Viwa QA

EXPERIMENT NUMBER 1

1. **What is a virtual machine (VM) in the context of Google Cloud Platform (GCP)?**
   * A VM in GCP is an instance of a virtualized computing environment provided by Google Compute Engine, allowing users to run applications on customizable virtual hardware.
2. **How do you create a VM instance in Google Cloud with specific CPU and memory configurations?**
   * To create a VM with specific CPU and memory settings, you can use the Google Cloud Console or the gcloud command-line tool to specify the desired machine type or create a custom machine type that fits your requirements.
3. **What are custom machine types in Google Compute Engine, and when would you use them?**
   * Custom machine types allow you to tailor the number of virtual CPUs and the amount of memory to your application's specific needs, providing flexibility beyond predefined machine types.
4. **How does selecting different machine types impact the performance and cost of a VM in GCP?**
   * Choosing a machine type with more CPUs and memory can enhance performance but will also increase costs. It's essential to balance resource requirements with budget constraints.
5. **What are preemptible VM instances in GCP, and how do they differ from standard instances?**
   * Preemptible instances are short-lived, cost-effective VMs suitable for batch processing and fault-tolerant workloads, whereas standard instances are regular, long-lived VMs.
6. **How can you modify the CPU and memory allocation of an existing VM instance in GCP?**
   * To change the CPU and memory allocation of an existing VM, you need to stop the instance, adjust its machine type or customize its resources, and then restart it.
7. **What role do instance templates play in deploying VMs with specific configurations in GCP?**
   * Instance templates define the configuration for VM instances, including CPU, memory, and other settings, enabling consistent deployment across multiple instances.
8. **How do you ensure high availability when deploying VMs in Google Cloud?**
   * High availability can be achieved by deploying VMs across multiple zones or regions, using managed instance groups, and setting up load balancing to distribute traffic.
9. **What are the considerations for selecting the appropriate region and zone when deploying a VM in GCP?**
   * Consider factors like latency requirements, data residency regulations, available services, and pricing when choosing a region and zone for your VM deployment.
10. **How can you monitor and optimize the performance of a VM in Google Cloud?**
    * Utilize Google Cloud's monitoring tools to track resource usage, set up alerts for performance thresholds, and adjust CPU and memory allocations as needed to optimize performance.

EXPERIMENT NUMBER 2

**1. What is Cloud Shell in Google Cloud?**

**Answer:**  
Cloud Shell is an online, interactive command-line interface provided by Google Cloud that allows users to manage their Google Cloud resources without requiring local installations. It provides a pre-configured environment with gcloud, Python, and other essential tools.

**2. What is gcloud and why is it used?**

**Answer:**  
gcloud is the command-line tool for Google Cloud Platform (GCP). It enables users to create, configure, and manage GCP resources such as virtual machines, storage, and databases from the terminal.

**3. How do you activate Cloud Shell?**

**Answer:**  
Cloud Shell can be activated by clicking on the **Cloud Shell** icon in the Google Cloud Console, which opens an interactive terminal session in the browser.

**4. What is the command to check the currently active project in gcloud?**

**Answer:**  
The command to check the active project is:

sh

CopyEdit

gcloud config get-value project

**5. How do you set a new active project in gcloud?**

**Answer:**  
To set a new active project, use the command:

sh

CopyEdit

gcloud config set project [PROJECT\_ID]

Replace [PROJECT\_ID] with the actual project ID.

**6. How can you list all available projects in Google Cloud using gcloud?**

**Answer:**  
Use the following command to list all projects:

sh

CopyEdit

gcloud projects list

**7. How do you authenticate and login to gcloud?**

**Answer:**  
To authenticate and login, use:

sh

CopyEdit

gcloud auth login

This opens a browser window where you can sign in with your Google account.

**8. How do you create a new virtual machine instance using gcloud?**

**Answer:**  
To create a VM instance, use:

sh

CopyEdit

gcloud compute instances create my-vm --zone=us-central1-a --machine-type=e2-medium

Here, my-vm is the instance name, us-central1-a is the zone, and e2-medium is the machine type.

