Cloud Computing Internship 2024
Research Assignment 1

Comparing Managed and Unmanaged Instance Groups in Google Cloud Platform (GCP)

1. Differences Between Managed and Unmanaged Instance Groups:

Primary Differences	
Managed Instance Groups (MIGs)	**Unmanaged Instance Groups**
• **Auto-Scaling:** MIGs support auto-scaling, automatically adding or removing instances based on load. • **Auto-Healing:** Instances are automatically replaced if they become unhealthy**.** • **Instance Templates:** Instances in MIGs are created using instance templates, ensuring uniformity. • **Rolling Updates:** MIGs support rolling updates, allowing for gradual updates to the instances without downtime. • **Load Balancing:** Integrated with load balancing to distribute traffic efficiently across instances.	• **Manual Management:** Instances are managed individually; no automated scaling or healing. • **Flexibility:** Allows for diverse configurations within the same group, as there is no need for a uniform template. • **Manual Updates:** Updates and maintenance need to be performed manually on each instance. • **Custom Load Balancing:** Requires manual setup for load balancing.
Impact on Management and Operation	
Managed Instance Groups (MIGs)	**Unmanaged Instance Groups**
• Simplifies management with automation for scaling, healing, and updates. • Reduces the need for manual intervention, enabling consistent performance and high availability. • Suitable for workloads that require high reliability and uptime.	• Offers flexibility for custom configurations but requires significant manual effort. • Suitable for heterogeneous workloads where instances need to be configured differently. • Ideal for scenarios where automated management is not a priority.

2. Use Cases and Scenarios:

When Managed Instance Groups are Beneficial:

- **High Availability Applications:** Applications requiring high uptime and reliability benefit from the auto-healing and rolling update features.
- **Scalable Services:** Web applications, APIs, and services with variable load benefit from auto-scaling.
- **Consistent Workloads:** Workloads that need uniform configurations for all instances.

Use Cases for Managed Instance Groups:

- **Web Hosting:** Auto-scaling to handle varying traffic loads.
- **Batch Processing:** Ensuring uniformity and reliability in processing jobs.
- **API Services:** Maintaining consistent performance and availability.

When Unmanaged Instance Groups are Preferred:

- **Custom Configuration Needs:** Applications requiring different configurations for each instance.
- **Legacy Systems:** Older applications that don't support modern management features.
- **Development and Testing:** Environments where diverse setups and manual control are necessary.

Use Cases for Unmanaged Instance Groups:

- **Complex Deployments:** Systems needing varied configurations.
- **Legacy Applications:** Applications that cannot be managed by automated tools.
- **Development Environments:** Where different setups are required for testing.

3. Cost Implications:

Managed Instance Groups:	Unmanaged Instance Groups:
• **Cost Efficiency through Automation:** Reduced operational costs due to automation in scaling, healing, and updates. • **Potential Higher Resource Usage:** Auto-scaling may lead to increased resource usage, impacting costs. • **Load Balancer Costs:** Integrated load balancing may add to the overall costs.	• **Manual Management Costs:** Higher operational costs due to manual management efforts. • **Potential Savings on Resource Usage:** More control over resource allocation, potentially reducing unnecessary usage. • **Custom Load Balancer Costs:** Additional costs for setting up and managing load balancers.

Overall Cost Efficiency:

- MIGs may offer better cost efficiency for large-scale, high-availability applications due to reduced manual efforts and improved performance.
- Unmanaged groups can be more cost-effective for smaller, custom-configured environments with lower resource needs.

4. Scalability:

Managed Instance Groups:	Unmanaged Instance Groups:
Auto-Scaling: Automatically adjusts the number of instances based on demand.	**Manual Scaling:** Requires manual intervention to add or remove instances.
Elasticity: Better suited for applications with fluctuating loads.	**Fixed Capacity:** Better for applications with predictable, steady workloads.
Disadvantages: Potential for over-scaling if not properly configured, leading to higher costs.	**Disadvantages:** Less responsive to sudden changes in demand, requiring constant monitoring.
provide seamless scalability and are ideal for dynamic workloads but may incur higher costs if not optimized	offer control and predictability but lack the flexibility to handle sudden demand spikes.

5. Maintenance Requirements:

-Managed Instance Groups:

- **Low Maintenance:** Automation reduces the need for manual interventions.
- **Consistent Updates:** Rolling updates ensure minimal downtime.
- **Impact:** Frees up administrative resources for other tasks, improving overall efficiency.

-Unmanaged Instance Groups:

- **High Maintenance:** Requires manual updates, scaling, and health checks.
- **Custom Maintenance Plans:** Need tailored maintenance plans for each instance.
- **Impact:** Increases operational workload, potentially leading to human errors and inconsistencies.

