



REPORT

Midterm, Cloud Computing

Akhmetkan Azhar
24MD0470
26.10.2024

Table of Contents

Executive Summary.....	2
Introduction.....	3
Project Objectives.....	4
Cloud Computing Overview.....	5
Google Cloud Platform: Core Services.....	6
Virtual Machines in Google Cloud.....	7
Storage Solutions in Google Cloud.....	10
Networking in Google Cloud.....	12
Identity and Security Management.....	13
Testing and Quality Assurance.....	14
Monitoring and Maintenance.....	15
Challenges and Solutions.....	16
Conclusion.....	17
References.....	18
Appendices.....	19

Executive Summary

This project developed cloud file platform based on Google Cloud Platform (GCP) in order to create safe and scalable file to exchange files. In the context of the rapid development of technologies and increasing the volume of data, the need for reliable and effective solutions for storing and sharing files becomes critical. The project is aimed at satisfying these requirements, providing users with a simple and intuitive interface for downloading, storing and sharing files.

In working on the project, key technologies of cloud computing were studied and applied, including virtual machines, cloud storage and access management. The main stages included the creation and configuration of virtual machines on Google Compute Engine, the implementation of the file storage system using Google Cloud Storage and setting up network infrastructure to ensure the security and high accessibility of the application. As a result of the project, several important results were achieved. First is data safety. The introduction of encryption and access control mechanisms provided protection of user data from unauthorized access. Second is scalability. The use of cloud technologies made it possible to scale resources depending on the growth of the number of users and the volume of stored data. Third is reliability, where setting up the balance and reservation of components increased the overall reliability of the application, minimizing the downtime. In addition, in the process of development, certain difficulties were identified and successfully resolved, such as the optimization of the network configuration and user access rights management. This made the project more effective and made it better to prepare for potential problems in the future.

In conclusion, the project demonstrated the high efficiency of cloud solutions in the field of application development. The use of Google Cloud Platform technologies has opened new horizons for further improvements and expansion of functionality, which will adapt to the changing needs of users and the market as a whole.

Introduction

Cloud calculations are a method of accessing users to computing resources via the Internet. This technology greatly simplifies the process of application development, providing high flexibility and scalability. Using cloud solutions, developers can focus on creating functionality without worrying about the technical details of the infrastructure, since it is easily controlled and adapted to changing requirements.

In this project, Google Cloud Platform (GCP) was chosen due to its powerful tools, ease of use and reliable capabilities that meet modern application development standards. The GCP offers a wide range of services such as virtual machines, cloud storage and network solutions, which makes it the perfect choice for the implementation of the application for exchanging files.

The choice of GCP is due not only to its technological advantages, but also by the support of various deployment models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These models provide greater flexibility in the management of resources and deployment of applications.

The purpose of the project is to create a safe, effective and scalable application for exchanging files that can satisfy the growing needs of users. Particular attention is paid to data security, ease of use and high accessibility, which are the necessary conditions for the successful operation of the cloud application in modern conditions of the digital economy.

Project Objectives

The main objectives of the project were to develop a safe and scalable file for exchanging files. This included the creation of an architecture that can easily integrate new functions and adapt to an increase in the number of users without a decrease in performance. The key task was to ensure the high availability and reliability of the application, which was achieved through the introduction of reservation and load balancing, minimizing downtime and ensuring uninterrupted operation.

The project also aimed at implementing an effective file management system that includes downloading, storing, downloading and deleting files, as well as the possibility of accessing access for users. Ensuring the safety of data has become a priority, which involves the use of encryption both when transmitting data and at rest, along with authentication and authorization mechanisms to protect information from unauthorized access.

The creation of an intuitive user interface was important to increase user satisfaction and reduce the number of appeals to the support service. The optimization of the application performance included testing and analysis to identify narrow places, which contributed to the improvement of the response and speed processing speed.

The project also provided for the development of a monitoring and reporting system that allows you to track the condition of the application, analyze its performance and collect usage statistics, which will help in the future improvement of the service. Support for multi-platformity was another important task, ensuring the accessibility of the application on various devices and platforms, including web browsers and mobile devices, which will increase the ease of use. All these goals are aimed at creating a quality and reliable product that can satisfy user needs and adapt to changes in their requirements.

Cloud Computing Overview

Cloud calculations are models for the provision of computing resources and services via the Internet, allowing users and organizations to use powerful technologies without the need for their local installation. There are several deployment models, each of which is aimed at satisfying various business needs. The first model is IaaS or Infrastructure as a Service. It offers users virtualized resources, such as servers, storage systems and network solutions. At the same time, users gain full control over their infrastructure, which allows them to control operating systems and applications. IaaS examples include Amazon EC2 and Google Compute Engine. The second model is PaaS or Platform as a Service. It provides the developers with a platform for creating, testing and deploying applications, freeing them from the need to manage the basic infrastructure. PaaS offers tools for development, databases and the environment, which allows you to focus on writing code. The PaaS examples include Google App Engine and Heroku. The third model is SaaS or Software as a Service. It provides access to software via the Internet, usually by subscription. Users can work with applications without installing them on their devices, which provides accessibility from any point on the Internet connection. SaaS examples include Google Workspace, Microsoft 365 and Salesforce.

Cloud services offer many advantages. Firstly, they can significantly reduce equipment costs, since users can avoid large initial investments in servers and other equipment, since cloud providers control all infrastructure. Secondly, cloud solutions provide flexibility in scale of resources: users can quickly increase or reduce the volume of necessary resources depending on the current business needs, which is especially relevant in conditions of changeable demand.

