



School of Information Technology and Engineering

**Cloud Application Development
Midterm Project**

Title: Deploying a Scalable Web Application on Google Cloud Platform

Done by: Altair Kabdrakhmanov
4th year student, ID: 21B030829,
Information Systems

Almaty, Fall 2024

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

The project aimed to develop a scalable web application a phonebook ensuring high performance and security. To achieve this, I utilized various technologies, including Google Cloud Platform (GCP) for hosting and serverless computing, Flask as the backend web framework, Docker for containerization, openAPI for cloud API, etc. The phonebook functionality allows users to add, update, retrieve, and delete contact information seamlessly. Through unit and integration testing, I ensured the application is robust and reliable by load testing.

2. Introduction

Google Cloud Platform (GCP) is a service that helps people build and run web applications in the cloud. It is important for creating scalable applications because it has features like auto-scaling, which means it can automatically adjust resources to handle more users or traffic. People choose GCP for its cost-effectiveness since you only pay for what you use, and it allows for faster development and deployment of applications. GCP also provides strong security to protect data. Also, one of the motivation it was Free Trail 300\$ that helped me to accomplish this work.

3. Project Objectives

The main objective is to use serverless architecture. This approach allows us to build and run the application without managing servers, which makes it easier and more efficient. Another goal of the project is to create a responsive application that works well on different devices, including smartphones, tablets, and computers. This means that users will have a good experience, no matter how they access the application. Finally, I aim to ensure high availability. This means that the application should always be accessible, so users can rely on it whenever they need it. By achieving these objectives, I want to deliver a smooth and reliable service that meets the needs of all users.

4. Google Cloud Platform Overview

Google Cloud Platform (GCP) has a strong architecture that helps developers create and deploy applications easily. The core services of GCP include computing, storage, and databases. For

computing, GCP offers services like Google Compute Engine, which lets you run virtual machines, and Google App Engine, which allows you to build and host applications without managing the servers. For storage, you can use Google Cloud Storage to save files and Google Cloud SQL for managing databases.

One big benefit of using GCP is its scalability. This means that you can easily grow your application as more users start to use it. GCP also provides high security, ensuring that your data and applications are safe. Additionally, GCP offers tools for monitoring and managing your applications, making it easier to keep everything running smoothly

5. Google Cloud SDK and Cloud Shell

Setup

To start working on Google Cloud you need to download and install Google Cloud SDK. Do the followings to have Google Cloud SDK on your machine:

- Go to this page “Install the gcloud CLI” - <https://cloud.google.com/sdk/docs/install>

These instructions are for installing the Google Cloud CLI. For information about installing additional components, such as gcloud CLI commands at the alpha or beta release level, see [Managing gcloud CLI components](#).

★ Note: If you are behind a proxy or firewall, see the [proxy settings](#) page for more information on installation.

Platform	Package	Size	SHA256 Checksum
macOS 64-bit	google-cloud-cli-darwin- Apple Silicon	53.6 MB	d5cf5912a84a1ef13417760282ebe9653851df ----- -----

- Select your operating machine. For my case it is MacOS
- Check you current version of Python. It must match with requirements for Google Cloud SDK
- Download your package. For my case it is Apple Silicon
- Extract this archive file on Home directory.
- To add Google Cloud SDK to your PATH. You need to use this command in root of folder where you extracted the archive. For my case it is HOME directory. The command: `./google-cloud-sdk/install.sh`

- Now you can run the `gcloud auth login` command and do authentication with your google cloud.

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
● (venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud auth login
Your browser has been opened to visit:
https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo_email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine_admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts_reauth&state=H1Kwa9BvDKMqldizisiPvff6RqBm&access_type=offline&code_challenge=8tfDRAf1hUxR9PmM8Pgh9Qm2ZhQ_zbFVX0E279ucCc&code_challenge_method=S256

You are now logged in as [altairkabdrakhmanov@gmail.com].
Your current project is [None]. You can change this setting by running:
$ gcloud config set project PROJECT_ID

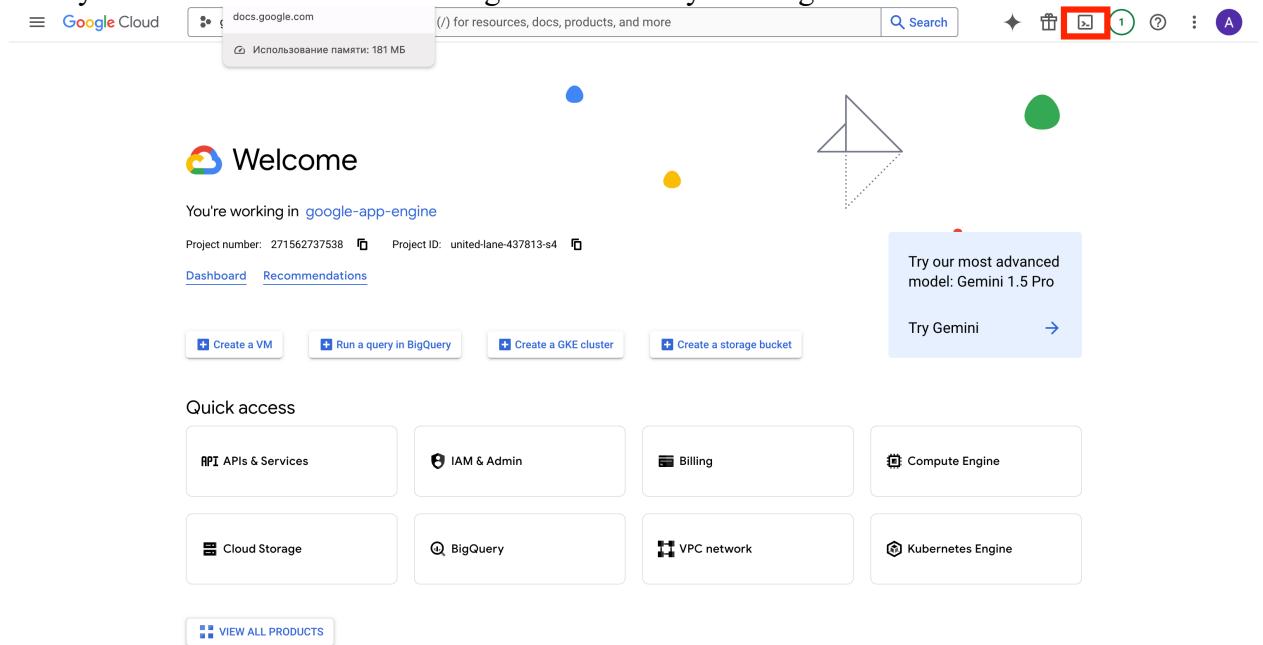
Updates are available for some Google Cloud CLI components. To install them,
please run:
$ gcloud components update
○ (venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm %

```

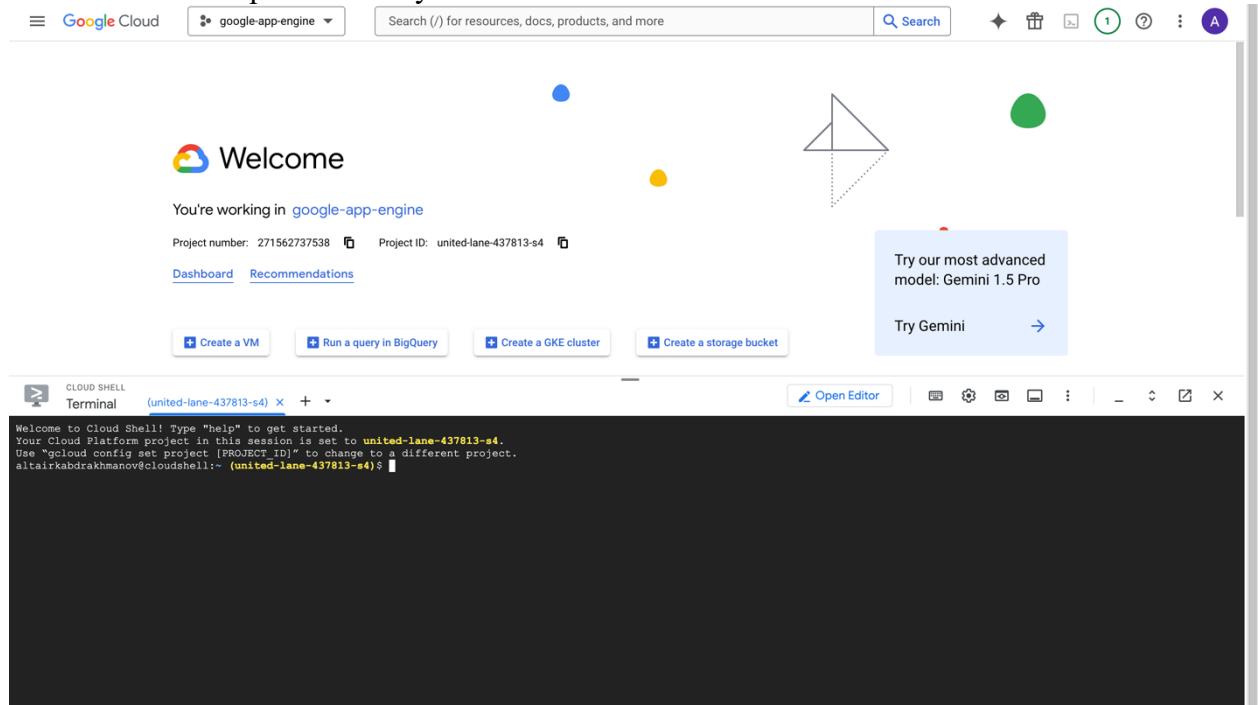
Ln 56, Col 51 Spaces: 4 UTF-8 LF Python ⌂ Go Live ⌂

Cloud Shell Usage

Also you can utilize Cloud Shell Usage that is already in Google Cloud Shell.



Click on this shell preview and you will obtain



So it has the same functionality as Google Cloud SDK but it is within Google Cloud Console.

6. Google App Engine Application Development

I created phonebook application with CRUD operations with using Flask python framework. Main endpoints: /contacts (get contacts, create contact), /contacts/{id} (update, create customers).

The code:

```
main.py
1  from flask import Flask, request, jsonify
2
3  app = Flask(__name__)
4
5  # In-memory phonebook and ID counter
6  phonebook = {}
7  next_contact_id = 1 # Start with ID 1
8
9  # Create a new contact
10 @app.route('/contacts', methods=['POST'])
11 def create_contact():
12     global next_contact_id
13     contact_data = request.json
14     contact = {
15         'id': next_contact_id,
16         'name': contact_data['name'],
17         'phone': contact_data['phone']
18     }
19     phonebook[next_contact_id] = contact
20     next_contact_id += 1 # Increment the ID for the next contact
21     return jsonify(contact), 201
22
23 # Read all contacts
24 @app.route('/contacts', methods=['GET'])
25 def get_contacts():
26     return jsonify(list(phonebook.values())), 200
27
28 # Read a specific contact by ID
29 @app.route('/contacts/<int:contact_id>', methods=['GET'])
30 def get_contact(contact_id):
31     contact = phonebook.get(contact_id)
32     if not contact:
33         return jsonify({'message': 'Contact not found'}), 404
34     return jsonify(contact), 200
35
36 # Update a contact
37 @app.route('/contacts/<int:contact_id>', methods=['PUT'])
38 def update_contact(contact_id):
39     contact = phonebook.get(contact_id)
40     if not contact:
41         return jsonify({'message': 'Contact not found'}), 404
42     updated_data = request.json
43     contact['name'] = updated_data.get('name', contact['name'])
44     contact['phone'] = updated_data.get('phone', contact['phone'])
45     return jsonify(contact), 200
46
47 # Delete a contact
48 @app.route('/contacts/<int:contact_id>', methods=['DELETE'])
49 def delete_contact(contact_id):
50     contact = phonebook.pop(contact_id, None)
51     if not contact:
52         return jsonify({'message': 'Contact not found'}), 404
53     return jsonify({'message': 'Contact deleted'}), 200
54
55 if __name__ == '__main__':
56     app.run(host='127.0.0.1', port=8080, debug=True)
```

Deployment

To deploy a phonebook application to google cloud engine. We need to do the followings:

- Create project in Google Cloud
- Activate Billing
- Activate Google App Engine

Create Project

Used this command to create my project: `gcloud projects create --name="CloudAppDev-Midterm-AK"`

```
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
● (venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud projects create --name="CloudAppDev-Midterm-AK"
No project ID provided.

Use [cloudappdev-midterm] Follow link (cmd + click) ID (Y/n)? y
Create in progress for [https://clouddre...com/v1/projects/cloudappdev-midterm-ak-439206].
Waiting for [operations/cp.6988521966428639315] to finish...done.
Enabling service [cloudapis.googleapis.com] on project [cloudappdev-midterm-ak-439206]...
Operation "operations/acat.p2-179348553544-a44e1bd2-f2e7-44bf-a039-e0ac92a9e47d" finished successfully.
○ (venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

Set Project

Used this command to set as working project `gcloud config set project cloudappdev-midterm-ak-439206`

Activate Billing Account

Go to google cloud console. Select “Billing”.

The screenshot shows the Google Cloud Welcome page for the project 'CloudAppDev-Midterm-AK'. At the top left, the project name is displayed with a dropdown arrow, which is highlighted with a red box. Below the header, there's a 'Welcome' section with a 'Cloud' icon. A message says 'You're working in CloudAppDev-Midterm-AK'. Underneath, it shows 'Project number: 179348553544' and 'Project ID: cloudappdev-midterm-ak-439206'. There are two navigation links: 'Dashboard' and 'Recommendations'. Below these are four buttons: 'Create a VM', 'Run a query in BigQuery', 'Create a GKE cluster', and 'Create a storage bucket'. To the right, there's a callout box with the text 'Try our most advanced model: Gemini 1.5 Pro' and a 'Try Gemini' button. In the 'Quick access' section, there are eight cards: 'API APIs & Services', 'IAM & Admin', 'Billing' (which is also highlighted with a red box), 'Compute Engine', 'Cloud Storage', 'BigQuery', 'VPC network', and 'Kubernetes Engine'. At the bottom left of this section is a 'VIEW ALL PRODUCTS' button.

Press the “Link Billing Account” and choose you billing.
Now your project ready for starting the deployment on Google Cloud.

Deploy Phonebook application

Activate Cloud Engine: [gcloud services enable compute.googleapis.com](https://cloud.google.com/compute/docs/quickstart-linux)

Go to google cloud console and create VM Instance with e2-micro.

The screenshot shows the 'Create an instance' dialog for a 'New VM instance'. On the left, there's a sidebar with options: 'New VM instance' (selected), 'New VM instance from template', 'New VM instance from machine image', and 'Marketplace'. The main area has a 'TRY NOW' button at the top. Below it, there are several configuration sections: 'Identity and API access' (with a 'Service accounts' dropdown set to 'Default compute service account'), 'Access scopes' (radio buttons for 'Allow default access', 'Allow full access to all Cloud APIs' (selected), and 'Set access for each API'), 'Firewall' (checkboxes for 'Allow HTTP traffic', 'Allow HTTPS traffic', and 'Allow Load Balancer Health Checks' (all selected)), 'Observability - Ops Agent' (checkbox for 'Install Ops Agent for Monitoring and Logging' (selected)), and 'Advanced options' (a collapsed section). On the right, there's a 'Monthly estimate' table:

Item	Monthly estimate
2 vCPU + 1 GB memory	\$6.11
10 GB balanced persistent disk	\$1.00
Logging	Cost varies
Monitoring	Cost varies
Total	\$7.11

At the bottom, there are 'CREATE', 'CANCEL', and 'EQUIVALENT CODE' buttons.

To deploy our application within Google Cloud Engine we need to create app.yaml file prior deploying. The app.yaml looks:

```
app.yaml
```

```
1 runtime: python39
2 handlers:
3   - url: /.*
4     script: auto
5
```

Used command to create requirements.txt: `pip freeze > requirements.txt`

```
requirements.txt
```

```
1 blinker==1.8.2
2 click==8.1.7
3 Flask==3.0.3
4 itsdangerous==2.2.0
5 Jinja2==3.1.4
6 MarkupSafe==3.0.2
7 Werkzeug==3.0.4
8
```

Used command to deploy: `gcloud app deploy`

```
(base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud app deploy
Services to deploy:
descriptor:          [/Users/kabdrakhman/Documents/KBTU/7th semester/Cloud-Application-Development/Midterm/app.yaml]
source:              [/Users/kabdrakhman/Documents/KBTU/7th semester/Cloud-Application-Development/Midterm]
target project:      [clouddappdev-midterm-ak-439206]
target service:      [default]
target version:      [20241020t113342]
target url:          [https://clouddappdev-midterm-ak-439206.df.r.appspot.com]
target service account: [clouddappdev-midterm-ak-439206@appspot.gserviceaccount.com]

Do you want to continue (Y/n)? y
Beginning deployment of service [default]...
[ Uploading 1 file to Google Cloud Storage ] =
```

File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://clouddappdev-midterm-ak-439206.df.r.appspot.com]

You can stream logs from the command line by running:
\$ gcloud app logs tail -s default

