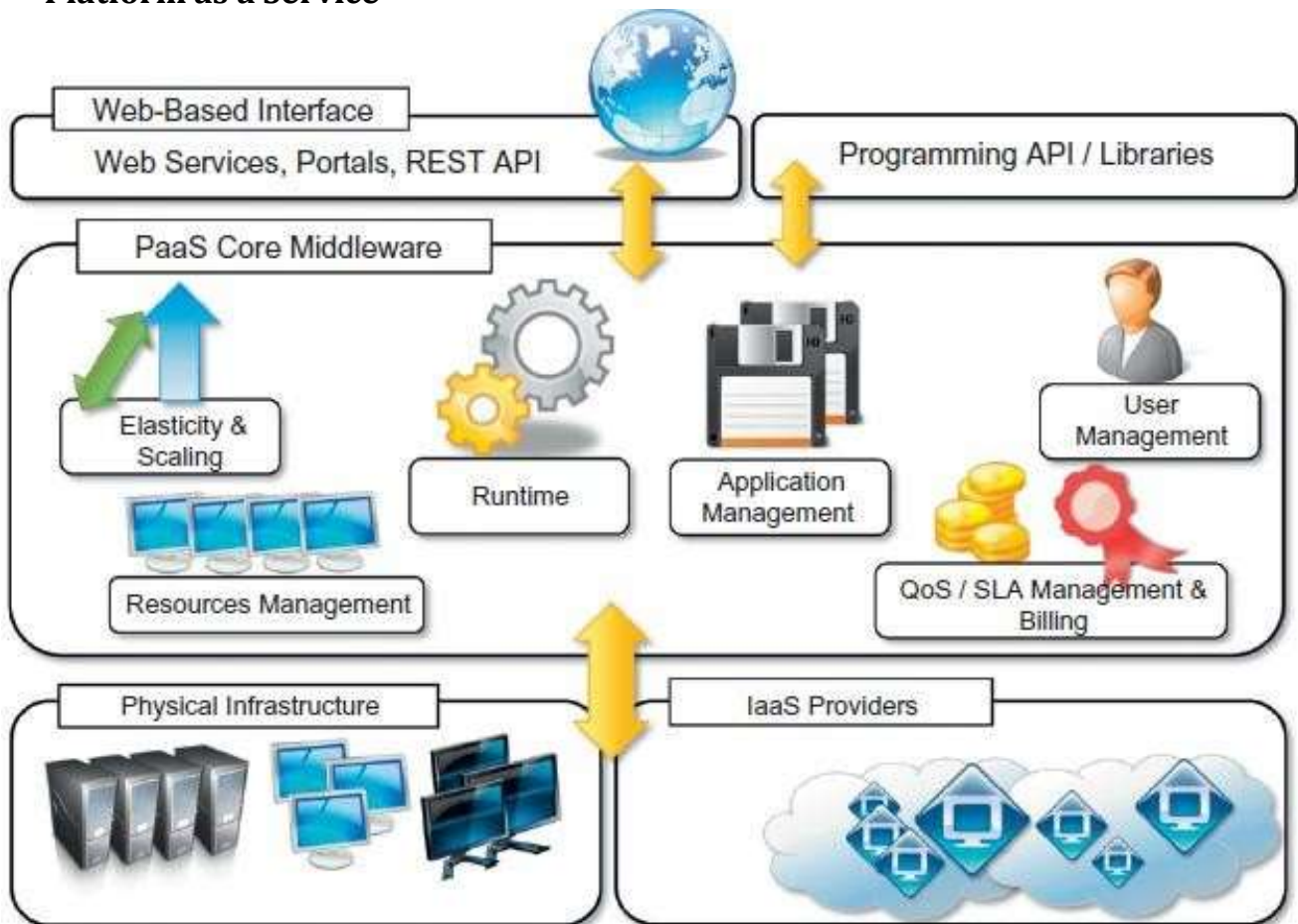


Module – 3

Introduction to Platform as a Service (PaaS), Popular PaaS Providers (e.g., Heroku, Google App Engine), Developing, Deploying, and Scaling Applications in the Cloud, Database as a Service (DBaaS), Microservices Architecture Serverless Computing

Platform as a Service



- Platform-as-a-Service (PaaS) solutions provide a development and deployment platform for running applications in the cloud.
- It constitute the middleware on top of which applications are built.
- PaaS implementations provide applications with a runtime environment and do not expose any service for managing the underlying infrastructure.
- Automate the process of deploying applications to the infrastructure, configuring application components, provisioning and configuring supporting technologies such as load balancers and databases, and managing system change based on policies set by the user.

- The core middleware is in-charge of managing the resources and scaling applications on demand or automatically, according to the commitments made with users.
- From a user point of view, the core middleware exposes interfaces that allow programming and deploying applications on the cloud. These can be in the form of a Web-based interface or in the form of programming APIs and libraries.
- It is possible to find integrated developed environments based on 4GL and visual programming concepts, or rapid prototyping environments where applications are built by assembling mash-ups and user-defined components and successively customized.
- Other implementations of the PaaS model provide a complete object model for representing an application and provide a programming language-based approach.
 - This approach generally offers more flexibility and opportunities but incurs longer development cycles.
 - PaaS solutions can offer middleware for developing applications together with the infrastructure or simply provide users with the software that is installed on the user premises.
- PaaS provider also owns large datacenters where applications are executed
- Pure PaaS, the middleware constitutes the core value of the offering.

Table provides a classification of the most popular PaaS implementations. It is possible to organize the various solutions into three wide categories: PaaS-I, PaaS-II, and PaaS-III.

