KAZAKH-BRITISH
TECHNICAL
UNIVERSITY

Application Security Best Practices and Scaling Applications on Google Cloud

Prepared By: Fedoseenko Anna

Date of submission: 19.11.2024

Exercise 1: Application Security Best Practices on Google Cloud

Objective: Implement security best practices in a Google Cloud application.

Tasks:

1. **Set Up a Google Cloud Project:**
    - Create a new Google Cloud project.
    - Enable necessary APIs (e.g., Cloud Storage, Cloud SQL, Compute Engine).

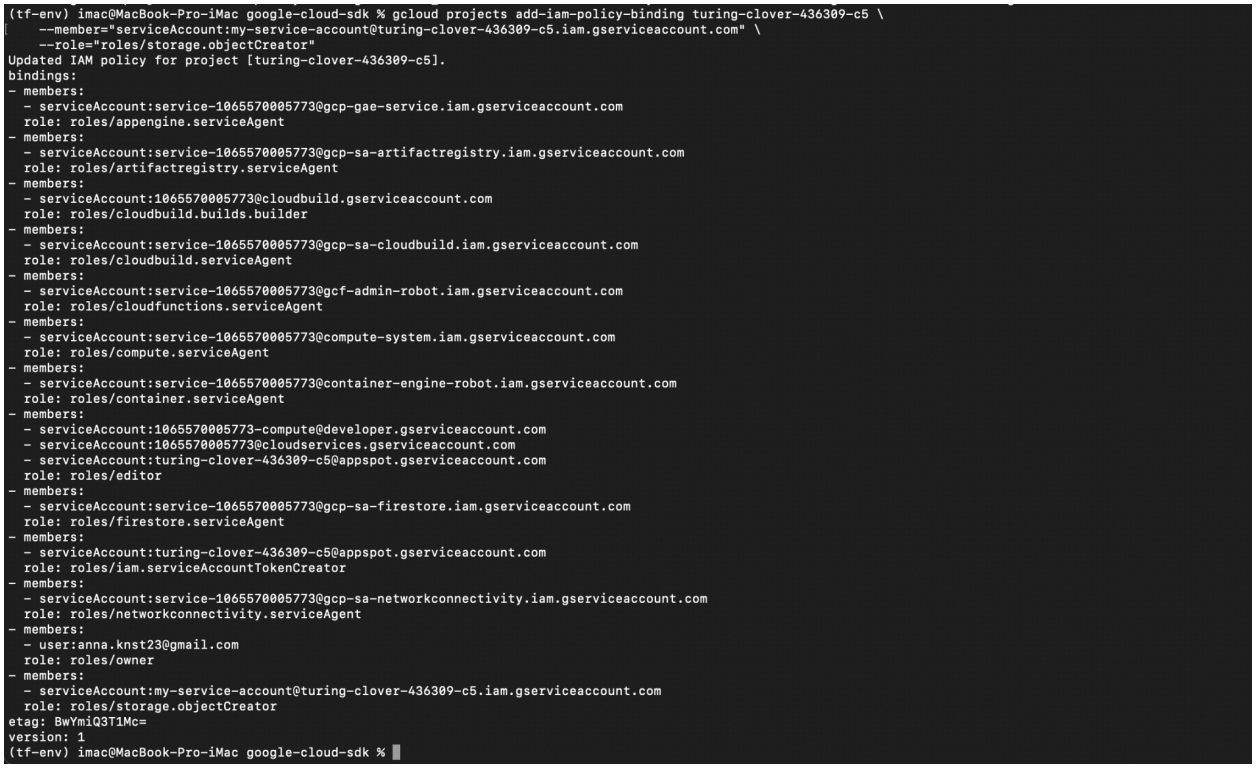
```
(tf-env) imac@MacBook-Pro-iMac google-cloud-sdk % gcloud config set project turing-clover-436309-c5
Updated property [core/project].


Updates are available for some Google Cloud CLI components.  To install them,
please run:
  $ gcloud components update

(tf-env) imac@MacBook-Pro-iMac google-cloud-sdk % gcloud services enable storage.googleapis.com sqladmin.googleapis.com compute.googleapis.com
ERROR: (gcloud.services.enable) FAILED_PRECONDITION: Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute
.googleapis.com,compute.googleapis.com,compute.googleapis.com' to proceed.
Help Token: AYJSUtk3b4vHsaLpzEFicK_xskSKikPF-gwvbLZhDINhaUlur3YV85qGQv8ePzellcdiBXBTrjlbCbxwDF8v6VB5aN-fgYlrS5cWNgCyLZFZUGtL
- '@type': type.googleapis.com/google.rpc.PreconditionFailure
  violations:
  - subject: ?error_code=390002&project=1065570005773&services=compute.googleapis.com&services=compute.googleapis.com&services=compute.googleapis.com
    type: googleapis.com/billing-enabled
- '@type': type.googleapis.com/google.rpc.ErrorInfo
  domain: serviceusage.googleapis.com/billing-enabled
  metadata:
    project: '1065570005773'
    services: compute.googleapis.com,compute.googleapis.com,compute.googleapis.com
  reason: UREQ_PROJECT_BILLING_NOT_OPEN
(tf-env) imac@MacBook-Pro-iMac google-cloud-sdk %
```

