VTAI - Todo App Deployment

Frontend Application - Angular + Material UI
Backend Application - Node JS + Postgres
Unit Testing - Karma & Jasmine

Frontend Application - Angular

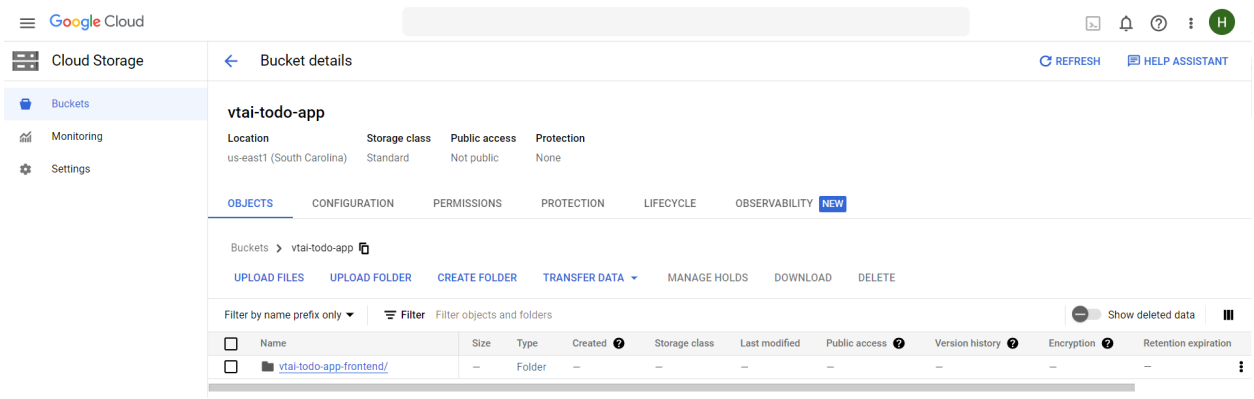
Configurations

Step 1: Create a Project in Google Cloud

Go to the Google Cloud console and click on the "Create Project" button. Give a preferred name for the project.

Step 2: Set up a Cloud Storage Bucket

Navigate to the "Storage" page. Click on the "Create Bucket" button and follow the prompts to create a new bucket.



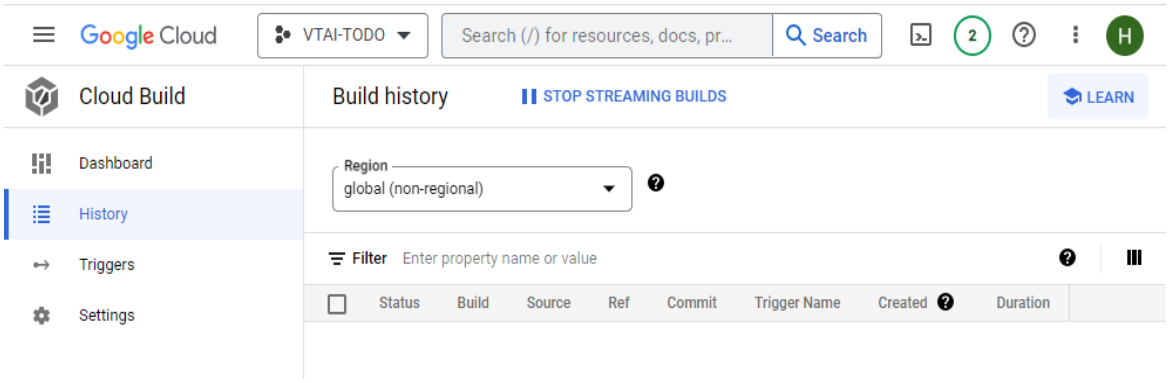
Automatic Deployment (With CI/CD)