Category	Description	Product Type	Vendors and Products
<i>PaaS-I</i>	Runtime environment with Web-hosted application development platform. Rapid application prototyping.	Middleware + Infrastructure Middleware + Infrastructure	Force.com Longjump
<i>PaaS-II</i>	Runtime environment for scaling Web applications. The runtime could be enhanced by additional components that provide scaling capabilities.	Middleware + Infrastructure Middleware Middleware + Infrastructure Middleware + Infrastructure Middleware + Infrastructure Middleware	Google AppEngine AppScale Heroku Engine Yard Joyent Smart Platform GigaSpaces XAP
<i>PaaS-III</i>	Middleware and programming model for developing distributed applications in the cloud.	Middleware + Infrastructure Middleware Middleware Middleware Middleware Middleware	Microsoft Azure DataSynapse Cloud IQ Manjrasof Aneka Apprenda SaaSGrid GigaSpaces DataGrid

Advantages of PaaS

- PaaS demands a lesser amount of time and average skills for management.
- The biggest benefit of PaaS over other cloud computing models is that it can catalyze the development of new applications.
- PaaS cloud model supports a number of programming languages that gives software developers a chance to execute multiple projects on a similar platform.
- PaaS offers software companies all the resources they require to develop applications and they don't need to hire any extra staff for doing this. All middleware and hardware are offered, upgraded and maintained by the vendor which means organizations don't need to hire dedicated staff to install servers and manage the operating system.
- In the case of PaaS, resourcing can be easily increased or decreased according to business needs. It is highly scalable. The database and web services are also perfectly integrated into the PaaS.
- The expenses involved in developing, testing and realizing apps are quite low when compared with other cloud-based models.
- The amount of time required for coding is also significantly low in case of PaaS.

Disadvantages of PaaS

- The client has zero control over the virtual machine that looks after the data.
- PaaS may incur you unexpected charges depend upon your requirement.
- PaaS cloud model does require basic coding skills and programming knowledge to deploy it successfully into the system.
- You may face a few data security concerns while deploying PaaS cloud model.
- In certain cases, compatibility issues for infrastructure can be observed as you can't make every single component cloud-enabled.
- You have to heavily rely on the support of the vendor.

When to choose PaaS:

- PaaS is effective for software developers who want to dedicate more time on coding, deploying and customizing applications.
- It is extremely beneficial when huge numbers of software developers are working on a single project.
- If you want to develop your own customized application then PaaS cloud model is the preferred choice over others.

- PaaS offers a perfect environment for developing, managing, testing and customizing apps that makes it an ideal choice for software companies.

Popular PaaS Providers:

- Windows Azure Cloud Services,
- Amazon AWS,
- Google App Engine
- OpenShift,
- Heroku,
- Apache Stratos,
- AWS Elastic Beanstalk,

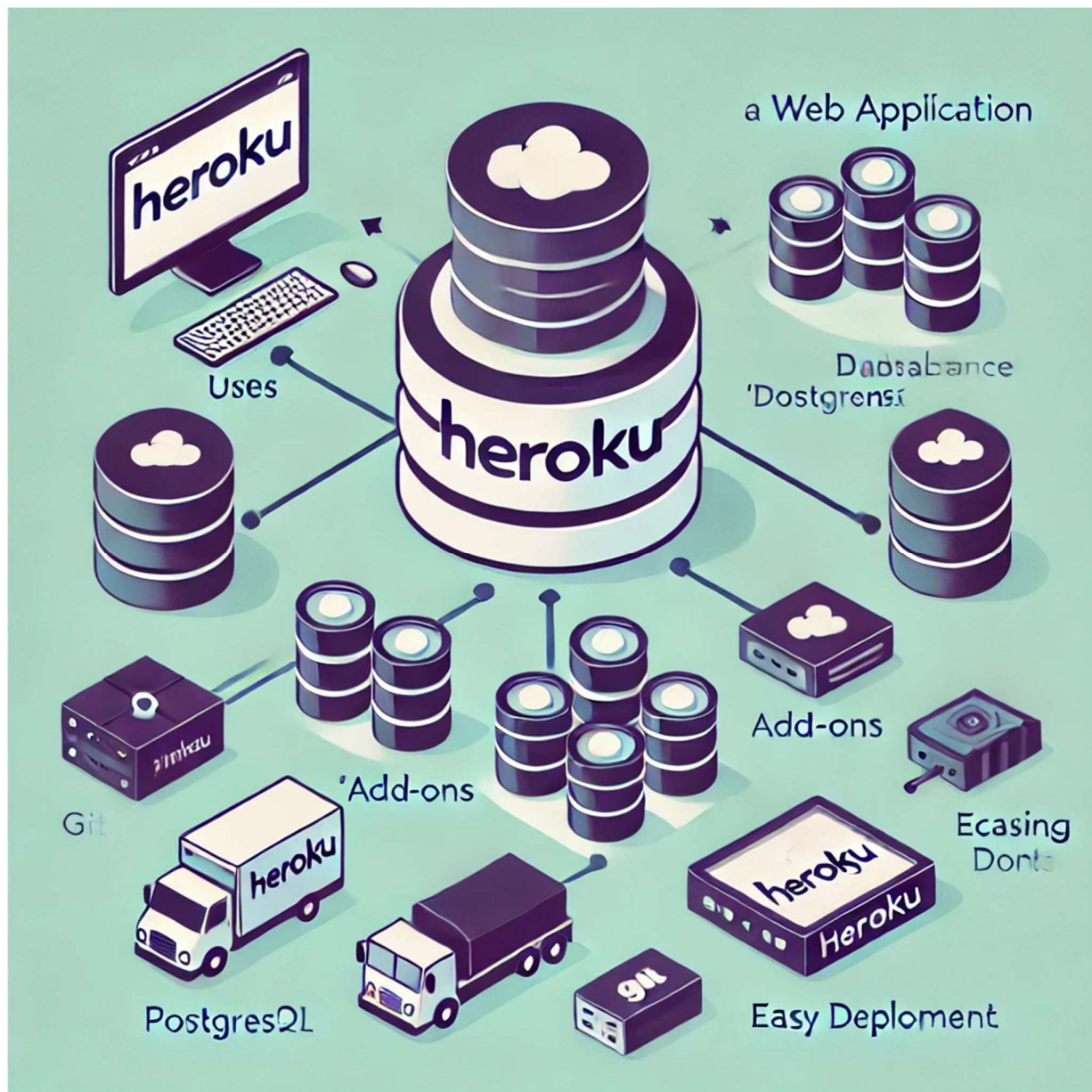
Heroku

- Heroku is a cloud platform as a service (PaaS) that allows developers to build, run, and operate applications entirely in the cloud.
- It supports several programming languages and frameworks, including Ruby, Python, Java, Node.js, PHP, and Go.
- Heroku abstracts away infrastructure management, letting developers focus on code while it handles scaling, server provisioning, and other operational aspects.