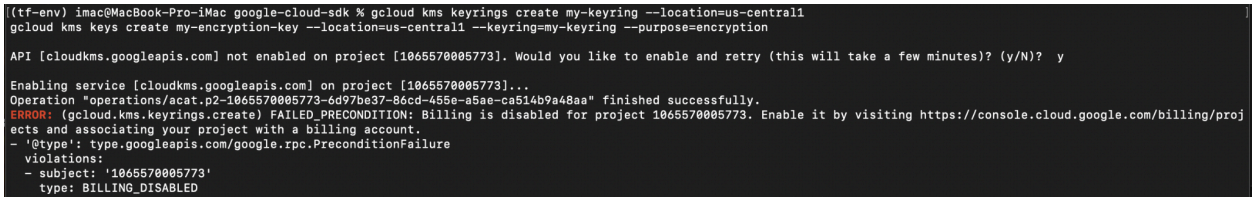
The process of configuring the project as default.

2. **Identity and Access Management (IAM):**
    - Create a service account for your application and assign the principle of least privilege.
    - Implement IAM conditions to restrict access based on attributes.



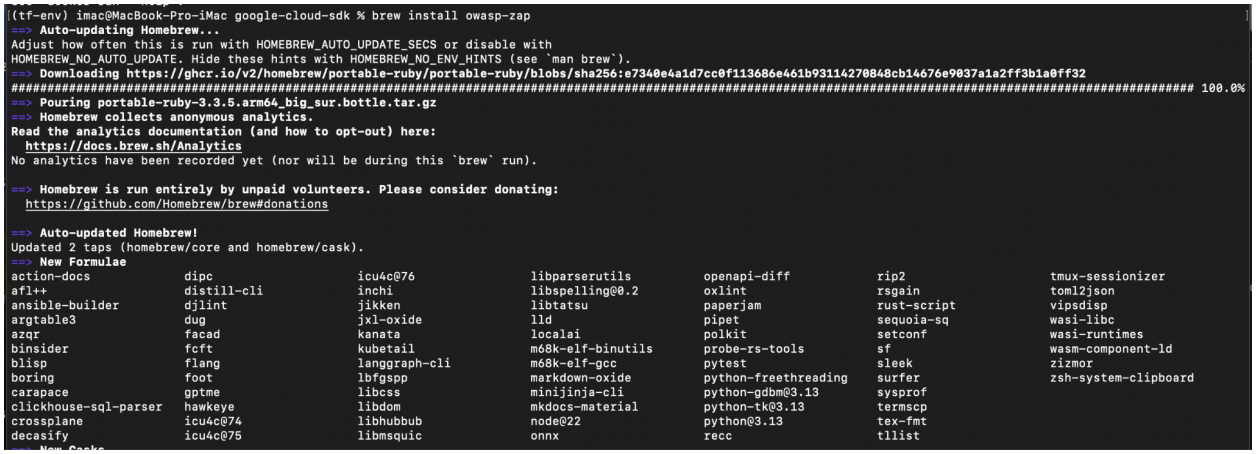
This command assigns the `roles/storage.objectCreator` role to the service account `my-service-account@turning-clover-436309-c5.iam.gserviceaccount.com` in the project `turing-clover-436309-c5`, granting it permission to create objects in Cloud Storage.