**9. How do you list all Compute Engine instances?**

**Answer:**  
To list all VM instances, run:

sh

CopyEdit

gcloud compute instances list

**10. How do you delete a Compute Engine instance using gcloud?**

**Answer:**  
To delete a VM instance, use:

sh

CopyEdit

gcloud compute instances delete my-vm --zone=us-central1-a

Replace my-vm with the actual instance name.

EXPERIMENT NUMBER 3

**1. What is a Cloud Function?**

**Answer:** A Cloud Function is a serverless, event-driven compute service that allows you to run code in response to events, such as file uploads to Cloud Storage, database changes, or HTTP requests, without managing infrastructure.

**2. What are the key steps to creating and deploying a Cloud Function in Google Cloud?**

**Answer:**

1. Write the function code (e.g., in Python or Node.js).
2. Specify the event trigger (e.g., Cloud Storage).
3. Deploy the function using the Google Cloud Console, gcloud CLI, or Terraform.
4. Test and monitor the function execution.

**3. How do you trigger a Cloud Function based on a Cloud Storage event?**

**Answer:** You can specify a Cloud Storage bucket as the event source and choose an event type such as:

* google.storage.object.finalize (file uploaded)
* google.storage.object.delete (file deleted)
* google.storage.object.archive (file archived)

**4. How do you deploy a Cloud Function using the gcloud CLI?**

**Answer:**

sh

CopyEdit

gcloud functions deploy my-function \

--runtime python310 \

--trigger-resource my-bucket \

--trigger-event google.storage.object.finalize

This command deploys a function named my-function in Python 3.10 that triggers on file uploads to my-bucket.

**5. What are the advantages of using Cloud Functions for automation?**

**Answer:**

* Serverless (no need to manage infrastructure)
* Scales automatically
* Event-driven execution
* Cost-efficient (pay only for execution time)

**6. How do you handle errors in a Cloud Function?**

**Answer:** Implement error handling using try-except blocks (Python) or try-catch (Node.js) and log errors using logging.error() for debugging in Cloud Logging.

**7. How can you test a Cloud Function locally before deployment?**

**Answer:** Use the functions-framework package:

1. Install it: pip install functions-framework
2. Run: functions-framework --target=my\_function --port=8080
3. Test using Postman or curl.

**8. How do you monitor and debug Cloud Functions?**

**Answer:** Use Google Cloud Logging and Cloud Monitoring to check logs and track function execution. You can view logs using:

sh

CopyEdit

gcloud functions logs read my-function

**9. How can you pass environment variables to a Cloud Function?**

**Answer:** Use the --set-env-vars flag during deployment:

sh

CopyEdit

gcloud functions deploy my-function --set-env-vars MY\_VAR=value

Inside the function, access it using os.environ.get("MY\_VAR") in Python.

**10. What are some common use cases of Cloud Functions with Cloud Storage?**

**Answer:**

* Image processing (resizing, compression)
* Data transformation (CSV to JSON conversion)
* Backup automation
* Content moderation (detecting inappropriate files)

EXPERIMENT NUMBER 4

**1. What is Google App Engine?**

**Answer:** Google App Engine (GAE) is a fully managed serverless platform that allows developers to build and deploy web applications without managing infrastructure. It supports automatic scaling based on traffic demand.

**2. What are the key steps to deploying a web application on App Engine?**

**Answer:**

1. Create a project in Google Cloud.
2. Write application code (e.g., using Flask, Node.js, or Java).
3. Define an app.yaml configuration file.
4. Deploy using gcloud app deploy.
5. Access the deployed app via the provided URL.

**3. What is the role of the app.yaml file in App Engine?**

**Answer:** The app.yaml file is the configuration file that specifies runtime, instance class, scaling type, environment variables, and other deployment settings. Example:

yaml

CopyEdit

runtime: python310

instance\_class: F2

automatic\_scaling:

min\_instances: 1

max\_instances: 10

target\_cpu\_utilization: 0.65

**4. How do you enable automatic scaling in App Engine?**

**Answer:** Automatic scaling is enabled by default in App Engine **Standard** environment. It can be configured in app.yaml:

yaml

CopyEdit

automatic\_scaling:

min\_instances: 1

max\_instances: 20

target\_cpu\_utilization: 0.75

This allows instances to scale dynamically based on demand.