Here are some key features of Heroku:

1. **Simple Deployment:** Applications can be deployed on Heroku using Git, GitHub, or Docker.
2. **Dynos:** Heroku runs applications on virtual containers called dynos, which are isolated, scalable units of compute power.
3. **Add-ons:** Heroku offers a marketplace of add-ons for databases, caching, logging, monitoring, and more, making it easy to extend app functionality.
4. **Scaling:** Heroku allows easy scaling of applications horizontally by adding more dynos to handle increased traffic.
5. **Supports Multiple Environments:** Heroku supports different stages of development like staging, production, and testing.
6. **Integrated Tools:** It provides tools for logging, performance monitoring, and managing security features.

Developers love Heroku for its simplicity, especially when starting new projects or managing smaller apps, although for larger applications, it may become more costly compared to other cloud platforms.



The diagram provides a visual representation of how Heroku operates. Here's a breakdown of its components:

1. **Users:** On the left side, users access the web application through their browsers or mobile devices. They interact with the application through HTTP requests.
2. **Web Application:** The core of the architecture is the application itself, which is hosted on Heroku's cloud platform. It can be written in various programming languages supported by Heroku (e.g., Ruby, Python, Java, Node.js, PHP).
3. **Dynos:** The application runs on Heroku's "dynos." Dynos are lightweight, virtualized containers where the application's code executes. These are the building blocks of Heroku,

and they can be scaled up or down to handle more or less traffic, as represented by the multiple dynos.

4. **Database (e.g., PostgreSQL):** The application often requires a database to store and retrieve data. Heroku provides easy integration with databases like PostgreSQL, which is commonly used. This connection is depicted by a direct link between the web application and the database.
5. **Add-ons:** Heroku offers a marketplace of add-ons, which are third-party services that extend the app's capabilities. These add-ons can include logging (e.g., Papertrail), caching (e.g., Redis), performance monitoring, or email services. The diagram shows these as connected services that enhance the functionality of the app.
6. **Scaling:** To manage increased traffic or heavier workloads, Heroku allows horizontal scaling by adding more dynos. This enables the app to distribute the load across multiple containers, as indicated by the multiple dynos running in parallel.
7. **Deployment:** On the right side, the diagram shows the deployment process, typically done via Git or Docker. Developers push their code to Heroku using Git commands or by deploying Docker images, making it easy to update and manage the app.

This structure simplifies the development and management of web applications by offloading operational tasks to Heroku's cloud environment.

Google App Engine

- Google App Engine (GAE) is a fully managed platform-as-a-service (PaaS) offering by Google Cloud, allowing developers to build, deploy, and scale applications without managing the underlying infrastructure.
- It is highly scalable and supports multiple programming languages, making it suitable for web applications, APIs, and microservices.

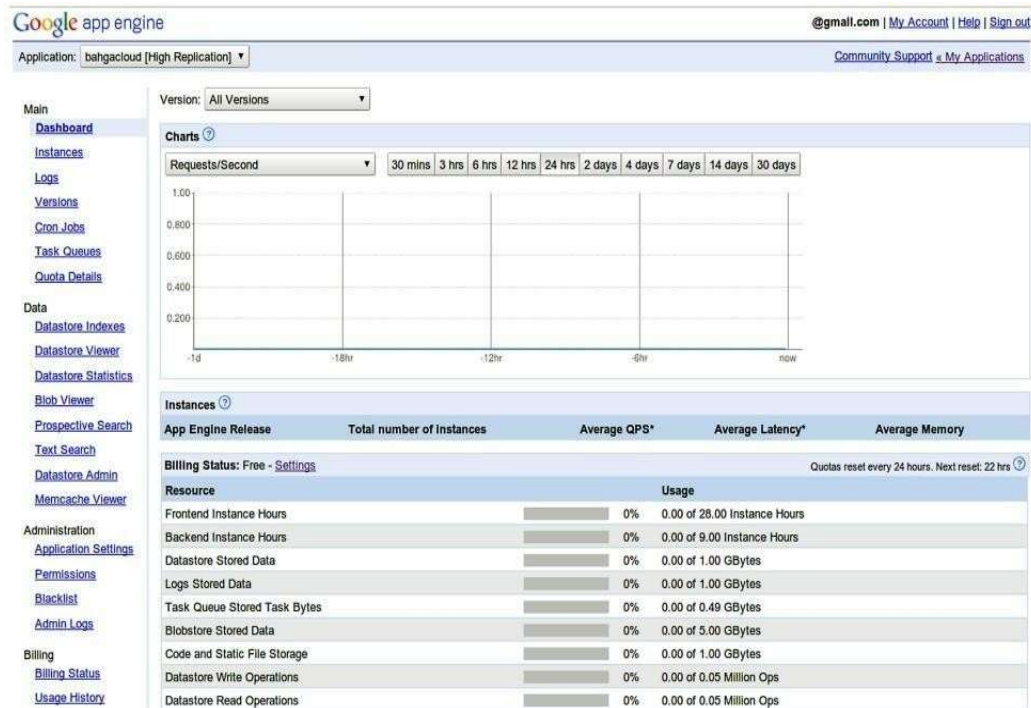
Key Features of Google App Engine:

1. **Automatic Scaling:** GAE automatically adjusts the number of instances running based on the application's traffic. This means the platform can handle sudden spikes in traffic without manual intervention.
2. **Supports Multiple Languages:** It natively supports languages like Python, Java, Node.js, Go, Ruby, and PHP. It also allows users to bring their own custom runtime.
3. **Managed Infrastructure:** App Engine abstracts infrastructure management, including load balancing, networking, and server management, so developers can focus solely on writing code.

4. Standard and Flexible Environments:
 - Standard Environment: Runs applications within specific language runtimes, with fast scaling and predictable environments.
 - Flexible Environment: Allows more flexibility, supporting Docker containers and custom runtime environments, giving you more control over infrastructure.
5. Integrated Google Cloud Services: App Engine integrates with other Google Cloud services like Cloud Datastore (for databases), Cloud Storage, Pub/Sub, and Stackdriver (for monitoring and logging).
6. Version Control and Traffic Splitting: GAE allows developers to manage different versions of their applications and route traffic between them, enabling safe rollouts and A/B testing.
7. Security: App Engine comes with built-in security features such as SSL, user authentication, and firewall rules.

