Google Cloud – Compute Services

Compute Engine:

What

Compute Engine is Google Cloud's Infrastructure as a Service (IaaS) offering. It lets you create and run virtual machines (VMs) on Google's infrastructure. You have full control over the operating system, hardware configuration, networking, and software stack.

Why (Use Cases)

- Custom workloads → Apps that don’t fit into managed or serverless models.

- High-performance computing (HPC) → Scientific, engineering, and data-intensive workloads.

- Legacy applications → Lift-and-shift migrations from on-premise to cloud.

- Web serving & batch jobs → Host websites, APIs, or scheduled jobs.

Key Features

- Wide range of predefined & custom machine types (CPU, memory, GPU, TPU).

- Supports Linux, Windows, and custom OS images.

- Multiple storage options → Persistent Disks, Local SSDs, Hyperdisk.

- Live migration → VMs can be moved without downtime during maintenance.

- Autoscaling & Instance Groups → Adjust number of VMs automatically.

- Global availability → Deploy in multiple regions/zones worldwide.

- Spot / Preemptible VMs → Very low-cost, short-lived instances.

Pricing Model

- Per-second billing (1-minute minimum).

- Costs = VM runtime + storage + networking.

- Sustained Use Discounts (SUDs) → automatic discount for long-running VMs.

- Committed Use Discounts (CUDs) → big savings (up to 70%) if you commit for 1–3 years.

- Spot VMs → cheapest, but can be terminated anytime.

Limitations

- You are responsible for managing the OS and software (patching, updates, monitoring).
- Security/firewall configuration is your responsibility.
- Not as simple as serverless (Cloud Run) or PaaS (App Engine).

Integration

- Cloud Storage → attach buckets for storing files.
- BigQuery → analyze data produced by workloads.
- Google Kubernetes Engine (GKE) → run container workloads alongside VMs.
- Cloud Monitoring & Logging → observe and manage VM health.

Best Practices

- Right-size machine types → don't over-provision; use autoscaling.
- Use CUDs for predictable workloads to save money.
- Apply strong IAM policies and firewall rules.
- Use service accounts & OS Login instead of root passwords.
- Deploy resources in the region closest to your users for lower latency.
- Enable Cloud Monitoring & Logging for observability.

App Engine:

What

App Engine is a fully managed Platform as a Service (PaaS) for developing and hosting applications. Google manages infrastructure, scaling, load balancing, monitoring, and security.

Why (Use Cases)

- Rapidly deploy applications without managing servers.
- Automatically scale apps in response to traffic.
- Focus on code, not infrastructure.

- Build web and mobile backends with supported runtimes (Python, Node.js, Java, Go, PHP, Ruby).

Key Features

- Standard & Flexible environments.
- Automatic scaling (0 to thousands of instances).
- Built-in monitoring, logging, and security.
- Integrated traffic splitting for safe rollouts.

Pricing Model

- Pay for instance hours + network + storage.
- Free tier available in Standard environment.

Limitations

- Standard environment has restricted runtimes and no root access.
- Flexible is costlier and has slower scaling.

Integration

- Cloud Storage, Cloud SQL, Datastore/Firestore, BigQuery.

Best Practices

- Use Standard for stateless, high-traffic apps.
- Use Flexible for custom runtimes or third-party dependencies.
- Avoid using App Engine for long-running batch jobs.

Cloud Run:

What

Cloud Run is a fully managed serverless compute platform for running containerized applications. It auto-scales based on incoming requests.

Why (Use Cases)

- Deploy containerized apps without managing servers.
- Run microservices and APIs.
- Event-driven applications (integrates with Pub/Sub, Eventarc).

- Stateless workloads that need auto-scaling.

Key Features

- Supports any language/framework that runs in a container.
- Automatic scaling (including scale-to-zero).
- Pay-per-use pricing model.
- Integration with Eventarc and IAM-based security.

Pricing Model

- Pay per CPU, memory, and requests used during execution.
- Scale-to-zero → no charges when idle.

Limitations

- Stateless only → no persistent connections.
- Timeout limits for requests (15 minutes).

Integration

- Pub/Sub, Cloud SQL, Firestore, Secret Manager.
- Triggered by Eventarc events.

Best Practices

- Use for microservices, APIs, and event-driven apps.
- Keep containers small for faster cold starts.
- Secure services with IAM roles.

Google Kubernetes Engine (GKE):

What

GKE is a managed Kubernetes service for running and scaling containerized applications. It automates cluster management while you manage workloads.

Why (Use Cases)

- Run microservices at scale with Kubernetes.
- Hybrid and multi-cloud workloads.

- Apps requiring fine-grained control over orchestration.
- Migrate on-prem Kubernetes apps to the cloud.

