

Cloud Application Development

Midterm Project Report

Deploying a Scalable Web Application on Google Cloud Platform

Yessimseit Manarbek 21B030812

Fall 2024 Almaty

Table of Contents

1. **Executive Summary**
2. **Introduction**
3. **Project Objectives**
4. **Google Cloud Platform Overview**
5. **Google Cloud SDK and Cloud Shell**
6. **Google App Engine**
7. **Building with Google Cloud Functions**
8. **Containerizing Applications**
9. **Managing APIs with Google Cloud Endpoints**
10. **Testing and Quality Assurance**
11. **Monitoring and Maintenance**
12. **Challenges and Solutions**
13. **Conclusion**
14. **References**
15. **Appendices**

1. Executive Summary

In this work, I created web applications according to the task - ToDo list. This application was deployed on the Google Cloud platform.

To create the basis of the application itself, I used the Flask framework and the Python language. To deploy a scalable web application, we used services such as the App Engine, Cloud Functions to create serverless functions, and Cloud Endpoints to operate the API. We also demonstrated the ability to deploy applications in the Kubernetes cluster and create docker images with subsequent storage in the Google Artifactory Container registry.

And we create user acceptance tests and quality assurance test.

As a result, we have successfully deployed a fault-tolerant, scalable web application.

2. Introduction

Google Cloud Platform - is one of the most popular cloud platforms in the world supported by Google Corporation. As we can understand cloud platforms offer ready-made solutions of different services and tools. This gives developers the ability to quickly deploy infrastructure, raise and customize it, deploy their web applications, work with ML and so on. This Platform as a Service approach allows people to use Google's computing power cheaply and quickly

when in turn Google's computing power is not idle and the company makes money. Thanks to GCP UI and gcloud command line tool we can conveniently work with such services as google kubernetes engine, compute engine, cloud storage, bigquery. For me GCP is much more convenient from its analogs like DigitalOcean, AWS. That's why I chose it, and also it is easy to activate a free trial period, which is not the case with AWS.

GCP makes it easy to quickly deploy scalable, fault-tolerant web applications. It is easy to customize API.

3. Project Objectives

Project objectives:

- 1) Develop To-do list Web-application using python, Flask.
- 2) Successfully deploying Application into Google App engine. Make sure it works
- 3) Create 2 cloud serverless functions for sending notifications and processing user inputs and success implement them to main To-Do application. Make sure it works
- 4) Containerize application using Docker and deploy application to Google Kubernetes Engine.
- 5) Create API with auth and monitoring of application using cloud Endpoints.
- 6) Test application using unit, integration, QAT.
- 7) Use monitoring tools for check performance and ensure uptime and reliability.

4. Google Cloud Platform Overview

GCP provides a huge number of services and technologies for solving various tasks. As part of our assignment and course, we will get acquainted with the following services:

- App engine for deploy web app
- Compute Engine for create VM
- Google Kubernetes to create kubernetes clusters
- Vertex AI to machine learning
- Cloud storage to create database for store data.
- VPC Network to create subnets and networks and configure network
- Cloud Endpoint for maintenance API
- Cloud Functions for create serverless functions

Thanks to all these features, we can create large-scale fault-tolerant cloud applications cheaply and quickly.

5. Google Cloud SDK and Cloud Shell

- **Setup:** Detail the installation and configuration of Google Cloud SDK.

- **Cloud Shell Usage:** Discuss how Cloud Shell was utilized for resource management and deployment tasks.

1) SETUP

Firstly we need install Google cloud SDK

Linux terminal command for install SDK:

```
sudo apt-get update
```

```
sudo apt-get install apt-transport-https ca-certificates gnupg curl
```

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /usr/share/keyrings/cloud.google.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
```

```
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
```

```
sudo apt-get update && sudo apt-get install google-cloud-cli
```

```
maes0624@ws-20432:~$ whoami
maes0624
maes0624@ws-20432:~$ gcloud version
Google Cloud SDK 495.0.0
alpha 2024.09.27
beta 2024.09.27
bq 2.1.8
bundled-python3-unix 3.11.9
core 2024.09.27
gcloud-crc32c 1.0.0
gke-gcloud-auth-plugin 0.5.9
gsutil 5.30
kubectl 1.29.8
maes0624@ws-20432:~$ |
```

Maes0624 – Manarbek Yessimseit

Then we run gcloud init command to initialize the SDK and authenticate with.

```
gcloud projects create PROJECT_ID --name="cloud-app-dev-yessimseit2" – command used for create new project
```

```
gcloud config set project cloud-app-dev-yessimseit2 - command used for set new created project
```

How you see our current project is cloud-app-dev-yessimseit2

```
Account: [manarbek04esim@gmail.com]
Project: [cloud-app-dev-yessimseit2]
Universe Domain: [googleapis.com]
```

2) Cloud Shell Usage

Unfortunately I didn't use google cloud shell because For me was enough use Linux terminal with gcloud cli.

6. Google App Engine

- **Application Development:** Describe the web application developed (e.g., features, framework used).
- **Deployment:** Step-by-step guide on deploying the application to App Engine.

1) Application Development

I create To Do web application via Flask framework.

In this application you can

- Create task
- View all task
- Update task
- Delete task

For example screenshot of main.py. This file contains application logic

```
main.py > notify
1  from flask import Flask, request, jsonify
2  import requests
3
4  app = Flask(__name__)
5
6  tasks = [
7      {"id": 1, "task": "Cloud-app-dev: Assignment 3", "done": False},
8      {"id": 2, "task": "Cloud-app-dev: Assignment 4", "done": False}
9  ]
10
11 @app.route('/')
12 def index():
13     return "To-DO list by Midterm tsk!"
14
15 @app.route('/tasks', methods=['GET'])
16 def get_tasks():
17     return jsonify(tasks)
18
19 @app.route('/tasks', methods=['POST'])
20 def add_task():
21     new_task = request.json
22     new_task['id'] = len(tasks) + 1
23     tasks.append(new_task)
24     user_data = {'user_data': new_task}
25     requests.post('https://europe-west1-cloud-app-dev-yessimseit2.cloudfunctions.net/process_user_data', json=user_data)
26     return jsonify(new_task), 201
27
28 @app.route('/tasks/<int:task_id>', methods=['PUT'])
29 def update_task(task_id):
30     # Implement update logic here
```

2) Deployment

Firstly we create app.yaml that contains necessary conf for app deployment to App engine.