Benefits:

- Ease of Deployment: Developers can deploy apps directly from command-line tools, IDEs, or CI/CD pipelines.
- Cost-Effective: You pay only for the resources you use, and Google offers a free tier for low-traffic apps.
- Focus on Development: Google handles the backend infrastructure, allowing developers to focus on building features rather than managing servers.
- Google App Engine is the platform-as-a-service (PaaS) from Google, which includes both an application runtime and web frameworks.
- Runtimes
 - App Engine provides runtime environments for Java, Python, PHP and Go programming language.
- Sandbox
 - Applications run in a secure sandbox environment isolated from other applications.
 - The sandbox environment provides a limited access to the underlying operating system.
- In short, Google App Engine is a powerful platform for developers who want to build and deploy applications without worrying about server management or scalability concerns.



Web Frameworks

- App Engine provides a simple Python web application framework called webapp2. App Engine also supports any framework written in pure Python that speaks WSGI, including Django, CherryPy, Pylons, web.py, and web2py.

Datastore

- App Engine provides a no-SQL data storage service.

Authentication

- App Engine applications can be integrated with Google Accounts for user authentication.

URL Fetch service

- URL Fetch service allows applications to access resources on the Internet, such as web services or other data.

Other services

- Email service
- Image Manipulation service – allows to resize, crop, rotate, flip and enhance images
- Memcache – (in-memory key-value cache) used for caching data items, that do not need persistent storage

- Task Queues – it helps applications to run in the background i.e breaking of work into small unit called tasks and enqueued
- Scheduled Tasks service – cron service for schedule Tasks that trigger events specified times or regular intervals

Windows Azure Web Sites

- Windows Azure Web Sites is a Platform-as-a-Service (PaaS) from Microsoft.
- Azure Web Sites allows you to host web applications in the Azure cloud.
- Shared & Standard Options.
 - In the shared option, Azure Web Sites run on a set of virtual machines that may contain multiple web sites created by multiple users.
 - In the standard option, Azure Web Sites run on virtual machines (VMs) that belong to an individual user.
- Azure Web Sites supports applications created in ASP.NET, PHP, Node.js and Python programming languages.
- Multiple copies of an application can be run in different VMs, with Web Sites automatically load balancing requests across them.

Developing Applications in Cloud

How to create a cloud-based application

Step 1.

- **Research** Conduct market research for your product and define your main competitors. This will help you later in marketing your product to the right audience and establishing your monetization model.
- Define your buyer personas and make a list of features your competitors already have.

Step 2.

- **Hire reliable developers** To build a cloud-based application, you need developers that have experience with cloud architectures and cloud computing.
- Building a cloud-based application is different from building a traditional software product, so make sure your software developers have the right skill set.

Step 3.

- **Build an architecture and features specification** The architecture is by far the most important part of your cloud-based product, so make sure you and your team have a clear understanding of how your product will work.

- Involve a business analyst along with a development team lead to plan out every part of your project from its architecture to each of its features.

Step 4.

- **Define your technology stack** A cloud-based product should have bulletproof security and be reliable, so you should pay close attention to your technology stack.
- The architecture should be scalable and maintainable, and all technologies should work well together. For the client part of your app, you'll need: HTML + CSSA JavaScript framework like Vue.js, Angular, or React For your server, you'll need a framework as well.
- Your choice of frameworks will depend on the programming language you choose. PHP (Laravel or Symfony)JavaScript (Node.js)Ruby (Ruby on Rails)Python (Django or Flask)You'll also need a database for storing your backend data.
- I advise you to combine several databases for backup purposes. Here are two common databases to choose from: MySQLPostgreSQLFor your server, you can choose Nginx or Apache. I also advise you to have more than one server: in case of downtime, you'll always have a backup.

Step 5.

- **Choose a monetization strategy** After you've done your competitor research, you should have an idea of your monetization strategy.
- Most cloud-based applications earn their money through subscriptions.
- This is convenient for clients, who don't need to pay every time a new product version is released, and it's also convenient for you, as you get a constant revenue stream and can predict your earnings.

Step 6.

- **Build an MVP** It's best to release your product as soon as possible instead of spending years polishing a product that may not meet market needs.
- Define the most important features for your product, implement them, and release your product to the market. Then use analytics to determine what features to implement next and what to improve.

Step 7.

- **Test and launch your product** Now that your MVP is ready, it's time to test each of its features and conduct a full regression test on all devices, operating systems, and browsers that are popular among your target audience.
- Launch your product along with your marketing campaign and start analyzing the market response.

Step 8.

- **Improve and update your app** Now it's time to improve your application and add new features so your users get value each time they renew their subscription.
- Apart from adding new functionality, you need to maintain existing functionality such as by updating libraries and updating your software to support new versions of operating systems or browsers.

Deployment

- Deployment design is an iterative process
 - Deployment Design
 - Performance evaluation
 - Deployment refinement

Deployment Design

- Deployment is created as per deployment configuration
- Variables includes
 - No of server in each tier
 - Computing
 - Memory
 - Storage
- Deployment is created by provisioning the cloud resource as per deployment configuration
- All the process are automated like launching , configuring , and deployment

