Kazakh-British Technical University

KBTU

Cloud Computing
Midterm report

Student: Khanfiyeva Elnara
Professor: Serek Azamat

2024

Table of Contents

Executive Summary

In this report it is provided an in depth analysis and description of the creating a file sharing application with the help of Google Cloud. This project aims to establish a platform to enable the user to upload, maintain and share a set of files in a managed cloud.The application will use virtual machines, storage solutions, networking capabilities, and security features.

The project uses all of Google Cloud's offerings with virtual machines (VMs) running on GCP's Compute Engine, Cloud Storage for file management, Virtual Private Cloud (VPC) for networking, and Identity and Access Management (IAM) for security. Also in this report, you will see the overview of the existing Google Cloud services, their description and ways to utilize them in your own projects. Furthermore you will see the detailed way of utilizing it in creating of the project you have read about earlier. I will also discuss the cloud computing basics, fundamentals and etc. for a full view of the technology.

Introduction

Cloud computing represents a significant advancement in the field of information technology, providing a model where data storage, management, and processing passed over the internet instead of on local servers or personal computers. This shift fundamentally changes how applications are developed, deployed, and maintained, bringing efficiencies and capabilities that were not feasible in traditional, on-premises infrastructures. In practice, using cloud as a provider of IT solutions is might be less budget.

At its core, cloud computing offers a range of benefits that make it especially appealing for modern application development. These advantages include scalability, flexibility, cost reduction, and streamlined resource management. Cloud services allow businesses and developers to deploy applications without the need for large investments in physical hardware or infrastructure. Instead, cloud providers manage these resources, enabling clients to concentrate on building, scaling, and optimizing applications without needing to worry about maintaining servers and data centers.

In contrast to traditional computing models, cloud computing can be deployed through various models, each defining the types of services provided by the cloud provider and the degree of control maintained by the user. The primary models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

The IaaS model provides essential resources such as virtual machines, networking, and storage, giving users a high degree of flexibility and control over their infrastructure while removing the responsibility of hardware maintenance. I like to imagine that it is the details for building a car. PaaS, on the other hand, offers a ready-to-use platform that supports the full lifecycle of application development, allowing developers to build and manage applications without concern for the underlying infrastructure details. SaaS delivers fully functional software applications over the internet, managed entirely by the service provider, enabling end users to access and use these applications without the need for installation, updates, or maintenance. Each model offers distinct advantages, and many cloud solutions employ a hybrid approach, combining elements from each model to create a more versatile and optimized service environment.

For this project, I primarily utilized IaaS resources through Google Cloud Platform (GCP), setting up an infrastructure to support my file-sharing application with essential computing power, storage, and networking capabilities. I chose it because I have experience using it, and I like it before. Let’s move on to the theoretical part**.**

Project Objectives

This project's primary objective is to develop a secure, scalable, and user-friendly cloud-based file-sharing application hosted on Google Cloud Platform (GCP).

As those are essential for modern collaboration, the project seeks to use GCP's services to create a solution that is accessible, efficient, and secure. To meet these goals, there were established the following specific objectives:

- Develop an application that can handle varying workloads by using Google Cloud's scalable infrastructure. In this project, a secure file-sharing application will be developed with robust encryption and access controls to ensure data security.
- The application will be designed to scale efficiently as more users join, maintaining optimal performance. Availability will be prioritized by minimizing downtime to enhance user accessibility.
- File management, including uploads, downloads, and storage, will be handled effectively through Google Cloud Storage.
- User permissions will be managed using Google Cloud Identity and Access Management (IAM) to control access rights securely.
- Additionally, a Virtual Private Cloud (VPC) with firewall rules will be implemented to provide a secure yet efficient network environment.
- Continuous monitoring and maintenance will be established to support consistency.

This approach to building the application on GCP is designed to ensure stable and secure file-sharing functionality, aligning with core requirements of reliability, security, and ease of use.

Cloud Computing Overview

Cloud computing has gained significant traction in recent years, offering numerous benefits for application development and deployment. The principles of cloud computing, including its deployment models, provide a flexible and scalable infrastructure for hosting applications and managing data. (Xiao & Xiao, 2012). This transition from the classic on premises models to the cloud environment makes the resource usage more efficient, easier available, and flexible than ever before, allowing organizations and developers to satisfy the growing and changing requirements without spending on an expensive hardware.

Essentially, the principles of cloud computing are grounded in few core characteristics highly distinct from traditional computing models. On demand self service, broad network access, resource pooling, rapid elasticity and measured service are examples of these. Each of these principles facilitates easy scaling up or down of computing resources for organizations. With on demand self service cloud resources are available on demand which eliminates delay in setting hardware manually. Cloud resources are available from any internet connected device in a virtually unlimited number of ways, allowing for flexibility and accessibility for remote or geographically distributed teams. Providers can use resource pooling to provide their service to multiple customers using the same hardware and so pool resources between multiple customers, thus making the service more economically viable. Cloud resources being elastic enables them to be rapidly adjusted to meet current demand times, and is especially useful for applications of varying workload. Finally, the measured service characteristic ensures that the resources are allocated and billed as the usage, thereby increasing the transparency and control on the cost.

