Moviebookingapplication

Java FSE1

Springboot-React-GCP

Backend - GitHub

Frontend - GitHub

GCP Deployment Steps

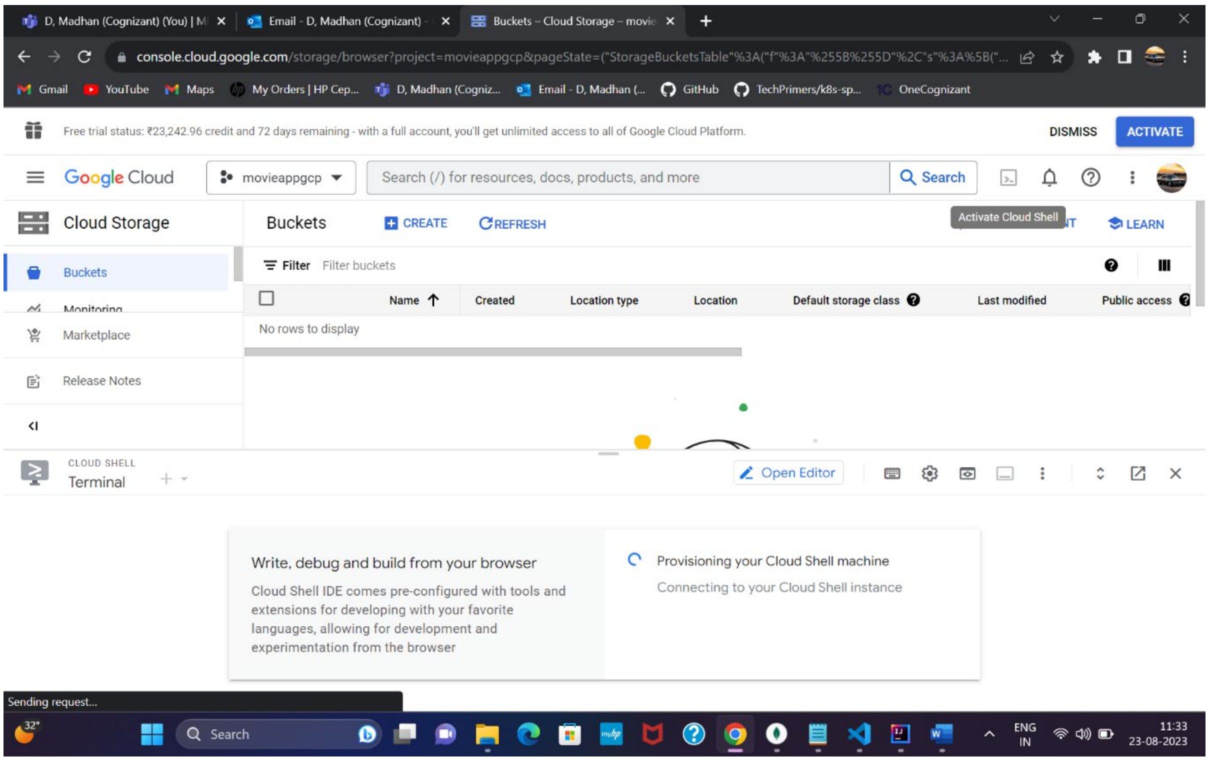
1. Create MongoDB atlas by selecting GCP as a service and create one connection

   After creating connection, you will get uri like this

   ```
   spring.data.mongodb.uri=mongodb+srv://test:test@cluster0.pbg51uc.mongodb.net/?retryWrites=true&w=majority
   spring.data.mongodb.database=movieapp
   ```

   Update this in application properties.