**5. What are the different scaling options available in App Engine?**

**Answer:**

1. **Automatic Scaling** – Scales based on request load (default).
2. **Manual Scaling** – Fixed number of instances running continuously.
3. **Basic Scaling** – Instances start on demand and shut down when idle.

**6. How do you deploy a web application to App Engine using the gcloud CLI?**

**Answer:**

1. Navigate to the project folder.
2. Run the deployment command:

sh

CopyEdit

gcloud app deploy

1. Confirm the deployment and wait for completion.
2. Access the app using:

sh

CopyEdit

gcloud app browse

**7. How do you monitor application performance in App Engine?**

**Answer:** Use **Google Cloud Monitoring and Logging** to track metrics such as CPU usage, request latency, and errors. View logs using:

sh

CopyEdit

gcloud app logs read

or check logs in **Google Cloud Console** under Operations > Logging.

**8. What is the difference between App Engine Standard and Flexible environments?**

**Answer:**

| **Feature** | **Standard Environment** | **Flexible Environment** |
| --- | --- | --- |
| **Scaling** | Automatic | Automatic & Manual |
| **Runtime** | Predefined (Python, Java, Node.js, etc.) | Custom (any runtime in Docker) |
| **Instance Start Time** | Fast (milliseconds) | Slower (minutes) |
| **Customization** | Limited | More control over VMs |

**9. How do you handle environment variables in App Engine?**

**Answer:** Define environment variables in app.yaml:

yaml

CopyEdit

env\_variables:

DATABASE\_URL: "mysql://user:password@host/db"

Access in Python using:

python

CopyEdit

import os

db\_url = os.getenv("DATABASE\_URL")

**10. What are the benefits of using App Engine for web applications?**

**Answer:**

* **Serverless** – No infrastructure management.
* **Automatic scaling** – Adjusts based on traffic.
* **Built-in security** – Google-managed security features.
* **Integrated monitoring** – Logs and performance tracking.
* **Supports multiple languages** – Python, Node.js, Java, Go, etc.

EXPERIMENT NUMBER 5

**1. What is Google Cloud Storage?**

**Answer:** Google Cloud Storage (GCS) is a scalable and secure object storage service that allows users to store and retrieve data such as images, videos, backups, and logs. It supports different storage classes based on access frequency and cost efficiency.

**2. What are the different storage classes in Google Cloud Storage?**

**Answer:**

1. **Standard** – Frequently accessed data (low latency, high availability).
2. **Nearline** – Data accessed once a month (lower cost).
3. **Coldline** – Data accessed once a year (even lower cost).
4. **Archive** – Long-term storage (cheapest, but high retrieval latency).

**3. How do you create a Cloud Storage bucket using the Google Cloud Console?**

**Answer:**

1. Open **Google Cloud Console**.
2. Navigate to **Cloud Storage** > **Create Bucket**.
3. Enter a **unique bucket name**.
4. Choose a **storage class** and **location**.
5. Set access controls (public or private).
6. Click **Create** to finalize the bucket creation.

**4. How do you create a bucket using the gsutil command-line tool?**

**Answer:** Use the following command:

sh

CopyEdit

gsutil mb gs://my-bucket-name

This creates a new storage bucket named my-bucket-name.

**5. How do you upload a file to Cloud Storage using gsutil?**

**Answer:** Use the gsutil cp command:

sh

CopyEdit

gsutil cp local-file.txt gs://my-bucket-name/

This uploads local-file.txt to the specified bucket.

**6. How do you download a file from a Cloud Storage bucket?**

**Answer:** Use the following command:

sh

CopyEdit

gsutil cp gs://my-bucket-name/my-file.txt .

This downloads my-file.txt from Cloud Storage to the local directory.

**7. How do you make a file publicly accessible in Cloud Storage?**

**Answer:** Use the following command to grant public read access:

sh

CopyEdit

gsutil acl ch -u AllUsers:R gs://my-bucket-name/my-file.txt

This allows anyone to access the file via a public URL.

**8. How do you list all the objects in a Cloud Storage bucket?**

**Answer:** Use the gsutil ls command:

sh

CopyEdit

gsutil ls gs://my-bucket-name/

This lists all files and folders in the specified bucket.