The appeal of cloud computing lies in these principles which make it appealing for organizations and developers to develop and implement, as it benefits the organization with scalability, cost effectiveness, improved flexibility and simplified resource management. With cloud computing, the faster innovation cycles are enabled because developers can design and run their apps without worrying about the infrastructure.Cloud computing model is further divided into some deployment models which offer different control power, flexibility, and management responsibilities. There are three main deployment models, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS).

In IaaS, we offer users the access to the fundamental computing resources like virtual machines, storage and networking to build and operate their own applications and systems. This offers extremely high flexibility for end users, who could install and configure their own software environments, but no longer need to have responsibility for the virtualized infrastructure. When the goal is to save costs and distribution of physical hardware but you still need control over your application environment, IaaS makes a lot of sense. IaaS — such as Google Compute Engine and Amazon EC2 — provides scalable virtual machines that users control so that they can be tailored to meet specific needs.

A higher level of abstraction is offered by PaaS which will take care of the whole platform of application development, deployment and management. Not only the underlying infrastructure, but essential development tools, middleware and the database management systems along with it, this model allows the developers to write and deploy the code, and primarily concentrate on creating and deploying the code rather than on infrastructure details. PaaS is especially well suited for development teams wanting to speed their software development process with some level of customisation. PaaS include examples like Google App Engine and Microsoft Azure App Services that centralize the application development, and host environment management demands.

SaaS is simply a service that delivers a fully functional application to the web over the internet on the service provider’s behalf. With these applications, users can access these without having to worry about installation, maintenance or updates. It’s a very convenient model for end users because it comes pre configured and ready to use without any setup or management, so it is excellent for productivity, collaboration and communication tools. Many SaaS applications take a subscription approach, allowing users to use the service in variable ways. Some popular examples of SaaS are Google Workspace, Salesforce, and Dropbox etc.

Together these deployment models provide a breadth of options for businesses and developers to select how much control and customization they would like to have. IaaS employs maximum flexibility, PaaS processes favorable development workflows, and SaaS offers easy access for the end users. Cloud computing is truly a foundational element of the modern information technology solutions due to various variants of these models, coupled with the basic principles of cloud computing, which make cloud services be able to satisfy the needs in different industries.

Google Cloud Platform: Core Services

GCP provides us with a plethora of services across numerous categories consisting of computing, storage, networking and databases. Here's a description of the main services relevant to many applications:

1. Computing Services
    - Google Compute Engine (GCE) is the Google infrastructure on which virtual machines do run. Here it supports the different operating systems and also suitable for the custom application. Virtual machines are used for computational processes that require the most of the core and CPU, RAM.
    - Google Kubernetes Engine (GKE) is a service which provides you with native Kubernetes deployment, management, and scaling of your containerized applications. This is easier for DevOPs managers, because it automates the processes of the orchestration making it easier to develop and deploy.
    - Google Cloud Functions is a serverless compute service, executing code when events occur, so developers don't need to manage servers.
    - Google App Engine is a service platform that allows developer to create and deploy the applications without worrying about the below infrastructure.
2. Storage Services
    - Google Cloud Storage is a scalable object storage service for unstructured data, e.g. images, videos and backups. It is a some kind of the AWS S3.
    - Google Filestore is a managed file storage service on which applications which need a file system interface can run, such as content management systems.
3. Networking Services
    - Virtual Private Cloud enables the user to create networks isolating his resources with control over IP address, subnets and routing.
    - Cloud Load Balancing aims are to transparently distribute incoming application traffic across multiple resources such that high availability is achieved.
    - Cloud CDN caches and deliver content from nearest location to user to facilitate content delivery.
4. Database Services
    - Cloud SQL is a MySQL and PostgreSQL support with an easy database administration.
    - Cloud Spanner is a horizontally scaled, strongly consistent database service available globally.

For this project I will be probably using these services such as Compute Engine for creating the VMs and etc. Also I will be using Cloud Storage for storing the files on the cloud.