In addition, cloud computing provides accessibility from anywhere in the world, allowing users to work with data and applications from any place with the Internet. This contributes to remote work and increases the overall mobility. Cloud providers also simplify management by taking on maintenance, updates and security issues, which allows companies to focus on their main business processes.

Finally, most cloud providers offer a high level of security, including data encryption, multifactorial authentication and regular backups, which protects information from unauthorized access. All these advantages make cloud calculations an attractive choice for organizations of various sizes, allowing them to optimize their business processes and increase work efficiency.

Google Cloud Platform: Core Services

Google Cloud Platform (GCP) offers a wide range of services that are ideal for the development of cloud applications. The main services that are relevant for this project include computing, storage, network and services for the management of identification and safety. In particular, GCP is provided by Google Compute Engine (GCE) and Google Kubernetes Engine (GKE), which allow users to create and control virtual machines (VM) with customizable configurations. GCE provides reliable control of VM, and GKE simplifies the deployment of containerized applications, which allows you to effectively use resources and scale. For storage of data, Google Cloud Storage (GCS) is used - this is a highly scheduled solution for object storage, which is highly available and reliable, which makes it suitable for application for file exchange. The GCP also offers Google Cloud SQL for relational databases and Google Firestore for NOSQL databases that can be selected depending on the application requirements. In terms of network services, GCP includes Virtual Private Cloud (VPC), which allows you to create isolated networks for applications and Cloud Load Balance, which distributes incoming traffic to ensure high accessibility and performance.

Identification and safety management is also an important aspect of GCP, which is carried out using Google Cloud Identity and Access Management (IAM). This service provides detailed control over the permits and roles of users, guaranteeing that only authorized users have access to certain resources. GCP also offers various security functions, including data encryption both at rest and in transmission, which enhances the overall security of the applications.

As part of this project, several key GCP services were selected based on their capabilities and compliance with the requirements. The Google Compute Engine was chosen to deploy the files for the exchange of files, since its flexible VM configurations provide the necessary distribution of resources and scalability. The Google Cloud Storage was indispensable for managing uploading, downloading and storing files, offering high availability and scalability to cope with various file sizes and access frequency. To insulate the application resources, Virtual Private Cloud (VPC) was created, which increased the level of security and provided a controlled network environment. Finally, Google Cloud Iam was introduced to control user roles and permits, providing safe access to the application.

Virtual Machines in Google Cloud

To initiate a virtual machine instance in the Google Cloud Console, the user first navigated to the Compute Engine section, enabled the service, and linked a billing account.

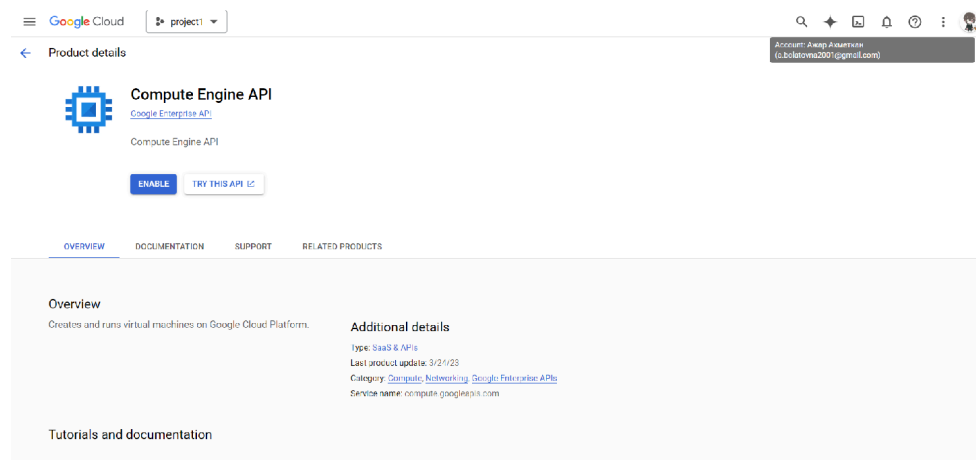


Image1.1

Subsequently, they clicked on "Create Instance" and selected the preferred operating system, region, and machine type to configure the virtual machine .

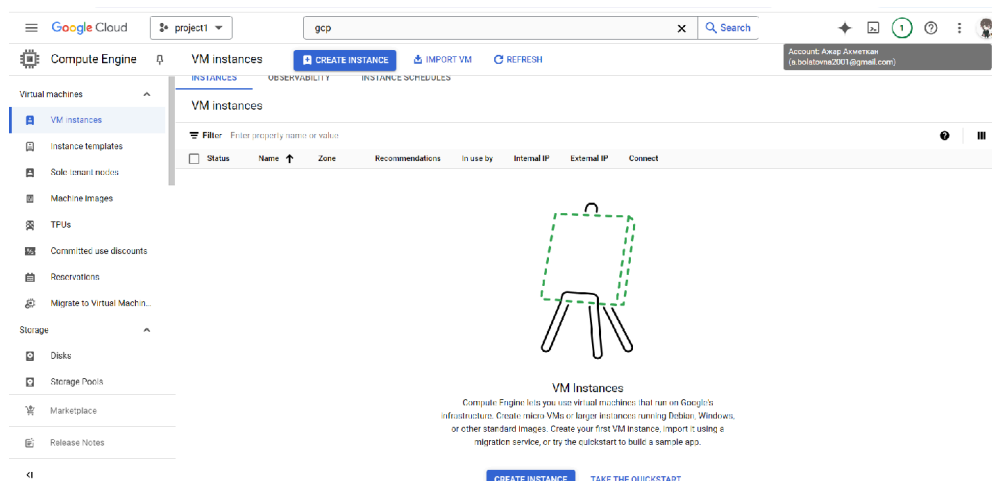


Image 1.2

After reviewing the configuration settings, they clicked the "Create" button, successfully launching the virtual machine. Finally, the new virtual machine instance was ready for use.