6. Real-World Applications:

-Examples of Managed Instance Groups:

- **E-commerce Platforms:** Handling variable traffic loads and ensuring high availability.
- **Streaming Services:** Scaling resources based on viewer count.
- **Financial Services:** Maintaining reliability and performance for transaction processing systems.

-Examples of Unmanaged Instance Groups:

- **Research Projects:** Environments where different configurations are essential for varied experiments.
- **Legacy Systems:** Supporting applications that cannot be reconfigured for managed environments.
- **Development and Testing:** Diverse setups for different stages of software development and testing.
  Analyzing Various Google Cloud Platform (GCP) Services

Analyzing Various Google Cloud Platform (GCP) Services

1. Key Features:

-Compute Engine:

- **Virtual Machines:** Provision of highly customizable virtual machines (VMs) with a wide range of CPU, memory, and storage options.
- **Persistent Disks:** High-performance persistent disks that can be resized and modified without downtime.
- **Preemptible VMs:** Cost-effective, short-term instances ideal for batch jobs and fault-tolerant workloads.
- **Custom Machine Types:** Ability to create VMs with custom vCPU and memory configurations.
- **Sustained Use Discounts:** Automatic discounts for VMs running for a significant portion of the month.
- **Security:** Integrated security features including Shielded VMs and VPC Service Controls.

-Cloud Functions:

- **Serverless:** No infrastructure management; scales automatically based on workload.
- **Event-Driven:** Executes code in response to events from various sources such as HTTP requests, Cloud Pub/Sub, and Cloud Storage.
- **Granular Billing:** Pay only for the actual usage, measured in milliseconds.
- **Support for Multiple Languages:** Supports popular programming languages like Node.js, Python, Go, and Java.
- **Integration:** Easily integrates with other GCP services for streamlined workflows.

-App Engine:

- **Platform as a Service (PaaS):** Fully managed platform that handles infrastructure, scaling, and security.
- **Automatic Scaling:** Scales applications automatically based on traffic.
- **Versioning and Traffic Splitting:** Allows deployment of multiple versions and gradual traffic migration between them.
- **Support for Multiple Languages and Frameworks:** Supports Java, Python, PHP, Go, and Node.js.
- **Built-in Services:** Includes built-in services for authentication, data storage, and messaging.

Feature Differentiation:	
Compute Engine:	Offers highly customizable VMs suitable for a wide range of workloads, from simple to complex.
Cloud Functions:	Ideal for event-driven, short-lived tasks without the need to manage servers.
App Engine:	Provides a fully managed platform with automatic scaling, ideal for web applications and APIs.

2. Pros and Cons:

	Pros:	Cons:
Compute Engine:	• Suitable for a wide range of workloads. • High flexibility and control over VM configurations. • Cost-effective options like preemptible VMs and sustained use discounts.	• Requires management of infrastructure and scaling. • More complex setup compared to serverless options.
Cloud Functions:	• No infrastructure management required. • Automatic scaling and granular billing. • Quick deployment and integration with other services.	• Limited to event-driven, stateless functions. • Cold start latency can affect performance.
App Engine:	• Fully managed platform with automatic scaling and built-in services. • Simplifies deployment and management of web applications. • Supports multiple programming languages and frameworks.	• Less control over the underlying infrastructure. • Higher costs for certain configurations and usage patterns.

3. Case Studies or Examples:

- Compute Engine:

- **Example:** Snap Inc. uses Compute Engine for its scalable, high-performance infrastructure to support Snapchat's growing user base.
- **Illustration:** Demonstrates the ability to handle large-scale, performance-intensive workloads with flexibility in configuration and cost management.

- Cloud Functions:

- **Example:** The New York Times uses Cloud Functions to manage and process real-time data pipelines for news content.
- **Illustration:** Highlights the benefits of serverless architecture for event-driven tasks and seamless integration with other cloud services.

- App Engine:

- **Example:** Khan Academy uses App Engine to deliver its educational content to millions of students globally.
- **Illustration:** Showcases the advantages of automatic scaling, managed services, and ease of deployment for web applications.

4. Improvements or Alternatives:

	Improvements:	Alternatives:
Compute Engine:	• Enhance management tools for easier infrastructure orchestration. • Improve integration with serverless and container services.	• **AWS EC2:** Similar VM service with a wide range of instance types and pricing options. • **Azure Virtual Machines:** Comparable service with integration into Microsoft's ecosystem.
Cloud Functions:	• Reduce cold start latency for improved performance. • Expand support for more programming languages and runtimes.	• **AWS Lambda:** Popular serverless computing service with extensive event sources. • **Azure Functions:** Serverless solution with robust integration with Azure services.
App Engine:	• Provide more customization options for underlying infrastructure. • Enhance cost management features for better predictability.	• **AWS Elastic Beanstalk:** Managed service for deploying and scaling web applications. • **Azure App Service:** PaaS offering for building, deploying, and scaling web apps.

By: Ashrakat Helmi Farouk