```

app.yaml > {} endpoints_api_service > rollout_strategy
1 runtime: python39
2 env: flex
3 service: default
4
5 handlers:
6 - url: /.+
7   script: auto
8
9 endpoints_api_service:
10   name: cloud-app-dev-yessimseit2.appspot.com
11   rollout_strategy: managed

```

Add: we need create python requirements.txt file for python lib

```

Flask==3.0.0
requests==2.32.3
functions-framework==3.8.1

```

then we use `gcloud app deploy` command for deploy application to Google platform.

Then gcloud provide for us <https://cloud-app-dev-yessimseit2.ew.r.appspot.com/> link through we have access to our application

7. Building with Google Cloud Functions

- **Serverless Functions:** Explain how Cloud Functions were created and integrated into the main application, including specific use cases.

I create 2 cloud functions for send notification and processing user inputs.

- 1) Send notification function send notify in logs and in future can help work with specify

```

send-notify > main.py > ...
1 import functions_framework
2
3 def send_notification(request):
4     request_json = request.get_json(silent=True)
5     if request_json and 'message' in request_json:
6         message = request_json['message']
7         print(f"Notification sent: {message}")
8         return "Notification sent successfully.", 200
9     else:
10        return "Invalid request. Please provide a message.", 400
11

```

This function send notify in logs and in future can help work with specify task.

Add: we need add requirement.txt with python lib

```

Flask==3.0.0

```

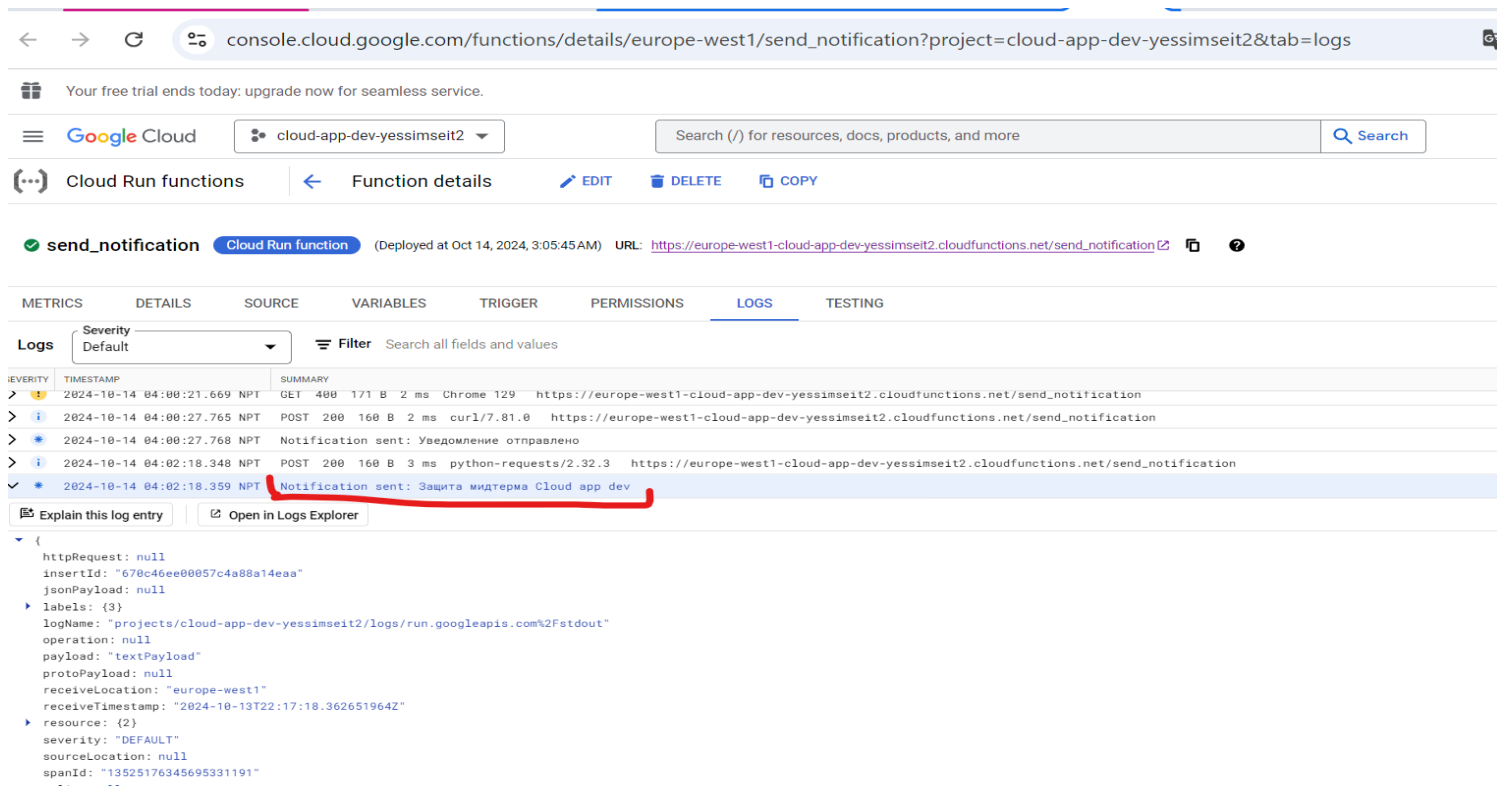
```
requests==2.32.3
functions-framework==3.8.1
```

Then we use next command to deploy function.