Prerequisites

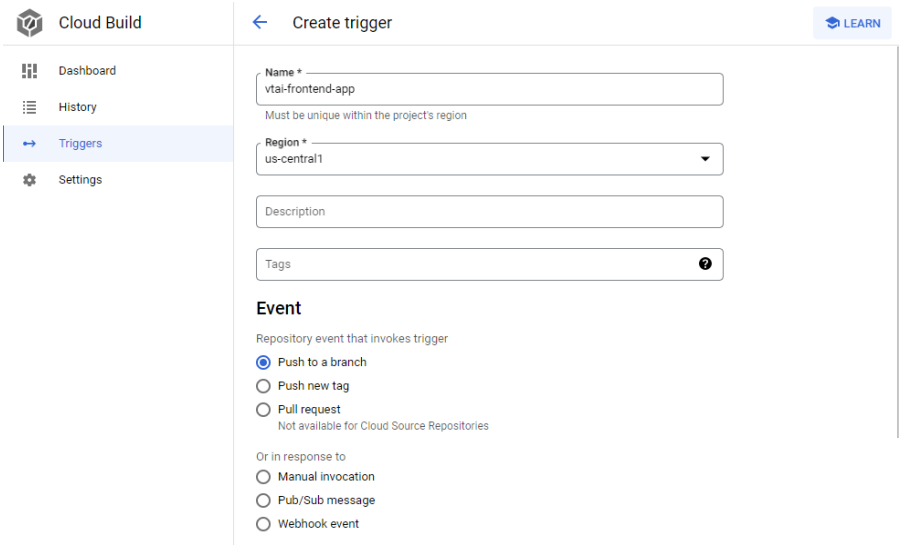
- A Google Cloud account
- A GitHub repository with your Angular application code
- A Google Cloud Storage bucket to host your application

Step 1: Set Up a Cloud Build Trigger

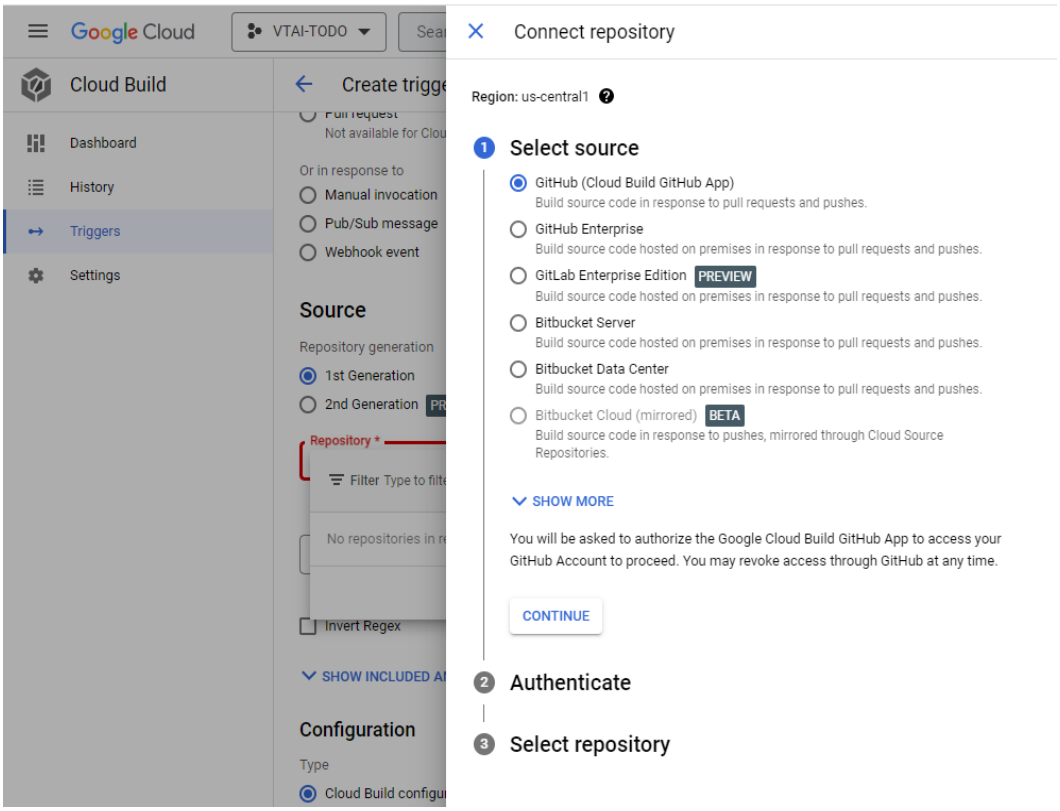
1. Go to your Google Cloud Console Project and select the project
2. Navigate to Cloud Build Page (If you haven't enabled API, **Enable** it. If you have already enabled you will navigate to the below page