**9. How do you delete a file and a bucket in Cloud Storage?**

**Answer:**

* To delete a file:

sh

CopyEdit

gsutil rm gs://my-bucket-name/my-file.txt

* To delete an empty bucket:

sh

CopyEdit

gsutil rb gs://my-bucket-name

**10. What are the security features available in Cloud Storage?**

**Answer:**

* **IAM Roles & Policies** – Controls who can access buckets and objects.
* **Bucket Lock** – Prevents modifications for compliance.
* **Encryption** – Data is encrypted at rest and in transit.
* **VPC Service Controls** – Restricts access to prevent data exfiltration.

EXPERIMENT NUMBER 6

**1. What is Cloud SQL for MySQL?**

**Answer:** Cloud SQL for MySQL is a fully managed relational database service that allows users to run MySQL databases on **Google Cloud** with automatic **backups, scaling, security, and high availability** without managing infrastructure.

**2. What are the benefits of using Cloud SQL for MySQL over a self-managed MySQL database?**

**Answer:**

* **Automated backups and updates**
* **High availability (HA) with failover support**
* **Automatic scaling of storage and performance tuning**
* **Built-in security with IAM-based access control**
* **Integration with Google Cloud services like BigQuery and Compute Engine**

**3. How do you create a Cloud SQL for MySQL instance using Google Cloud Console?**

**Answer:**

1. Open **Google Cloud Console**.
2. Go to **SQL** > Click **Create Instance**.
3. Choose **MySQL** as the database engine.
4. Configure **machine type, storage, and HA settings**.
5. Set root password and network access.
6. Click **Create** to launch the instance.

**4. How do you connect to a Cloud SQL for MySQL instance?**

**Answer:**

* **Using gcloud CLI:**

sh

CopyEdit

gcloud sql connect my-instance --user=root

* **Using MySQL Client:**

sh

CopyEdit

mysql -h <INSTANCE\_IP> -u root -p

* **Using Cloud SQL Proxy:** Recommended for secure connections.

**5. What are the high availability (HA) features in Cloud SQL?**

**Answer:**

* **Automatic failover**: If the primary instance fails, a standby replica takes over.
* **Data replication**: Uses synchronous replication between primary and standby instances.
* **Regional HA deployment**: Ensures databases are available even during zonal failures.

**6. How does Cloud SQL handle automated backups?**

**Answer:**

* Cloud SQL **automatically takes daily backups** and allows manual backups.
* You can **configure retention policies** for backup storage.
* **Point-in-time recovery (PITR)** helps restore data to a specific timestamp using binary logs.

**7. How can you scale a Cloud SQL instance?**

**Answer:**

* **Vertical scaling**: Increase CPU, RAM, or storage using the Google Cloud Console or CLI.
* **Horizontal scaling**: Use **read replicas** to distribute read queries across multiple instances.

**8. How do you secure a Cloud SQL for MySQL database?**

**Answer:**

* Use **IAM roles** for access control.
* Enable **SSL/TLS encryption** for secure connections.
* Restrict access with **authorized networks and private IP**.
* Enable **database auditing** and **Cloud SQL Insights** for monitoring.

**9. What is the difference between Cloud SQL read replicas and failover replicas?**

**Answer:**

| **Feature** | **Read Replica** | **Failover Replica** |
| --- | --- | --- |
| **Purpose** | Distributes read queries | Ensures high availability |
| **Data Sync** | Asynchronous | Synchronous |
| **Failover** | No automatic failover | Automatic failover in case of failure |

**10. How do you migrate an existing MySQL database to Cloud SQL?**

**Answer:**

* **Using Database Migration Service (DMS)** for minimal downtime.
* **Using mysqldump** for exporting and importing data:

sh

CopyEdit

mysqldump -u root -p --databases mydb > backup.sql

gcloud sql import sql my-instance gs://my-bucket/backup.sql

* **Using Cloud SQL Auth Proxy** for secure migration.

EXPERIMENT NUMBER 7

**1. What is Google Cloud Pub/Sub?**

**Answer:** Cloud Pub/Sub is a **fully managed messaging service** that allows applications to send and receive messages asynchronously, enabling **real-time communication** between distributed systems. It follows the **publish-subscribe** pattern.