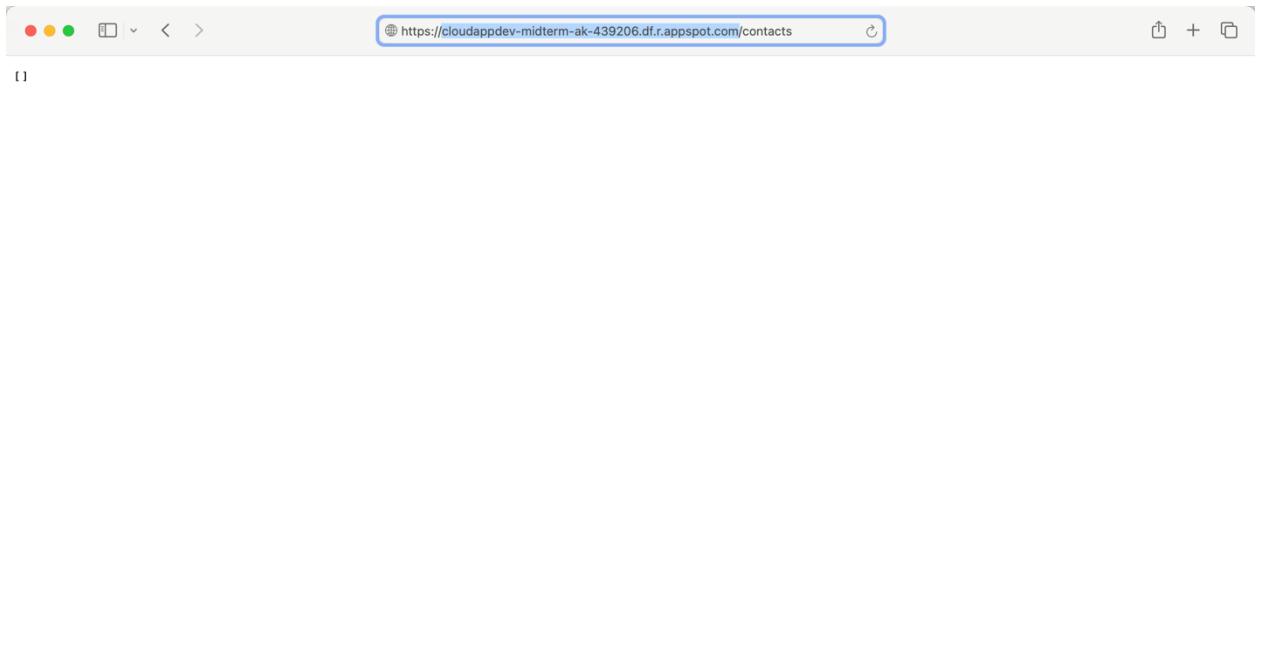
To view your application in the web browser run:
\$ gcloud app browse

```
(base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

Ln 4, Col 1 Span

Checking the result

Go to page - <https://clouddappdev-midterm-ak-439206.df.r.appspot.com/contacts>



7. Building with Google Cloud Functions

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

1. Input Validation Cloud Function

Code:

```
input_validation > ls input_validation.js > ...
1 // Define the input validation function
2 exports.inputValidation = (req, res) => {
3   // Check if content type is JSON
4   if (req.get('content-type') !== 'application/json') {
5     return res.status(415).json({ error: 'Unsupported Media Type' });
6   }
7
8   const requestBody = req.body;
9
10  // Validate the presence of 'name' and 'phone' in the request body
11  if (!requestBody.name && !requestBody.phone) {
12    return res.status(400).json({ error: 'Name and phone not provided!' });
13  }
14
15  if (!requestBody.name) {
16    return res.status(400).json({ error: 'Name not provided!' });
17  }
18
19  if (!requestBody.phone) {
20    return res.status(400).json({ error: 'Phone not provided!' });
21  }
22
23  // Input data are valid any
24  return res.status(200).json({ message: 'Input data are validated.' });
25};
26
```

```
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS zsh - input_validation + ~ □ ⌂ ⌂ ... ^ ×
ingressSettings: ALLOW_ALL
maxInstanceCount: 100
maxInstanceRequestConcurrency: 1
revision: inputvalidation-00001-goc
service: projects/clouddappdev-midterm-ak-439206/locations/us-central1/services/inputvalidation
serviceAccountEmail: 17934853544-compute@developer.gserviceaccount.com
timeoutSeconds: 60
uri: https://inputvalidation-zqyj4gtxea-uc.a.run.app
state: ACTIVE
updateTime: '2024-10-20T07:39:16.165727399Z'
url: https://us-central1-clouddappdev-midterm-ak-439206.cloudfunctions.net/inputValidation
Ln 26, Col 1 Spaces: 2 UTF-8 LF {} JavaScript ⌂ Go Live ⌂
```

Command:

```
gcloud functions deploy inputValidation \
--runtime nodejs18 \
--trigger-http \
--allow-unauthenticated
```

Result:

```
(base) kabdrakhman@MacBook-Pro-Altair input_validation % gcloud functions deploy inputValidation \
--runtime nodejs18 \
--trigger-http \
--allow-unauthenticated

As of this Cloud SDK release, new functions will be deployed as 2nd gen  functions by default. This is equivalent to currently deploying new with the --gen2 flag. Existing 1st gen functions will not be impacted and will continue to deploy as 1st gen functions.
You can disable this behavior by explicitly specifying the --gen2 flag or by setting the functions/gen2 config property to 'off'.
To learn more about the differences between 1st gen and 2nd gen functions, visit:
https://cloud.google.com/functions/docs/concepts/version-comparison
Preparing function...done.
X Deploying function...
[Build] Logs are available at [https://console.cloud.google.com/cloud-build/builds;region=us-central1/296ccbf4-7b38-4e99-9bc7-5d1904449dd6?project=1793485535]
44
  / [Service]
  . [ArtifactRegistry]
  . [Healthcheck]
  . [Triggercheck]
Completed with warnings:
[WARNING] *** Improve build performance by generating and committing package-lock.json.
You can view your function in the Cloud Console here: https://console.cloud.google.com/functions/details/us-central1/inputValidation?project=clouappdev-midterm-ak-439206

buildConfig:
  automaticUpdatePolicy: {}
  build: projects/179348553544/locations/us-central1/builds/296ccbf4-7b38-4e99-9bc7-5d1904449dd6
  dockerRegistry: ARTIFACT_REGISTRY
  dockerRepository: projects/clouappdev-midterm-ak-439206/locations/us-central1/repositories/gcf-artifacts
  entryPoint: inputValidation
  runtime: nodejs18
  serviceAccount: projects/clouappdev-midterm-ak-439206/serviceAccounts/179348553544-compute@developer.gserviceaccount.com
  source:
    storageSource:
      bucket: gcf-v2-sources-179348553544-us-central1
      generation: '1729409914892412'
    object: inputValidation/function-source.zip
  sourceProvenance:
    resolvedStorageSource:
      bucket: gcf-v2-sources-179348553544-us-central1
      generation: '1729409914892412'
    object: inputValidation/function-source.zip
  createTime: '2024-10-20T07:38:35.078Z109832'
  environment: GEN_2
  labels:
    deployment-tool: cli-gcloud
  name: projects/clouappdev-midterm-ak-439206/locations/us-central1/functions/inputValidation
  serviceConfig:
    allTrafficLocatesRevision: true
    availableMemory: '0.166G'
    availableMemory: 256M
    environmentVariables:
      LOG_EXECUTION_ID: 'true'
    ingressSettings: ALLOW_ALL
    maxInstanceCount: 1
    maxPendingRequestConcurrency: 1
    revision: inputValidation-00001-goc
    service: projects/clouappdev-midterm-ak-439206/locations/us-central1/services/inputvalidation
    serviceAccountEmail: 179348553544-compute@developer.gserviceaccount.com
    timeoutSeconds: 60
  uri: https://inputValidation-zqyj4gfxea-uc.a.run.app
  state: DEPLOYED
  updateTime: '2024-10-20T07:39:16.165Z727390Z'
  url: https://us-central1-clouappdev-midterm-ak-439206.cloudfunctions.net/inputValidation
(base) kabdrakhman@MacBook-Pro-Altair input_validation %
```

Integration with input_validation cloud function with phonebook application:

```
python main.py
 1  from flask import Flask, request, jsonify
 2  import requests
 3
 4  app = Flask(__name__)
 5
 6  # In-memory phonebook and ID counter
 7  phonebook = {}
 8  next_contact_id = 1 # Start with ID 1
 9
10 # URL for the Cloud Function (Follow link (cmd + click) to your actual Cloud Function URL)
11 CLOUD_FUNCTION_URL = 'https://inputvalidation-zqyj4gfxea-uc.a.run.app'
12
13 # Function to validate the request using the Cloud Function
14 def validate_request(contact_data):
15     try:
16         response = requests.post(
17             CLOUD_FUNCTION_URL,
18             json=contact_data,
19             headers={'Content-Type': 'application/json'}
20         )
21         # Return the JSON response and status code
22         return response.json(), response.status_code
23     except requests.RequestException as e:
24         return {'error': 'Failed to connect to validation service'}, 500
25
26 # Create a new contact
27 @app.route('/contacts', methods=['POST'])
28 def create_contact():
29     global next_contact_id
30     contact_data = request.json
31
32     # Validate the request data using the Cloud Function
33     validation_result, status_code = validate_request(contact_data)
34     if status_code != 200:
35         # If validation fails, return the validation error response
36         return jsonify(validation_result), status_code
37
38     # Proceed to create the contact if validation succeeds
39     contact = {
40         'id': next_contact_id,
41         'name': contact_data['name'],
42         'phone': contact_data['phone']
43     }
44     phonebook[next_contact_id] = contact
45     next_contact_id += 1 # Increment the ID for the next contact
46     return jsonify(contact), 201
47
```

Result:

HTTP Cloud-App / Midterm / **CreateCustomer**

POST http://{{BASE_URL}}/contacts

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {  
2   "name": "$randomFullName"  
3 }
```