```
gcloud functions deploy send_notification --gen2 --runtime=python39 --trigger-http --allow-unauthenticated --region=europe-west1 --source=./directory/
```

Now I want to demonstrate how it working:

In google cloud console we see my function and see HTTP response and how you my notification working



The screenshot shows the Google Cloud Console interface for a Cloud Run function named 'send_notification'. The 'LOGS' tab is active, displaying a list of log entries. The most recent entry, timestamped 2024-10-14 04:02:18.359 NPT, shows a POST request from curl/7.81.0 to the function's URL. The log message is 'Notification sent: Защита мидтерма Cloud app dev'. Below the log entry, the full log data is expanded, showing details like httpRequest, insertId, labels, and resource information.

For test we use curl command with POST http method.

```
maes0624@ws-20432:~$ curl -X POST https://cloud-app-dev-yessimseit2.ew.r.appspot.com/tasks -H "Content-Type: application/json" -d '{"task": "Train 4 ama", "done": false, "user_data": "Check working 1"}' ^C
maes0624@ws-20432:~$ curl -X POST https://cloud-app-dev-yessimseit2.ew.r.appspot.com/send_notification -H "Content-Type: application/json" -d '{"message": "4 am check"}' | jq .
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100    58  100    33  100    25      32      24  0:00:01  0:00:01 --:--:--   56
{
  "message": "Notification sent."
}
maes0624@ws-20432:~$ |
```

Integrate this function with my main application

```
@app.route('/send_notification', methods=['POST'])
def notify():
    message = request.json.get('message')
    requests.post('https://europe-west1-cloud-app-dev-yessimseit2.cloudfunctions.net/send_notification', json={'message': message})
    return jsonify({"message": "Notification sent."}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

2) Process user data function

Firstly we create this function

```
1  import functions framework
2
3  def process_user_data(request):
4      request_json = request.get_json(silent=True)
5      if request_json and 'user_data' in request_json:
6          user_data = request_json['user_data']
7          print(f"Received user data: {user_data}")
8          return "User data processed successfully.", 200
9      else:
10         return "Invalid request. Please provide user data.", 400
11
```

This function receive user data and validate that, if we don't have user data it send error

Next step it's deploy function to cloud function

```
gcloud functions deploy process_user_data --gen2 --runtime=python39 --trigger-http --allow-unauthenticated --region=europe-west1 --source=./directory/
```

For check working we use curl command with HTTP method

```
maes0624@ws-20432:~$ curl -X POST https://cloud-app-dev-yessimseit2.ew.r.appspot.com/tasks -H "Content-Type: application/json" -d '{"task": "Train 4 ama", "done": false, "user_data": "check working 1"}' | jq .
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total   Spent    Left     Speed
100 143    100   73 100    70    213    204  --:--:-- --:--:-- --:--:--   418
{
  "done": false,
  "id": 4,
  "task": "Train 4 ama",
  "user_data": "check working 1"
}
maes0624@ws-20432:~$ |
```


process_user_data

Cloud Run function

(Deployed on Oct 14, 2024, 3:30:13 AM)

URL: https://europe-west1-cloud-app-dev-yessimseit2.cloudfunctions.net/process_user_data

METRICS

DETAILS

SOURCE

VARIABLES

TRIGGER

PERMISSIONS

LOGS

TESTING

logs

Severity

Default

Filter

Search all fields and values

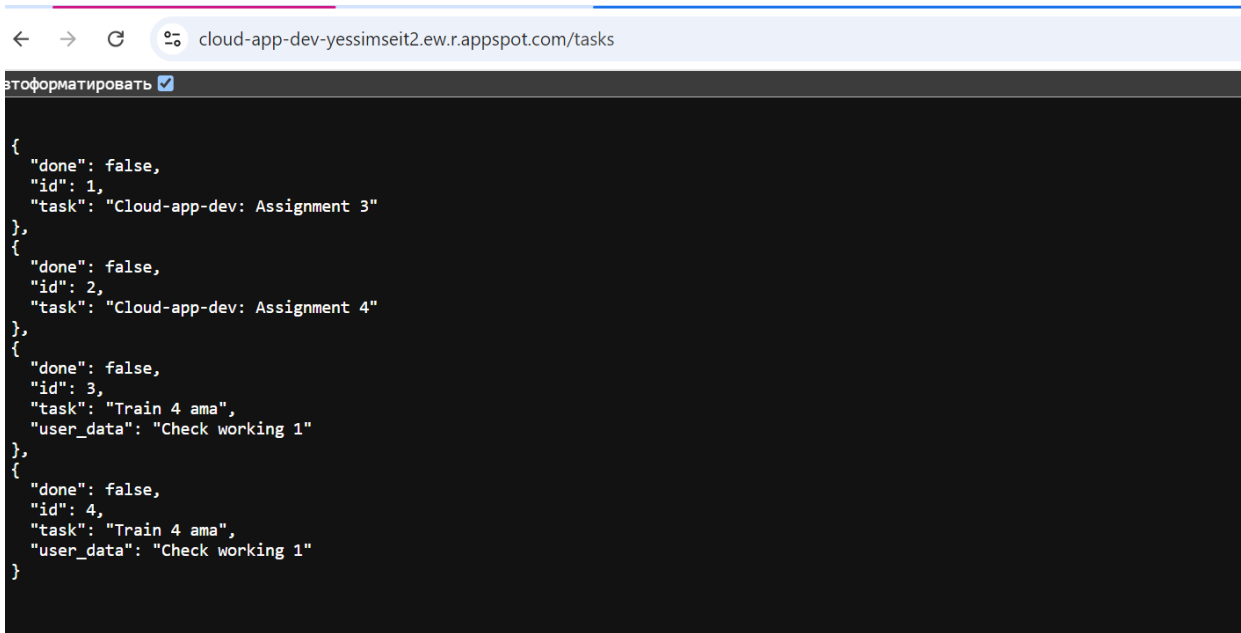
ENTRY	TIMESTAMP	SUMMARY
	2024-10-14 04:42:03.253 NPT	POST 200 162 B 2 ms python-requests/2.32.3 https://europe-west1-cloud-app-dev-yessimseit2.cloudfunctions.net/process_user_data
	2024-10-14 04:42:03.260 NPT	Received user data: {'task': 'Train 4 ama', 'done': False, 'user_data': 'Check working 1', 'id': 3}
	2024-10-14 04:42:31.606 NPT	POST 200 162 B 7 ms python-requests/2.32.3 https://europe-west1-cloud-app-dev-yessimseit2.cloudfunctions.net/process_user_data
	2024-10-14 04:42:31.614 NPT	Received user data: {'task': 'Train 4 ama', 'done': False, 'user_data': 'Check working 1', 'id': 4}