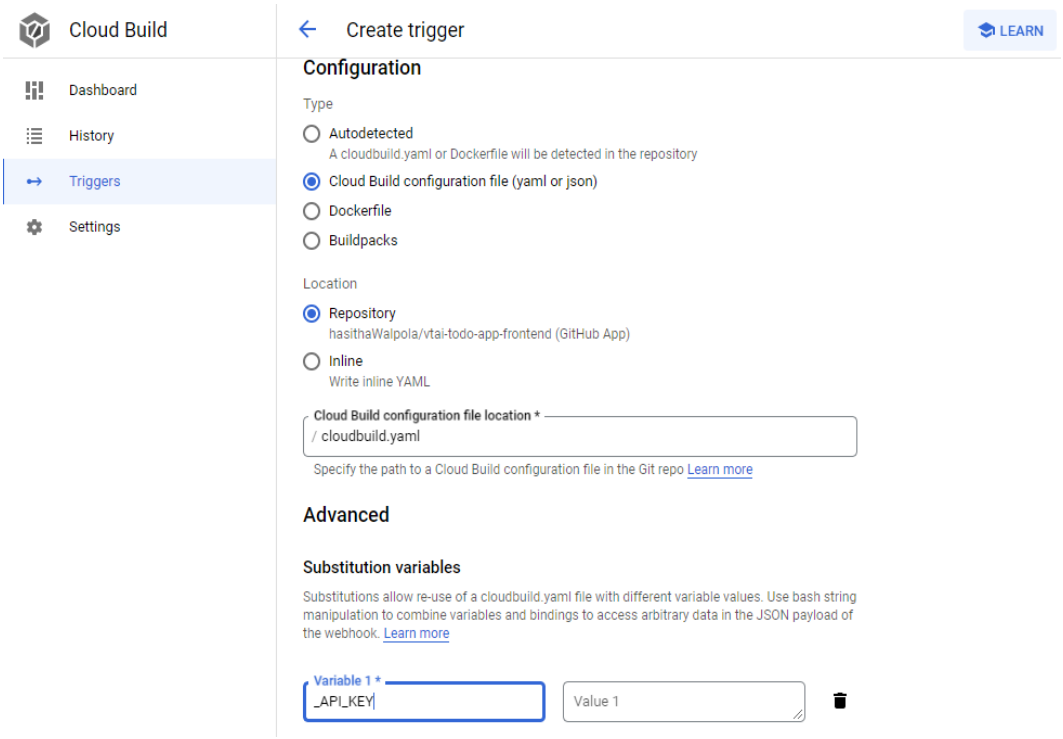
3. Next, go to the Trigger section and Create a Trigger
    a. Name trigger and set event



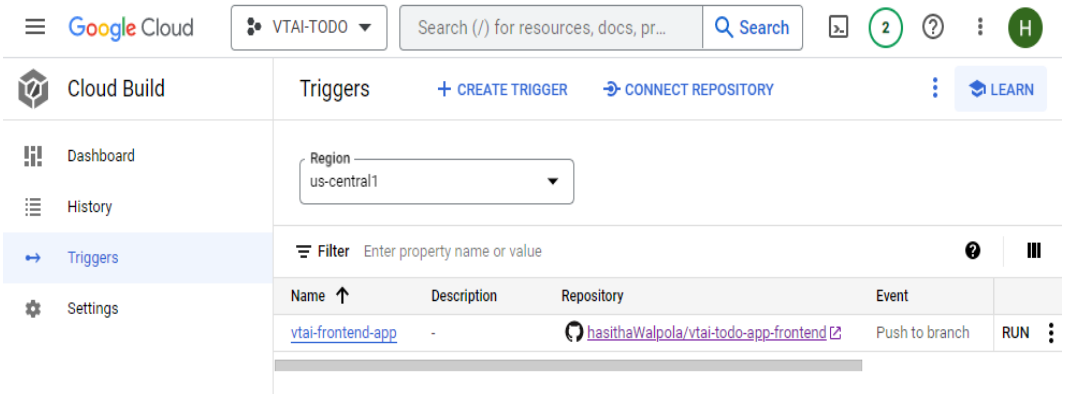
b. Connect Github Repository (Configure Github from this URL)



c. Configuration with Environment Variables(You can set environment variables If you need)