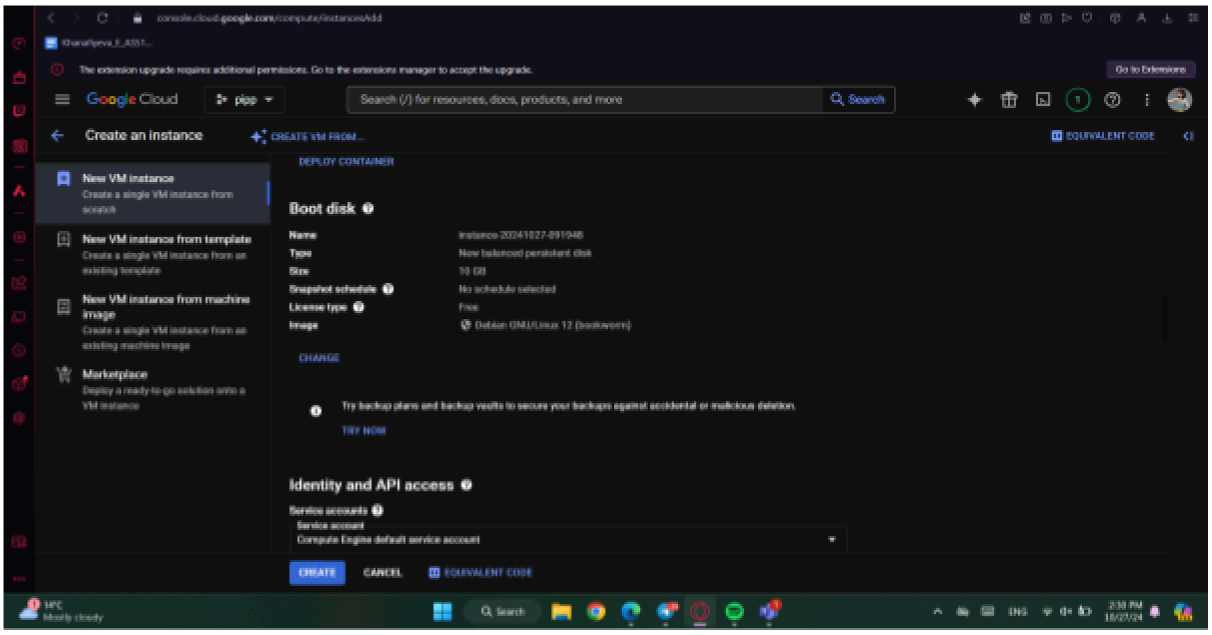
Virtual Machines in Google Cloud

Objectives of this part include:

- Set up and configure virtual machines using Google Compute Engine.
- Deploy the backend of the file sharing application on a VM, ensuring proper resource allocation and scaling options.

The first step I took in the deployment process is setting up virtual machines through Google Compute Engine. In the Google Console, I went to Google Console, select "Create Instance," and choose an appropriate machine type, e.g., asia-east1 (Taiwan), based on the expected application load. I have chosen the E2 type, because it's most suitable and universal at the same time, and cheap - e2-medium (2 vCPU, 1 core, 4 GB memory). Next I chose the Debian 12 as the boot disk image, because, again, it's well known for me, easy to use and stable OS.

:



Boot disk

Name	instance-20241027-091948
Type	New balanced persistent disk
Size	10 GB
Snapshot schedule	No schedule selected
License type	Free
Image	Debian GNU/Linux 12 (bookworm)

Try backup plans and backup vaults to secure your backups against accidental or malicious deletion.

Identity and API access

Service accounts

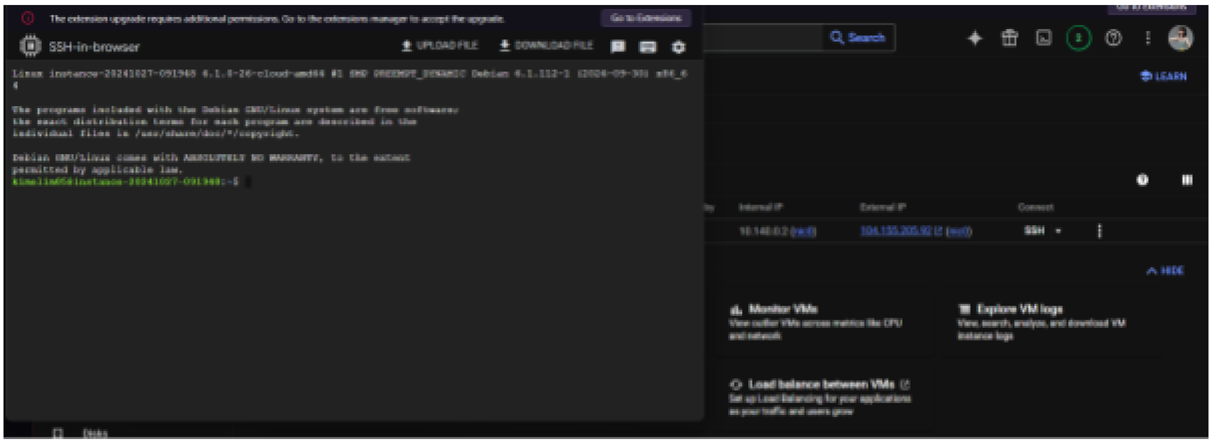
Service account: Compute Engine default service account

Figure 1. VM creation.