2. Push the updated code to your git repo.
3. Now go to GCP console and activate the cloud shell



Free trial status: ₹23,242.96 credit and 72 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Write, debug and build from your browser

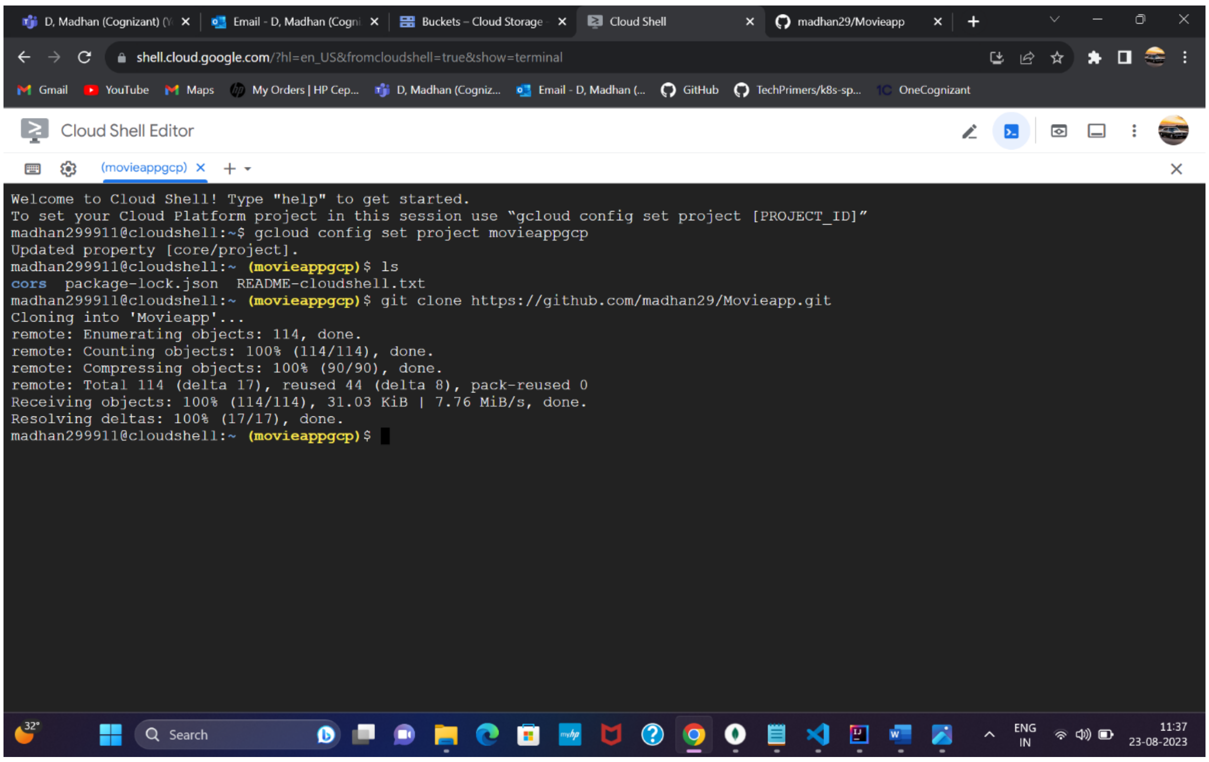
Cloud Shell IDE comes pre-configured with tools and extensions for developing with your favorite languages, allowing for development and experimentation from the browser

Provisioning your Cloud Shell machine

Connecting to your Cloud Shell instance

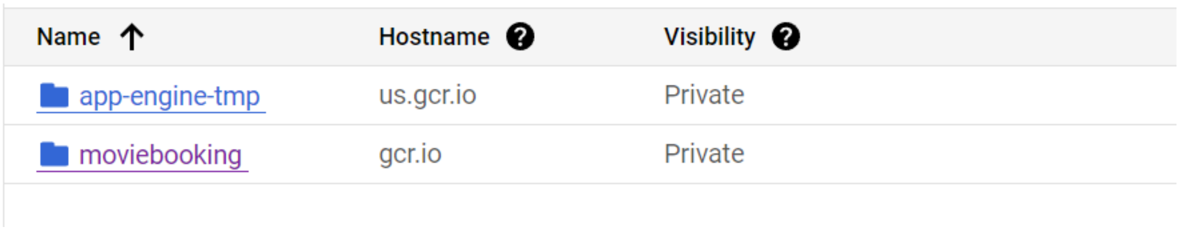
4. In cloud shell select your project by using following command if it's not selected default
   gcloud config set project [PROJECT_ID]