Explain this log entry

Open in Logs Explorer

```
{
  httpRequest: null
  insertId: "670c505b00095e72205e6015"
  jsonPayload: null
  labels: {3}
  logName: "projects/cloud-app-dev-yessimseit2/logs/run.googleapis.com%2Fstdout"
  operation: null
  payload: "textPayload"
  protoPayload: null
  receiveLocation: "europe-west1"
  receiveTimestamp: "2024-10-13T22:57:31.619707568Z"
  resource: {2}
  severity: "DEFAULT"
  sourceLocation: null
  spanId: "13491126296403909686"
  split: null
}
```

How you see this function success working and we can add new tasks to To-DO list



```
{
  "done": false,
  "id": 1,
  "task": "Cloud-app-dev: Assignment 3"
},
{
  "done": false,
  "id": 2,
  "task": "Cloud-app-dev: Assignment 4"
},
{
  "done": false,
  "id": 3,
  "task": "Train 4 ama",
  "user_data": "Check working 1"
},
{
  "done": false,
  "id": 4,
  "task": "Train 4 ama",
  "user_data": "Check working 1"
}
```

8. Containerizing Applications

- **Docker Overview:** Discuss the containerization process using Docker.
- **GKE Deployment:** Detail the steps taken to deploy the containerized application on GKE.

1) **Docker Overview:** **Docker use this tool** you can run container with you containerized application. It helps run application on various system. For this we need create docker Image of our application that it will assemble into one code, libraries, and all conf files and instructions necessary to create container.

Steps:

Create DockerFile → Use docker build command to create image → run container with our image using docker run command.

After that we create docker image we need store them in container registry. We use google cloud registry.

Using this command we push docker image to

`docker push europe-west1-docker.pkg.dev/cloud-app-dev-yessimseit2/cloud-app-dev/to-do:1.0`


```

provideClusterInfo: true
maes0624@ws-20432:~/.kube$ kubectl get namespaces
NAME                STATUS    AGE
default             Active   5m33s
gke-managed-cim     Active   5m
gke-managed-system  Active   4m48s
gmp-public          Active   4m34s
gmp-system          Active   4m34s
kube-node-lease     Active   5m33s
kube-public         Active   5m33s
kube-system         Active   5m33s
maes0624@ws-20432:~/.kube$ kubectl create namespace cloud-app-dev-todo
namespace/cloud-app-dev-todo created
maes0624@ws-20432:~/.kube$ ksn --namespace=cloud-app-dev-todo
Context "gke_cloud-app-dev-yessimseit2_europe-west1_cloud-app-dev-cluster" modified.
maes0624@ws-20432:~/.kube$ kaf /home/maes0624/gcloud/cloud-app-dev/midterm/deployment.yaml
deployment.apps/midterm-todo created
maes0624@ws-20432:~/.kube$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
midterm-todo-6c8c6d4d54-k6pmw      0/1     ContainerCreating   0           5s
midterm-todo-6c8c6d4d54-pq24b      0/1     ContainerCreating   0           5s
maes0624@ws-20432:~/.kube$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
midterm-todo-6c8c6d4d54-k6pmw      1/1     Running     0           86s
midterm-todo-6c8c6d4d54-pq24b      1/1     Running     0           86s
maes0624@ws-20432:~/.kube$

```

Here we

- 1) Create out namespace on kubernetes cluster
- 2) Set current namespace as default
- 3) We create deployment.yaml file that contains the logic of deploying our application. And using **kubectl apply -f** command deploy that deployment.

```

deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] ports > {} 0
io.k8s.api.apps.v1.Deployment (v1@deployment.json)
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: midterm-todo
5    labels:
6      app: midterm-todo
7      assingment: midterm
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       app: cloud-app-todo
13   template:
14     metadata:
15       labels:
16         app: cloud-app-todo
17     spec:
18       containers:
19         - name: midterm-todo-app
20           image: europe-west1-docker.pkg.dev/cloud-app-dev-yessimseit2/cloud-app-dev/to-do:1.0
21           ports:
22             - containerPort: 8080
23

```

- 4) For have access to our application on cluster we create service LoadBalancer. And using **kubectl apply -f** command deploy that service.

```
service.yaml > {} spec > type
io.k8s.api.core.v1.Service (v1@service.json)
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: midterm-todo-service
5    labels:
6      app: midterm-todo
7      assingment: midterm
8  spec:
9    selector:
10     app: cloud-app-todo
11    ports:
12     - port: 80
13       targetPort: 8080
14     type: LoadBalancer
```