d. The created trigger can see in the trigger section



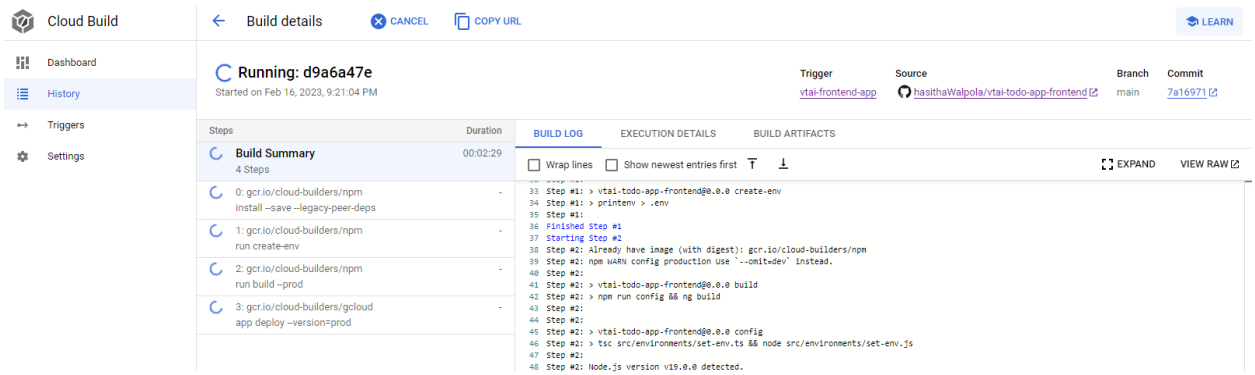
Step 2: Configure Cloud Build to Build and Deploy Your Application

To do this, create a "cloudbuild.yaml" file in the root directory of your Angular application. In this file, specify the build steps to install dependencies, build your application, and deploy it to your bucket. Here's a configuration file:

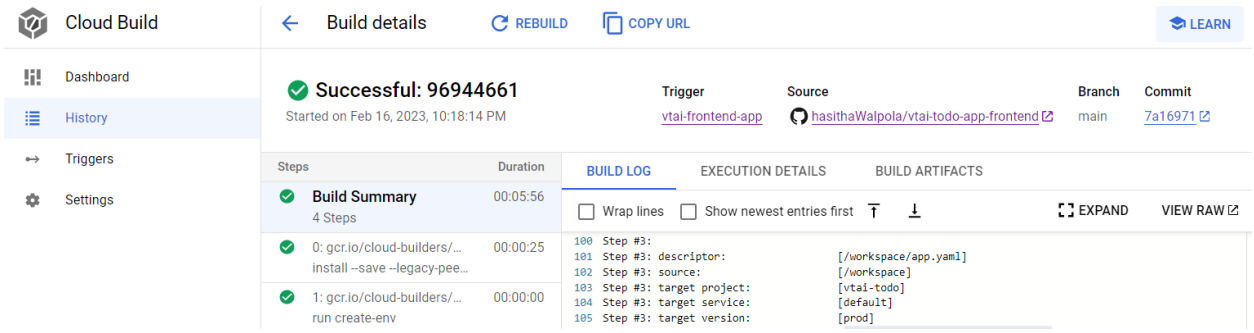
```yaml
steps:

  # Install node packages
  - name: 'gcr.io/cloud-builders/npm'
    args: [ 'install', '--save', '--legacy-peer-deps']

    # Create Environment file
  - name: 'gcr.io/cloud-builders/npm'
    args: [ 'run', 'create-env' ]
    env:
    - 'API_KEY=${_API_KEY}'

  # Build productive files
  - name: 'gcr.io/cloud-builders/npm'
    args: [ 'run', 'build', '--prod' ]

  # Deploy to google cloud app egnine
  - name: 'gcr.io/cloud-builders/gcloud'
    args: ['app', 'deploy', '--version=prod']
```