I will connect to the VM by SSH. You can see it shown on the *figure 2*.:

sudo apt update

Then for installation of python "sudo apt install python3 python3-pip -y". next installing all of the needed dependencies: "pip3 install Flask google-cloud-storage", cheating the app: "mkdir ~/file_sharing && cd ~/file_sharing" .

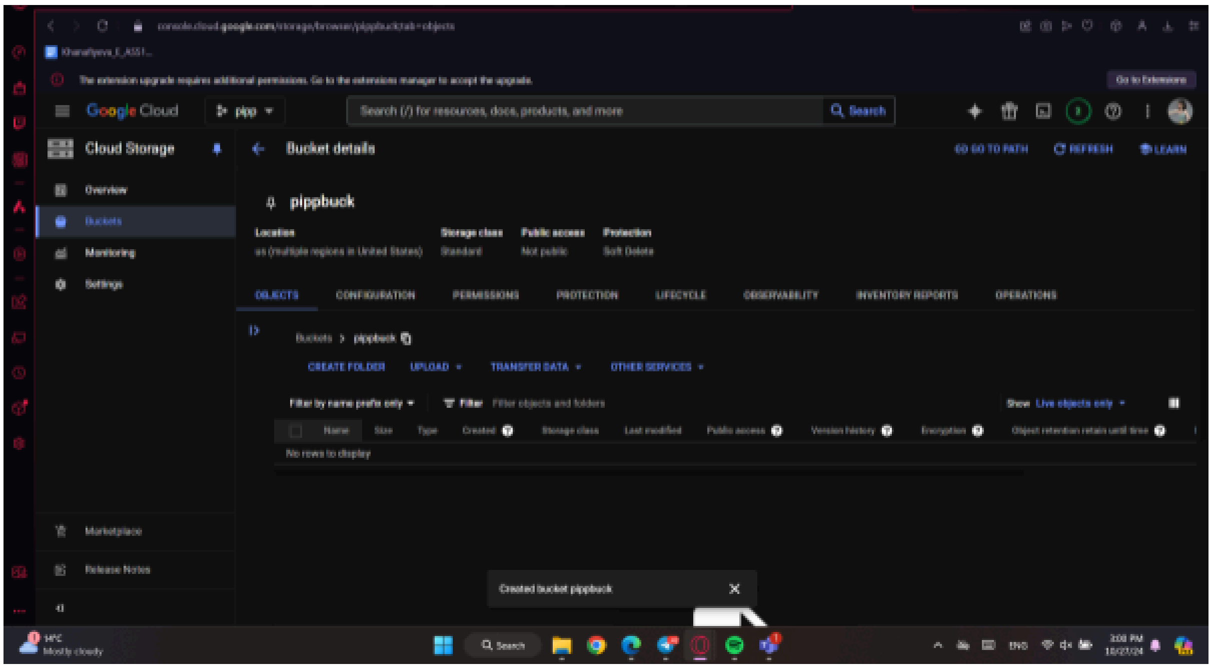


```
Linux instance-20241027-091948 6.1.0-26-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_6
4

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kimelim05@instance-20241027-091948:~$
```

Figure 2. SSH connection.

Storage Solutions in Google Cloud

After the creation we need to create the bucket in Cloud Storage. Mine is named “Pippbuck”, and is configured to be in the “asia-east-1” and multi regionality.