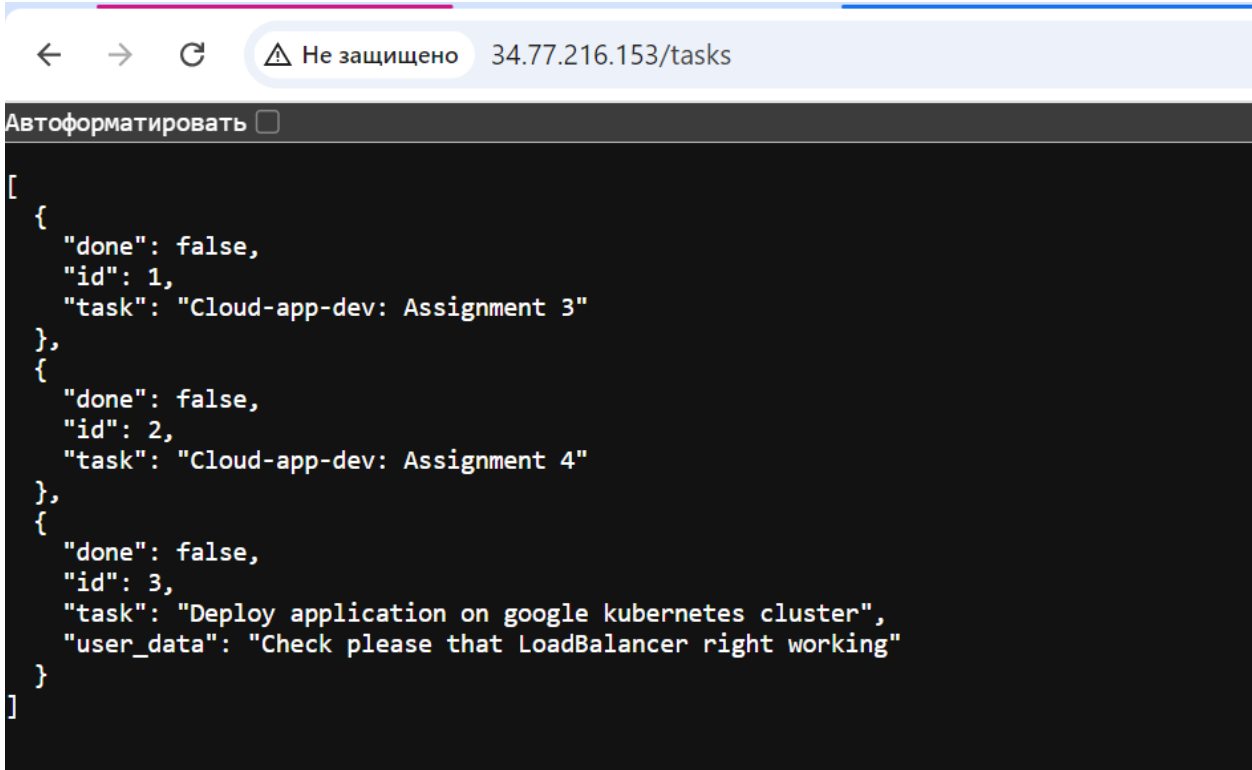
```
maes0624@ws-20432:~/k8e$ kaf /home/maes0624/gcloud/cloud-app-dev/midterm/service.yaml
service/midterm-todo-service created
maes0624@ws-20432:~/k8e$ kgs
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
midterm-todo-service LoadBalancer  34.118.232.98  <pending>      80:31204/TCP     9s
maes0624@ws-20432:~/k8e$ kgs
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
midterm-todo-service LoadBalancer  34.118.232.98  34.77.216.153  80:31204/TCP     60s
maes0624@ws-20432:~/k8e$
```

We have LoadBalancer Ip ADDRESS and try check working of our application.

```
← → ↺ ⚠ Не защищено 34.77.216.153/tasks
Автоформатировать ☐
[
  {
    "done": false,
    "id": 1,
    "task": "Cloud-app-dev: Assignment 3"
  },
  {
    "done": false,
    "id": 2,
    "task": "Cloud-app-dev: Assignment 4"
  }
]
```

To fully verify that our application is running in the kubernetes cluster, we check its operation using curl requests for our functions

```
maes0624@ws-20432:~$ curl -X POST http://34.77.216.153/tasks -H "Content-Type: application/json" -d '{"message": "check docker", "done": false, "id": 1, "task": "Cloud-app-dev: Assignment 3"}'
{"done": false, "id": 1, "task": "Cloud-app-dev: Assignment 3"}
maes0624@ws-20432:~$ curl -X POST http://34.77.216.153/tasks -H "Content-Type: application/json" -d '{"message": "check docker", "done": false, "id": 2, "task": "Cloud-app-dev: Assignment 4"}'
{"done": false, "id": 2, "task": "Cloud-app-dev: Assignment 4"}
maes0624@ws-20432:~$ curl -X POST http://34.77.216.153/send_notification -H "Content-Type: application/json" -d '{"message": "Application Working on Kubernetes cluster"}'
{"message": "Notification sent."}
maes0624@ws-20432:~$
```



Google Cloud cloud-app-dev-yessimselt2 Search (/) for resources, docs, products, and more

Cloud Run functions Function details EDIT DELETE COPY

process_user_data Cloud Run function (Deployed at Oct 14, 2024, 3:30:13 AM) URL: https://europe-west1-cloud-app-dev-yessimselt2.cloudfunctions.net/process_user_data View in Cloud Run process-user-data

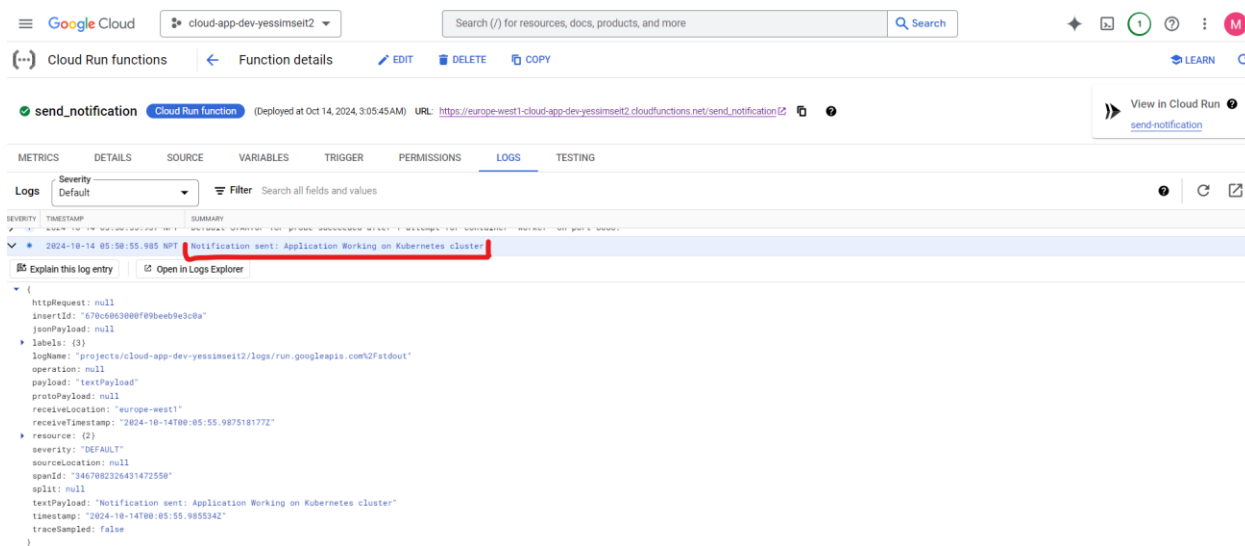
METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Logs Severity Default Filter Search all fields and values