Step 3: Trigger Your Build and Deploy Process 😃

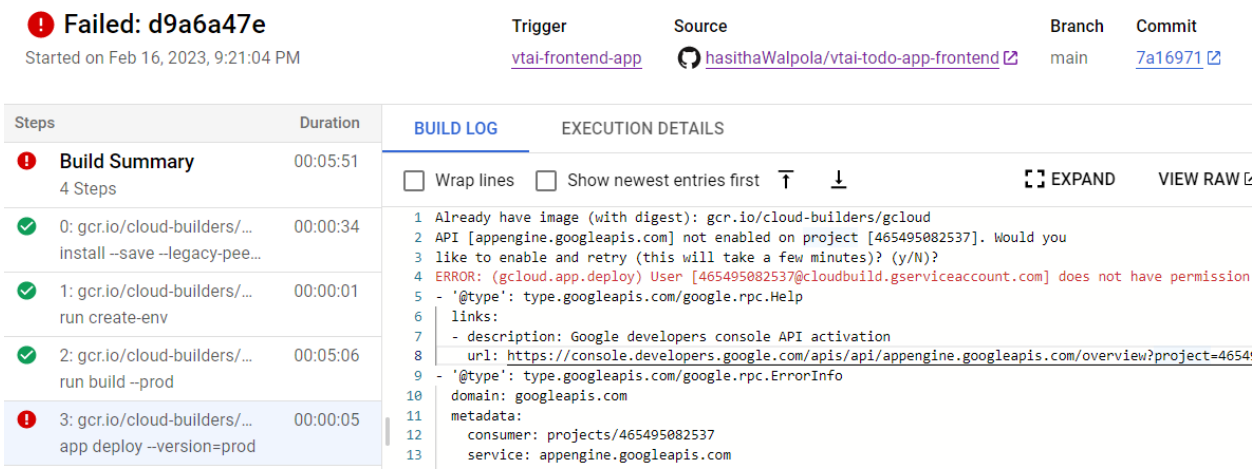
With everything set up, now you can trigger the build and deploy process by pushing changes to your GitHub repository. Cloud Build will automatically detect the changes and trigger a build process. Once the build is complete, your Angular application will be automatically deployed to your Google Cloud



If the build is a success you can see a screen like below and you can view the application by navigating to the mentioned URL build log



PS : If any error occurred 😣

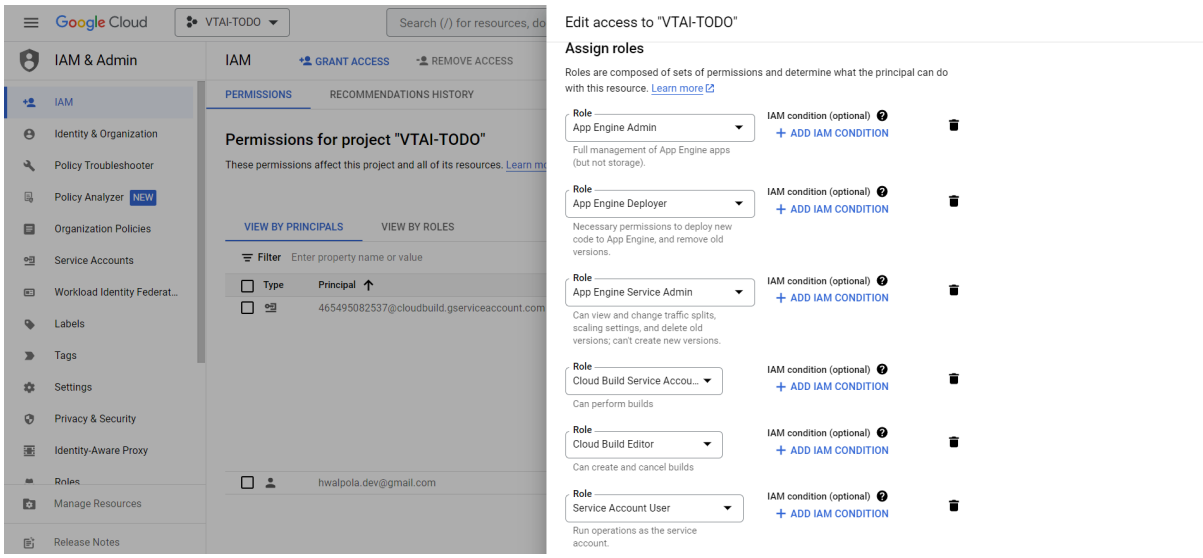


Most of the time this issue occurs when the Cloud Build service account doesn't have enough permission.