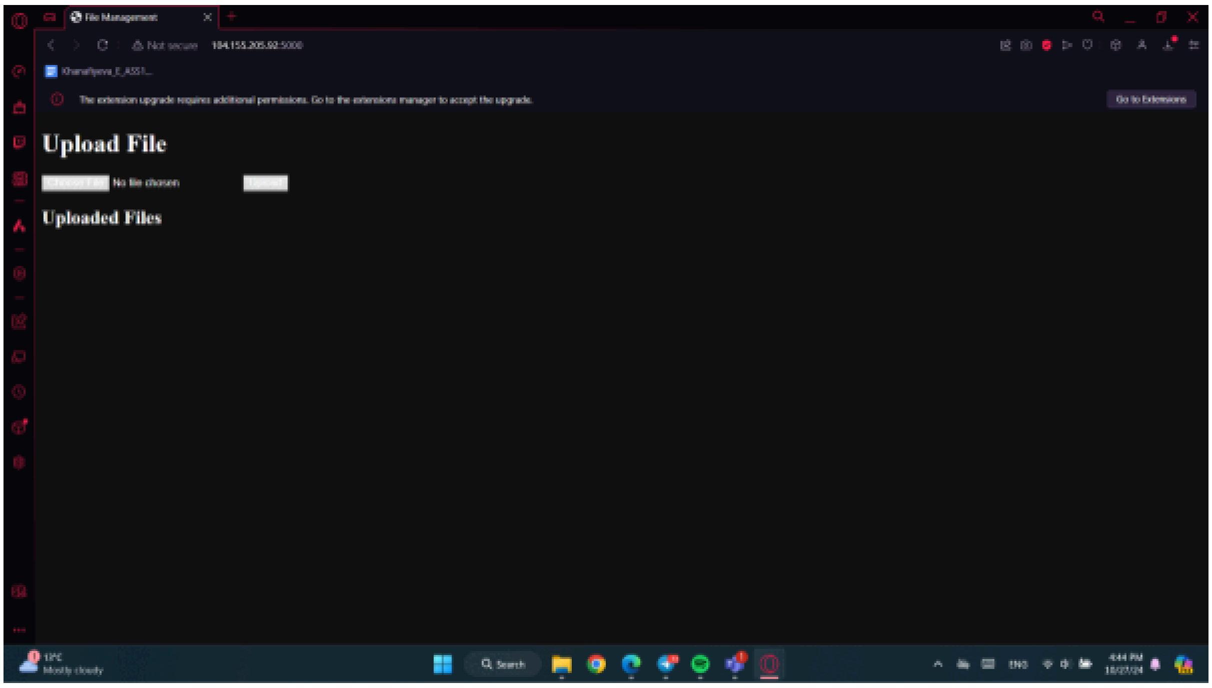
```
pippbuck
Location                                  Storage class   Public access   Protection
us (multiple regions in United States)    Standard        Not public      Soft Delete

No rows to display

Created bucket pippbuck
```

Figure 3. Created bucket in Cloud Storage.

Before that, I have created the app on python - flask app, with most simple functions of uploading and downloading and deleting.