Body Cookies Headers (5) Test Results Status: 400

Pretty Raw Preview Visualize **JSON**

```
1 {  
2   "error": "Phone not provided!"  
3 }
```

Send notification cloud function

Function log_event for adding logging when doing action CRUD

```

9
10 # URL for the Cloud Function (replace with your actual Cloud Function URL)
11 CLOUD_INPUT_VALIDATION = 'https://inputvalidation-zqyj4gfxea-uc.a.run.app'
12 CLOUD_FUNCTION_LOGGING_URL = 'https://logevents-zqyj4gfxea-uc.a.run.app'
13
14 # Function to validate the request using the Cloud Function
15 def validate_request(contact_data):
16     try:
17         response = requests.post(
18             CLOUD_INPUT_VALIDATION,
19             json=contact_data,
20             headers={'Content-Type': 'application/json'}
21         )
22         # Return the JSON response and status code
23         return response.json(), response.status_code
24     except requests.RequestException as e:
25         return {'error': 'Failed to connect to validation service'}, 500
26
27 # Function to log events using the Cloud Function
28 def log_event(contact_data, method):
29     try:
30         response = requests.post(
31             CLOUD_FUNCTION_LOGGING_URL,
32             json={
33                 'name': contact_data.get('name'),
34                 'phone': contact_data.get('phone'),
35                 'method': method # Include the HTTP method to indicate the action
36             },
37             headers={'Content-Type': 'application/json'}
38         )
39         if response.status_code != 200:
40             print(f"Failed to log {method} action.")
41         else:
42             print(f"{method} action logged successfully.")
43     except requests.RequestException:
44         print("Failed to connect to logging service.")
45

```

Create a new contact with logging

```

46 # Create a new contact
47 @app.route('/contacts', methods=['POST'])
48 def create_contact():
49     global next_contact_id
50     contact_data = request.json
51
52     # Validate the request data using the Cloud Function
53     validation_result, status_code = validate_request(contact_data)
54     if status_code != 200:
55         # If validation fails, return the validation error response
56         return jsonify(validation_result), status_code
57
58     # Proceed to create the contact if validation succeeds
59     contact = {
60         'id': next_contact_id,
61         'name': contact_data['name'],
62         'phone': contact_data['phone']
63     }
64     phonebook[next_contact_id] = contact
65     next_contact_id += 1 # Increment the ID for the next contact
66
67     # Log the creation event
68     log_event(contact_data, 'POST')
69
70     return jsonify(contact), 201
71

```

Update contact with logging

```
85  # Update a contact
86  @app.route('/contacts/<int:contact_id>', methods=['PUT'])
87  def update_contact(contact_id):
88      contact = phonebook.get(contact_id)
89      if not contact:
90          return jsonify({'message': 'Contact not found'}), 404
91      updated_data = request.json
92      contact['name'] = updated_data.get('name', contact['name'])
93      contact['phone'] = updated_data.get('phone', contact['phone'])
94
95      # Log the creation event
96      log_event(updated_data, 'PUT')
97
98  return jsonify(contact), 200
```

Delete contact with logging

```
100 # Delete task from To-Do List
101 @app.route('/delete/<string:todo_id>', methods=['DELETE'])
102 @require_api_key
103 def delete_todo(todo_id):
104     global todos
105     todo_to_delete = next((todo for todo in todos if todo['id'] == todo_id), None)
106
107     if not todo_to_delete:
108         return jsonify({"error": "Todo not found."}), 404
109     else:
110         requests.post(
111             SEND_NOTIFICATION_URL,
112             headers={'Content-Type': 'application/json'},
113             json={
114                 "todo": todo_to_delete,
115                 "action": "deleted"
116             }
117         )
118
119     print(f"Deleting todo with id: {todo_id}")
120     todos = [todo for todo in todos if todo['id'] != todo_id]
121     return jsonify({"message": "Todo deleted successfully"})
122
```

Result

The screenshot shows the Google Cloud Platform interface for a Cloud Run function named "logEvents". The logs tab is selected, displaying a list of log entries. A red box highlights the last four entries, which correspond to the API calls shown in the code snippets above:

- POST 200 224 B 3 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run.app/
- PUT 500 176 B 88 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run.app/
- PUT 500 176 B 4 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- PUT 500 176 B 3 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- DELETE 200 224 B 3 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run.app/
- DELETE 200 224 B 5 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- DELETE 200 224 B 151 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- PUT 200 227 B 5 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- PUT 200 227 B 3 ns PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- DELETE 200 224 B 3 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- DELETE 200 224 B 5 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/
- DELETE 200 224 B 1776 ms PostmanRuntime/7.36.1 https://logevents-zqyj4gfxea-uc.a.run/app/

8. Containerizing Applications

Working with Dockerfile

With step that need to be taken is to create Dockerfile for building image and further push to google cloud registry

The terminal window shows the Dockerfile content on the left and its execution log on the right. The log shows the Docker daemon starting a container, running gunicorn, and finally exiting.

```
#!/bin/bash
# Use the official Python image as a base
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the requirements file into the container
COPY requirements.txt .

# Install the required packages
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code into the container
COPY main.py .

# Expose the port the app runs on
EXPOSE 8080

# Command to run the application using gunicorn
CMD ["gunicorn", "--bind", "0.0.0.0:8080", "main:app"]
```

(base) kabdrakhman@MacBook-Pro-Altair Midterm % docker run -p 8080:8080 phonebook-app

[2024-10-20 10:51:00 +0000] [1] [INFO] Starting gunicorn 20.1.0

[2024-10-20 10:51:00 +0000] [1] [INFO] Listening at: http://0.0.0.0:8080 (1)

[2024-10-20 10:51:00 +0000] [1] [INFO] Using worker: sync

[2024-10-20 10:51:00 +0000] [7] [INFO] Booting worker with pid: 7

[2024-10-20 10:52:31 +0000] [1] [INFO] Handling signal: int

[2024-10-20 10:52:31 +0000] [7] [INFO] Worker exiting (pid: 7)

POST action logged successfully.

POST action logged successfully.

POST action logged successfully.

POST action logged successfully.

PUT action logged successfully.

[2024-10-20 10:52:31 +0000] [1] [INFO] Shutting down: Master

(base) kabdrakhman@MacBook-Pro-Altair Midterm % docker run -p 8080:8080 phonebook-app

[2024-10-20 10:52:43 +0000] [1] [INFO] Starting gunicorn 20.1.0

[2024-10-20 10:52:43 +0000] [1] [INFO] Listening at: http://0.0.0.0:8080 (1)

[2024-10-20 10:52:43 +0000] [1] [INFO] Using worker: sync

[2024-10-20 10:52:43 +0000] [7] [INFO] Booting worker with pid: 7

[2024-10-20 10:53:22 +0000] [1] [INFO] Handling signal: term

POST action logged successfully.

POST action logged successfully.

POST action logged successfully.

POST action logged successfully.

Command to build image with tag phonebook-app `docker build -t phonebook-app .`

The terminal window shows the `docker build` command output. It lists various layers being built, their sizes, and the total size of the image.

```
(base) kabdrakhman@MacBook-Pro-Altair Midterm % docker build -t phonebook-app .
```