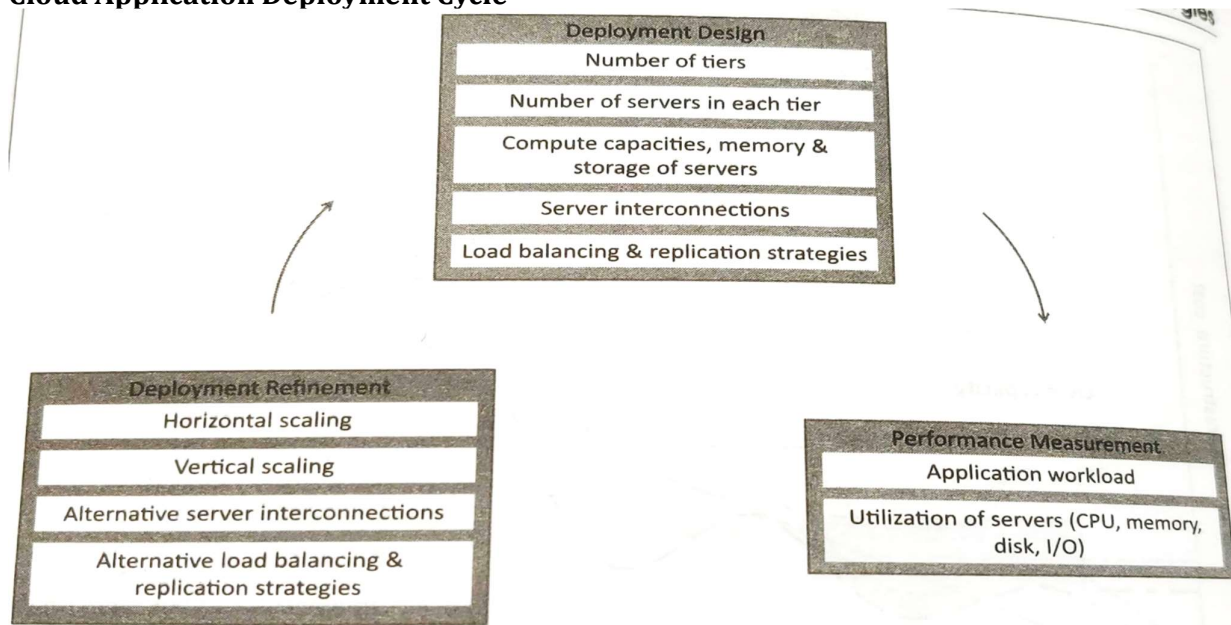
Performance Evaluation

- Once deployment is done , check the application meets the performance requirement
- This step involves the monitoring the workload , measuring various parameters such as response time and throughput
- In addition to this utilization of servers (CPU , memory , disk, IO etc) in each tier is monitored.

Deployment Refinement

- After evaluating the performance of the application , deployments are refined so that the application can meet the performance requirement.
- Various alternatives can exist in this step like
 - Vertical scaling
 - Horizontal scaling
 - Alternative Load balancing
 - Alternative server interconnections
 - Replication strategies

Cloud Application Deployment Cycle



Scaling

- Cloud Scalability is a strategic resource allocation operation.
- Cloud scalability is the ability of the system's infrastructure to handle growing workload requirements while retaining a consistent performance adequately.
- Scale up
 - Vertical scale up
 - Horizontal Scale up
- Scalability is an important factor for the business whose resource demands are increasing slowly and predictably.

Terms related to scalability

Scale Up: Increasing / Adding resources in existing server.

Scale Down: Decreasing / Taking out resources from the existing server.

Scale Out: Adding new or extra Servers in the cluster.

Scale In: Taking out added servers from cluster.

Types of Scaling:

Vertical Scaling – Scale Up

This ability to add resources to accommodate increasing workload volumes is **vertical scaling**. It can resize your server with no change in your code.

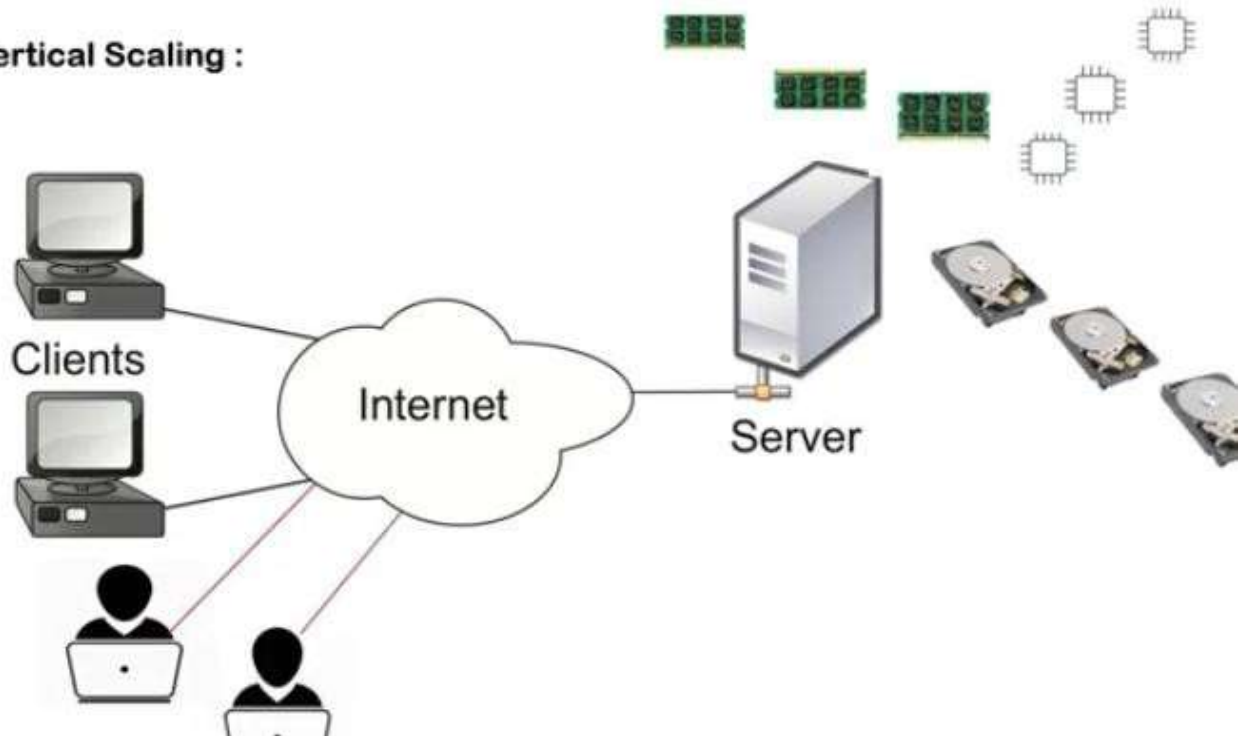
Horizontal Scaling – Scale Out

It is the addition of nodes to the existing infrastructure to accommodate additional workload volumes.