**2. What are the key components of Cloud Pub/Sub?**

**Answer:**

1. **Publisher** – Sends messages to a Pub/Sub topic.
2. **Topic** – A named resource to which messages are sent.
3. **Subscriber** – A system that receives messages from a topic via a **subscription**.
4. **Subscription** – A connection between a topic and a subscriber.
5. **Message** – Data sent from publisher to subscriber.

**3. How does Cloud Pub/Sub ensure message delivery?**

**Answer:** Cloud Pub/Sub follows an **at-least-once delivery** model, meaning messages are **persisted** until they are **successfully acknowledged** by the subscriber. If a subscriber fails to process a message, it will be **resent** until acknowledged.

**4. What are the different types of Pub/Sub message delivery methods?**

**Answer:**

1. **Push Subscription** – Messages are automatically sent to an HTTP/HTTPS endpoint.
2. **Pull Subscription** – Subscribers manually request messages from the Pub/Sub service.

**5. How do you create a topic and subscription using the gcloud CLI?**

**Answer:**

* Create a **topic**:

sh

CopyEdit

gcloud pubsub topics create my-topic

* Create a **subscription**:

sh

CopyEdit

gcloud pubsub subscriptions create my-sub --topic=my-topic

**6. How do you publish and receive messages in Cloud Pub/Sub?**

**Answer:**

* **Publish a message:**

sh

CopyEdit

gcloud pubsub topics publish my-topic --message="Hello, Pub/Sub!"

* **Pull a message (Pull subscription):**

sh

CopyEdit

gcloud pubsub subscriptions pull my-sub --auto-ack

**7. What are the benefits of using Cloud Pub/Sub?**

**Answer:**

* **Scalability** – Handles millions of messages per second.
* **Asynchronous processing** – Decouples services for better performance.
* **Global availability** – Works across multiple regions.
* **Guaranteed message delivery** – Stores messages until acknowledged.

**8. How can Cloud Pub/Sub be integrated with other Google Cloud services?**

**Answer:**

* **Cloud Functions** – Trigger serverless functions when messages arrive.
* **Cloud Dataflow** – Process and analyze streaming data.
* **Cloud Run** – Trigger containerized applications.
* **BigQuery** – Ingest streaming data for analytics.

**9. What is a Dead Letter Topic (DLT) in Cloud Pub/Sub?**

**Answer:** A **Dead Letter Topic (DLT)** is used to handle messages that **fail processing multiple times**. Unprocessed messages are moved to the **DLT for later analysis** and debugging.

**10. What are some real-world use cases of Cloud Pub/Sub?**

**Answer:**

* **Real-time event processing** (e.g., IoT sensor data streaming).
* **Log aggregation** across distributed applications.
* **Fraud detection** in financial transactions.
* **Messaging between microservices** in a cloud-native application.

EXPERIMENT NUMBER 8

**1. What is a VPC (Virtual Private Cloud) in Google Cloud?**

**Answer:** A **VPC (Virtual Private Cloud)** is a **software-defined network** in Google Cloud that allows users to define **subnets, firewall rules, and routes** to control network traffic and securely connect cloud resources.

**2. What are the benefits of using multiple VPC networks in Google Cloud?**

**Answer:**

* **Resource isolation**: Separate environments for production, testing, and development.
* **Improved security**: Different VPCs with separate firewall rules.
* **Compliance and governance**: Ensures regulatory and organizational requirements.
* **Custom routing policies**: Manage traffic between applications and regions efficiently.
* **Multi-tenancy support**: Enables different teams or projects to have dedicated networks.

**3. How do you create a new VPC network using the gcloud CLI?**

**Answer:**

sh

CopyEdit

gcloud compute networks create my-vpc --subnet-mode=custom

This command creates a **custom-mode** VPC network named my-vpc.