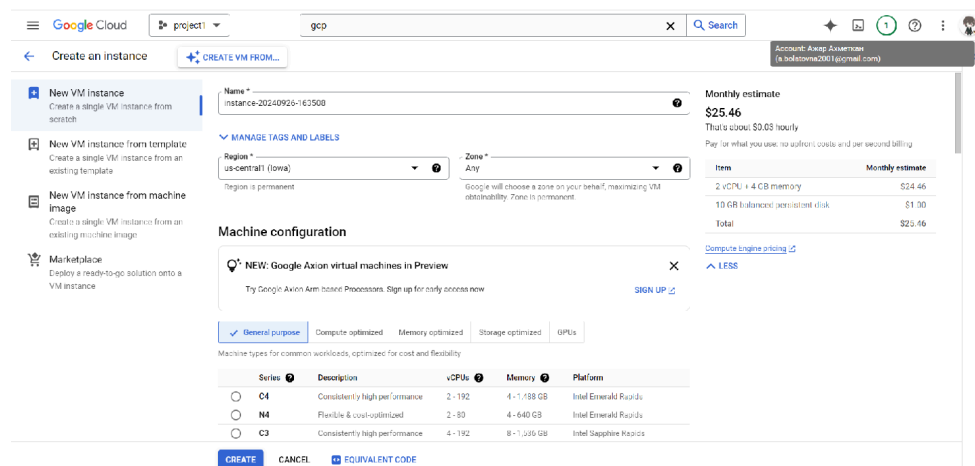


Image 1.3

After the instance was created, the user connected to it via SSH to begin setting up the necessary software and dependencies for their application.

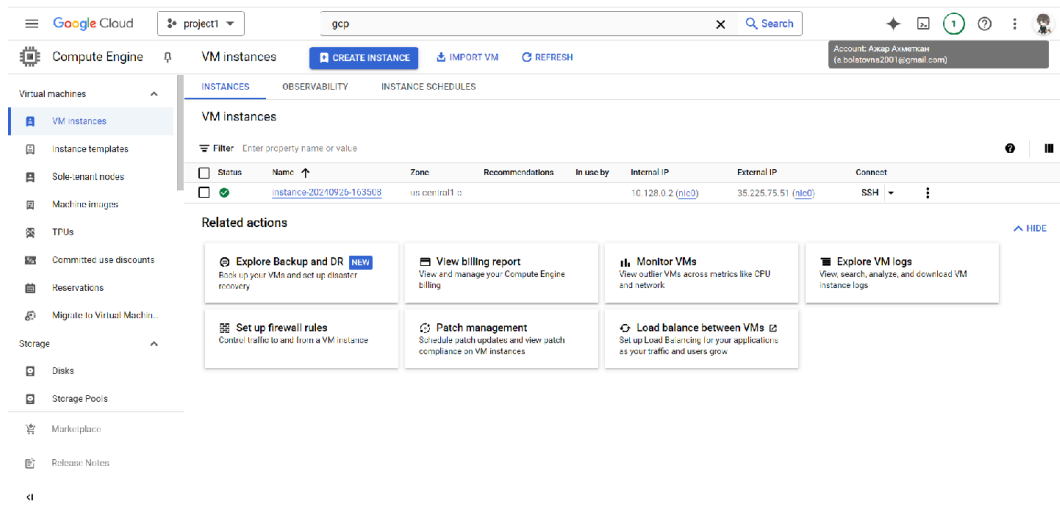


Image 1.4

They proceeded to update the package manager with the command:

sudo apt-get update

and installed required packages, ensuring the environment was prepared for deployment. This initial setup paved the way for deploying the backend of the file-sharing application effectively.

