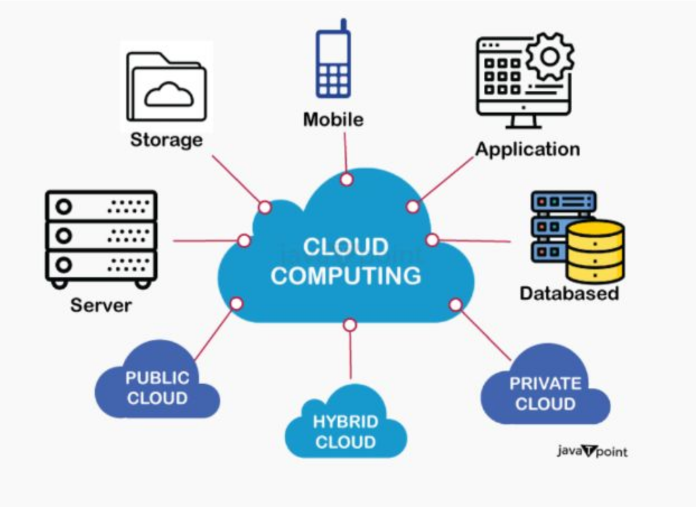
GCP and Cloud for Data Engineering

by Mateusz Wasylow

Cloud Computing: The Foundation ☁️

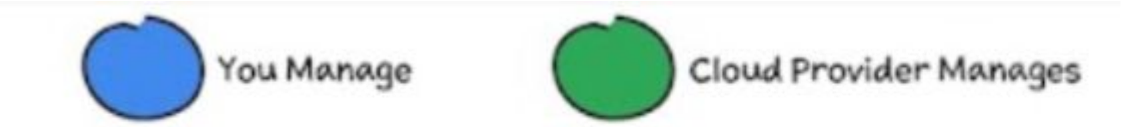
- **Definition:** Delivery of various services (storage, processing, databases, networking, software) over the internet.
- **Providers:** Google Cloud Platform (GCP), Amazon Web Services (AWS), Microsoft Azure.
- **Key Concepts:**
    - **Scalability:** Easily adjust resources based on demand.
    - **Cost-efficiency:** Pay-as-you-go model, no upfront capital investment.
    - **Reliability:** Robust disaster recovery and backup solutions.
    - **Accessibility:** Access resources anywhere with an internet connection.
    - **Performance:** High-performance hardware and global networks.

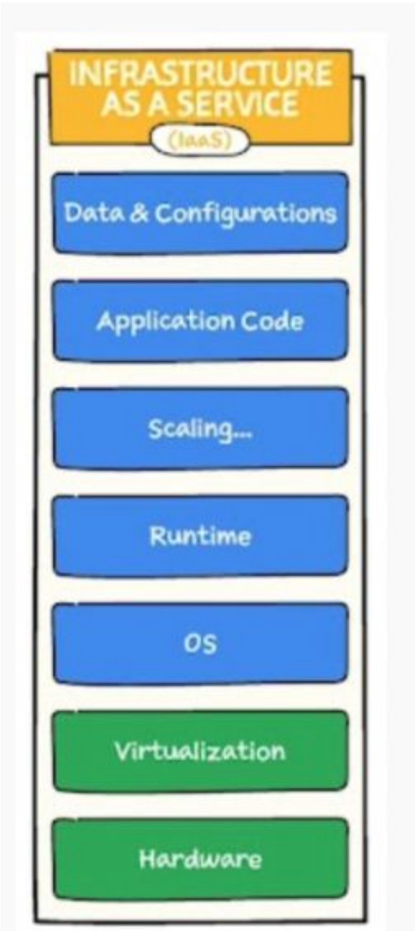


Cloud Service Models: Who Manages What? 🤔

Infrastructure as a Service (IaaS)

- **You Manage:** Data, Applications, Runtime, OS.
- **Cloud Provider Manages:** Virtualization, Hardware.
- **Description:** Virtualized computing resources (compute, storage, networking). You control the OS and above.
- **Examples:** Google Compute Engine, Amazon EC2, Microsoft Azure VMs.