Here is the code of my app:

```
app = Flask(__name__)
storage_client = storage.Client()
bucket_name = os.getenv("pippbuck")
bucket = storage_client.bucket(bucket_name)
@app.route('/')
def index():
    files = list_blobs()
    return render_template('index.html', files=files)

def list_blobs():
    """Lists all the blobs in the bucket."""
    blobs = bucket.list_blobs()
    return [blob.name for blob in blobs]

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400

    filename = secure_filename(file.filename)
    blob = bucket.blob(filename)
    blob.upload_from_file(file)

    return redirect(url_for('index'))

@app.route('/download', methods=['GET'])
def download_file():
    filename = request.args.get('filename')
    if not filename:
        return "No filename provided", 400

    blob = bucket.blob(filename)
    if not blob.exists():
        return "File not found", 404

    with tempfile.NamedTemporaryFile() as temp_file:
        blob.download_to_filename(temp_file.name)
        return send_file(temp_file.name, as_attachment=True,
download_name=filename)

@app.route('/delete', methods=['POST'])
def delete_file():
    filename = request.form.get('filename')
    if not filename:
        return "No filename provided", 400

    blob = bucket.blob(filename)
    if not blob.exists():
        return "File not found", 404

    blob.delete()
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```



The extension upgrade requires additional permissions. Go to the extensions manager to accept the upgrade.

Upload File

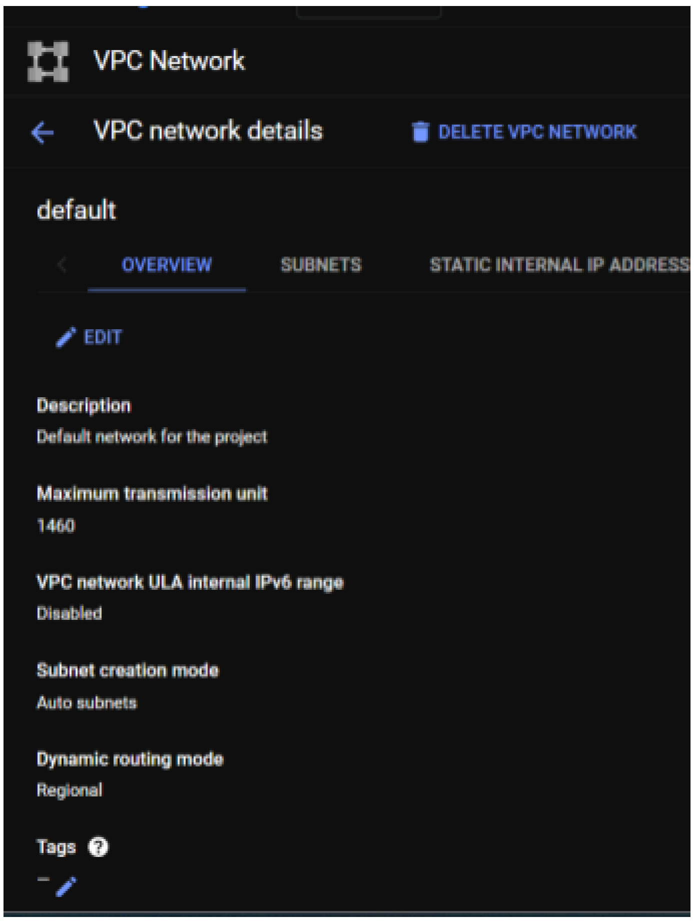
No file chosen

Uploaded Files

Figure 4. My App.

Networking in Google Cloud

The creation and configuration of a Virtual Private Cloud (VPC) are critical steps in establishing a secure and efficient networking environment for an application.
Firstly. i determined the IR addresses for IPv4 as 0.0.0.0/0. The default, I also created the rule to allow http traffic on the 5000 port (tcp). Also the main part was enabling load balancers on the VPC settings. By establishing a well-structured VPC environment and implementing stringent security protocols, organizations can effectively manage their cloud resources while safeguarding against potential threats.



default

Description
Default network for the project

Maximum transmission unit
1460

VPC network ULA internal IPv6 range
Disabled

Subnet creation mode
Auto subnets

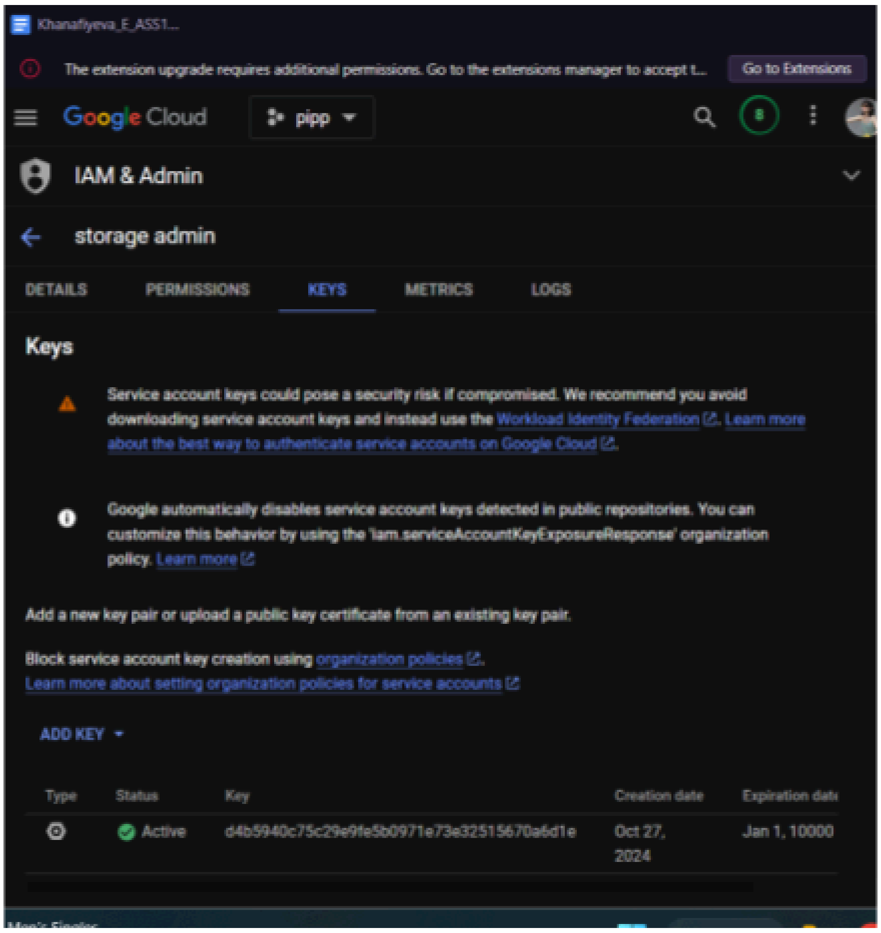
Dynamic routing mode
Regional

Tags
–

Figure 5. VPC Network settings.

Identity and Security Management

In my project, I implemented Identity and Access Management (IAM) to effectively manage user roles and permissions, ensuring that access to resources within the cloud environment is both secure and streamlined. In my project, taking advantage of IAM serves to effectively manage user roles and permissions to provide easy access to resources as well as giving a sense of security with which resources can be accessed in the cloud environment. At first I defined a set of roles based on the needs of specific application, define different users and what responsibilities they have and what degree of access they need.



The extension upgrade requires additional permissions. Go to the extensions manager to accept t...