Make sure to Enable **App Engine Admin API**

You can assign roles for the account by navigating to IAM admin page

- **App Engine Deployer (deploy new code)**
- **Storage Object Viewer (list images)**
- **Storage Object Creator (upload the image)**
- **Cloud Build Editor (create build)**
- **App Engine Service Admin (promoting the new version)**



Manual Deployment (Without CI/CD)

Step 1: Build Your Angular Application

The next step is to build your Angular application

- Run “ng build” command in the project directory. This will create a "dist" directory containing the compiled files for your application.

```
D:\Projects\vtai-todo-app-frontend>ng build
√ Browser application bundle generation complete.
√ Copying assets complete.
√ Index html generation complete.
D:\Projects\vtai-todo-app-frontend>
Initial Chunk Files           | Names                  |  Raw Size | Estimated Transfer Size
main.b054cc886c9f84d3.js      | main                   | 774.65 kB |               169.16 kB
styles.02265c3a5250fecc.css   | styles                 | 153.27 kB |                11.14 kB
polyfills.db44d88161dbef46.js | polyfills              |  33.09 kB |                10.65 kB
runtime.605e372001995de9.js   | runtime                |   2.74 kB |                 1.31 kB

                              | Initial Total          | 963.75 kB |               192.26 kB

Lazy Chunk Files              | Names                  |  Raw Size | Estimated Transfer Size
893.9e054932b51b415b.js       | layouts-main-main-module |  15.81 kB |                 4.13 kB
173.258d582314c775d7.js       | layouts-auth-auth-module |  11.16 kB |                 2.35 kB
common.5f0a75260fc9936d.js    | common                 | 511 bytes |               308 bytes

Build at: 2023-02-16T14:20:25.163Z - Hash: 4b5ba179d9550a6b - Time: 7039ms
```

Step 2: Upload Your Application to the Cloud Storage Bucket

You can do this by using the "gsutil" command-line tool or by using the Google Cloud Console. (Follow the below process if you haven’t installed Google Cloud CLI)

- Run the following command to upload files through Google Cloud CLI

```
Gsutil -m rsync -r dist/ gs://<bucket-name>
```

```
D:\Projects\vtai-todo-app-frontend>gsutil -m rsync -r dist/ gs://vtai-todo-app
Building synchronization state...
Starting synchronization...
Copying file://dist\vtai-todo-app-frontend\3rdpartylicenses.txt [Content-Type=text/plain]...
Copying file://dist\vtai-todo-app-frontend\styles.02265c3a5250fecc.css [Content-Type=text/css]...
Copying file://dist\vtai-todo-app-frontend\893.9e054932b51b415b.js [Content-Type=application/javascript]...
Copying file://dist\vtai-todo-app-frontend\favicon.ico [Content-Type=image/x-icon]...
Copying file://dist\vtai-todo-app-frontend\common.5f0a75260fc9936d.js [Content-Type=application/javascript]...
Copying file://dist\vtai-todo-app-frontend\index.html [Content-Type=text/html]...
Copying file://dist\vtai-todo-app-frontend\runtime.605e372001995de9.js [Content-Type=application/javascript]...
Copying file://dist\vtai-todo-app-frontend\173.258d582314c775d7.js [Content-Type=application/javascript]...
Copying file://dist\vtai-todo-app-frontend\polyfills.db44d88161dbef46.js [Content-Type=application/javascript]...
Copying file://dist\vtai-todo-app-frontend\main.b054cc886c9f84d3.js [Content-Type=application/javascript]...
/ [10/10 files][  1.0 MiB/  1.0 MiB] 100% Done
Operation completed over 10 objects/1.0 MiB.
```

Installing Google Cloud SDK

1. Download the official google cloud SDK file form [here]

2. Run the installation file
3. Run `gcloud -v` to verify the installation

Initializing Google Cloud SDK

1. Run `gcloud init`
2. Accept the option to login and log into the account
3. Pickup the project

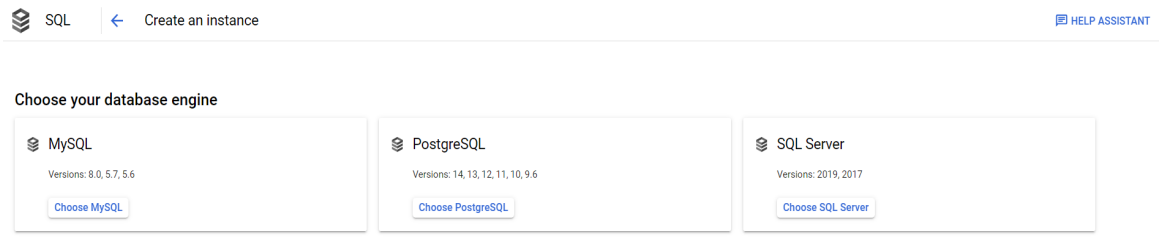
Step 3: Configure the Cloud Storage Bucket for Static Website Hosting