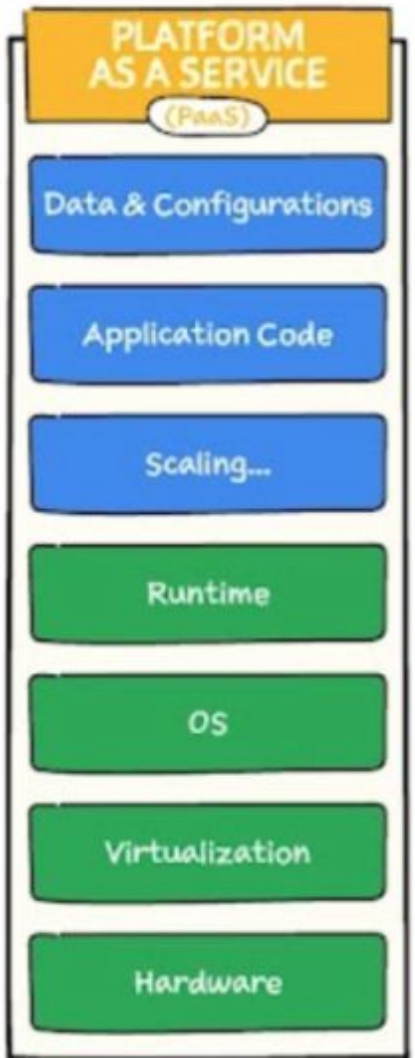


Cloud Service Models: Who Manages What? 🤔

Platform as a Service (PaaS)

- **You Manage:** Data, Applications.
- **Cloud Provider Manages:** Runtime, OS, Virtualization, Hardware.
- **Description:** A platform to develop, run, and manage applications without underlying infrastructure worries. You focus on code.
- **Examples:** Google App Engine, Microsoft Azure App Service.

You Manage
Cloud Provider Manages



Cloud Service Models: Who Manages What? 🤔

Software as a Service (SaaS)

- **You Manage:** (Nothing, it's ready to use)
- **Cloud Provider Manages:** Data, Applications, Scaling, Runtime, OS, Virtualization, Hardware.
- **Description:** Entire software application stack delivered. Provider manages everything, including security and updates.
- **Examples:** Google Workspace (Gmail, Docs), Salesforce, Microsoft 365.

Cloud Service Models: An Overview 📊

- **Traditional On-Premises:** You manage everything.
- **IaaS:** You manage up to the OS.
- **Containers as a Service (CaaS):** (Implicitly fits between IaaS/PaaS) You manage application code and data, provider handles runtime, OS, etc., with containers as the unit of deployment.
- **PaaS:** You manage application code and data.
- **Function as a Service (FaaS):** You manage *just* the function code and data.
- **SaaS:** You manage nothing.

Benefits of Cloud Computing ✅

- **Cost Savings:** Reduce capital expenditure on hardware.
- **Flexibility:** On-demand resources for varying workloads.
- **Disaster Recovery:** Robust, often built-in, backup and recovery.
- **Automatic Updates:** Systems and applications always up-to-date.
- **Collaboration:** Enhance teamwork with cloud-based tools.

Challenges of Cloud Computing ⚠️

- **Security and Privacy:** Data breaches, control over sensitive information.
- **Downtime:** Potential service outages (though high uptime guarantees exist).
- **Compliance:** Complexities with regulations (GDPR, HIPAA).
- **Vendor Lock-in:** Risk of dependency on a single provider.
- **Cost Management:** Costs can escalate without proper monitoring.

Why Cloud Computing? Key Use Cases 🚀

- **Serverless Computing:** Build apps without managing servers, pay only for compute time. (e.g., Cloud Functions, Cloud Run)
- **Edge Computing:** Computation and storage closer to data source for reduced latency.
- **AI and ML Integration:** Cloud providers offer ML/AI services, avoiding infrastructure investment.
- **Hybrid and Multi-Cloud Strategies:** Leverage multiple environments for flexibility, risk management, and optimization.