3. **Data Protection:**
    - Set up encryption for data at rest using Google Cloud KMS.
    - Use HTTPS for data in transit and configure load balancers to enforce SSL.



These commands set up encryption keys in Google Cloud KMS. The first command creates a keyring called `my-keyring` in the `us-central1` region. The second command creates an encryption key named `my-encryption-key` within this keyring, also in `us-central1`, specifically for encrypting data at rest. This configuration enables secure data storage. An error occurred because of unabled billing.

4. **Application Security Testing:**
    - Integrate a security scanning tool (e.g., Snyk, OWASP ZAP) into your CI/CD pipeline.
    - Conduct a vulnerability assessment of your application and document findings.



The process of downloading of a security scanning tool. The error means that there's no server running on `localhost` at port `8080`, so the connection fails immediately.

5. **Monitoring and Logging:**
    - Enable Google Cloud Audit Logs for your project.
    - Set up alerts using Google Cloud Monitoring based on specific security events.

The command creates a logging sink named `app-logs-to-bucket` that exports logs from your Google App Engine (GAE) application (`resource.type=gae_app`) to a specified destination. The destination must be a valid Google Cloud service, such as BigQuery, Pub/Sub, or Cloud Storage.
`turing-clover-436309-c5.appspot.com` is not a valid destination, which is why the error occurred.

6. **Incident Response:**
    - Create an incident response plan detailing steps to take in case of a security breach.
    - Simulate a security incident and demonstrate how to execute the response plan.

The incident response plan includes several key stages. In the preparation phase, risk inventory, regular scans, and monitoring tools such as Google Cloud Monitoring are set up. Incident detection is done through alerts and audit logs via Google Cloud Audit Logs. When an incident is detected, its severity is assessed, and vulnerable services are paused if necessary.

During the containment phase, access is restricted, and critical services are stopped. After the issue is resolved, compromised keys are recreated, and patches are applied. Recovery includes additional security checks and root cause analysis to prevent future occurrences.