5. Clone your git repo here – git clone your repo URL.

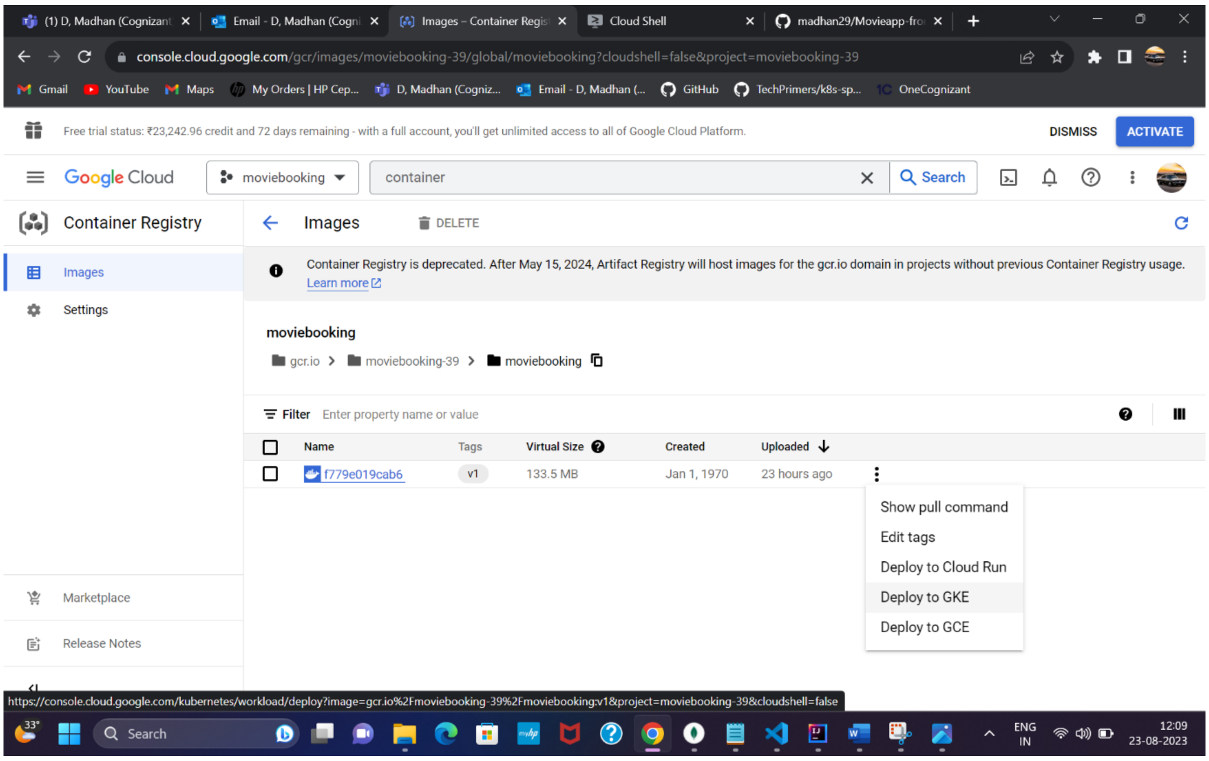

shell.cloud.google.com/?hl=en_US&fromcloudshell=true&show=terminal
Cloud Shell Editor
(movieappgcp)

```
Welcome to Cloud Shell! Type "help" to get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
madhan299911@cloudshell:~$ gcloud config set project movieappgcp
Updated property [core/project].
madhan299911@cloudshell:~ (movieappgcp)$ ls
cors  package-lock.json  README-cloudshell.txt
madhan299911@cloudshell:~ (movieappgcp)$ git clone https://github.com/madhan29/Movieapp.git
Cloning into 'Movieapp'...
remote: Enumerating objects: 114, done.
remote: Counting objects: 100% (114/114), done.
remote: Compressing objects: 100% (90/90), done.
remote: Total 114 (delta 17), reused 44 (delta 8), pack-reused 0
Receiving objects: 100% (114/114), 31.03 KiB | 7.76 MiB/s, done.
Resolving deltas: 100% (17/17), done.
madhan299911@cloudshell:~ (movieappgcp)$
```

6. Now select your file using cd your_filename.
7. Run **mvn clean install**, to build your backend.
8. After build completed you need to create docker image for your backend application. Use this command
   mvn com.google.cloud.tools:jib-maven-plugin:build -Dimage=gcr.io/your gcp project-id /your_containername:v1
9. After running this command your docker image will be created and stored in container registry in GCP.