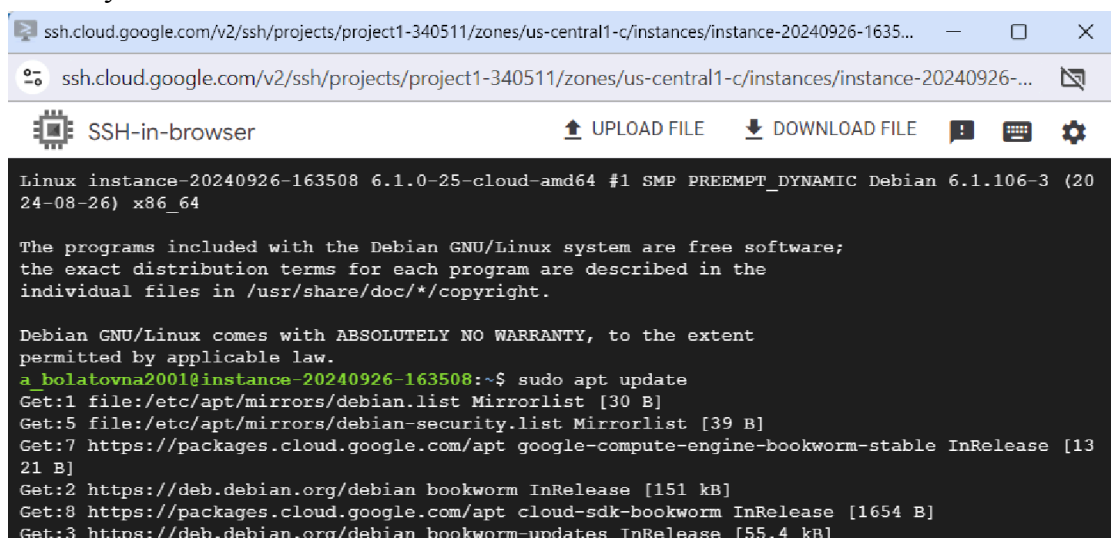


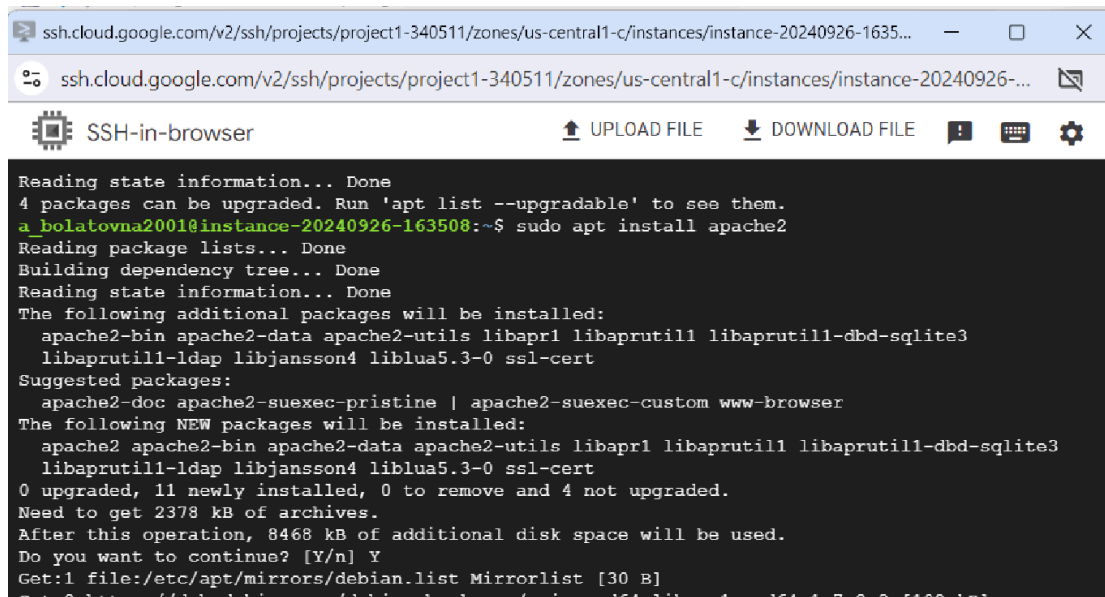
Image 1.5

Then, the user should install the necessary software for their application, such as Node.js or Python, using the command:

sudo apt-get install <package-name>

To deploy the backend of the file-sharing application, the user can upload application files to the VM through SCP or by cloning from a repository. After navigating to the application directory, they should set up the environment—using virtual environments for Python or npm for Node.js—and start the application with commands like

node app.js
python app.py.

The image shows a terminal window titled "SSH-in-browser" with a URL bar indicating it's connected to a Google Cloud instance. The terminal output shows the process of installing Apache2 using the apt package manager. It lists additional packages to be installed, suggested packages, and the disk space requirements. The user is prompted to continue with the installation, and the process begins to download files from the mirrorlist.

```
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
a_bolatovna2001@instance-20240926-163508:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libjansson4 liblua5.3-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libjansson4 liblua5.3-0 ssl-cert
0 upgraded, 11 newly installed, 0 to remove and 4 not upgraded.
Need to get 2378 kB of archives.
After this operation, 8468 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
```

Image 1.6

By going to "Instance Groups" in the Google Cloud Console, the user can configure auto-scaling policies if necessary. They can specify auto-scaling parameters based on CPU utilization or other metrics, and they can create an instance group based on the virtual machine. Lastly, it's critical that the user use Google Cloud Monitoring tools to track and improve the virtual machine's performance. Depending on the performance and load of the application, resource adjustments, like raising CPU or memory, can be made. These steps will help the user deploy the backend of their file-sharing application, efficiently set up and configure a virtual machine using Google Compute Engine, and guarantee appropriate resource allocation and scaling options.

Storage Solutions in Google Cloud

To implement file storage using Google Cloud Storage (GCS), the user begins by creating a storage bucket. This involves navigating to the “Storage” section in the Google Cloud Console and clicking on “Create Bucket.”

