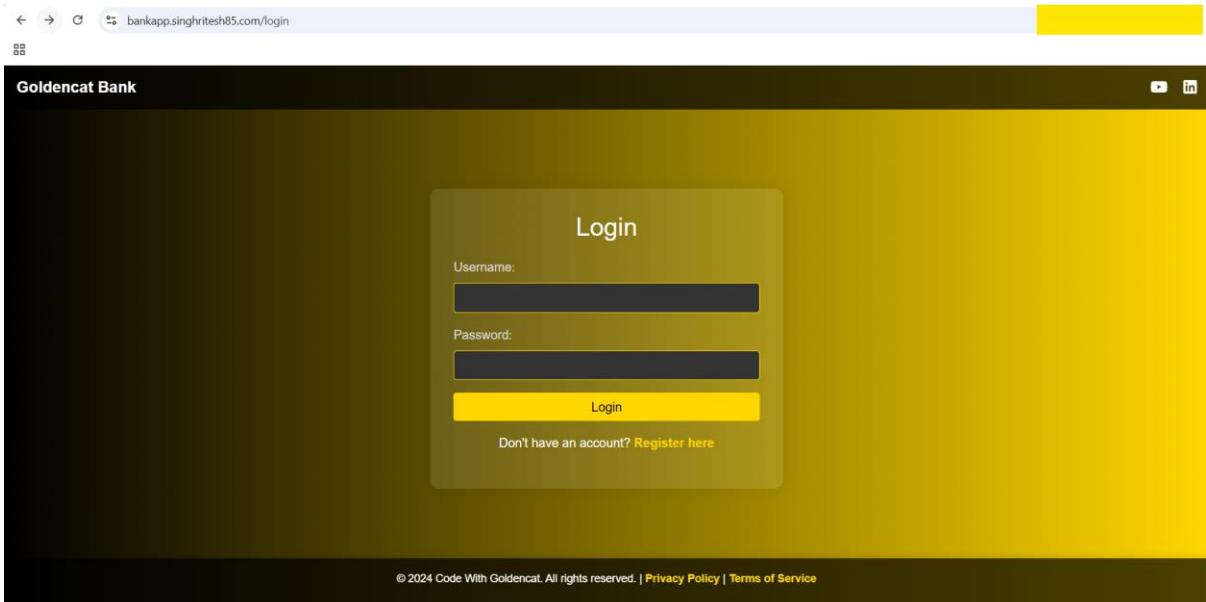
[+ Add item](#)

Create [Cancel](#)

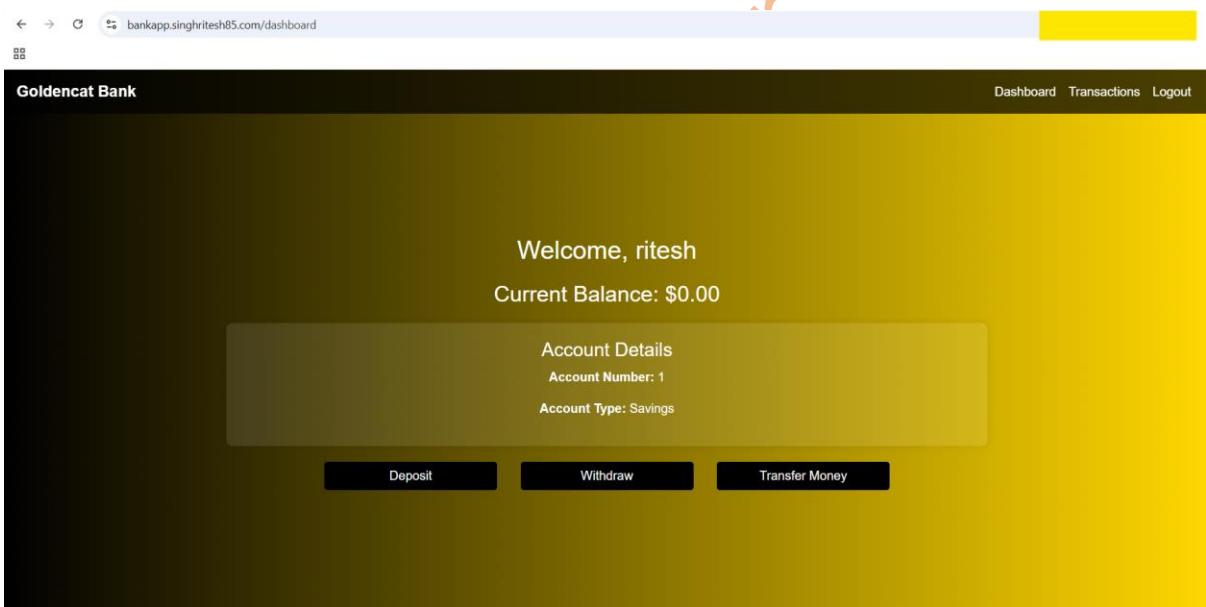
Record sets

<input type="checkbox"/>	DNS name	Type	TTL (seconds)	Record data	?	☰
<input type="checkbox"/>	3.singhritesh85.com.	CNAME	300	3.singhritesh85.com.	▼	
<input type="checkbox"/>	bankapp.singhritesh85.com.	A	300	bankapp.singhritesh85.com.	▼	
<input type="checkbox"/>	gitlab.singhritesh85.com.	CNAME	300	gitlab.singhritesh85.com.	▼	
<input type="checkbox"/>	nexus.singhritesh85.com.	CNAME	300	nexus.singhritesh85.com.	▼	
<input type="checkbox"/>	singhritesh85.com.	NS	21600	singhritesh85.com.	▼	
<input type="checkbox"/>	singhritesh85.com.	SOA	21600	singhritesh85.com.	▼	
<input type="checkbox"/>	sonarqube.singhritesh85.com.	CNAME	300	sonarqube.singhritesh85.com.	▼	