You can do this by going to the bucket details page in the Google Cloud Console and selecting the "Static website hosting" tab. Enable the "Static website hosting" option and specify the main page and error page for your website. Follow the official document

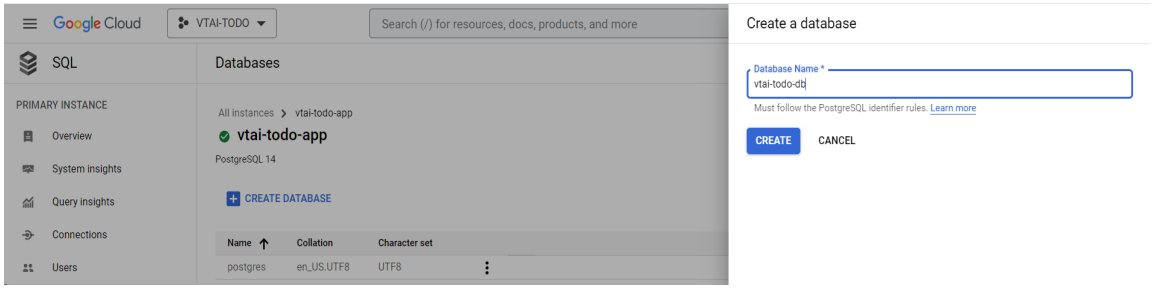
Backend Application - Node JS + PostgreSQL

Setup Database

1. Go to Google Cloud Console Project and Navigate to the SQL section
2. Then create an instance by selecting PostgreSQL



3. Enable **Compute Engine API**
4. Enter the **Instance Id** and **Password**
5. Select the Database version
6. Create the instance
7. Create Database by navigating to the **Database** section



8. Create User by navigating to the **User** section

Once you completed above steps you can you can find the connection name from the **Overview** tab

We need following fields to setup the database connection with Node JS server

- Connection Name
- Database Name
- User
- Password

Setup Node JS Project

In the .env file store the above fields (This file not push to Github Repository)

If we deploy the server using ci/cd we can store these credentials in github secrets

Automatic Deployment (With CI/CD)

Step 1: Create a Service Account and Set Up Authentication

Go to the IAM & Admin page and click on **Service account**s. Click on the "Create Service Account" button and follow the prompts to create a new service account. Give the service account the necessary permissions to deploy your application.

I added following permissions

- App Engine Admin
- App Engine Deployer
- App Engine Service Admin
- Cloud Build Editor
- Service Account User
- Storage Admin
- Storage Object Admin
- Storage Object Creator
- Storage Object Viewer

Step 2: Set Up a GitHub Actions Workflow

Create a new file named "deploy.yml" in the ".github/workflows" directory of your GitHub repository. In this file, specify the workflow to install dependencies, build your application, and deploy it to your Google Cloud Storage bucket. Here's a workflow file:

```yaml
name: Deploy to GAE

on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]

  workflow_dispatch:

jobs:
  deploy:
    name: Deploying to Google Cloud
    runs-on: ubuntu-latest

    steps:
    - name: Checkout
      uses: actions/checkout@v2

    - name: Load Config Files
      run: |
        echo "${{secrets.VTAI_CONFIG}}" | base64 --decode > .env

    - name: Deploy to App Engine
      id: deploy
      uses: google-github-actions/deploy-appengine@v0.2.0      You, 3 days ago • chore: u
      with:
        deliverables: app.yaml
        version: v1
        project_id: ${{ secrets.GCP_PROJECT }}
        credentials: ${{ secrets.GCP_CREDENTIALS }}

    - name: Test
      run: curl "${{ steps.deploy.outputs.url }}"
```