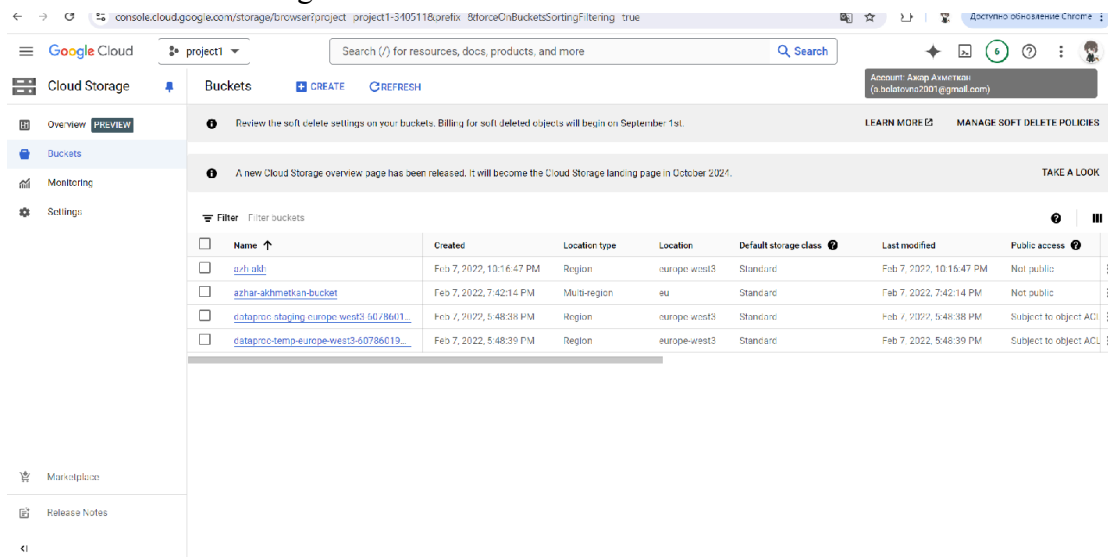


Image 2.1

After that, the user gives the bucket a distinctive name, decides on a suitable location, and sets the default storage class—for example, Standard for frequently accessed files—according to how frequently the data will be accessed. Before clicking "Create" to complete the setup, it is also crucial to set permissions and access controls to decide who can view or edit the files in the bucket.

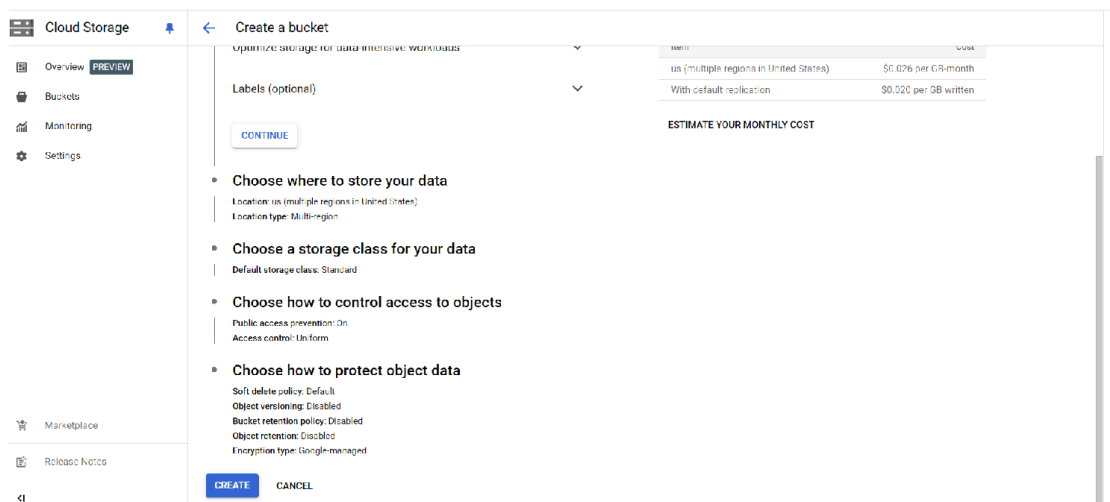


Image 2.2

The user then configures permissions for Identity and Access Management (IAM) to guarantee safe access. To do this, go to the console's "IAM & Admin" section and assign roles to users or service accounts that will need access to the bucket. For example, Storage Object Admin can be assigned to users who need complete control over the objects.

After that, the user creates a file management system with capabilities for managing, downloading, and uploading files. Creating an interface or API endpoint to allow users to upload files to the GCS bucket is part of this system architecture. The user can use any

programming language, like Python or Node.js, to upload files by utilizing the Google Cloud Storage client library.

For example, a Python function can be implemented to upload files using the following code:

```
from google.cloud import storage
def upload_file(bucket_name, source_file_name, destination_blob_name):
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_filename(source_file_name)
```

By ensuring that permissions are checked before granting access, the user implements a similar function to enable users to download files. An illustration of a file download function could resemble this:

```
def download_file(bucket_name, source_blob_name, destination_file_name):
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(source_blob_name)
    blob.download_to_filename(destination_file_name)
```

The user also creates features that let users view their stored files by listing them in the bucket. Additionally, they incorporate features for renaming and deleting files, making sure that these actions adhere to permissions and are appropriately recorded. The user uses the monitoring tools in Google Cloud Console to keep tabs on expenses and usage in order to maintain effective storage use. In order to optimize storage costs, they can also configure lifecycle rules in GCS to automatically delete or transition files according to their age or access patterns. These procedures enable the user to successfully use Google Cloud Storage for file storage, creating a reliable system for file management, uploading, and downloading that guarantees effective access to and utilization of storage resources.

Networking in Google Cloud

The user starts by setting up a Virtual Private Cloud (VPC) in Google Cloud in order to efficiently manage networking for the application. This entails going to the Google Cloud Console's "VPC network" section.