Google Cloud Platform (GCP) 🚀

- A comprehensive suite of cloud computing services from Google.
- Offers tools for computing, storage, data analytics, machine learning, and application development.
- **Key Services include:**
    - **Compute:** Compute Engine, App Engine, Kubernetes Engine, **Cloud Run**, Cloud Functions.
    - **Storage:** Cloud Storage, Cloud Bigtable, Cloud SQL.
    - **Databases:** Cloud Spanner, Firestore.
    - **Big Data:** BigQuery, Dataflow, Dataproc.
    - **Machine Learning:** Vertex AI (AI Platform, AutoML).
    - **Networking:** Virtual Private Cloud (VPC), Cloud Load Balancing.
    - **Management Tools:** Cloud Logging, Cloud Monitoring.
    - **CI/CD & DevOps: Cloud Build**, **Artifact Registry**.

Our Data Pipeline Stack 📦

- **Data Sources:** Where our raw data originates.
- **Data Ingestion:** Batch or Stream processing into the cloud.
- **Storage & Processing:** Storing raw and intermediate data.
- **Transformation & Modeling:** Cleaning, enriching, and structuring data.
- **Consumption:** Analytics, Machine Learning, Data Products.

Compute for Microservices: Google Cloud Run 🏃‍♀️

- **Definition:** Fully managed serverless platform for deploying **containerized applications**.
- **Why for Microservices?** Perfect for your FastAPI apps and Python scripts!
- **Key Features:**
    - **Serverless:** No infrastructure to manage.
    - **Container-based:** Deploy any language/library packaged in Docker.
    - **Automatic Scaling:** Scales from zero to thousands based on traffic.
    - **Pay-per-use:** Only pay when your code is running.
    - **Event-driven:** Triggered by HTTP, Pub/Sub, Cloud Storage, etc.
- **Use Cases in Data Pipelines:**
    - Deploying individual ingestion, transformation, or ML prediction services.
    - Creating scalable APIs for data products.

Serverless Functions: Google Cloud Functions 🛠️

- **Definition:** Serverless execution environment for building and connecting cloud services.
- **Key Features:**
  - **Serverless Architecture:** Scales automatically, no infrastructure management.
  - **Event-Driven:** Triggered by HTTP requests, Pub/Sub messages, Cloud Storage changes.
  - **Multi-language Support:** Node.js, Python, Go, Java, etc.
- **Use Cases:**
  - Real-time data processing (e.g., file arrival in storage).
  - Scheduled tasks (e.g., daily reports).
  - Lightweight APIs and webhooks.
- **Functions vs. Run:** Use Functions for simpler, single-purpose event handlers; Cloud Run for more complex, full-fledged microservices.

Data Storage & Processing: Cloud Storage ☁️

- **Definition:** Object storage service offering high availability, scalability, and durability.
- **Key Features:**
  - **Storage Classes:** Standard, Nearline, Coldline, Archive for varying access needs.
  - **Global Accessibility:** Data accessible globally with strong consistency.
  - **Security:** Encryption at rest and in transit, IAM for access control.
  - **Lifecycle Management:** Automated rules for data retention and deletion.
- **Use Cases:**
  - **Data Lakes:** Centralized storage for raw and processed data.
  - **Backup and Recovery:** Reliable storage for backups and archival.
  - Content delivery for web and mobile applications.

Data Lake & Data Warehouse: Google BigQuery 🔍

- **Definition:** Fully managed, serverless data warehouse that enables fast SQL queries on massive datasets.
- **BigQuery as a Data Lakehouse:**
  - Can directly store **raw, semi-structured, and structured data**.
  - Supports querying data directly from Cloud Storage (federated queries) or ingesting it into native BigQuery tables.
  - Eliminates the need for a separate data lake for many use cases, allowing you to query raw and transformed data in one place.
- **Key Features:**
  - **Serverless:** Scales automatically, no infrastructure management.
  - **SQL Interface:** Familiar SQL syntax for querying.
  - **Performance:** Optimized for large-scale data analysis (columnar storage, parallel execution).
  - **BigQuery ML:** Built-in machine learning capabilities.
  - **Data Transfer Service:** Automates data transfer from various sources.
- **Use Cases:**
  - Storing **raw data** for historical analysis and audit.
  - Real-time and batch analytics on transformed data.
  - Business Intelligence (integration with Looker, Tableau).
  - Training and deploying ML models directly.

