

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.

The screenshot shows the Google Cloud Storage console interface. On the left, there's a sidebar with 'Cloud Storage' selected, and a sub-menu with 'Buckets', 'Monitoring', and 'Settings'. The main area is titled 'Bucket details' for the bucket named 'vtai-todo-app'. It shows metadata: Location (us-east1 (South Carolina)), Storage class (Standard), Public access (Not public), and Protection (None). Below this, there are tabs for 'OBJECTS', 'CONFIGURATION', 'PERMISSIONS', 'PROTECTION', 'LIFECYCLE', and 'OBSERVABILITY' (marked as 'NEW'). The 'OBJECTS' tab is active, showing a list of objects. At the top of the list, there's a breadcrumb 'Buckets > vtai-todo-app' and a toolbar with buttons: 'UPLOAD FILES', 'UPLOAD FOLDER', 'CREATE FOLDER', 'TRANSFER DATA', 'MANAGE HOLDS', 'DOWNLOAD', and 'DELETE'. Below the toolbar, there's a filter bar with 'Filter by name prefix only' and a 'Filter' button. The table below has columns: Name, Size, Type, Created, Storage class, Last modified, Public access, Version history, Encryption, and Retention expiration. One object is listed: 'vtai-todo-app-frontend/' with a size of '-', type of 'Folder', and other fields as '-'. There's also a 'Show deleted data' toggle and a 'Help Assistant' icon in the top right.

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration
vtai-todo-app-frontend/	-	Folder	-	-	-	-	-	-	-

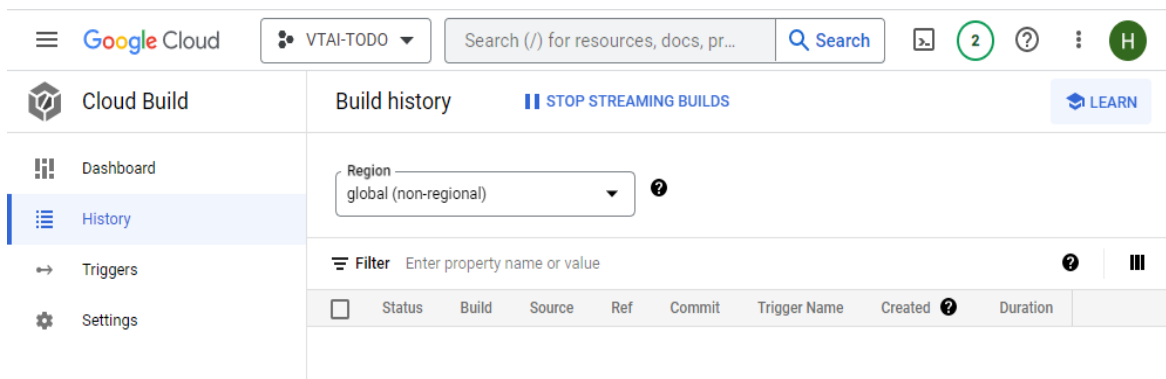
Automatic Deployment (With CI/CD)

Prerequisites

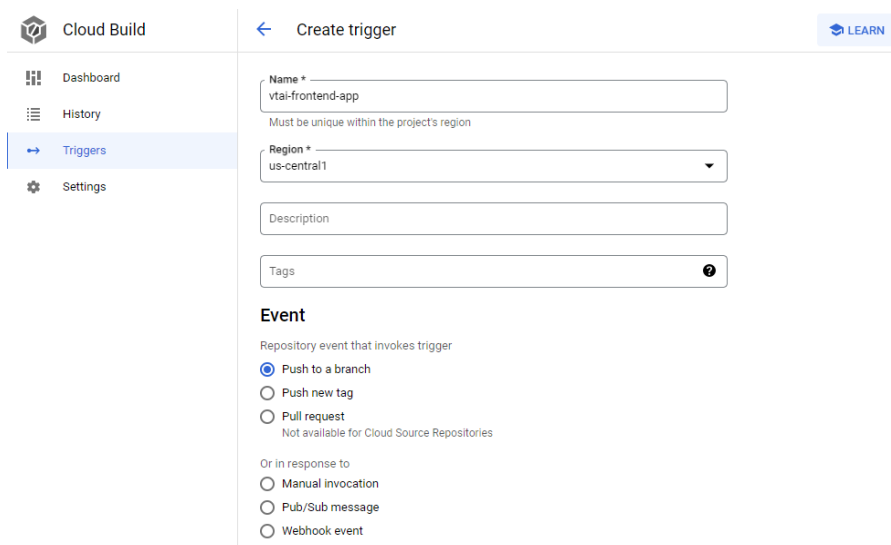
- A Google Cloud account
- A GitHub repository with your Angular application code
- A Google Cloud Storage bucket to host your application