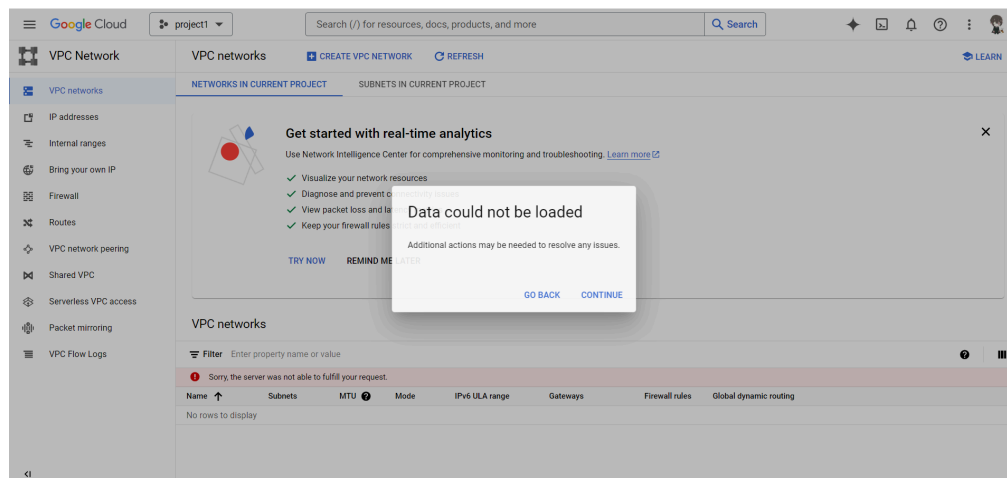


Image 3.1

After choosing to create a new VPC, the user specifies its name, region, and network mode. Because it automatically creates subnets in each region, the "Automatic" mode is appropriate for the majority of applications. The user can choose "Custom" mode and manually configure the necessary subnets, though, if greater subnet control is required.

The user sets up firewall rules after the VPC is created to guarantee safe access to the virtual machines and services inside the VPC. In order to accomplish this, go to the "Firewall rules" section and select "Create Firewall Rule." The name, action (allow or deny), traffic direction (ingress or egress), and the protocols and ports that will be impacted are all specified by the user. For instance, the user can configure the protocols to permit traffic on ports 80 and 443 for HTTP and HTTPS.

The user can also specify IP ranges for permitted sources, which can include all IPs (0.0.0.0/0) for public access, particular IP addresses, or ranges. To reduce security risks, the principle of least privilege must be applied when defining these rules. The user configures load balancing after setting up the firewall rules in order to maximize performance and guarantee the application's high availability. To accomplish this, go to the Google Cloud Console's "Load balancing" section and choose "Create Load Balancer." The user follows the instructions to configure frontend settings, backend services, and health checks after selecting the appropriate load balancer type: HTTP, HTTPS, TCP, or UDP based on the needs of the application. By choosing the instance groups that will manage incoming traffic, the user designates the backend service. While frontend settings, such as IP address and port configurations, specify how users will access the application, health checks are set up to guarantee that traffic is only routed to healthy instances. By taking these actions, the user effectively establishes a Virtual Private Cloud, sets up load balancing, and configures necessary firewall rules to improve the application's security and performance, guaranteeing a stable networking environment in Google Cloud.

Identity and Security Management

The application has implemented the Identity and Access Management (IAM) system in Google Cloud to properly manage access and protect data. First, the user creates an IAM role, which defines access rights for different users and groups. In the Google Cloud Console, he goes to the IAM and Administration section,

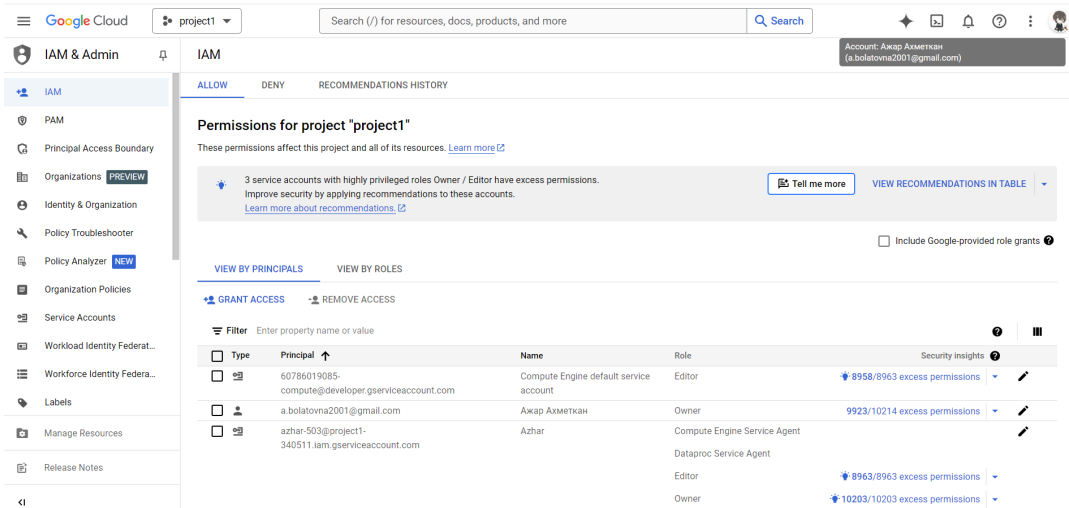


Image 4.1

adding new users and assigning them the appropriate roles, such as "Viewer", "Editor" or "Owner". This allows you to set clear levels of access to resources and manage user rights depending on their roles. There are user roles for more granular access control with specific permissions required to perform specific tasks. This helps avoid excessive rights and supports the principle of least privilege, which significantly reduces security risks.