**4. What is the difference between Auto mode and Custom mode VPCs?**

**Answer:**

| **Feature** | **Auto Mode VPC** | **Custom Mode VPC** |
| --- | --- | --- |
| Subnet Creation | Automatically created in all regions | Manually created |
| Control | Less control over subnets | Full control over subnet configuration |
| Use Case | Quick setup, simple workloads | Advanced network setups, security-conscious environments |

**5. How do you connect multiple VPC networks in Google Cloud?**

**Answer:**  
You can connect multiple VPCs using:

1. **VPC Peering** – Direct private communication between VPCs.

sh

CopyEdit

gcloud compute networks peerings create my-peering --network=my-vpc --peer-network=other-vpc

1. **Cloud VPN** – Secure encrypted tunnels between networks.
2. **Cloud Interconnect** – High-bandwidth private connectivity for hybrid cloud setups.

**6. What is VPC Peering, and how does it work?**

**Answer:** VPC Peering allows **private connectivity** between two VPC networks without using public IPs. It enables low-latency, high-bandwidth connections while keeping traffic **within Google’s network infrastructure**.

**7. What are the limitations of VPC Peering?**

**Answer:**

* **No transitive peering** – Traffic cannot pass through multiple VPCs.
* **Different organizations cannot peer VPCs** (unless using Shared VPC).
* **No overlapping IP ranges** – VPCs must have unique CIDR blocks.

**8. What is a Shared VPC, and how does it differ from VPC Peering?**

**Answer:**

* **Shared VPC**: Allows multiple projects to share a single VPC while maintaining separate billing and IAM controls.
* **VPC Peering**: Connects separate VPC networks but does not share administration.

**9. How does Google Cloud handle inter-VPC communication without public IPs?**

**Answer:** By using **VPC Peering, Private Google Access, or Cloud VPN**, services in different VPCs can communicate privately without exposing resources to the public internet.

**10. What are some real-world use cases of multiple VPC networks?**

**Answer:**

* **Multi-tier applications** (e.g., separating frontend, backend, and database layers).
* **Isolated dev, test, and production environments**.
* **Hybrid cloud networking** (e.g., connecting on-premise networks with Google Cloud).
* **Multi-tenant SaaS applications** requiring separate VPCs for different customers.

EXPERIMENT NUMBER 9

**1. What is Google Cloud Monitoring?**

**Answer:** Google Cloud Monitoring is a **managed service** that helps track, visualize, and alert on the **performance, health, and availability** of cloud resources, applications, and infrastructure.

**2. What are the key features of Cloud Monitoring?**

**Answer:**

* **Metrics collection** – Gathers system and application performance data.
* **Dashboards** – Visual representation of resource health.
* **Alerting policies** – Notifies users of critical issues.
* **Uptime checks** – Monitors availability of services.
* **Integration** – Works with **Cloud Logging, Cloud Trace, and Cloud Profiler**.

**3. How do you access Cloud Monitoring in Google Cloud Console?**

**Answer:**

1. Open **Google Cloud Console**.
2. Navigate to **Operations** > **Monitoring**.
3. View **Dashboards, Metrics Explorer, Uptime Checks, and Alerts**.

**4. What types of metrics can Cloud Monitoring track?**

**Answer:**

* **System Metrics** – CPU, memory, disk usage, network traffic.
* **Application Metrics** – Custom application logs, error rates.
* **Cloud Service Metrics** – Load Balancer latency, Cloud SQL queries, VM uptime.

**5. How do you create an alerting policy in Cloud Monitoring?**

**Answer:**

1. Open **Cloud Monitoring** > **Alerting**.
2. Click **Create Policy**.
3. Define **conditions** (e.g., CPU usage > 80%).
4. Select **notification channels** (email, Slack, PagerDuty, etc.).
5. Click **Create** to activate the alert.

**6. What are uptime checks in Cloud Monitoring?**

**Answer:**  
Uptime checks **monitor the availability** of applications or endpoints by sending HTTP, TCP, or ICMP ping requests and alerting when the service becomes **unreachable**.

**7. How do you monitor a virtual machine (VM) instance using Cloud Monitoring?**

**Answer:**

* **Enable Cloud Monitoring agent** on the VM:

sh

CopyEdit

curl -sSO https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh

sudo bash add-monitoring-agent-repo.sh

sudo apt-get install stackdriver-agent

sudo service stackdriver-agent start

* View VM metrics in **Cloud Monitoring > Metrics Explorer**.