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](#))

Google Cloud VTAI-TODO Search

Cloud Build

Dashboard History Triggers Settings

Create trigger

Full request Not available for Cloud Build

Or in response to

Manual invocation Pub/Sub message Webhook event

Source

Repository generation

1st Generation 2nd Generation

Repository * Filter Type to filter

No repositories in repository

Invert Regex

SHOW INCLUDED AND EXCLUDED REPOSITORIES

Configuration

Type

Cloud Build configuration file

Connect repository

Region: us-central1

1 Select source

GitHub (Cloud Build GitHub App) Build source code in response to pull requests and pushes.

GitHub Enterprise Build source code hosted on premises in response to pull requests and pushes.

GitLab Enterprise Edition PREVIEW Build source code hosted on premises in response to pull requests and pushes.

Bitbucket Server Build source code hosted on premises in response to pull requests and pushes.

Bitbucket Data Center Build source code hosted on premises in response to pull requests and pushes.

Bitbucket Cloud (mirrored) BETA Build source code in response to pushes, mirrored through Cloud Source Repositories.

SHOW MORE

You will be asked to authorize the Google Cloud Build GitHub App to access your GitHub Account to proceed. You may revoke access through GitHub at any time.

CONTINUE

2 Authenticate

3 Select repository

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

Cloud Build

Dashboard History Triggers Settings

Create trigger

LEARN

Configuration

Type

Autodetected A cloudbuild.yaml or Dockerfile will be detected in the repository

Cloud Build configuration file (yaml or json)

Dockerfile

Buildpacks

Location

Repository hasithaWaipola/vtai-todo-app-frontend (GitHub App)

Inline Write inline YAML

Cloud Build configuration file location * / cloudbuild.yaml

Specify the path to a Cloud Build configuration file in the Git repo [Learn more](#)

Advanced

Substitution variables

Substitutions allow re-use of a cloudbuild.yaml file with different variable values. Use bash string manipulation to combine variables and bindings to access arbitrary data in the JSON payload of the webhook. [Learn more](#)

Variable 1 * _API_KEY

Value 1

d. The created trigger can see in the trigger section

The screenshot shows the Google Cloud Triggers page. The left sidebar contains navigation links: Cloud Build, Dashboard, History, Triggers (selected), and Settings. The main content area is titled 'Triggers' and includes a '+ CREATE TRIGGER' button and a 'CONNECT REPOSITORY' button. A 'Region' dropdown menu is set to 'us-central1'. Below this is a 'Filter' input field. A table lists the triggers:

Name	Description	Repository	Event	Actions
vtai-frontend-app	-	hasithaWalpola/vtai-todo-app-frontend	Push to branch	RUN