In addition to setting up IAM, additional measures have been implemented to protect data. To ensure the security of information both during transmission and at the storage stage, encryption has been used. All data transmitted between clients and servers is encrypted using the HTTPS protocol, which protects it from interception. Built-in encryption is also used for data stored in Google Cloud Storage, providing additional protection.

Additionally, the user has set up data access control using security policies that restrict access to certain resources based on user roles. This allows precise control over who has the right to view, modify, or delete data, protecting sensitive information. Thus, implementing IAM and applying comprehensive security measures ensures reliable access control and data protection, which is critical to the stability and security of an application in Google Cloud.

Testing and Quality Assurance

Testing and maintenance of quality are key stages in the development of the application, since they help to identify and correct errors, ensuring its reliability and functionality. The testing process consists of several important steps. The first step is unit tests. At this stage, the developers create and launch tests for individual application modules to check the correctness of each function and method. These tests allow you to detect errors in the early stages of development, which guarantees the correct functioning of the components and facilitates the further completion of the code. The next step is integration tests. This process is aimed at checking the interaction between different components of the application. Integration tests are necessary to confirm that the modules work correctly together and correctly exchange data. This is especially important for identifying problems that may occur when combining individual parts of the system. The third step is load testing. As part of this stage, tests are carried out to evaluate the performance of the application at a high level of traffic. Excessive tests help to understand how the system copes with a large number of simultaneously connected users and requests, which is critical to ensure the stability and reliability of the application in real operating conditions.

Thus, testing and maintenance of quality-from unit testing to load testing-contribute to the creation of a reliable and effective application. These processes help developers make sure that the application meets high quality standards, which, in turn, increases user satisfaction and contributes to the successful operation of the product in the market.

Monitoring and Maintenance

Monitoring and maintenance play a critical role in ensuring the stability and performance of the application. These processes are necessary to maintain its effective work and a quick reaction to emerging problems. As a result, the stages of monitoring and maintenance - from setting up monitoring systems to planning regular updates - play an important role in ensuring reliable operation of the application and effective resource management. This, in turn, helps to increase the satisfaction of users and the stability of the system.

To achieve reliable operation and performance of the application, various tools and strategies are used. One of the key tools is Google Cloud Monitoring. This service allows in real time to monitor the state of the application and its performance, providing data on indicators such as loading the processor, the use of RAM, the state of disk space and network traffic. With it, you can create individual panels to visualize these metrics, which simplifies the identification of narrow places and anomalies. Another important tool is Stackdriver Logging, which integrates with Google Cloud Monitoring. It collects data magazines from applications and services, which allows developers to analyze them to diagnose and understand the behavior of the application. The setting of notifications based on certain events in the magazines helps to proactively discover problems. In addition, it is worth considering the use of Application Performance Monitoring (APM) Tools, such as New Relic or Datadog. These tools provide a deep analysis, providing information about transactions, response time and interaction with users, which helps to identify slow components and optimize user experience. To control the accessibility of the application is to use Monitoring Services, such as Pingdom or UptimeRobot. They regularly check whether the application is available and whether it works correctly, sending notification in case of problems.

As for service strategies, regular updates play an important role. It is necessary to create a schedule for planned updates, including the installation of software updates, security patches and improvements for both the application and its infrastructure. This helps to protect the system from vulnerabilities and guarantee compatibility with the latest technologies. Also important are periodic performance reviews. Regular assessments of the metrics of performance and the state of the application allow you to make reasonable decisions on the distribution of resources, needs in scale and possible optimization. It is recommended to implement backup plans for copying and recovery. An effective backup strategy guarantees the regular maintenance of critical information and the possibility of its recovery in case of data loss. Regular testing of recovery procedures is also necessary to confirm their reliability. Finally, the active collection and analysis of feedback from users can provide valuable information about the performance and convenience of the application. These data will help in planning future updates and improvements.

In general, a combination of effective monitoring tools and proactive service strategies is critical of maintaining the performance and reliability of the application. Constant metrics and regular updates contribute to the creation of a comfortable experience for users and rapidly respond to emerging problems.

Challenges and Solutions

There were a few simple but significant challenges during the project. One of the main challenges was billing management. Google Cloud charges for its services, and sometimes the costs exceeded expectations. This created some financial difficulties and required budget revisions, which complicated the process.

There were times when the format did not meet the requirements, which prevented further work. To solve these problems, time had to be spent on re-uploads and correcting formats, which delayed the overall process. In addition, setting up virtual machines using Google Compute Engine also caused some difficulties. Sometimes it was difficult to properly allocate resources and ensure scalability. To solve this problem, attention was paid to assessing the needs of the application, which helped to choose the right configurations.

Despite all these difficulties, it was possible to adapt to the situation and find solutions, which ultimately allowed us to successfully complete the project of creating a file sharing application on the Google Cloud platform.

Conclusion

This task analyzed an understanding of the principles and advantages of cloud computing, with an emphasis on deployment models, such as infrastructure as a service (IAAS), a platform as a service (PAAS) and software as a service (SAAS). Awareness of these concepts is critical of understanding how cloud services improve the process of application development, offering flexibility, scalability and cost reduction. In the course of the work, the key services of Google Cloud Platform (GCP) were studied, including computing, storage and network services, as well as databases. For the project, the main services such as Google Compute Engine were selected for control of virtual machines, Google Cloud Storage for storing files and Google Cloud IAM for security management, which made it possible to create an effective file exchange application.