Google Cloud Build: Your CI/CD Engine 🛠️

- **Definition:** Serverless CI/CD platform that executes your builds on GCP.
- **Key Features:**
  - **Serverless:** No build servers to provision or manage.
  - **Integrated:** Connects to Cloud Source Repositories, GitHub, Bitbucket, etc.
  - **Customizable Build Steps:** Define your process using `cloudbuild.yaml` files.
  - **Docker Image Building:** Ideal for building container images for Cloud Run.
- **Use Cases:**
  - Automatically build Docker images for microservices on code push.
  - Run automated unit and integration tests.
  - Prepare and push artifacts to Artifact Registry.

GitHub Actions for CI/CD with GCP 🐙

- **Definition:** Automate, customize, and execute your software development workflows directly in your repository.
- **Integration with GCP:** GitHub Actions can directly interact with GCP services using official actions or the `gcloud CLI`.
- **Key Features:**
  - **Event-driven:** Triggered by commits, pull requests, scheduled events, etc.
  - **YAML Workflows:** Define pipelines directly in your GitHub repository.
  - **Secrets Management:** Securely store GCP credentials (e.g., a service account key as a GitHub Secret).
  - **Community & Official Actions:** Leverage pre-built actions for common tasks.
- **Use Cases for Data Pipelines:**
  - Triggering **Cloud Build** to execute your GCP-specific build steps.
  - Directly deploying Docker images to **Artifact Registry**.
  - Deploying new revisions to **Cloud Run** after successful tests.
  - Running `gcloud` commands for infrastructure as code or data operations.

Google Artifact Registry: Managing Your Artifacts 📦

- **Definition:** Universal package manager for storing, managing, and securing your build artifacts.
- **Key Features:**
  - **Universal:** Supports Docker images, npm, Maven, Python packages, etc.
  - **Integrated:** Seamlessly with Cloud Build, Cloud Run, GKE.
  - **Security:** Vulnerability scanning, fine-grained access control (IAM).
  - **Regional & Global:** Choose location for performance and compliance.
- **Use Cases:**
  - Storing Docker images of microservices (built by Cloud Build).
  - Central repository for all pipeline deployable components.
  - Ensuring version control and traceability of deployed applications.

CI/CD Workflow for Data Microservices 🔄

1. **Code Commit:** Developer pushes code to source repository (e.g., GitHub).
2. **Trigger Cloud Build:** A `cloudbuild.yaml` file defines steps:
   - Build Docker image for the microservice.
   - Run unit/integration tests.
3. **Push to Artifact Registry:** Cloud Build pushes the built Docker image to Artifact Registry.
4. **Deploy to Cloud Run:** A successful build (or manual trigger) deploys the new image revision to Cloud Run.
5. **Run Pipeline:** Orchestrator (e.g., Cloud Workflows, Cloud Scheduler + Pub/Sub) triggers the deployed microservices.
6. **Monitor:** Cloud Logging and Cloud Monitoring provide observability.

Deploying Cloud Functions to GCP ⚡

- **1. Write Your Function Code:**
  - Create a function in a supported language (e.g., Python, Node.js).
  - Define an **entry point** and specify dependencies (`requirements.txt`).
  - *Example (Python):* `def hello_http(request): return 'Hello!'`
- **2. Define the Trigger:**
  - Cloud Functions are **event-driven**. Specify what invokes them.
  - **Common Triggers:** HTTP, Cloud Pub/Sub message, Cloud Storage event, Firestore change, Cloud Scheduler.
- **3. Deploy the Function:**
  - Deploy your code, specifying runtime, entry point, and trigger type.
  - The platform automatically packages and containerizes your function.