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:

```
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 engine
  - 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

The screenshot shows the Google Cloud Build 'Build details' page for a build with ID **d9a6a47e**. The build is in a 'Running' state, started on Feb 16, 2023, at 9:21:04 PM. The interface includes a sidebar with 'Dashboard', 'History', 'Triggers', and 'Settings'. The main content area shows the build's progress with a 'Build Summary' table and a 'BUILD LOG' section. The 'Build Summary' table lists 4 steps: 0. gcr.io/cloud-builders/npm install --save --legacy-peer-deps, 1. gcr.io/cloud-builders/npm run create-env, 2. gcr.io/cloud-builders/npm run build --prod, and 3. gcr.io/cloud-builders/gcloud app deploy --version=prod. The 'BUILD LOG' section shows the execution details for each step, including commands like 'create-env', 'build', and 'deploy'.

Steps	Duration	BUILD LOG	EXECUTION DETAILS	BUILD ARTIFACTS
Build Summary 4 Steps	00:02:29	<input type="checkbox"/> Wrap lines <input type="checkbox"/> Show newest entries first <input type="text"/> <input type="button" value="⌵"/>		<input type="button" value="EXPAND"/> <input type="button" value="VIEW RAW"/>
0. gcr.io/cloud-builders/npm install --save --legacy-peer-deps	-	33 Step #1: > vtai-todo-app-frontend@0.0.0 create-env 34 Step #1: > printenv > .env 35 Step #1: 36 Finished Step #1		
1. gcr.io/cloud-builders/npm run create-env	-	37 Starting Step #2 38 Step #2: Already have image (with digest): gcr.io/cloud-builders/npm 39 Step #2: npm WARN config production Use '--omit=dev' instead. 40 Step #2: 41 Step #2: > vtai-todo-app-frontend@0.0.0 build 42 Step #2: > npm run config && ng build 43 Step #2: 44 Step #2: 45 Step #2: > vtai-todo-app-frontend@0.0.0 config 46 Step #2: > tsc src/environments/set-env.ts && node src/environments/set-env.js 47 Step #2: 48 Step #2: Node.js version v19.0.0 detected.		
2. gcr.io/cloud-builders/npm run build --prod	-			
3. gcr.io/cloud-builders/gcloud app deploy --version=prod	-			

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

The screenshot shows the Google Cloud Build 'Build details' page for a build with ID **96944661**. The build is in a 'Successful' state, started on Feb 16, 2023, at 10:18:14 PM. The interface includes a sidebar with 'Dashboard', 'History', 'Triggers', and 'Settings'. The main content area shows the build's progress with a 'Build Summary' table and a 'BUILD LOG' section. The 'Build Summary' table lists 4 steps: 0. gcr.io/cloud-builders/npm install --save --legacy-peer-deps, 1. gcr.io/cloud-builders/npm run create-env, 2. gcr.io/cloud-builders/npm run build --prod, and 3. gcr.io/cloud-builders/gcloud app deploy --version=prod. The 'BUILD LOG' section shows the execution details for each step, including commands like 'create-env', 'build', and 'deploy'.

Steps	Duration	BUILD LOG	EXECUTION DETAILS	BUILD ARTIFACTS
Build Summary 4 Steps	00:05:56	<input type="checkbox"/> Wrap lines <input type="checkbox"/> Show newest entries first <input type="text"/> <input type="button" value="⌵"/>		<input type="button" value="EXPAND"/> <input type="button" value="VIEW RAW"/>
0. gcr.io/cloud-builders/npm install --save --legacy-peer-deps	00:00:25	100 Step #3: 101 Step #3: descriptor: 102 Step #3: source: 103 Step #3: target project: 104 Step #3: target service: 105 Step #3: target version:		[/workspace/app.yaml] [/workspace] [vtai-todo] [default] [prod]
1. gcr.io/cloud-builders/npm run create-env	00:00:00			

PS : If any error occurred 😞

❗ Failed: d9a6a47e

Started on Feb 16, 2023, 9:21:04 PM

Trigger

[vtai-frontend-app](#)

Source

[hasithaWalpola/vtai-todo-app-frontend](#)

Branch

main

Commit

[7a16971](#)

Steps	Duration	BUILD LOG	EXECUTION DETAILS