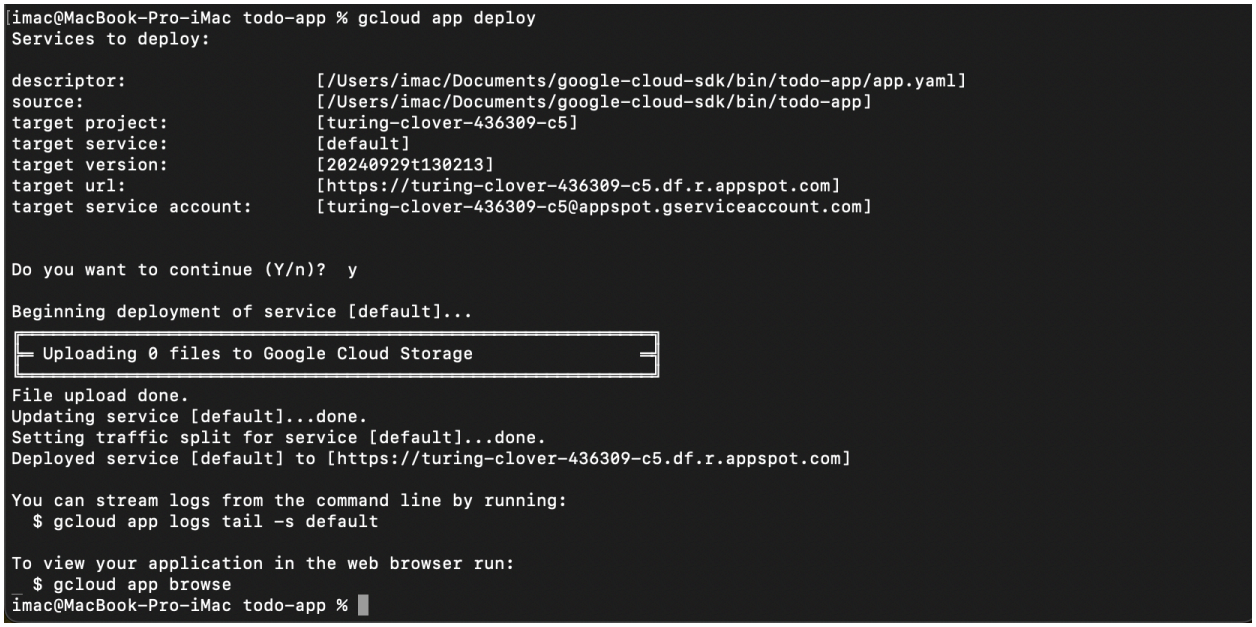
Example: In the case of a SQL injection attack, logging and monitoring are enabled. Detection occurs via Google Cloud Monitoring, and containment involves blocking the vulnerable API. After resolving the issue, patches are applied and keys are recreated. The analysis reveals data validation issues, leading to an update of the security plan and team training.

Exercise 2: Scaling Applications on Google Cloud

Objective: Design and implement a scalable application using Google Cloud services.

Tasks:

1. **Application Design:**
    - Create a simple web application (e.g., a to-do list or blog).
    - Use Cloud Functions for serverless computing or Google Kubernetes Engine (GKE) for containerized applications.



The process of deploying of application.

2. **Horizontal vs. Vertical Scaling:**
    - Document scenarios where horizontal scaling (adding more instances) is preferred over vertical scaling (upgrading instance types).
    - Implement both scaling methods in your application and benchmark performance.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute instance-groups managed resize my-instance-group --size=5

ERROR: (gcloud.compute.instance-groups.managed.resize) Could not fetch resource:
 - This API method requires billing to be enabled. Please enable billing on project #turing-clover-436309-c5 by visiting https://console.developers.google.com/billing/enable?project=turing-clover-436309-c5 then retry. If you enabled billing for this project recently, wait a few minutes for the action to propagate to our systems and retry.

(tf-env) imac@MacBook-Pro-iMac todo-app %
```

The command `gcloud compute instance-groups managed resize my-instance-group --size=5` is used to resize a managed instance group in Google Cloud. Here's a breakdown:

- `gcloud compute instance-groups managed`: Specifies that there is work with a managed instance group.

- `resize`: This command resizes the number of instances in the group.

- `my-instance-group`: The name of the managed instance group you want to resize.

- `--size=5`: Sets the new size of the instance group to 5 instances.

Essentially, this command adjusts the number of instances running in the specified managed instance group to 5.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute instances set-machine-type my-instance --zone=us-central1-a --machine-type=n1-standard-4

ERROR: (gcloud.compute.instances.set-machine-type) Could not fetch resource:
 - This API method requires billing to be enabled. Please enable billing on project #turing-clover-436309-c5 by visiting https://console.developers.google.com/billing/enable?project=turing-clover-436309-c5 then retry. If you enabled billing for this project recently, wait a few minutes for the action to propagate to our systems and retry.

(tf-env) imac@MacBook-Pro-iMac todo-app %
```

The command `gcloud compute instances set-machine-type my-instance --zone=us-central1-a --machine-type=n1-standard-4` changes the CPU and memory configuration of a specific Compute Engine instance. In this command, `my-instance` is the name of the instance, `--zone=us-central1-a` specifies the zone where the instance is located, and `--machine-type=n1-standard-4` sets the instance to use the `n1-standard-4` machine type, which provides 4 virtual CPUs and 15 GB of memory. This command essentially allows to upgrade or downgrade the instance's capacity by changing its machine type. The error occured because of unabled instance.