Layer	Size
[+/-] Building 92.83 MB (9/9)	0.05
>= [internal] load build definition from Dockerfile	0.05
>> transferring dockerfile: 259B	0.05
>= [internal] load .dockerrcignore	0.05
>> transferring context: 2B	4.95
>= [internal] load metadata for docker.io/library/python:3.9@sha256:ed8b9dd4e9f89c111f4bdb85a5ff8cf9fe22796a298449380b15f627d9914095	70.55
>> resolve docker.io/library/python:3.9@sha256:ed8b9dd4e9f89c111f4bdb85a5ff8cf9fe22796a298449380b15f627d9914095	0.05
>> sha256:ab47617e2fa0@eccda877994317c17d2656c57fd45508e594c73a423b6a36a7f 2.33KB / 2.33KB	0.05
>> sha256:427912ae15dffabdd248ce07be4646e46bbef5d494fe6d66960ac0442305e160 6.31KB / 6.31KB	0.05
>> sha256:cb5594266b1ba9f9ad6855b00d9c5c98e721001eb15218eda673e548e04fd8f 64.39MB / 64.35MB	36.25
>> sha256:ed8b9dd4e9f89c111f4bdb85a5ff8cf9fe22796a298449380b15f627d9914095 10.35KB / 10.35KB	0.05
>> sha256:c1e0e7fb956a07c7b90256aa16ccb0550a34d0625d1d23c5b1a76e92a58d01e 49.58MB / 49.58MB	21.55
>> sha256:95b894d63c771a6056bc65ff25192b251413259b07c7b90256aa16ccb0550a34d0625d1d23c5b1a76e92a58d01e 23.59MB / 23.59MB	9.85
>> sha256:59d4884f85282b9a352dbcfd2cc0d73a6e0b151be84375ce9279de1c553 202.64MB / 202.64MB	65.75
>> extracting sha256:c1e0e7fb956a07c7b90256aa16ccb0550a34d0625d1d23c5b1a76e92a58d01e	1.45
>> sha256:b9c5056b1dd6ab028f8244043ab688985cb313087a4ca1694b989fbbe0dc320 6.24MB / 6.24MB	26.05
>> extracting sha256:95b894d63c771a6056bc65ff25192b251413259b07d160b0076f0f5d7975dc39	0.45
>> sha256:4c747ee882ad97a20f5ea77851097f46c057338955788ea76347820a4a0f149 19.27MB / 19.27MB	34.85
>> sha256:27f16a1c33fd8a521de4f9d96ea4bb028f8244043ab688985cb313087a4ca1694b989fbbe0dc320	36.05
>> extracting sha256:c5b594266b1ba9f9ad6855b00d9c5c98e721001eb15218eda673e548e04fd8f	1.75
>> extracting sha256:59d4884f85282b9a352dbcfd2cc0d73a6e0b151be84375ce9279de1c553	4.05
>> extracting sha256:b9c5056b1dd6ab028f8244043ab688985cb313087a4ca1694b989fbbe0dc320	0.25
>> extracting sha256:4c747ee882ad97a20f5ea77851097f46c057338955788ea76347820a4a0f149	0.55
>> extracting sha256:27f16a1c33fd8a521de4f9d96ea4bb028f8244043ab688985cb313087a4ca1694b989fbbe0dc320	0.05
> [internal] load build context	0.35
>> transferring context: 22.67MB	0.35
>> [2/4] WORKDIR /app	0.55
>> [3/4] COPY . .	0.15
>> [4/4] RUN pip install --no-cache-dir Flask requests gunicorn Werkzeug MarkupSafe Jinja2 itsdangerous click blinker	16.65
>> exporting to image	0.15
>> exporting layers	0.15
>> writing image sha256:0a4fa72ad94aa759e7de8355eb3957e7abcb14e7c479d58015c4b746abbfeaf6	0.05
>> naming to docker.io/library/phonebook-app	0.05

Prior pushing image to GCR, I tested locally with this command: `docker run -p 8080:8080 phonebook-app`

Pushing image to GCR

To have permission to add images into your remote docker registry on google cloud you need to execute this command:

```
gcloud auth configure-docker
```

To push my docker images I used this command to tag phonebook with my google cloud registry (GCR):

```
docker tag phonebook-app gcr.io/cloudappdev-midterm-ak-439206/phonebook-app:latest
```

Only then I could push image to GCR

```
docker push gcr.io/cloudappdev-midterm-ak-439206/phonebook-app:latest
```

Working with K8S

To create kubernetes cluster used code:

```
gcloud container clusters create midterm-cluster \
```

```
--zone us-central1-a \
```

```
--num-nodes 1
```

```
● (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud container clusters create midterm-cluster \
  --zone us-central1-a \
  --num-nodes 1
  Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass
the '--no-enable-ip-alias' flag
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migration instructions.
Note: Your Pod address range ('--cluster-ip-v4-cidr') can accommodate at most 1008 node(s).
Creating cluster midterm-cluster in us-central1-a... Cluster is being health-checked [Kubernetes Control Plane is healthy]...done.
Created [https://container.googleapis.com/v1/projects/cloudappdev-midterm-ak-439206/zones/us-central1-a/clusters/midterm-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/gcloud/us-central1-a/midterm-cluster?project=cloudappdev-midterm-ak-439206
CRITICAL: ACTION REQUIRED: gke-gcloud-auth-plugin, which is needed for continued use of kubectl, was not found or is not executable. Install gke-gcloud-auth-plugin for
use with kubectl by following https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl#install_plugin
kubeconfig entry generated for midterm-cluster.
NAME          LOCATION      MASTER_VERSION    MASTER_IP      MACHINE_TYPE   NODE_VERSION     NUM_NODES  STATUS
midterm-cluster  us-central1-a  1.30.5-gke.1014001  34.136.50.77  e2-medium       1.30.5-gke.1014001  1          RUNNING
○ (base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

To install kubectl

```
gcloud components install gke-gcloud-auth-plugin
```

Command to enable cluster in kubectl config file

```
gcloud container clusters get-credentials midterm-cluster --zone us-central1-a
```

Result

```
No resources found in default namespace.
● (base) kabdrakhman@MacBook-Pro-Altair Midterm % kubectl get namespaces
  NAME          STATUS  AGE
  default        Active  5m38s
  gke-managed-cim  Active  5m14s
  gke-managed-system  Active  5m2s
  gmp-public      Active  4m47s
  gmp-system       Active  4m48s
  kube-node-lease  Active  5m38s
  kube-public       Active  5m38s
  kube-system       Active  5m38s
○ (base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

Create deployment and service yaml file with our application

Create separate namespace for our application

```
kubectl create namespace phonebook
```

9. Managing APIs with Google Cloud Endpoints

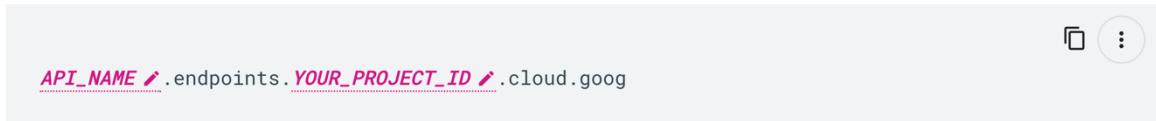
API Setup

To set up API using Cloud Endpoints we need to specify openAPI yaml file. Below you can find it.

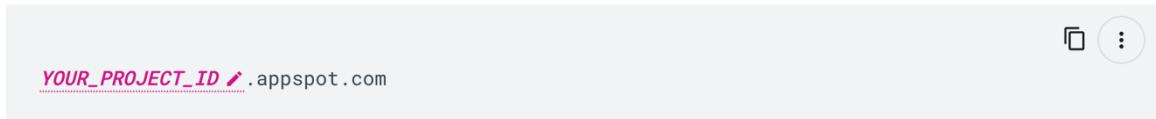
```
openapi.yaml
1  swagger: '2.0'
2  info:
3    title: Phonebook API
4    description: API for managing contacts in a phonebook.
5    version: 1.0.0
6    host: "cloudappdev-midterm-ak-439206.appspot.com"
7    schemes:
8      - https
9      - http
10   paths:
11     /contacts:
12       get:
13         summary: Get all contacts
14         operationId: getContacts
15         responses:
16           '200':
17             description: List of contacts
18           '500':
19             description: Internal server error
20         security:
21           - api_key: []
22
23       post:
24         summary: Create a new contact
25         operationId: createContact
26         parameters:
27           - in: body
28             name: contact
... . . .
```

The host name must in this format: “project_id.appspot.com”, otherwise it doesn’t work. Below official documentation.

- If you are using the `cloud.goog` domain, confirm that the value for the `host` field is in the following format, and that the project ID is correct:



- If you are using the `appspot.com` domain (supported for App Engine only), confirm that the `host` field is in the following format, and that the project ID is correct:



Command to deploy endpoint:

`gcloud endpoints services deploy openapi.yaml`

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS zsh + □ ×
For detailed information on this command and its flags, run:
  gcloud projects --help
● (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud projects list
PROJECT_ID          NAME          PROJECT NUMBER
cloudappdev-midterm-ak-439206  CloudAppDev-Midterm-AK  179348553544
● (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud endpoints services deploy openapi.yaml
Waiting for async operation operations/services.cloudappdev-midterm-ak-439206.appspot.com-0 to complete...
Waiting for async operation operations/serviceConfigs.cloudappdev-midterm-ak-439206.appspot.com:91a972d5-6796-4eac-9e2b-171aac348202 to complete...
Operation finished successfully. The following command can describe the Operation details:
  gcloud endpoints operations describe operations/serviceConfigs.cloudappdev-midterm-ak-439206.appspot.com:91a972d5-6796-4eac-9e2b-171aac348202

Waiting for async operation operations/rollouts.cloudappdev-midterm-ak-439206.appspot.com:59c57a59-db67-4b94-baf4-5856133b9539 to complete...
Operation finished successfully. The following command can describe the Operation details:
  gcloud endpoints operations describe operations/rollouts.cloudappdev-midterm-ak-439206.appspot.com:59c57a59-db67-4b94-baf4-5856133b9539

Enabling service [cloudappdev-midterm-ak-439206.appspot.com] on project [cloudappdev-midterm-ak-439206]...
Operation "operations/acat.p2-179348553544-d0c92474-5c6c-4b72-8e7d-864d94ad010" finished successfully.