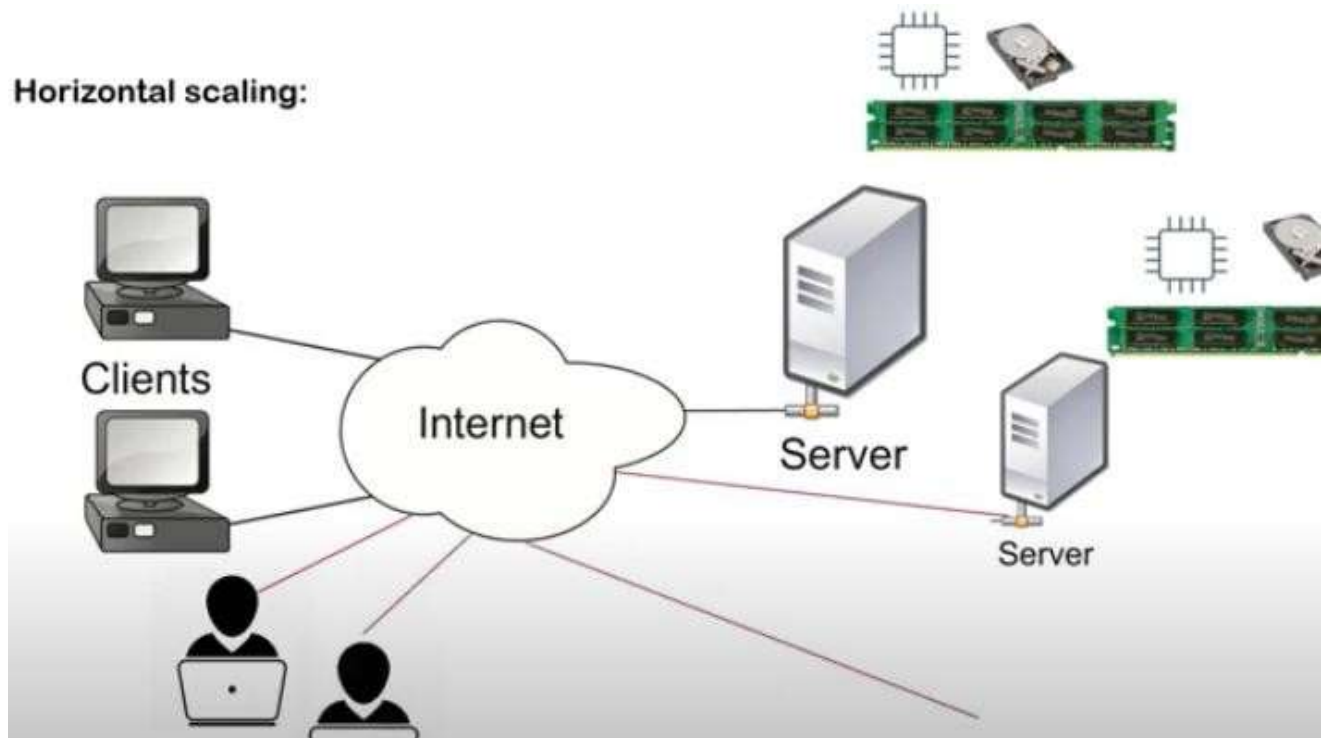
Diagonally Scaling

It combines the scaling up and scaling down. Scaling down is the removal of storage resources as the requirements decrease.

Vertical Scaling :



Horizontal scaling:



Benefits of cloud scalability :

- **Performance:**
It facilitates performance. Scalable architecture has the ability to handle the bursts of traffic and heavy workloads that will come with business growth.
- **Cost Efficient:**
One can allow business to grow without making any expensive changes in the current setup. This reduces the cost implications of storage growth making scalability in the cloud very cost effective.
- **Easy and Quick:**
Scaling up or scaling out in the cloud is simpler; you can commission additional VMs with a few clicks and after the payment is processed, the additional resources are available without any delay.
- **Capacity:**
With scalability, you don't have to worry about additional capacity needs. It ensures that with the continuous growth of your business the storage space in cloud grows as well.

Cloud Elasticity

- Cloud Elasticity is a tactical resource allocation operation.
- It provides the necessary resources required for the current task and handles varying loads for short periods
- Cloud elasticity is a system's ability to manage available resources according to the current workload requirements dynamically.
- Because of the pay-per-use pricing model of modern cloud platforms, cloud elasticity is a cost-effective solution for a business with a dynamic workload.

What is the need of cloud elasticity?

- **Over-provisioning**
- **Under-provisioning**

Difference:

Scalability	Elasticity
"Increasing" the capacity to meet the "increasing" workload	"Increasing or reducing" the capacity to meet the "Increasing or reducing" workload
In a scaling environment, the available resources may exceed to meet the "future demands"	In the elasticity environment, the available resources matches the "current demands" as closely as possible
Scalability adapts only to the "workload increase" by "provisioning" the resources in an "incremental manner"	Elasticity adopts to both the "workload increase" as well as "workload decrease" by provisioning and " <u>deprovisioning</u> " resources in an "automatic" manner
Scalability enables a corporate to meet expected demands for services with "long-term" "strategic needs"	Elasticity enables a corporate to meet unexpected changes in the demand for services with "short-term" , tactical needs

Deployment & Management Services

- Cloud base deployment & management services allow you to easily deploy and manage applications in the cloud. These services automatically handle deployment tasks such as capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- Amazon Elastic Beanstalk
 - Amazon provides a deployment service called Elastic Beanstalk that allows you to quickly deploy and manage applications in the AWS cloud.
 - Elastic Beanstalk supports Java, PHP, .NET, Node.js, Python, and Ruby applications.
 - With Elastic Beanstalk you just need to upload the application and specify configuration settings in a simple wizard and the service automatically handles instance provisioning, server configuration, load balancing and monitoring.
- Amazon CloudFormation
 - Amazon CloudFormation is a deployment management service from Amazon.
 - With CloudFront you can create deployments from a collection of AWS resources such as Amazon Elastic Compute Cloud, Amazon Elastic Block Store, Amazon Simple Notification Service, Elastic Load Balancing and Auto Scaling.

- A collection of AWS resources that you want to manage together are organized into a stack.

Database Services

- A database service is a managed offering that allows users to store, retrieve, and manage data in a structured or unstructured format, typically over the internet.
- These services eliminate the need to manually set up and maintain database infrastructure, allowing users to focus on application development while the service provider handles database management, scalability, backups, and security.