The project included the configuration of virtual machines for the deployment of the application bachelor, which ensured the correct distribution of resources and the possibility of scale. The file storage system was successfully implemented, which provides convenient loading and downloading. The creation of a Virtual Private Cloud (VPC) contributed to reliable control of network resources, and setting up the rules of firewall and load balancing increased both performance and safety. It also emphasized the value of the Identity and Access Management (IAM) to monitor resolutions and roles of users, along with the introduction of the best practices of encryption and data protection.

So, this task not only provided practical experience with GCP services, but also demonstrated the importance of cloud computing in the modern development of applications. The acquired knowledge and skills will become useful for future projects in the field of cloud solutions, as technologies continue to contribute to the creation of scalable and safe applications.

References

- Google Cloud. *Google Cloud Compute Engine documentation*. Retrieved from <https://cloud.google.com/compute/docs>
- Google Cloud. *Google Cloud Storage documentation*. Retrieved from <https://cloud.google.com/storage/docs>
- Google Cloud. *Identity and Access Management (IAM) documentation*. Retrieved from <https://cloud.google.com/iam/docs>
- Google Cloud. *Networking in Google Cloud documentation*. Retrieved from <https://cloud.google.com/networking/docs>
- Google Cloud. *Google Cloud Monitoring*. Retrieved from <https://cloud.google.com/monitoring>
- Google Cloud. *Stackdriver Logging documentation*. Retrieved from <https://cloud.google.com/logging/docs>
- IBM Cloud. *Understanding IaaS, PaaS, and SaaS*. Retrieved from <https://www.ibm.com/cloud/learn/iaas-paas-saas>
- Cloudflare. *What is monitoring?* Retrieved from <https://www.cloudflare.com/learning/how-to/what-is-monitoring/>
- Guru99. *Software Testing Types: A Detailed Guide*. Retrieved from <https://www.guru99.com/software-testing.html>
- Software Testing Help. *What is Load Testing?* Retrieved from <https://www.softwaretestinghelp.com/load-testing/>
- Amazon Web Services. *What is unit testing?* Retrieved from <https://aws.amazon.com/what-is/unit-testing/#:~:text=Unit%20testing%20is%20the%20process,test%20for%20each%20code%20unit.>
- The CTO Club. *Best application monitoring software*. Retrieved from <https://thectoclub.com/tools/best-application-monitoring-software/>
- StackShare. *Pingdom vs. UptimeRobot*. Retrieved from <https://stackshare.io/stackups/pingdom-vs-uptimerobot#:~:text=Pingdom%20provides%20features%20like%20transaction,with%20limited%20advanced%20features%20available.>
- Microsoft Azure. *Virtual Machines*. Retrieved from <https://azure.microsoft.com/en-us/services/virtual-machines/>
- Microsoft Azure. *App Service*. Retrieved from <https://azure.microsoft.com/en-us/services/app-service/>
- Heroku. *Heroku Platform*. Retrieved from <https://www.heroku.com/>
- Google Workspace. *What is Google Workspace?* Retrieved from <https://workspace.google.com/>

Appendices

Appendix A: Overview of Cloud Service Models

Service Model	Definition	Notable Examples	Typical Applications
IaaS	Infrastructure as a Service; offers virtualized computing resources over the internet.	Amazon EC2, Google Compute Engine, Microsoft Azure VMs	Suitable for hosting websites, data storage, and backup solutions.
PaaS	Platform as a Service; provides hardware and software tools via the internet, primarily for application development.	Google App Engine, Heroku, Microsoft Azure App Service	Used for developing and deploying applications without managing the underlying infrastructure.
SaaS	Software as a Service; delivers software applications over the internet on a subscription basis.	Google Workspace, Salesforce, Dropbox	Commonly used for collaboration tools, customer relationship management (CRM), and email services.

Appendix B: Overview of Cloud Service Models

Service	Description	Configuration Details
Google Compute Engine	Virtual machines used for the application backend	n1-standard-1 VM, 2 CPUs, 7.5 GB RAM, Ubuntu 20.04 OS.
Google Cloud IAM	Manages user permissions and roles for enhanced security.	Roles assigned: Viewer, Editor, and a custom role for admin access.
Google Cloud Storage	Object storage solution for handling files.	Storage class: Standard, multi-region location.
Google VPC	Isolated network for secure application operation.	Custom subnet with firewall rules for traffic control.

Appendix C: Example Code Snippets for Google Cloud Storage Operations

Uploading a File to Google Cloud Storage:

```
from google.cloud import storage
def upload_file(bucket_name, source_file_name, destination_blob_name):
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_filename(source_file_name)
```

Downloading a File from Google Cloud Storage:

```
def download_file(bucket_name, source_blob_name, destination_file_name):  
    storage_client = storage.Client()  
    bucket = storage_client.bucket(bucket_name)  
    blob = bucket.blob(source_blob_name)  
    blob.download_to_filename(destination_file_name)
```

Listing Files in a Bucket:

```
def list_files(bucket_name):  
    storage_client = storage.Client()  
    blobs = storage_client.list_blobs(bucket_name)  
    for blob in blobs:  
        print(blob.name)
```

Deleting a File from Google Cloud Storage:

```
def delete_file(bucket_name, blob_name):  
    storage_client = storage.Client()  
    bucket = storage_client.bucket(bucket_name)  
    blob = bucket.blob(blob_name)  
    blob.delete()
```