Service Configuration [2024-10-20r0] uploaded for service [cloudappdev-midterm-ak-439206.appspot.com]

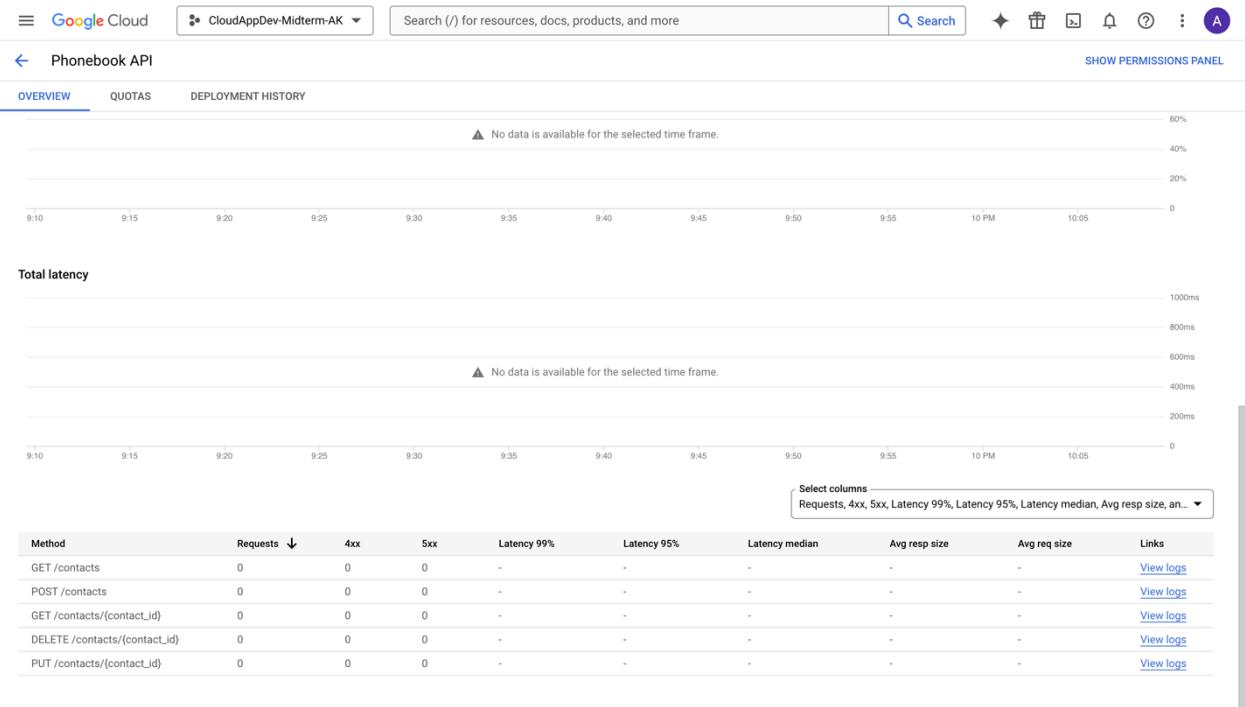
To manage your API, go to: https://console.cloud.google.com/endpoints/v2/api/cloudappdev-midterm-ak-439206.appspot.com/overview?project=cloudappdev-midterm-ak-439206
○ (base) kabdrakhman@MacBook-Pro-Altair Midterm %

```

Ln 6, Col 37 Spaces: 2 UTF-8 LF YAML ⚡ Go Live

As result we can monitor now Phonebook API by this link -

<https://console.cloud.google.com/endpoints/v2/api/cloudappdev-midterm-ak-439206.appspot.com/overview?project=cloudappdev-midterm-ak-439206>.



Security and Monitoring:

Describe how authentication and monitoring were implemented.

Used this command to create api key:

```
gcloud services api-keys create --display-name="secret-key"
```

```
● (base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud services api-keys create --display-name="secret-key"
Operation [operations/akmf.p7-179348553544-8ed4fca4-801e-4158-b29b-ecb42c1d309e] complete. Result: {
  "@type": "type.googleapis.com/google.api.apikeys.v2.Key",
  "createTime": "2024-10-20T17:13:38.110786Z",
  "displayName": "secret-key",
  "etag": "W/"MmFoNmvi9GwQx474zoAKXw=="",
  "keyString": "AIzaSyAV8oOpvDrC44gW9f0tSXZEYEW7JmVwLE",
  "name": "projects/179348553544/locations/global/keys/24424fb-504e-446b-b92e-d058fc58dc68",
  "uid": "24424fb-504e-446b-b92e-d058fc58dc68",
  "updateTime": "2024-10-20T17:13:38.131614Z"
}
○ (base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

After obtaining the key we do the followings:

- Creating wrapper function

```

10  # Security key
11  API_KEY = "AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE"
12
13  # URL for the Cloud Function (replace with your actual Cloud Function URL)
14  CLOUD_INPUT_VALIDATION = 'https://inputvalidation-zqyj4gfxea-uc.a.run.app'
15  CLOUD_FUNCTION_LOGGING_URL = 'https://logevents-zqyj4gfxea-uc.a.run.app'
16
17  # Secret key check
18  def secret_key(func):
19      def wrapper(*args, **kwargs):
20          secret_key = request.headers.get('secret_key')
21          if secret_key != API_KEY:
22              return jsonify({"error": "Unauthorized: Invalid API Key"}), 401
23          return func(*args, **kwargs)
24      wrapper.__name__ = func.__name__
25      return wrapper
26

```

- Adding this function to each method

```

87  # Read all contacts
88  @app.route('/contacts', methods=['GET'])
89  @require_api_key
90  def get_contacts():
91      return jsonify(list(phonebook.values())), 200
92
93  # Read a specific contact by ID
94  @app.route('/contacts/<int:contact_id>', methods=['GET'])
95  @require_api_key
96  def get_contact(contact_id):
97      contact = phonebook.get(contact_id)
98      if not contact:
99          return jsonify({'message': 'Contact not found'}), 404
100     return jsonify(contact), 200
101
102  # Update a contact
103  @app.route('/contacts/<int:contact_id>', methods=['PUT'])
104  @require_api_key
105  def update_contact(contact_id):
106      contact = phonebook.get(contact_id)
107      if not contact:
108          return jsonify({'message': 'Contact not found'}), 404
109      updated_data = request.json
110      contact['name'] = updated_data.get('name', contact['name'])
111      contact['phone'] = updated_data.get('phone', contact['phone'])
112
113  # Log the creation event
114  log_event(updated_data, 'PUT')
115
116  return jsonify(contact), 200
117
118  # Delete a contact
119  @app.route('/contacts/<int:contact_id>', methods=['DELETE'])
120  @require_api_key
121  def delete_contact(contact_id):
122      contact = phonebook.pop(contact_id, None)
123      if not contact:
124          return jsonify({'message': 'Contact not found'}), 404
125

```

- Adding secret key as security in openapi.yaml file

```

95
96     delete:
97         summary: Delete a contact
98         operationId: deleteContact
99         parameters:
100            - in: path
101                name: contact_id
102                type: integer
103                required: true
104         responses:
105            '200':
106                description: Contact deleted successfully
107            '404':
108                description: Contact not found
109            '500':
110                description: Internal server error
111         security:
112            - api_key: []
113
114     securityDefinitions:
115       api_key:
116           type: apiKey
117           name: secret_key
118           in: header
119

```

- Do gcloud app deploy and gcloud endpoints services deploy openapi.yaml

Result:

Without key in header

The screenshot shows a Postman request for a 'GET' operation to 'http://{{BASE_URL}}/contacts'. The 'Headers' tab is selected, showing 'Content-Type: application/json'. The 'Body' tab is selected, showing 'This request does not have a body'. The 'Tests' tab shows a failing test: `expect(response).to.have.status(200)` with the message 'Assertion failed: expected 200 but got 401 Unauthorized'. The 'JSON' response pane shows a JSON object with an 'error' key: `{"error": "Unauthorized: Invalid API Key"}`.