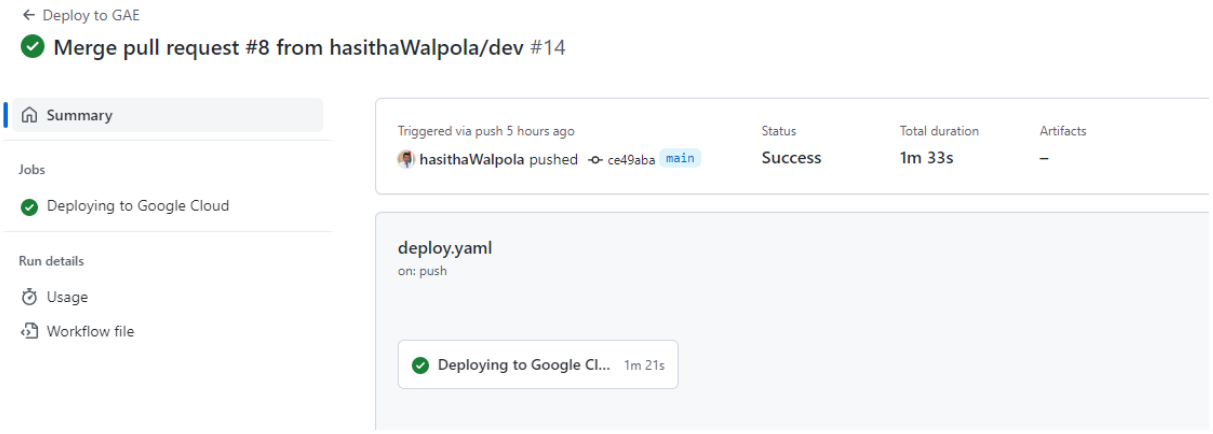
Make sure to add project details and credentials in the Github secrets. I added Database credentials into the secrets as well

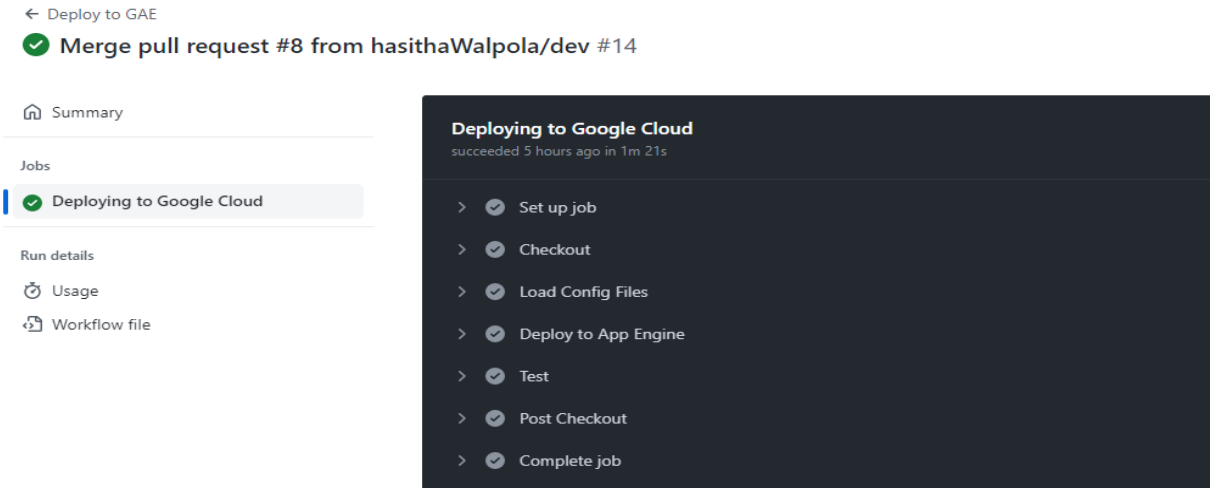
You can find the place to add secrets under the settings section in the repository.

Step 4: Trigger Your Build and Deploy Process

With everything set up, you can now trigger the build and deploy process by pushing changes to your GitHub repository. GitHub Actions will automatically detect the changes and trigger a build process. Once the build is complete, your Node.js application will be automatically deployed to your Google Cloud

If all went smoothly you can see the pipeline like below screenshot





Manual Deployment (Without CI/CD)

If you already setup the Google Cloud CLI in your local machine, and initialize the project Then you just need to run the below command in your project directory

- Open terminal and run `gcloud app deploy`
- You can browse the app by run `gcloud app browse`