**8. How does Cloud Monitoring integrate with Cloud Logging?**

**Answer:**

* Cloud Monitoring collects **metrics** while Cloud Logging captures **logs**.
* Logs-based **metrics and alerts** can be created to detect issues from logs.
* Logs and metrics are unified in **Cloud Operations Suite**.

**9. What are some real-world use cases for Cloud Monitoring?**

**Answer:**

* **Tracking server performance** (CPU, memory, and disk usage).
* **Monitoring Kubernetes workloads** (GKE cluster health).
* **Detecting network bottlenecks** in cloud applications.
* **Generating alerts** for database failures or high latency.

**10. What are some best practices for using Cloud Monitoring effectively?**

**Answer:**

* **Use custom dashboards** to visualize key metrics.
* **Set up alerting policies** for critical failures.
* **Enable uptime checks** for APIs and web applications.
* **Integrate with third-party tools** (Slack, PagerDuty, Opsgenie).
* **Use logs-based metrics** for deeper analysis.

EXPERIMENT NUMBER 10

**1. What is Google Kubernetes Engine (GKE)?**

**Answer:** Google Kubernetes Engine (GKE) is a **managed Kubernetes service** that simplifies the deployment, management, and scaling of containerized applications on Google Cloud.

**2. What are the benefits of using GKE over self-managed Kubernetes?**

**Answer:**

* **Automatic scaling** of clusters and workloads.
* **Built-in monitoring and logging** with Cloud Operations.
* **Load balancing and networking** with built-in Ingress support.
* **Security features** like IAM, Workload Identity, and private clusters.
* **Easy cluster upgrades** and auto-repair mechanisms.

**3. How do you create a GKE cluster using the gcloud CLI?**

**Answer:**

sh

CopyEdit

gcloud container clusters create my-cluster --num-nodes=3 --region=us-central1

This creates a **3-node cluster** in the **us-central1** region.

**4. What are the key components of a Kubernetes cluster?**

**Answer:**

* **Master Node** – Manages the cluster (API Server, Controller Manager, Scheduler, etc.).
* **Worker Nodes** – Run application workloads (Pods).
* **Pods** – Smallest deployable unit, contains one or more containers.
* **Deployments** – Manages and scales Pod replicas.
* **Services** – Exposes Pods to internal or external traffic.

**5. How do you deploy a containerized application to GKE?**

**Answer:**

1. **Push the Docker image** to Google Container Registry (GCR):

sh

CopyEdit

gcloud auth configure-docker

docker tag my-app gcr.io/my-project/my-app:v1

docker push gcr.io/my-project/my-app:v1

1. **Deploy the application using Kubernetes manifest (deployment.yaml)**:

sh

CopyEdit

kubectl apply -f deployment.yaml

**6. How do you expose a Kubernetes application to the internet?**

**Answer:**

* Use a **LoadBalancer** service:

sh

CopyEdit

kubectl expose deployment my-app --type=LoadBalancer --port=80

* This assigns an **external IP** to the application.

**7. How do you scale a deployment in Kubernetes?**

**Answer:**

* Manually scale replicas:

sh

CopyEdit

kubectl scale deployment my-app --replicas=5

* Use **Horizontal Pod Autoscaler (HPA)** to scale automatically:

sh

CopyEdit

kubectl autoscale deployment my-app --cpu-percent=80 --min=2 --max=10

**8. How do you check the status of a deployed application in GKE?**

**Answer:**

* View running Pods:

sh

CopyEdit

kubectl get pods

* Check service status:

sh

CopyEdit

kubectl get services

* View logs of a specific Pod:

sh

CopyEdit

kubectl logs my-pod

**9. What is a ConfigMap and Secret in Kubernetes?**

**Answer:**

* **ConfigMap** stores **configuration data** (e.g., environment variables).
* **Secret** stores **sensitive data** (e.g., passwords, API keys).
* Example of creating a Secret:

sh

CopyEdit

kubectl create secret generic my-secret --from-literal=password=1234

**10. What are some real-world use cases for GKE?**

**Answer:**

* **Microservices deployment** for scalable applications.
* **Machine learning workloads** using TensorFlow on Kubernetes.
* **Batch processing and CI/CD pipelines** with Jenkins.
* **Hybrid and multi-cloud applications** with Anthos.