Name ↑	Hostname	Visibility
app-engine-tmp	us.gcr.io	Private
moviebooking	gcr.io	Private

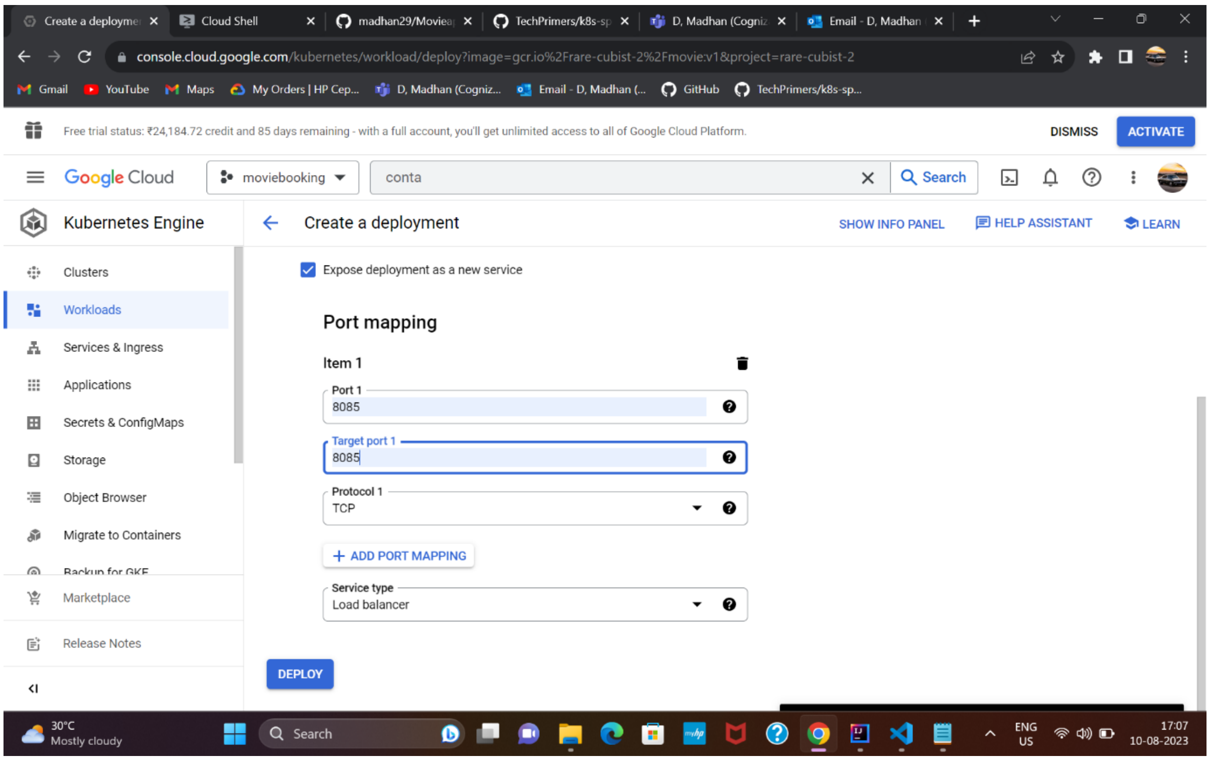

Container Registry is deprecated. After May 15, 2024, Artifact Registry will host images for the gcr.io domain in projects without previous Container Registry usage. Learn more

moviebooking

gcr.io > moviebooking-39 > moviebooking

Name	Tags	Virtual Size	Created	Uploaded
f779e019cab6	v1	133.5 MB	Jan 1, 1970	23 hours ago

From Here to choose to deploy to GKE.