Key Features

- Autopilot mode (Google manages nodes).
- Standard mode (you manage nodes).
- Automatic upgrades, patching, and monitoring.
- Horizontal Pod Autoscaler (HPA).
- Integration with Load Balancing, Monitoring, IAM.

Pricing Model

- Cluster management fee + worker nodes (VMs).
- Autopilot pricing = pay per Pod resources.

Limitations

- More complex than App Engine or Cloud Run.
- Requires Kubernetes knowledge.

Integration

- Cloud Build, Artifact Registry, Monitoring, Cloud Storage.
- Hybrid/multi-cloud support via Anthos.

Best Practices

- Use Autopilot for minimal operations overhead.
- Apply Pod Security Policies and IAM.
- Use autoscaling and monitoring to optimize costs.
- Keep workloads stateless; use Cloud Storage/Databases for persistence.

Cloud Functions:

What

Cloud Functions is a serverless execution environment for building and connecting cloud services. You write simple, single-purpose functions that are attached to events from your cloud infrastructure and services.

Why (Use Cases)

- Event-driven workloads (e.g., file upload to Cloud Storage, Pub/Sub message).
- Lightweight APIs and microservices.
- Automation and backend tasks.
- Real-time processing (IoT, logs, triggers).

Key Features

- Supports multiple languages (Node.js, Python, Go, Java, .NET, Ruby, PHP).
- Automatic scaling down to zero when idle.
- Pay only for actual compute time used.
- Integrates with Eventarc for 90+ event sources.

Pricing Model

- Billed based on number of invocations, compute time, and memory allocation.
- Free tier available each month.

Limitations

- Execution timeout (up to 60 minutes in 2nd Gen).
- Stateless — no persistent connections.

Integration

- Cloud Storage, Pub/Sub, Firestore, Firebase, Eventarc, Cloud Logging.

Best Practices

- Use 2nd Gen for higher concurrency and more event sources.
- Secure with IAM and least privilege service accounts.
- Minimize cold start delays (configure min instances if needed).

Bare Metal Solution:

What

Bare Metal Solution provides dedicated physical servers near Google Cloud regions, optimized for specialized workloads like Oracle Database.

Why (Use Cases)

- Running Oracle workloads in proximity to Google Cloud.
- Applications requiring low latency to on-premises systems.
- Regulatory or licensing restrictions that require bare metal.

Key Features

- Dedicated physical machines (not virtualized).
- Direct, low-latency connection to Google Cloud services.
- High-performance networking and storage.

Pricing Model

- Subscription-based pricing (monthly or yearly commitments).

Limitations

- Limited availability (only in select regions).
- Managed separately from other Compute services.

Integration

- Connects to Google Cloud via low-latency interconnects.
- Works with Cloud Storage, BigQuery, and other services for hybrid deployments.

Best Practices

- Use for Oracle and other specialized workloads only.
- Plan capacity in advance due to hardware provisioning.

VMware Engine:

What

Google Cloud VMware Engine is a fully managed VMware service that lets you run VMware workloads natively on Google Cloud infrastructure.

Why (Use Cases)

- Lift-and-shift migration of VMware workloads to Google Cloud.
- Extend on-premises VMware environments.

- Disaster recovery and hybrid cloud setups.

Key Features

- Native VMware stack (vSphere, vCenter, vSAN, NSX-T, HCX).
- Private, dedicated VMware environment on Google Cloud.
- High-speed connectivity to other Google Cloud services.

Pricing Model

- Dedicated nodes billed monthly or yearly.
- Committed use contracts offer discounts.

Limitations

- Limited flexibility compared to re-architecting apps into cloud-native models.
- Higher cost compared to serverless/VM options.

Integration

- Integrates with Compute Engine, Cloud Storage, BigQuery, and Anthos.
- Use HCX for live migration.

Best Practices

- Use for migration without refactoring.
- Combine with cloud-native services for modernization over time.

Batch:

What

Batch is a fully managed service for running batch computing workloads at scale on Google Cloud.

Why (Use Cases)

- High-performance computing (HPC) and large-scale batch jobs.
- Data processing pipelines (rendering, simulations, genomics).
- Cost-effective execution of parallel workloads.

Key Features

• Fully managed job scheduling and execution.

• Scales across Compute Engine VMs automatically.

• Support for containerized and script-based workloads.

Pricing Model

• Pay only for resources used (VMs, storage, networking).

• Works with preemptible and spot VMs for cost savings.

Limitations

• Designed for batch, not interactive workloads.

• Limited real-time processing capabilities.

Integration

• Integrates with Cloud Storage, Pub/Sub, BigQuery, and Cloud Logging.

• Can use Compute Engine and GPUs for specialized workloads.

Best Practices

• Use spot/preemptible VMs to minimize costs.

• Break down large jobs into smaller parallel tasks.