SEVERITY	TIMESTAMP	SUMMARY
✓	2024-10-14 05:58:10.417 NPT	Received user data: {'task': 'Deploy application on google kubernetes cluster', 'done': False, 'user_data': 'Check please that LoadBalancer right working', 'id': 3}

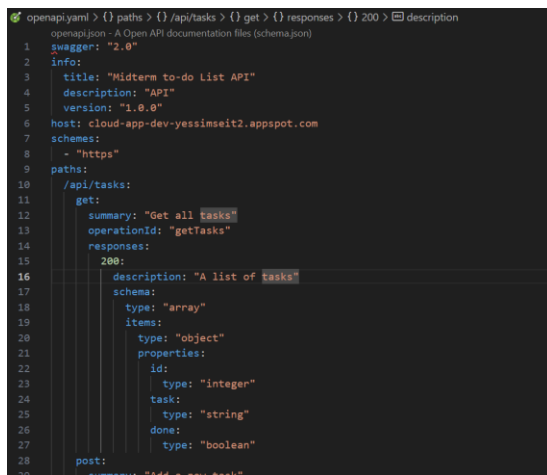
Explain this log entry Open in Logs Explorer

```
{
  httpRequest: null
  insertId: "e78c8e360805f80bcc9ef4"
  jsonPayload: null
  labels: {3}
  logName: "projects/cloud-app-dev-yessimselt2/logs/run.googleapis.com%2fstdout"
  operation: null
  payload: "textPayload"
  protoPayload: null
  receiveLocation: "europe-west1"
  receiveTimestamp: "2024-10-14T08:05:10.422578436Z"
  resource: {2}
  severity: "DEFAULT"
  sourceLocation: null
  spanId: "72693409113700356930"
  spanName: null
  textPayload: "Received user data: {'task': 'Deploy application on google kubernetes cluster', 'done': False, 'user_data': 'Check please that LoadBalancer right working', 'id': 3}"
  timestamp: "2024-10-14T08:05:10.417675Z"
  traceSampled: false
}
```



9. Managing APIs with Google Cloud Endpoints

- **API Setup:** Explain the process of setting up an API using Cloud Endpoints.
- **Security and Monitoring:** Describe how authentication and monitoring were implemented

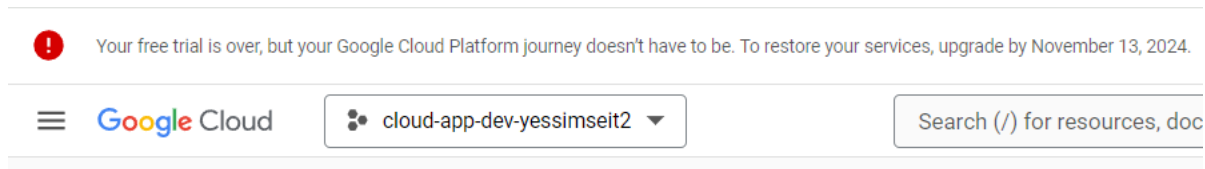


Firstly we need create Openapi.yaml conf file. Here we describe our api management methods, swagger, info about your API, host (choose your project), schemes with https for safety, after that you write your pathes (Endpoints) and describe them including method type, parameters, responses and security.

Then we need deploy api and app using next commands.

```
gcloud endpoints services deploy openapi.yaml.
gcloud app deploy
```

WARNING: At this point, my trial period ended, so I couldn't really continue and I'll have to explain the meaning just in words.



2) Security and Monitoring

For add security we need create api keys. Then restrict key to use only with certain API with command. Add new features into our application files. And deploy new changed application.

All used commands:

```
gcloud services api-keys create --display-name="to-do-api"
gcloud services api-keys list
gcloud api-keys update api_key_uid --api targets=service=endpoints.googleapis.com
gcloud endpoints services deploy openap1.yaml
gcloud app deploy
```

And we need add security into main.py files

```
85         type: "boolean"
86     delete:
87         summary: "Delete a task"
88         operationId: "deleteTask"
89         parameters:
90             - name: "task_id"
91               in: "path"
92               required: true
93               type: "integer"
94         responses:
95             200:
96                 description: "Task deleted"
97             404:
98                 description: "Todo not found"
99         security:
100             - api_key: []
101     securityDefinitions:
102         api_key:
103             type: apiKey
104             name: api-key
105             in: header
```

We need function that validate api key

```
def require_api_key(func):
    def wrapper(*args, **kwargs):
        api_key = request.headers.get('api-key')
        if api_key != API_KEY:
            return jsonify({"error": "Unauthorized: Invalid API Key"}), 401
        return func(*args, **kwargs)
```



```

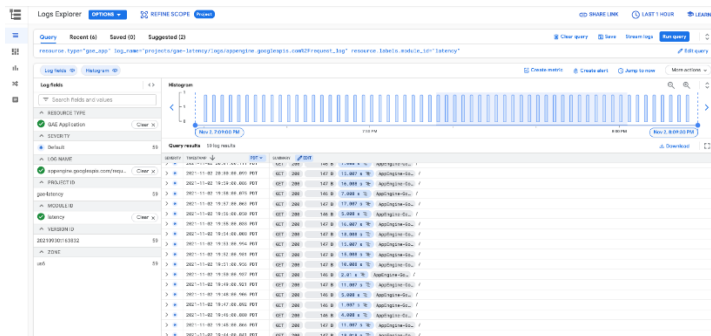
wrapper.__name__ = func.__name__
return wrapper