Monitoring & Logging: Observability for Your Pipeline 👀

- **Definition:** Essential for understanding the health, performance, and behavior of your data pipeline and microservices.
- **Google Cloud Logging:**
  - **Centralized Log Management:** Aggregates logs from all your GCP services (Cloud Run, Cloud Functions, BigQuery, etc.).
  - **Log Explorer:** Powerful interface to search, filter, and analyze logs.
  - **Log-based Metrics:** Create metrics from log entries to monitor specific events.
  - **Export Logs:** Export logs to BigQuery for long-term analysis or to Pub/Sub for real-time processing.
- **Google Cloud Monitoring:**
  - **Metrics Collection:** Gathers metrics (CPU usage, memory, network traffic, custom metrics) from your GCP resources.
  - **Dashboards:** Create custom dashboards to visualize key metrics.
  - **Alerting:** Set up alerts to notify you of critical issues (e.g., microservice errors, high BigQuery job failures).
  - **Uptime Checks:** Monitor the availability of your public-facing services (e.g., Cloud Run endpoints).
- **Why for Data Pipelines?**
  - **Troubleshooting:** Quickly identify errors or bottlenecks in your microservices chain.
  - **Performance Optimization:** Monitor resource usage to optimize cost and speed.
  - **Proactive Issue Detection:** Get alerted before small issues become big problems.

Other Key GCP Concepts 🔑

- **Projects:** Fundamental organizational unit in GCP, containing all resources.
  - Each project has a unique ID and number.
  - Forms the basis for enabling services, managing APIs, and billing.

Other Key GCP Concepts 🔑

- **Regions & Zones:**
  - **Regions:** Specific geographic locations (e.g., `europe-west1`).
  - **Zones:** Isolated locations within a region (e.g., `europe-west1-a`).
  - **Benefit:** Enhance availability and resilience by deploying across multiple regions/zones.
  - **Factors to Consider:** Latency, redundancy, compliance, cost.

Identity and Access Management (IAM) 🛡️

- **Definition:** Framework to ensure the right individuals/services have appropriate access to resources.
- **"Who, What, Where":** Who (identity) has what access (role) to which resource (policy).
- **Principals (Identities):**
  - **Users:** Individual human users.
  - **Groups:** Collections of users.
  - **Service Accounts: Crucial for our microservices!** Accounts for applications and VMs to authenticate and make API requests.
- **Resources:** Projects, Compute Engine instances, Cloud Storage buckets, BigQuery datasets, etc.
- **Roles:**
  - **Primitive Roles:** Owner, Editor, Viewer (broad).
  - **Predefined Roles:** Specific sets of permissions (e.g., `roles/storage.objectViewer`).
  - **Custom Roles:** User-defined roles with tailored permissions.
- **IAM Policies:** Define access at project, folder, or organization level (inherited by resources).
- **Best Practice: Principle of Least Privilege** – grant only necessary permissions.

IAM for Local Development & Microservices 🧑‍💻

- **Microservices on Cloud Run:** Automatically use the attached service account's identity. Grant this service account **specific roles** to access BigQuery, Cloud Storage, etc.
- **Local Docker Testing:**
  - **Recommended:** Use **Application Default Credentials (ADC)**. Each student authenticates their `gcloud CLI` (e.g., `gcloud auth application-default login`). Then, mount their local ADC file into the Docker container. This uses their user identity temporarily.
  - **Less Recommended:** If absolutely necessary, use a service account **key file**. This file must be treated like a password, *never* committed to Git, and securely mounted into the container. Emphasize the risks!

Billing: Understanding GCP Costs 💰

- **Definition:** Tools to track, understand, pay, and optimize your GCP spending.
- **Billing Account:** Linked to projects to manage costs.
- **Cost Management:** Monitoring tools to manage spending.
- **Budgets and Alerts:** Set up notifications when spending approaches or exceeds limits.
- **Best Practice:** Regularly review billing reports and set up alerts to prevent unexpected costs.