With key

HTTP Cloud-App / Midterm / Contacts

GET http://{{BASE_URL}}/contacts

Headers (8)

Key	Value	Description	Bulk Edit	Pr
<input checked="" type="checkbox"/> secret-key	AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE			
Key	Value	Description		

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 445 ms Size: 413 B Save as ex:

Pretty Raw Preview Visualize JSON ↻

```

1 [
2   {
3     "id": 1,
4     "name": "David Koelpin",
5     "phone": "418-461-9626"
6   },
7   {
8     "id": 2,
9     "name": "Mrs. Barry Howell",
10    "phone": "673-980-1965"
11  },
12  {
13    "id": 3,
14    "name": "Fredrick Graham",
15    "phone": "218-409-9699"
16  }
17 ]

```

10. Testing and Quality Assurance

Unit testing:

```

testing > 🐍 unit-test.py
  1 import unittest
  2 from flask import json
  3 from main import app, phonebook # Import the Flask app and the phonebook dictionary
  4
  5 class PhonebookAppUnitTests(unittest.TestCase):
  6     def setUp(self):
  7         self.app = app.test_client()
  8         self.app.testing = True
  9         # Create a contact to use in tests
10         self.contact_data = {"name": "Test User", "phone": "1234567890"}
11         response = self.app.post('/contacts', json=self.contact_data, headers={'secret_key': 'AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE'})
12         self.contact_id = response.get_json()["id"]
13
14     def test_create_contact(self):
15         contact_data = {"name": "New User", "phone": "0987654321"}
16         response = self.app.post('/contacts', json=contact_data, headers={'secret_key': 'AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE'})
17         self.assertEqual(response.status_code, 201)
18         self.assertIn("id", json.loads(response.data))
19
20     def test_get_all_contacts(self):
21         response = self.app.get('/contacts', headers={'secret_key': 'AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE'})
22         self.assertEqual(response.status_code, 200)
23         self.assertGreater(len(json.loads(response.data)), 0)
24
25     def test_get_contact(self):
26         response = self.app.get(f'/contacts/{self.contact_id}', headers={'secret_key': 'AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE'})
27         self.assertEqual(response.status_code, 200)

```

Result:

```

● (base) kabdrakhman@MacBook-Pro-Altair Midterm % python -m unittest testing.unit-test

POST action logged successfully.
POST action logged successfully.
.POST action logged successfully.
DELETE action logged successfully.
.POST action logged successfully.
.POST action logged successfully.
.POST action logged successfully.
Failed to log PUT action.
.

-----
Ran 5 tests in 7.977s

OK
○ (base) kabdrakhman@MacBook-Pro-Altair Midterm %

```

Integration testing:

```

testing > 🐍 integration-test.py
 1  import unittest
 2  from flask import json
 3  from main import app
 4  import requests_mock
 5
 6  # Define your secret key
 7  SECRET_KEY = "AIzaSyAV8oQPuVDrC44gW9f0tSXZEYEW7JmVwLE"
 8
 9  class PhonebookAppIntegrationTests(unittest.TestCase):
10      def setUp(self):
11          self.app = app.test_client()
12          self.app.testing = True
13
14          # Create a contact with the secret_key header
15          response = self.app.post('/contacts',
16                                  json={"name": "Test Name", "phone": "1234567890"},
17                                  headers={'secret_key': SECRET_KEY})
18
19          print(response.data) # Debug print
20
21          # Check if the response is successful
22          if response.status_code == 201: # 201 for created
23              self.contact_id = response.get_json()["id"]
24          else:
25              self.contact_id = None # Handle the case when the contact isn't created
26
27  @requests_mock.Mocker()
28  def test_create_contact_integration(self, mock_requests):
29      mock_requests.post('https://inputvalidation-zqyj4gfbea-uc.a.run.app', json={}, status_code=200)
30      mock_requests.post('https://logevents-zqyj4gfbea-uc.a.run.app', json={}, status_code=200)
31
32      response = self.app.post('/contacts',
33                              json={"name": "Integration Test Name", "phone": "0987654321"},
34                              headers={'secret_key': SECRET_KEY})
35      self.assertEqual(response.status_code, 201) # 201 for created
36      self.assertIn("id", json.loads(response.data))
37
38  @requests_mock.Mocker()
39  def test_get_contact_integration(self, mock_requests):
40      response = self.app.get(f'/contacts/{self.contact_id}', headers={'secret_key': SECRET_KEY})
41      self.assertEqual(response.status_code, 200)
42      self.assertIn("name", json.loads(response.data))
43
44  @requests_mock.Mocker()
45  def test_update_contact_integration(self, mock_requests):
46      mock_requests.post('https://inputvalidation-zqyj4gfbea-uc.a.run.app', json={}, status_code=200)

```

Ln 7, Col 54 Spaces:

Result:

```
(venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm % python -m unittest testing.integration-test
POST action logged successfully.
b'{"id":1,"name":"Test Name","phone":"1234567890"}\n'
POST action logged successfully.
.POST action logged successfully.
b'{"id":3,"name":"Test Name","phone":"1234567890"}\n'
DELETE action logged successfully.
.POST action logged successfully.
b'{"id":4,"name":"Test Name","phone":"1234567890"}\n'
.POST action logged successfully.
b'{"id":5,"name":"Test Name","phone":"1234567890"}\n'
PUT action logged successfully.

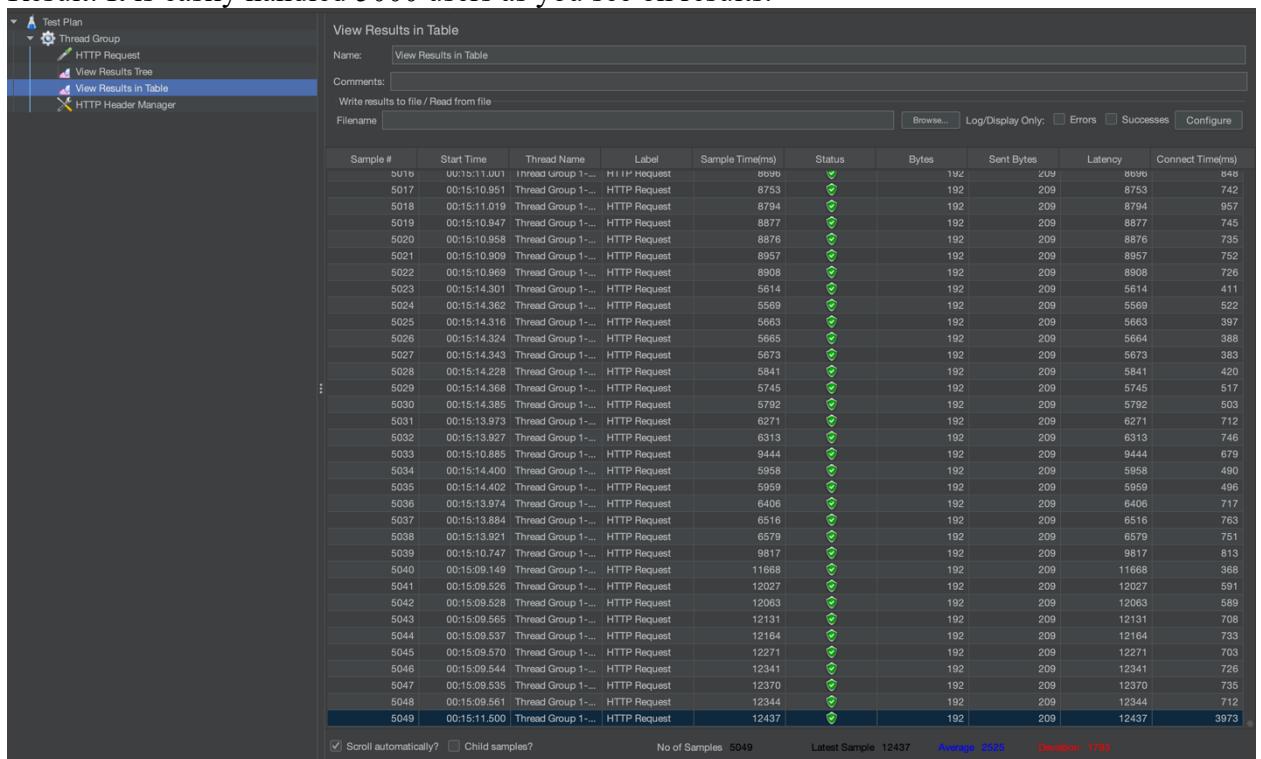
-----
Ran 4 tests in 9.724s

OK
(venv) (base) kabdrakhman@MacBook-Pro-Altair Midterm %
```

Outline the testing methodologies used, including unit tests, integration tests, and load testing results.