```

2) For implement monitoring we need go to console.google.com → Google Cloud Endpoints

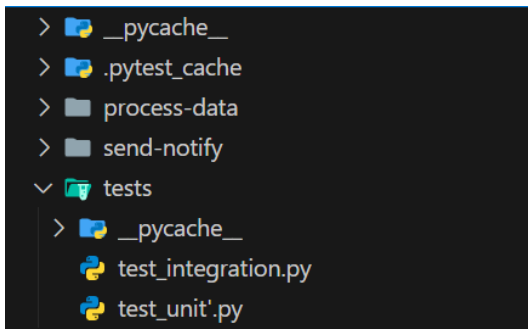
There you find you api and there will be monitoring in Logs explorer

Usually it seems like



10. Testing and Quality Assurance

Testing the operation of an application is a very important step in creating applications



We have created a unit and integration tests. Integration testing is a testing, where individual pieces of code are combined and tested in a group. It performs after unit testing

```

1 import unittest
2 import requests
3
4 class LocalAppIntegrationTestCase(unittest.TestCase):
5
6     BASE_URL = 'http://localhost:8080'
7
8     def test_full_task_lifecycle(self):
9
10         new_task = {"task": "Integration Test Task", "done": False}
11         response = requests.post(f'{self.BASE_URL}/tasks', json=new_task)
12         self.assertEqual(response.status_code, 201)
13         created_task = response.json()
14         self.assertIn("Integration Test Task", response.text)
15
16         task_id = created_task['id']
17
18         updated_task = {"task": "Updated Integration Test Task", "done": True}
19         response = requests.put(f'{self.BASE_URL}/tasks/{task_id}', json=updated_task)
20         self.assertEqual(response.status_code, 200)
21         self.assertIn("Updated Integration Test Task", response.text)
22
23         response = requests.delete(f'{self.BASE_URL}/tasks/{task_id}')
24         self.assertEqual(response.status_code, 200)
25         self.assertIn("Задача удалена", response.text)
26
27 if __name__ == '__main__':
28     unittest.main()
29

```

```

maes0624@WS-20432:~/gcloud/cloud-app-dev/midterm$ python -m unittest tests/test_unit\'.py
.....
Ran 6 tests in 0.031s

OK
maes0624@WS-20432:~/gcloud/cloud-app-dev/midterm$

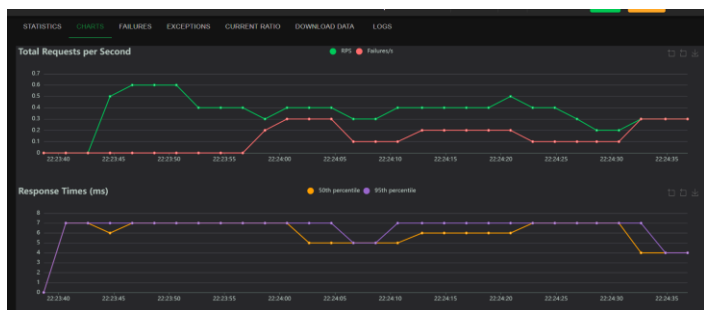
```

We also created load tests using the locust library. Then load testing was performed. Load testing is testing that is necessary to verify the functionality of the application. Using it, you can find out how many requests the system will be able to process in an n-th amount of time. Unfortunately, due to the fact that my application stopped working in Google cloud, the only way I could check it was through a local method.

```

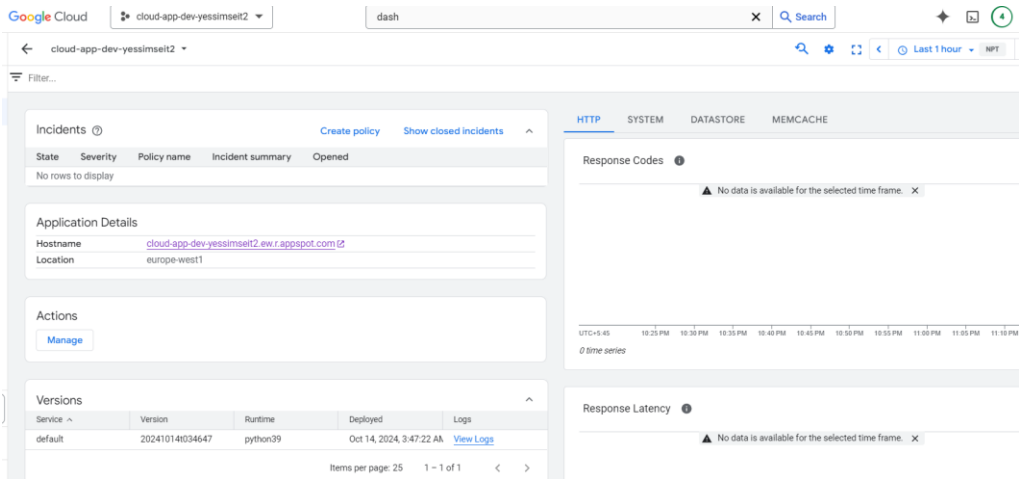
locust.py 1 WebsiteUser 2 delete_task
1 from locust import HttpUser, between, task
2
3 class WebsiteUser(HttpUser):
4     wait_time = between(1, 5)
5
6     @task
7     def get_tasks(self):
8         """simulate a user fetching tasks"""
9         self.client.get("/tasks")
10
11     @task
12     def add_task(self):
13         """simulate a user adding a new task"""
14         self.client.post("/tasks", json={"task": "Load Test Task", "done": False})
15
16     @task
17     def update_task(self):
18         """simulate a user updating a task"""
19         self.client.put("/tasks/1", json={"task": "Updated Load Test Task", "done": True})
20
21     @task
22     def delete_task(self):
23         """simulate a user deleting a task"""
24         self.client.delete("/tasks/1")
25

```



11. Monitoring and Maintenance

For monitoring we use Google cloud Monitoring. We creating dashboard with widgets that show logs of my application and endpoint and functions. **unfortunately, due to the fact that my trial period has passed, I was unable to make a dashboard with my web application.** I decided to add at least the dashboard of my project here. As you can see there are no indicators due to the fact that the application has stopped working