storage admin

Keys

Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use the Workload Identity Federation. Learn more about the best way to authenticate service accounts on Google Cloud.

Google automatically disables service account keys detected in public repositories. You can customize this behavior by using the 'iam.serviceAccountKeyExposureResponse' organization policy. Learn more

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts

Type	Status	Key	Creation date	Expiration date
⚙	✓ Active	d4b5940c75c29e9fe5b0971e73e32515670a6d1e	Oct 27, 2024	Jan 1, 10000

Figure 6. IAM Control setup.

IAM roles and policies were carefully defined with access controls, using the principle of least privileged in order to let only authorized ones access the sensitive information.

Testing and Quality Assurance

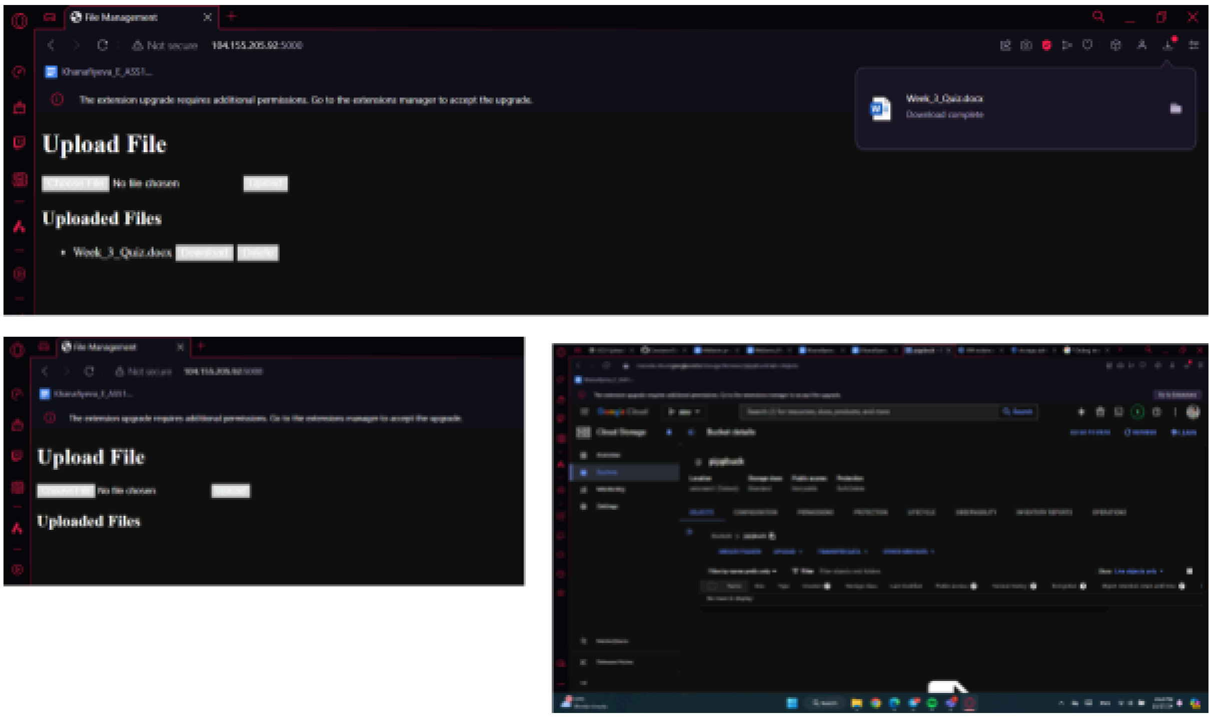

File Management
Not secure 104.155.205.92:5000
The extension upgrade requires additional permissions. Go to the extensions manager to accept the upgrade.
Week_3_Quiz.docx
Download complete
Upload File
Choose File No file chosen Upload
Uploaded Files
Week_3_Quiz.docx Download Delete
Cloud Storage
Bucket details
pipjback

Figure 7. Testing the app.

I prefer manual testing, and because the app is small and easy, I didn’t see the need to write the unit tests, but of course I did some test. Here on the figure you can see the screenshots of the app and the bucket to follow the process of the upload and etc.

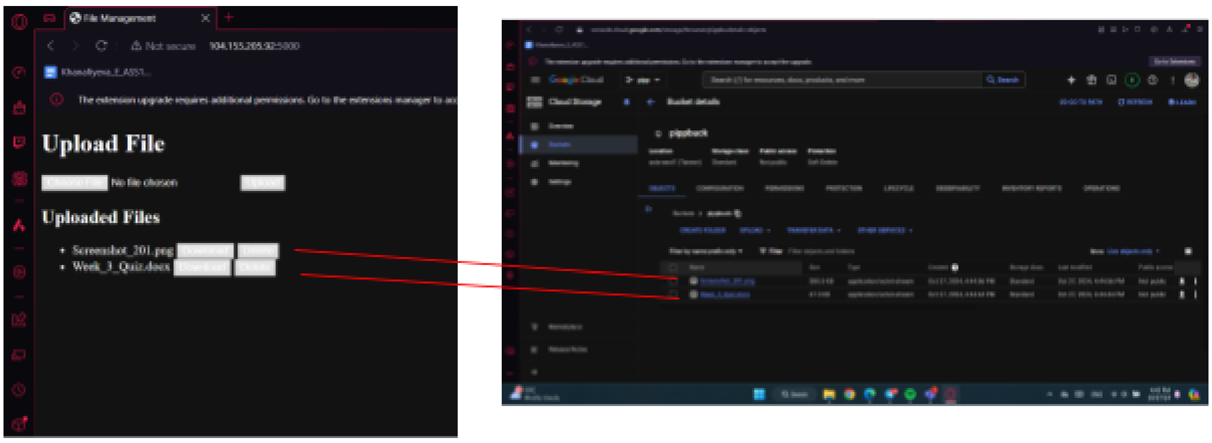

File Management
Not secure 104.155.205.92:5000
Upload File
Choose File No file chosen Upload
Uploaded Files
Screenshot_201.png Download Delete
Week_3_Quiz.docx Download Delete
Cloud Storage
Bucket details
pipjback

Figure 8. Testing the app.

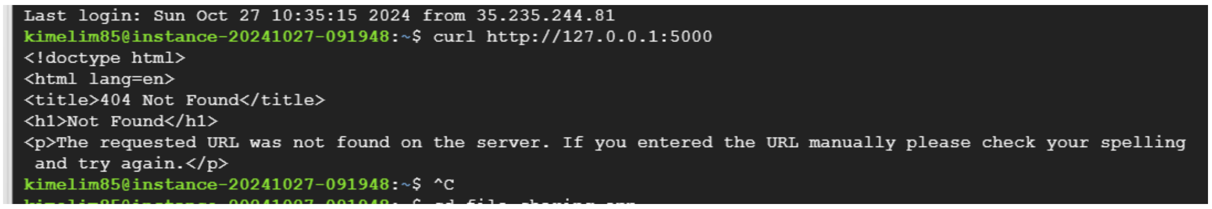
Everything works, is secure, you’re able to delete,upload, download and see what’s you’ve sent.

Monitoring and Maintenance

The monitoring tools I could use on my project for tracking application performance and health was many. For example, Google Cloud Monitoring gives you real time visible resource use, application latency, or error rate, that would allow for the setting up custom dashboards and alerts to proactively manage performance bottlenecks and possible issues. I created a routine strategy for maintenance of the system including regular system updates and performance reviews. Cloud Monitoring checks the applications and infrastructure health on a scheduled bases to be operating efficiently. Furthermore, the establishment of automated alerts relating to anomalies will provide the ability to respond to issues quickly on a reliable, high performing application environment.

Challenges and Solutions

Sometimes the most difficult is the easiest things. For me the challenge was to not forget to secure what you have to secure at first, so I will not come back to reconfigure them or even delete and recreate. Then I went into another problem of the allowing the access to my app. for some reason my request went blocked, and I had to figure it out. I also had a problem of 5000 port being used, and i had to kill the process in terminal. You can see it on the figure:



```
Last login: Sun Oct 27 10:35:15 2024 from 35.235.244.81
kimelim85@instance-20241027-091948:~$ curl http://127.0.0.1:5000
<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling
 and try again.</p>
kimelim85@instance-20241027-091948:~$ ^C
```

Figure 9. Errors.

Conclusion

Finally, the new development of a file-sharing application in the cloud based on the Google Cloud Platform is successfully shown in this midterm project. The application is designed to be secure, scalable and user friendly because it leverages key services such as Google Compute Engine, Cloud Storage, and IAM. Configurations of the Virtual Private Cloud (VPC), along with firewall rules further solidify the application's security, as well as network performance.

Further, maintenance and reliability strategies for ongoing monitoring are laid out for the application's reliability and responsiveness. During the project, we ran into challenges, but these were good learning experiences on the technical aspects that helped to sharpen my technical skills and understand better what cloud computing. Not only does this project highlight the power of GCP to build cloud application but also the criticality of the planning and execution when building cloud application. Time will tell what more optimizations and enhancements are achieved, all further to work in cloud computing.

References

Google Cloud *Documentation*, Google Compute Engine. https://cloud.google.com/compute

Google Cloud IAM *Documentation*. https://cloud.google.com/iam

Appendices

index.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>File Management</title>
</head>
<body>
    <h1>Upload File</h1>
    <form method="post" action="/upload"
enctype="multipart/form-data">
        <input type="file" name="file">
        <input type="submit" value="Upload">
    </form>

    <h2>Uploaded Files</h2>
    <ul>
        {% for file in files %}
            <li>
                {{ file }}
                <form action="/download" method="get"
style="display:inline;">
                    <input type="hidden" name="filename" value="{{
file }}">
                    <input type="submit" value="Download">
                </form>
                <form action="/delete" method="post"
style="display:inline;">
                    <input type="hidden" name="filename" value="{{
file }}">
                    <input type="submit" value="Delete">
                </form>
            </li>
        {% endfor %}
    </ul>
</body>
```

```
</html>
```