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



Then registered with a user and checked its entry in the mysql pod and found the same as shown in the screenshot attached below.



```
[gitlab-runner@bankapp-gitlab-runner ~]$ kubectl exec -it mysql-primary-0 -n mysql -- bash
Defaulted container "mysql" out of: mysql, preserve-logs-symlinks (init)
I have no name!@mysql-primary-0:/$ mysql -h localhost -u root --password
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
I have no name!@mysql-primary-0:/$ mysql -h localhost -u root --password
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 849
Server version: 8.4.0 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'dat
bases' at line 1
mysql> show databases;
```

```

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

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

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

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

mysql>

```

For your reference I am providing here the .gitlab-ci.yml file for deployment of mysql pods and bankapp respectively.

These two .gitlab-ci.yml files are also present in the GitHub Repo
<https://github.com/singhritesh85/helm-repo-for-bitnami-GKE.git> and
<https://github.com/singhritesh85/Bank-App-GKE.git>.

```
default:  
tags:  
- gitlab-runner-1  
  
stages:  
- deployment_of_mysql_pods_to_gke_cluster  
  
deployment_to_gke:  
stage: deployment_of_mysql_pods_to_gke_cluster  
script:  
- helm upgrade --install mysql bitnami/mysql/ -f bitnami/mysql/values.yaml --create-namespace --  
namespace mysql --set  
secondary.replicaCount=1,primary.persistence.enabled=true,primary.persistence.size=1Gi,architectu  
re=replication,secondary.persistence.enabled=true,secondary.persistence.size=1Gi,global.storageCla  
ss=standard-  
rwo,primary.service.type=ClusterIP,auth.rootPassword=$MYSQL_ROOT_PASSWORD,auth.database=$  
MYSQL_DATABASE  
- kubectl get pods -n mysql  
when: manual  
  
# only:  
# - main # Or your desired branch
```

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

```
stages:
- build_and_sonarqube_analysis
- nexus_artifact_upload
- docker_image_build
- deployment_to_gke_cluster
```

```
build_job:
stage: build_and_sonarqube_analysis
before_script:
- export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.15.0.6-2.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 -Dsonar.qualitygate.wait=true -Dsonar.qualitygate.timeout=600 -Dsonar.projectKey=bankapp -Dsonar.projectName='bankapp' -Dsonar.host.url=https://sonarqube.singhritesh85.com -Dsonar.login=$SONARQUBE_TOKEN
```

```
artifacts:
```

```
paths:
- target/*.jar
```

```
upload_job:
```

```
stage: nexus_artifact_upload
script:
- echo "Uploading artifact to Nexus..."
- curl -v -u $NEXUS_USER:$NEXUS_PASSWORD --upload-file target/*.jar https://nexus.singhritesh85.com/repository/maven-snapshot/
```

```
dependencies:
```

- build_job

docker_image:

stage: docker_image_build

before_script:

- export JAVA_HOME="/usr/lib/jvm/java-17-openjdk-17.0.15.0.6-2.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 us-central1-docker.pkg.dev/wise-trainer-244916/bankapp-gcr-dev/dexter:latest -f Dockerfile-Project-1 .

- docker push us-central1-docker.pkg.dev/wise-trainer-244916/bankapp-gcr-dev/dexter:latest

dependencies:

- upload_job

deployment_to_gke:

stage: deployment_to_gke_cluster

script:

- git clone

[https://deployment:\\$HELM_REPO_FOR_GKE_ACCESS_TOKEN@gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git](https://deployment:$HELM_REPO_FOR_GKE_ACCESS_TOKEN@gitlab.singhritesh85.com/deployment/helm-repo-for-gke.git)

- helm upgrade --install bankapp helm-repo-for-gke/folo --create-namespace --namespace=bankapp --set image.repository=us-central1-docker.pkg.dev/wise-trainer-244916/bankapp-gcr-dev/dexter,image.tag=latest,replicaCount=1,service.type=ClusterIP,service.port=80

- kubectl get pods -n bankapp

dependencies:

- docker_image

only:

- main # Or your desired branch

In the worst scenario if the Access Token which you had created for the GitLab Project/Repository was compromised then for security reasons rotate it but do not forget to change it wherever you used it in your project.

For security reasons as per your Organisation you need to Rotate it on a Periodic basis but do not forget to change it on those places wherever you used it.

Kumar Singh

Source Code: <https://github.com/singhritesh85/Bank-App-GKE.git>

GitHub Repo: <https://github.com/singhritesh85/DevOps-Project-BankApp-CICD-using-GitLab-GCP-and-AWS.git>

Helm Chart: <https://github.com/singhritesh85/helm-repo-for-bitnami-GKE.git>

<https://github.com/singhritesh85/helm-repo-for-GKE.git>

Terraform Script: <https://github.com/singhritesh85/terraform-google-cloud-platform.git>

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

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