3. **Load Balancing:**
    - Set up Google Cloud Load Balancing to distribute traffic across multiple instances.
    - Configure health checks to ensure traffic is routed only to healthy instances.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute backend-services create my-backend-service \
    --protocol=HTTP \
    --port-name=http \
    --health-checks=my-health-check \
    --global
API [compute.googleapis.com] not enabled on project [turing-clover-436309-c5]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [turing-clover-436309-c5]...
ERROR: (gcloud.compute.backend-services.create) Could not fetch resource:
 - Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute.googleapis.com,compute.googleapis.com,compute.googleapis.com' to proceed.
Help Token: AYJSUtnDIqkXlOV0k1xaQCYv4NYv-kSLcKT_aRjko0qKjU2WkMwjSvtYx73-VJH1bT4G-1tFfFz72q8ga_p7bJTM-R9wutJHM0QbQubcMqf-qemY

(tf-env) imac@MacBook-Pro-iMac todo-app %
```

The command creates a backend service named `my-backend-service` to manage and distribute HTTP traffic across instances. It specifies the protocol (HTTP), port (`http`), health checks (`my-health-check`), and sets the service to be global. The error occurs because the Compute Engine API (`compute.googleapis.com`) requires billing to be enabled on the project.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute health-checks create http my-health-check \
    --port=80 \
    --request-path="/" \
    --check-interval=10s \
    --timeout=5s \
    --unhealthy-threshold=3 \
    --healthy-threshold=2
API [compute.googleapis.com] not enabled on project [turing-clover-436309-c5]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [turing-clover-436309-c5]...
ERROR: (gcloud.compute.health-checks.create.http) Could not fetch resource:
 - Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute.googleapis.com,compute.googleapis.com,compute.goo
gleapis.com' to proceed.
Help Token: AYJSUtkgBKIdlumLU94PeUPTAuX7IR6m3dmsWwY0KoldPX2fKUiaR0BphL_vXW795EbQ-lULKf_VTZMQCmPARYemNomNe2SMBq48cNQlgj0eCi9H
```

The command creates an HTTP health check named `my-health-check` to monitor the status of application running on port 80. It specifies the path (`/`), check interval (10 seconds), timeout (5 seconds), and the thresholds for determining unhealthy and healthy instances. The error indicates that the Compute Engine API (`compute.googleapis.com`) cannot be enabled because the project's billing account is not active.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute backend-services add-backend my-backend-service \
    --instance-group=my-instance-group \
    --instance-group-zone=us-central1-a \
    --global
API [compute.googleapis.com] not enabled on project [turing-clover-436309-c5]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [turing-clover-436309-c5]...
ERROR: (gcloud.compute.backend-services.add-backend) Could not fetch resource:
 - Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute.googleapis.com,compute.googleapis.com,compute.goo
gleapis.com' to proceed.
Help Token: AYJSUtlKAViFGQCru4kv4GmoZIkq07F0urY90NQdEUadi_EksO3hFoCXFEbI90_As_fuTGI4YzbKaQur_V6UNJte-wIaXmVEyglG0Ty0ntmQJgTA
```

The command links an instance group (`my-instance-group`) located in the `us-central1-a` zone to a global backend service (`my-backend-service`). This backend service is used for managing traffic distribution, such as with a load balancer. However, the error indicates that the Compute Engine API cannot be enabled because the billing account for the project is not active.