Types of Database Services:

1. **Relational Database Services (RDS):** These services store data in structured formats using tables, rows, and columns, and are managed through SQL (Structured Query Language). Common examples include:
 - Amazon RDS (supports MySQL, PostgreSQL, Oracle, SQL Server)
 - Google Cloud SQL
 - Azure SQL Database
2. **NoSQL Database Services:** NoSQL services store unstructured or semi-structured data, often using key-value pairs, documents, or graphs. They are ideal for applications with variable data models or high-traffic environments. Examples include:
 - Amazon DynamoDB (key-value and document database)
 - MongoDB Atlas (document database)
 - Google Firestore (NoSQL cloud database)
3. **Data Warehousing Services:** These services are designed for storing and querying large datasets, often used for analytics and business intelligence. Examples include:
 - Amazon Redshift
 - Google BigQuery
 - Azure Synapse Analytics
4. **In-Memory Databases:** In-memory databases store data in RAM rather than on disk, making them extremely fast. They are used for caching and real-time applications. Examples include:
 - Amazon ElastiCache (supports Redis and Memcached)
 - Azure Cache for Redis

5. **Graph Database Services:** These databases store data in a graph format, focusing on relationships between data points. They are used in social networks, fraud detection, and recommendation engines. Examples include:

- Amazon Neptune
- Neo4j Aura

Key Features of Database Services:

1. **Managed Infrastructure:** Providers handle the operational aspects like database setup, patching, backups, scaling, and failover, so users can focus on data and application logic.
2. **Automatic Scaling:** Many database services offer automatic scaling capabilities to handle varying loads, scaling up or down based on demand.
3. **Backup and Recovery:** Managed database services typically include automatic backups and the ability to restore to specific points in time.
4. **Security:** Database services offer built-in security features such as encryption at rest and in transit, access control, and integration with identity management systems.
5. **High Availability:** Services often include replication, failover, and disaster recovery features to ensure data is available even in case of failure.
6. **Global Reach:** Cloud-based database services allow data to be distributed across multiple geographic locations, ensuring low-latency access and data redundancy.

Benefits of Database Services:

- **Simplicity:** Users don't need to manage hardware, software, or security patches, making database management much easier.
- **Cost Efficiency:** Pay-as-you-go pricing means you only pay for the resources you use, with options for reserved pricing for long-term savings.
- **Scalability:** These services are designed to scale effortlessly, supporting both small applications and large, enterprise-grade systems.
- **Reliability:** Built-in features like backups, replication, and failover ensure your data is safe and always accessible.

Popular Database Services:

- **Amazon RDS:** A versatile relational database service supporting multiple database engines.
- **Google Cloud SQL:** Managed relational database service for MySQL, PostgreSQL, and SQL Server.

- **Amazon DynamoDB:** A fast and scalable NoSQL database for key-value and document use cases.
- **MongoDB Atlas:** A popular managed service for the document-based MongoDB database.
- **Amazon Redshift:** A fully managed data warehouse service optimized for high-performance analytics.

Database services allow businesses and developers to access powerful database capabilities without the complexity of managing infrastructure, ensuring performance, security, and reliability.

Database Services – Amazon RDS

- Amazon RDS (Relational Database Service) is a managed database service provided by Amazon Web Services (AWS) that simplifies the setup, operation, and scaling of relational databases in the cloud.
- It automates common database tasks like provisioning, patching, backups, and scaling, allowing developers and administrators to focus on their applications rather than database maintenance.

Key Features of Amazon RDS:

1. **Support for Multiple Database Engines:** Amazon RDS supports several popular relational database engines, including:
 - Amazon Aurora (MySQL and PostgreSQL-compatible)
 - MySQL
 - PostgreSQL
 - MariaDB
 - Oracle
 - Microsoft SQL Server
2. **Automated Backups:** RDS provides automatic backups and point-in-time recovery, ensuring your data is protected. You can restore a database to any point within a specified retention period (from 1 to 35 days).
3. **High Availability with Multi-AZ Deployments:** For critical applications, RDS can be configured for Multi-AZ (Availability Zone) deployments. This ensures high availability by automatically replicating the database to a standby instance in another availability zone. In case of a failure, RDS automatically fails over to the standby instance.

4. Scalability:

- Storage Scaling: Storage for RDS can automatically increase as needed, ensuring you don't run out of space.
- Read Replicas: You can create read replicas to handle increased read traffic and offload work from the primary database.
- Vertical Scaling: You can increase CPU, memory, or storage capacity by resizing the database instance to meet growing application needs.

5. Security:

- Encryption: Data can be encrypted at rest using AWS Key Management Service (KMS).
- VPC Integration: You can run RDS within an Amazon Virtual Private Cloud (VPC), which isolates your database from the internet.
- IAM Integration: AWS Identity and Access Management (IAM) policies can be used to control access to RDS instances.

6. **Monitoring and Performance Optimization:** RDS integrates with Amazon CloudWatch to provide performance metrics, allowing you to monitor CPU utilization, read/write performance, memory usage, and more. It also provides Performance Insights to help optimize database queries and performance.

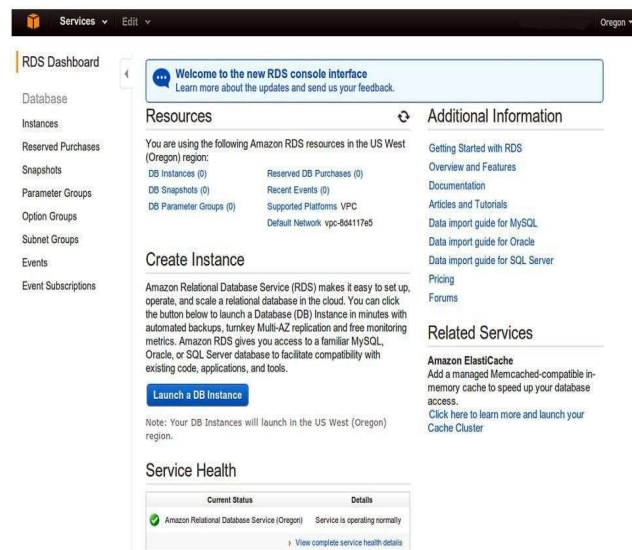
7. **Automated Patching and Maintenance:** RDS handles software patching of the database engine to keep it up to date with the latest versions and security patches.

8. **Cost-Effective Pricing:** RDS offers pay-as-you-go pricing, so you only pay for what you use. You can also use Reserved Instances for long-term commitments to save costs.