❗ Build Summary	00:05:51	<input type="checkbox"/> Wrap lines <input type="checkbox"/> Show newest entries first	EXPAND VIEW RAW
4 Steps			
✓ 0: gcr.io/cloud-builders/...	00:00:34	install --save --legacy-pee...	1 Already have image (with digest): gcr.io/cloud-builders/gcloud 2 API [appengine.googleapis.com] not enabled on project [465495082537]. Would you 3 like to enable and retry (this will take a few minutes)? (y/N)? 4 ERROR: (gcloud.app.deploy) User [465495082537@cloudbuild.gserviceaccount.com] does not have permission
✓ 1: gcr.io/cloud-builders/...	00:00:01	run create-env	5 - '@type': type.googleapis.com/google.rpc.Help 6 links: 7 - description: Google developers console API activation 8 url: https://console.developers.google.com/apis/api/appengine.googleapis.com/overview?project=4654
✓ 2: gcr.io/cloud-builders/...	00:05:06	run build --prod	9 - '@type': type.googleapis.com/google.rpc.ErrorInfo 10 domain: googleapis.com 11 metadata: 12 consumer: projects/465495082537 13 service: appengine.googleapis.com
❗ 3: gcr.io/cloud-builders/...	00:00:05	app deploy --version=prod	

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)**

Google Cloud

VTAI-TODO

Search (/) for resources, do

IAM & Admin

IAM

GRANT ACCESS

REMOVE ACCESS

PERMISSIONS

RECOMMENDATIONS HISTORY

Permissions for project "VTAI-TODO"

These permissions affect this project and all of its resources. [Learn more](#)

VIEW BY PRINCIPALS

VIEW BY ROLES

Filter

Enter property name or value

☐ Type

☐ Principal

465495082537@cloudbuild.gserviceaccount.com

☐ hwalpola.dev@gmail.com

Edit access to "VTAI-TODO"

Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. [Learn more](#)

Role

App Engine Admin

IAM condition (optional)

+ ADD IAM CONDITION

Full management of App Engine apps (but not storage).

Role

App Engine Deployer

IAM condition (optional)

+ ADD IAM CONDITION

Necessary permissions to deploy new code to App Engine, and remove old versions.

Role

App Engine Service Admin

IAM condition (optional)

+ ADD IAM CONDITION

Can view and change traffic splits, scaling settings, and delete old versions; can't create new versions.

Role

Cloud Build Service Accou...

IAM condition (optional)

+ ADD IAM CONDITION

Can perform builds

Role

Cloud Build Editor

IAM condition (optional)

+ ADD IAM CONDITION

Can create and cancel builds

Role

Service Account User

IAM condition (optional)

+ ADD IAM CONDITION

Run operations as the service account.

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

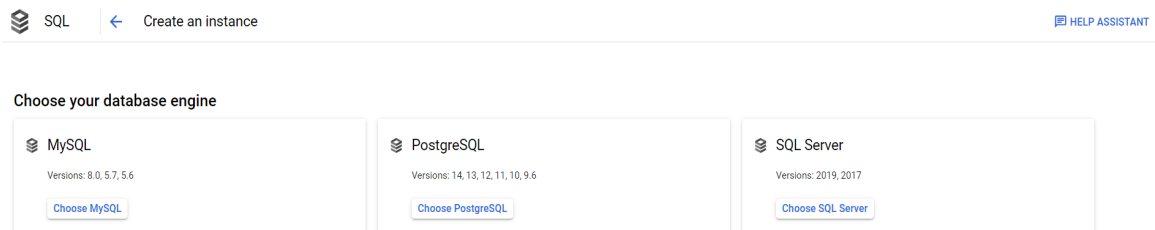
Step 3: Configure the Cloud Storage Bucket for Static Website Hosting

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](#)

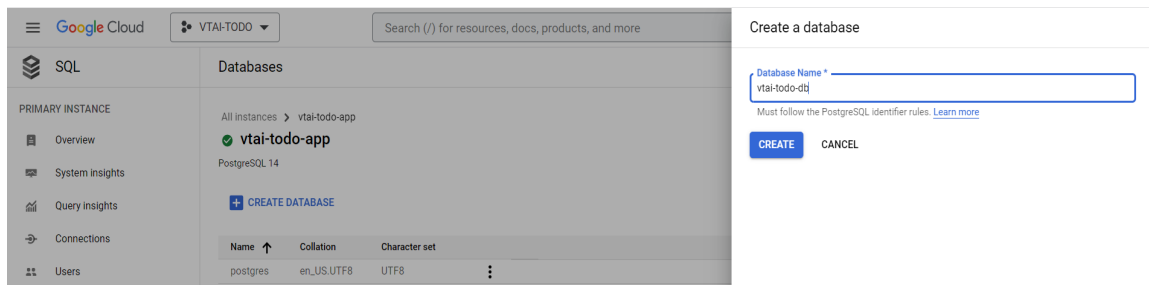
Backend Application - Node JS + PostgreSQL

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 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 accounts**. 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:

```
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
        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

← Deploy to GAE

✓ Merge pull request #8 from hasithaWalpola/dev #14

Summary

Jobs

✓ Deploying to Google Cloud

Run details

Usage

Workflow file

Triggered via push 5 hours ago

hasithaWalpola pushed · ce49aba · main

Status: Success

Total duration: 1m 33s

Artifacts: —

deploy.yaml

on: push

✓ Deploying to Google Cl... 1m 21s

← Deploy to GAE

✓ Merge pull request #8 from hasithaWalpola/dev #14

Summary

Jobs

✓ Deploying to Google Cloud

Run details

Usage

Workflow file

Deploying to Google Cloud

succeeded 5 hours ago in 1m 21s

- > ✓ Set up job
- > ✓ Checkout
- > ✓ Load Config Files
- > ✓ Deploy to App Engine
- > ✓ Test
- > ✓ Post Checkout
- > ✓ Complete job

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`