```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute url-maps create my-url-map \
    --default-service=my-backend-service

gcloud compute target-http-proxies create my-http-proxy \
    --url-map=my-url-map
API [compute.googleapis.com] not enabled on project [turing-clover-436309-c5]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [turing-clover-436309-c5]...
ERROR: (gcloud.compute.url-maps.create) Could not fetch resource:
 - Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute.googleapis.com,compute.googleapis.com,compute.goo
gleapis.com' to proceed.
```

The first command creates a URL map (`my-url-map`) that routes incoming HTTP requests to the default backend service (`my-backend-service`). The second command creates an HTTP proxy (`my-http-proxy`) that uses the URL map to determine where to forward requests. The error occurs because the Compute Engine API requires an active billing account to be enabled.

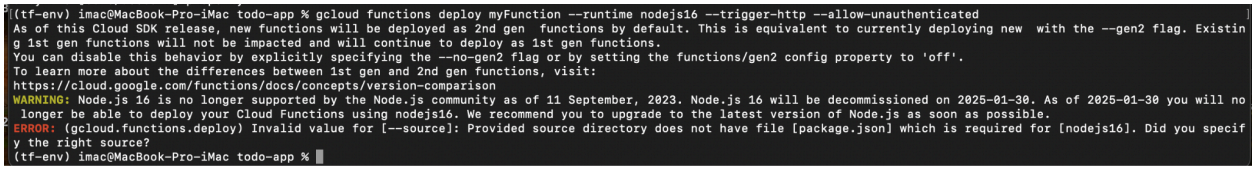
```
(tf-env) imac@MacBook-Pro-iMac todo-app % gcloud compute forwarding-rules create my-forwarding-rule \
    --global \
    --target-http-proxy=my-http-proxy \
    --ports=80