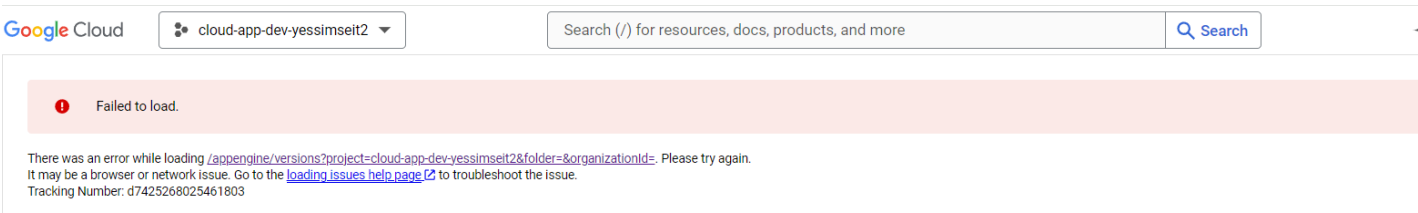


2)Maintenance :

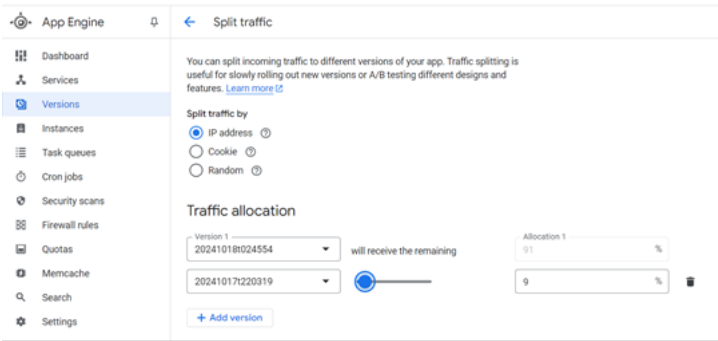
We can migrate from one version to another. We can enter App engine → Versions.

Here you will have the opportunity to manage the versions of our application. We could launch both the new and the old version at the same time. To maintain fault tolerance, we can distribute traffic between our versions, for example, between the new and old versions. And when we are sure that the new version is working, fully disable the old version.

For me it not working

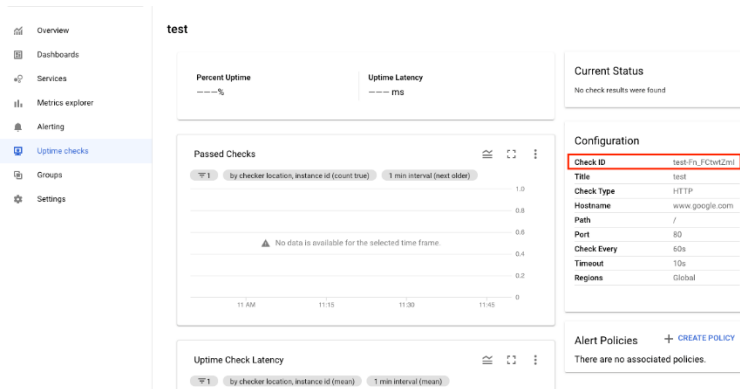


But for you it will be look like that



If we go Cloud monitoring → Uptime checks

We will get to the panel that will help you create a system for checking uptime. It checks the availability of our application in different regions. We pass the URL of the link to verify the operation of our application and our functions. We are setting up an alert system, if something does not work somewhere, or our system is at peak performance, the system will notify us about it. Thanks to this, we can check and be sure that our application is working correctly and quickly fix it at the right time.



12. Challenges and Solutions

The most important problem I faced was that in the middle of working on the project, I ended the free trial period since I started it back in the summer. Because of this, I could not fully work with all the services and tools of the cloud platform using a real example. And my web application and the kubernetes cluster turned off, respectively. Because of this, I could not check the work of monitoring and testing. I couldn't solve this problem, so as the teacher said, I just started describing the steps of the solution in words

The midterm itself was heavy, I had never deployed applications to Google cloud before. At first, it was difficult to create cloud run functions, it was unclear how to create them correctly and integrate them with your main application.

The second problem is the creation of API and endpoint functions. It was hard to set them up and try to deploy them.

13. Conclusion

In conclusion, I would like to say that it was a cool experience deploying a web application in Google cloud using all its services and tools. As a result, we got a fully working TO-DO web application that allows you to create a task list, create and add them, update, get a list and delete tasks. I have integrated functions through the cloud run functions service to send notifications and verify client data when adding new tasks. A very big achievement was the creation of the docker image and the launch of the kubernetes cluster. We also added authentication to our web application, which increased the security of the application. We

have tested the operation of our application through unit, integration, and load tests. We have added monitoring tools so that you can monitor the load and added an alert to quickly fix the situation if anything happens.

I believe that modern developers need to have experience working with cloud platforms. More and more companies are switching to using Google cloud platform services and tools.

14. References

1. Install the gcloud CLI
<https://cloud.google.com/sdk/docs/install>
2. Python 3 Runtime Environment. App Engine
<https://cloud.google.com/appengine/docs/standard/python3/runtime>
3. Create a Cloud Run function by using the Google Cloud CLI
<https://cloud.google.com/functions/docs/create-deploy-gcloud#functions-clone-sample-repository-python>
4. Docker Documentation
<https://www.docker.com/>
5. Google Kubernetes Engine (GKE) Documentation
<https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview>
6. Getting started with Cloud Endpoints for the App Engine flexible environment with ESP
https://cloud.google.com/endpoints/docs/openapi/get-started-app-engine?_gl=1*1xr3iqw*_up*MQ..&gclid=CjwKCAjw68K4BhAuEiwAylp3kiy63Cz_zsM4Em3dBWihFNK0a2or2xcU1JBWtDSnXlg5FrvpDf2ENhoCUCwQAvD_BwE&gclid=aw.ds

15. Appendices

I have described each of my steps in detail, so I think that you can not add anything else.