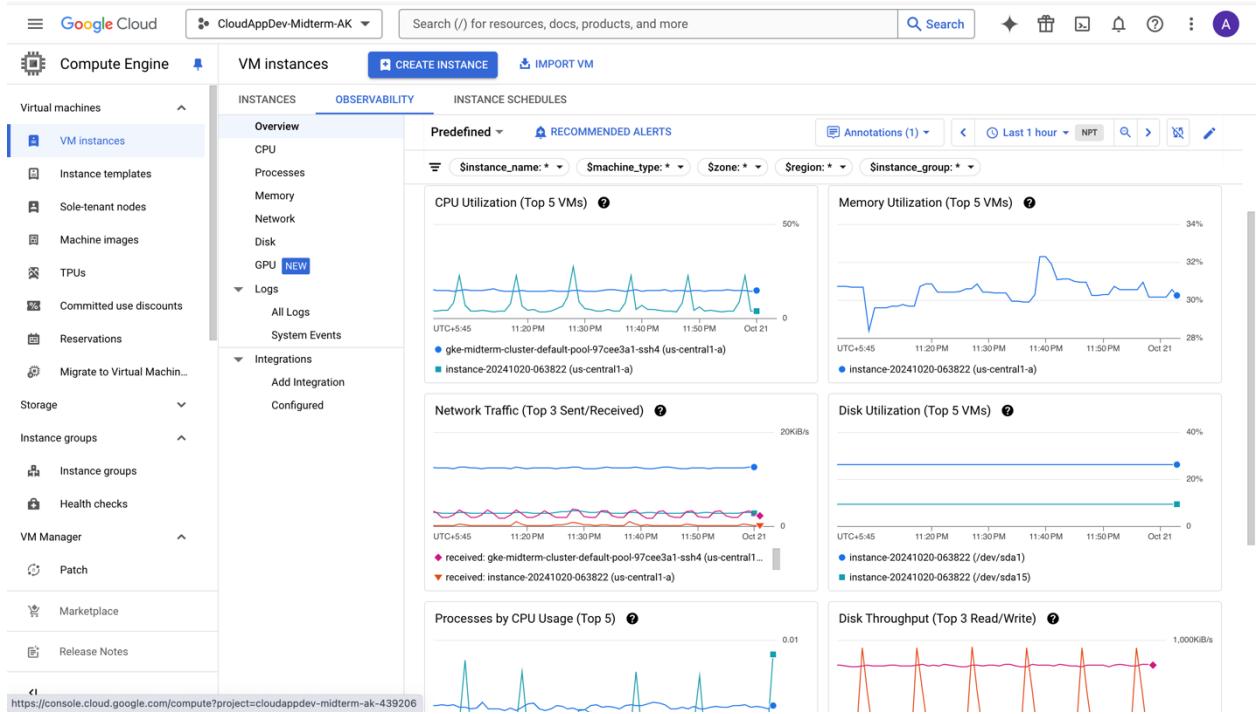
For load testing I used open source tool Jmeter with 5000 users request in 20 seconds.

Result: It is easily handled 5000 users as you see on results.

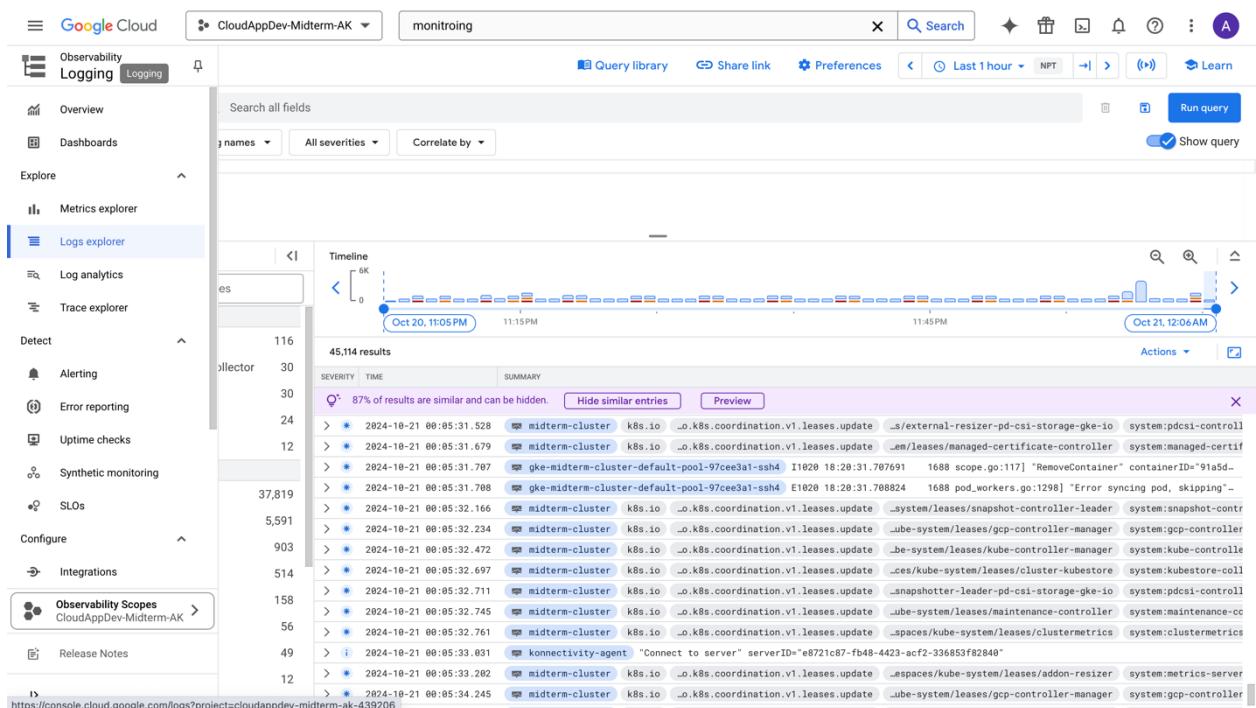


11. Monitoring and Maintenance

Google cloud already implemented monitoring features on their console.cloud.google.com website. As you can see here is example monitoring of compute engine



The most easy-to-use is observability and its logging where collected all logs from all cluster, vm intances, application and etc. Also it provides opportunity to create logging alerts.



12. Challenges and Solutions

During this midterm project I had several problems. Look at the most five problems:

- Forget to enable functions or apis/services
- Problem with deploying to Google Cloud App Engine. Solution: requirements.txt
- No access to Cloud Functions because I used another account accidentally
- Working with GCR, didn't know about tagging as project
- With openapi to write host as project_id.appspot.com

13. Conclusion

In this project, I developed a responsive web application using Google Cloud Platform (GCP), focusing on deploying a phonebook service. The architecture of the project includes core GCP services such as Cloud Run for serverless functions, Kubernetes for container orchestration, and Docker for application containerization. The application was built with a focus on security by implementing a header for API key validation. I utilized OpenAPI for defining the cloud API, which enhanced the documentation and usability of the service. During development, I faced challenges such as ensuring the application's scalability and high availability. To overcome these challenges, I conducted unit and integration testing, along with load testing to validate performance under various conditions. Overall, the project successfully demonstrated the effectiveness of the chosen technologies and met the objectives set for a robust cloud-based application.

14. References

1. Google Cloud SDK - <https://cloud.google.com/sdk/docs/install>
2. Google Compute Engine - <https://cloud.google.com/products/compute?hl=ru>
3. Cloud Run functions - <https://cloud.google.com/functions?hl=ru>
4. Docker Documentation - <https://www.docker.com/>
5. Google Kubernetes Engine (GKE) - <https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview>
6. Cloud Endpoints -<https://cloud.google.com/endpoints/docs/openapi/get-started-app-engine>
7. API keys - <https://cloud.google.com/docs/authentication/api-keys>
8. Jmeter - <https://jmeter.apache.org/>

15. Appendices

Project structure

The screenshot shows a code editor interface with several tabs and panes. The main pane displays a Python file named `main.py` containing code related to logging events and creating contacts. Below the code editor are several status messages from a terminal or log viewer, indicating the deployment of a service to Google App Engine. The sidebar on the left shows a file tree for a project named "MIDTERM". The bottom right corner of the screen shows the number "27".

```
def log_event(contact_data, method):
    try:
        if response.status_code != 200:
            print(f"Failed to log {method} action.")
        else:
            print(f"{method} action logged successfully.")
    except requests.RequestException:
        print("Failed to connect to logging service.")

# Create a new contact
@app.route('/contacts', methods=['POST'])
def check_secret_key():
    def create_contact():
        global next_contact_id
        contact_data = request.json

        # Validate the request data using the Cloud Function
        validation_result, status_code = validate_request(contact_data)
        if status_code != 200:
            # If validation fails, return the validation error response
            return jsonify(validation_result), status_code

        # Proceed to create the contact if validation succeeds
        contact = {
            'id': next_contact_id,
            'name': contact_data['name'],
            'phone': contact_data['phone']
        }
        next_contact_id += 1
        return jsonify(contact), 201

    return create_contact()

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

(base) kabdrakhman@MacBook-Pro-Altair Midterm % gcloud endpoints services deploy openapi.yaml
Waiting for async operation operations/serviceConfigs.cloudappdev-midterm-ak-439206.appspot.com:33f77edc-e931-4478-a758-c8220f491059 to complete...
Operation finished successfully. The following command can describe the Operation details:
gcloud endpoints operations describe operations/serviceConfigs.cloudappdev-midterm-ak-439206.appspot.com:33f77edc-e931-4478-a758-c8220f491059
Waiting for async operation operations/rollouts.cloudappdev-midterm-ak-439206.appspot.com:4fff6deb-65f8-4f3c-af90-f595108a5ca7 to complete...
Operation finished successfully. The following command can describe the Operation details:
gcloud endpoints operations describe operations/rollouts.cloudappdev-midterm-ak-439206.appspot.com:4fff6deb-65f8-4f3c-af90-f595108a5ca7
Service Configuration [2024-10-20T11] uploaded for service [cloudappdev-midterm-ak-439206.appspot.com]
To manage your API, go to: https://console.cloud.google.com/endpoints/api/cloudappdev-midterm-ak-439206.appspot.com/overview?project=cloudappdev-midterm-ak-439206