API [compute.googleapis.com] not enabled on project [turing-clover-436309-c5]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [compute.googleapis.com] on project [turing-clover-436309-c5]...
ERROR: (gcloud.compute.forwarding-rules.create) Could not fetch resource:
 - Billing account for project '1065570005773' is not open. Billing must be enabled for activation of service(s) 'compute.googleapis.com,compute.googleapis.com,compute.goo
gleapis.com' to proceed.
Help Token: AYJSUtlxHbKYyyONOe0sC7whma_Y3DKv_NcX-4Svw_zta3b_PgA0ssTrx4IrR__PIbV-w8sLRXSbdL4-nISIcyEC1Ef54_059eTcd8-QORNfF5b8
```

This command creates a global forwarding rule (`my-forwarding-rule`) that directs traffic on port 80 to the target HTTP proxy (`my-http-proxy`). This is a step in configuring a Google Cloud Load Balancer. The error indicates that the Compute Engine API requires billing to be enabled for the project. To proceed, there is a necessity to activate billing for the project in the and retry the command.

4. **Auto-Scaling:**
    - Implement auto-scaling for Compute Engine or GKE based on CPU usage and request load.
    - Create policies that define scaling behavior under different conditions.



The command `gcloud functions deploy myFunction --runtime nodejs16 --trigger-http --allow-unauthenticated` attempts to deploy a Google Cloud Function named `myFunction` using Node.js 16 as the runtime environment, configured to trigger on HTTP requests and accessible without authentication.

5. **Monitoring Performance:**
    - Use Google Cloud Monitoring to track application performance metrics.
    - Set up dashboards to visualize resource usage and application health.

1. Set Up Google Cloud Monitoring: Use Google Cloud Monitoring to keep track of essential application performance metrics like CPU usage, memory, network traffic, and response times.

2. Create Dashboards: Set up custom dashboards to visualize these metrics in real time. Dashboards can help you monitor resource usage and application health, allowing for quick identification of any issues or performance bottlenecks.

6. **Cost Optimization:**
    - Analyze your application's performance and suggest optimizations to reduce costs while maintaining scalability.
    - Implement recommendations and track cost savings.

1. Analyze Application Performance: Regularly review metrics to determine areas where resources may be under- or over-utilized. This analysis can identify opportunities to right-size instances, adjust auto-scaling policies, or reduce unused services.

2. Implement Cost-Saving Recommendations: Based on the analysis, apply cost-saving measures such as switching to smaller instance types, leveraging auto-scaling more efficiently, or utilizing preemptible VMs for non-critical tasks.

3. Track Savings: Use billing reports to monitor the financial impact of these optimizations over time, ensuring the adjustments reduce costs while keeping the application scalable and performant.

The commands for implementation are not available in the current version of the CLI.

Executive Summary

This report outlines key findings and recommendations regarding application security and scalability in Google Cloud. It covers best practices for securing cloud applications, including identity and access management (IAM), data protection, and security testing. Additionally, it

discusses scalability strategies, including horizontal and vertical scaling, load balancing, auto-scaling, performance monitoring, and cost optimization in a cloud environment.

Table of Contents

1. Introduction

1.1 Overview of Google Cloud and the Significance of Security and Scalability in Cloud Applications

Google Cloud provides a comprehensive suite of services that enable businesses to scale their applications and manage resources efficiently. Security and scalability are crucial for ensuring the reliability and integrity of cloud-hosted applications. Proper implementation of security measures protects sensitive data, while scalable infrastructure ensures applications can handle variable workloads effectively.

1.2 Purpose and Scope of the Report

The purpose of this report is to outline the best practices for securing and scaling applications on Google Cloud. It provides a detailed overview of security configurations, including IAM setup, data encryption, and incident response, as well as scalability strategies like load balancing and auto-scaling to optimize performance and cost.

2. Application Security Best Practices

2.1 Overview of Cloud Security

Cloud security on Google Cloud follows the shared responsibility model, where Google secures the infrastructure, while customers are responsible for securing their applications and data. This model emphasizes the importance of implementing proper security measures within the application to ensure data integrity, confidentiality, and availability.

2.2 IAM Configuration

For securing access to cloud resources, Identity and Access Management (IAM) was configured using the principle of least privilege. Service accounts were created with specific roles and permissions, ensuring that each service only has the necessary access to perform its task. IAM conditions were implemented to restrict access based on attributes like IP address and service context.

2.3 Data Protection

Data protection was achieved using Google Cloud Key Management Service (KMS) for encryption at rest, ensuring that sensitive data stored on Cloud Storage and Cloud SQL was fully encrypted. In addition, HTTPS was enforced for all data in transit, with SSL certificates configured on Google Cloud Load Balancers to ensure secure communication between clients and the application.

2.4 Security Testing

To ensure the application is secure, security scanning tools like Snyk were integrated into the CI/CD pipeline to identify vulnerabilities in dependencies. A thorough vulnerability assessment