Create a deployment

Expose deployment as a new service

Port mapping

Item 1

Port 1: 8085

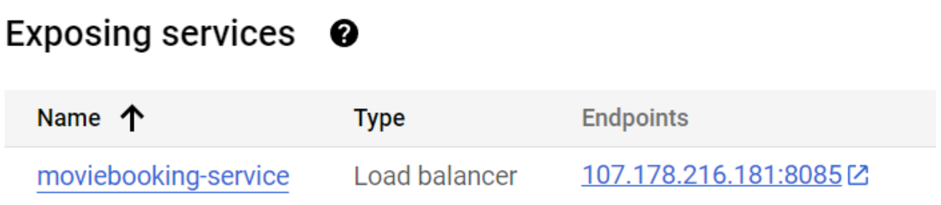
Target port 1: 8085

Protocol 1: TCP

Service type: Load balancer

Here Choose your port which your backend running and expose it and click deploy.

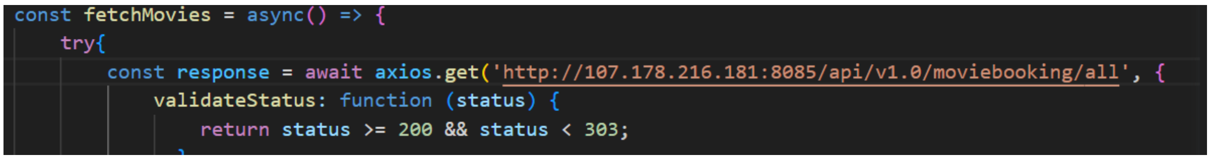
Once it deployed to GKE you will get external IP



Exposing services

Name	Type	Endpoints
moviebooking-service	Load balancer	107.178.216.181:8085

From using this IP you need to update the frontend code like this.



```
const fetchMovies = async() => {
    try{
        const response = await axios.get('http://107.178.216.181:8085/api/v1.0/moviebooking/all', {
            validateStatus: function (status) {
                return status >= 200 && status < 303;
```

After updating all your endpoint you need to create build file for your frontend using

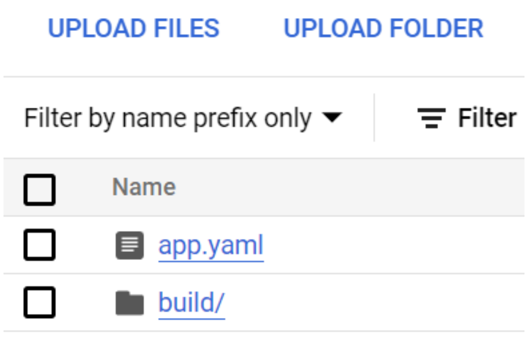
npm run build command this will give a build file which needs to deploy on GCP.

After that you need to create app.yaml file for deploying it your application to app engine.

app.yaml

```
runtime: nodejs16
handlers:
  # Serve all static files with url ending with a file extension
  - url: /(.*\..+)$
    static_files: build/\1
    upload: build/(.*\..+)$
  # Catch all handler to index.html
  - url: /.*
    static_files: build/index.html
    upload: build/index.html
    http_headers:
      Access-Control-Allow-Origin: "*"
```

Then go to Cloud Storage and create new bucket



Name
app.yaml
build/

and update your build folder and app.yaml file to your bucket after go to cloud shell create new folder using mkdir foldername command.

Then you need to copy your bucket to newly created folder using the below command

gsutil rsync -r gs://yourbucket ./yourfolder

after that select your folder by run cd yourfoldername.

After selecting your folder run **gcloud app deploy** your application to App engine.

After deploying your application to app engine, you will get URL for accessing your application.

https://moviebooking-39.uc.r.appspot.com/ like this. (By using this link, you will get SSL certificate issue)

Make sure to run using http instead of using https.

And you will CORS error after changing it to http for fix this you need to disable cors in google chrome using the below command.

cd **C:\Program Files (x86)\Google\Chrome\Application**

chrome.exe --user-data-dir="C:/Chrome dev session" --disable-web-security

C:\Program Files (x86)\Google\Chrome\Application – select your chrome installed path.

C:/Chrome dev session – change a folder in your drive.

That's all now your application will work without any issue.

Will attach YouTube tutorials I followed to deploy this

Backend deployment – Click here

Frontend deployment – Click here

Disable cors – Click here

GitHub link for GKE commands – Click here

Thanks