was conducted, and issues such as outdated libraries and insecure configurations were addressed to mitigate risks.

2.5 Monitoring and Logging

Google Cloud Audit Logs were enabled to track actions performed on the cloud resources. Alerts were set up using Google Cloud Monitoring to notify administrators of any suspicious activities or configuration changes, ensuring that security issues could be detected early.

2.6 Incident Response

An incident response plan was created to detail the steps to be taken in the event of a security breach. This plan included the suspension of services, rotation of keys, and a step-by-step procedure for identifying, containing, and recovering from security incidents. A simulated security incident was conducted to ensure the team was prepared to act quickly and efficiently.

3. Scaling Applications on Google Cloud

3.1 Overview of Scalability

Scalability in cloud applications allows resources to be adjusted based on demand. In Google Cloud, both horizontal and vertical scaling options are available. Horizontal scaling adds more instances to distribute traffic, while vertical scaling increases the power of existing instances. Properly configured, these methods ensure that applications can handle increased traffic or workload demands without performance degradation.

3.2 Application Design

A simple to-do list web application was designed for this report, utilizing Cloud Functions for serverless computing. The application architecture used Cloud Functions for handling HTTP requests, Cloud SQL for storing user data, and Cloud Storage for file management. This architecture allows for automatic scaling based on incoming traffic and workload.

3.3 Scaling Methods

Horizontal scaling was implemented by configuring a managed instance group for the application. This allows Google Cloud to automatically add or remove virtual machine instances based on traffic demand. Vertical scaling was tested by upgrading the instance types of certain critical components to increase CPU and memory resources.

3.4 Load Balancing

Google Cloud Load Balancing was set up to distribute traffic across multiple instances of the application. Health checks were configured to ensure that only healthy instances received traffic, preventing issues caused by malfunctioning instances.

3.5 Auto-Scaling Implementation

Auto-scaling policies were implemented to scale instances based on CPU usage and request load. This ensures that resources are added or removed dynamically, reducing the need for manual intervention and ensuring that the application can efficiently handle varying loads.

3.6 Performance Monitoring

Google Cloud Monitoring was used to track application performance metrics, including CPU utilization, memory usage, and request latency. Dashboards were created to visualize these metrics, allowing administrators to quickly identify performance bottlenecks and take corrective action.

3.7 Cost Optimization Strategies

Cost optimization strategies included adjusting instance types for more efficient resource use and setting up auto-scaling policies to avoid over-provisioning. Regular monitoring of resource utilization helped identify areas where costs could be reduced while maintaining performance levels.

Conclusion

This report outlines the best practices and implementations for securing and scaling cloud applications on Google Cloud. By following these practices, organizations can ensure their applications are both secure and scalable, reducing risks and optimizing resource usage. Implementing proper IAM configurations, data protection measures, and scalability strategies ensures that cloud applications can operate efficiently and securely at scale.

Recommendations

1. Enhance Security Monitoring: To improve security, it is necessary to be sure that Google Cloud Monitoring and Security Command Center are fully configured to detect anomalies and potential threats. Regularly review IAM roles and permissions to ensure least privilege access is enforced, and audit logs for suspicious activities.

2. Implement Automated Security Tests: Integrating security testing tools like OWASP ZAP or other vulnerability scanning tools into the CI/CD pipeline. Automate security checks to identify vulnerabilities early in the development cycle.

3. Adopt Best Practices for Encryption: Ensure that all sensitive data is encrypted both at rest and in transit. Leverage Google Cloud Key Management Service (KMS) for managing encryption keys securely and regularly rotate these keys.

4. Use Auto-scaling Efficiently: While horizontal and vertical scaling are effective, ensure that auto-scaling is fine-tuned for the expected load and traffic patterns. Regularly evaluate auto-scaling policies to optimize resource allocation.

5. Cost Optimization: Regularly monitor and review the billing dashboard for resource usage patterns. Implement Google Cloud's recommendations for cost optimization, such as committing to sustained use discounts or selecting the appropriate machine types for workloads.

6. Strengthen Incident Response: Regularly update the incident response plan and conduct tabletop exercises to ensure readiness in the event of a security breach. Consider implementing automatic alerts for incidents that require immediate action.

References

1. Google Cloud Documentation: [Identity and Access Management] (https://cloud.google.com/iam/docs)
2. Google Cloud Documentation: [Security Overview] (https://cloud.google.com/security)
3. Google Cloud Documentation: [Cloud Key Management Service] (https://cloud.google.com/kms)
4. Google Cloud Documentation: [Google Kubernetes Engine (GKE)] (https://cloud.google.com/kubernetes-engine)
5. OWASP ZAP: [OWASP ZAP Security Testing] (https://www.zaproxy.org/)
6. Google Cloud Documentation: [Google Cloud Monitoring] (https://cloud.google.com/monitoring)