Benefits of Amazon RDS:

- Ease of Use: RDS simplifies database management with automated backups, scaling, and maintenance.
- Reliability: Multi-AZ deployments provide built-in high availability and failover mechanisms.
- Scalability: RDS can scale up or down depending on the application's workload.
- Security: AWS ensures robust security controls, including encryption, VPC isolation, and identity management.
- Cost-Effective: You can optimize costs by using reserved instances, automatic scaling, and paying only for the resources you need.

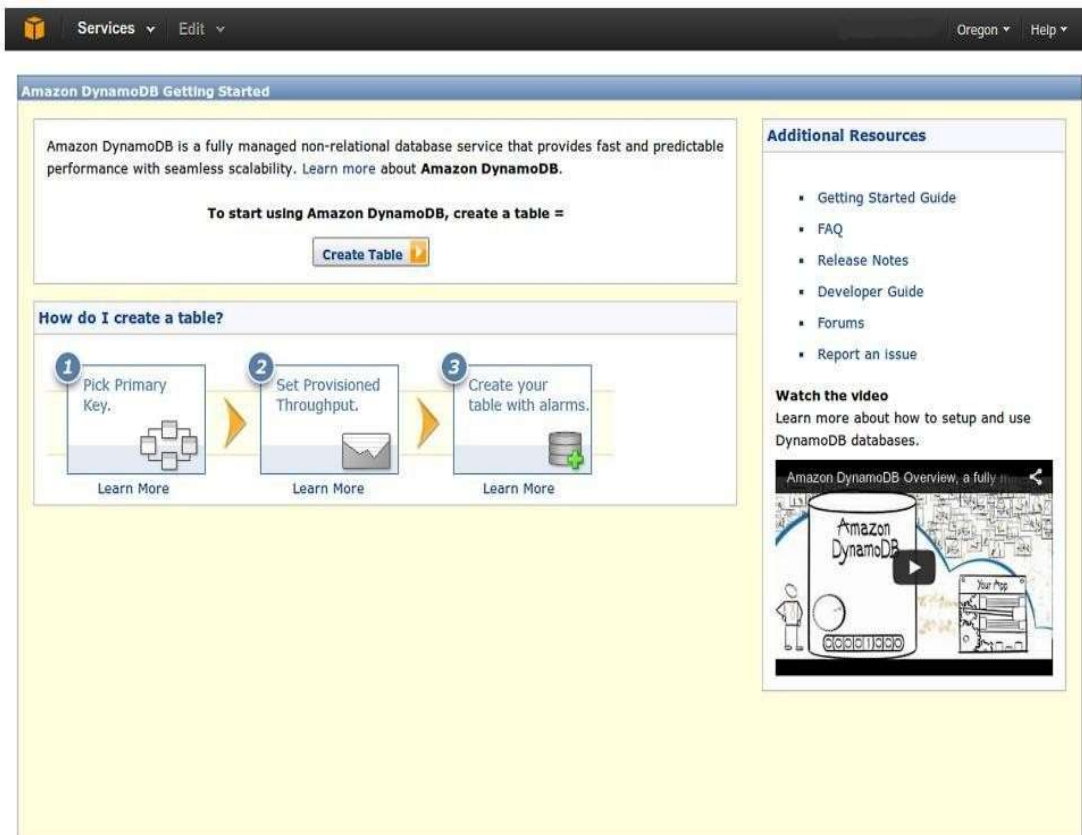
- Amazon Relational Database Service (RDS) is a web service that makes it easy to setup, operate and scale a relational database in the cloud.
- Launching DB Instances
- The console provides an instance launch wizard that allows you to select the type of database to create (MySQL, Oracle or SQL Server) database instance size, allocated storage, DB instance identifier, DB username and password. The status of the launched DB instances can be viewed from the console.
- Connecting to a DB Instance
- Once the instance is available, you can note the instance end point from the instance properties tab. This end point can then be used for securely connecting to the instance.



Database Services – Amazon Dynamo DB

- Amazon Dynamo DB is the non- relational (No-SQL) database service from Amazon.
- Data Model
 - The DynamoDB data model includes include tables, items and attributes.
 - A table is a collection of items and each item is a collection of attributes.
 - To store data in DynamoDB you have to create a one or more tables and specify how much throughput capacity you want to provision and reserve for reads and writes.

- Fully Managed Service
 - DynamoDB is a fully managed service that automatically spreads the data and traffic for the stored tables over a number of servers to meet the throughput requirements specified by the users.
- Replication
 - All stored data is automatically replicated across multiple availability zones to provide data durability.



Database Services– Google Cloud SQL

- Google Cloud SQL is a fully managed relational database service provided by Google Cloud. It allows you to run and manage relational databases in the cloud with ease.
- Cloud SQL supports popular database engines such as MySQL, PostgreSQL, and SQL Server, making it versatile for many use cases.

Key Features of Google Cloud SQL:

1. Managed Infrastructure:
 - Cloud SQL automates common database tasks such as provisioning, patch management, backups, and failover, reducing the operational overhead for users.

2. Support for Multiple Relational Databases:

- MySQL, PostgreSQL, and SQL Server are natively supported, providing a familiar environment for developers and database administrators.

3. Scalability:

- Cloud SQL provides automatic scaling of storage, and users can manually scale CPU and memory to accommodate workload demands. This is particularly useful for growing applications.

4. High Availability (HA):

- For mission-critical applications, Cloud SQL supports high availability configurations with automatic failover. If an instance fails, Google Cloud automatically promotes a standby replica to minimize downtime.

5. Automated Backups and Point-in-Time Recovery:

- Cloud SQL automatically performs daily backups, and you can enable point-in-time recovery, allowing you to restore your database to any point within the last seven days.

6. Security:

- Data can be encrypted in transit using SSL, and at rest using AES-256 encryption.
- Cloud SQL integrates with Google Cloud IAM (Identity and Access Management) for role-based access control.
- It supports private IP connections, which allows you to connect securely from within a Virtual Private Cloud (VPC).

7. Integration with Google Cloud Services:

- Cloud SQL can be easily integrated with other Google Cloud services, such as Google Kubernetes Engine (GKE), BigQuery (for data analytics), App Engine, and Cloud Functions.

8. Read Replicas:

- You can create read replicas to distribute read traffic across multiple instances, improving performance for high-read workloads.

9. Compliance and Security Standards:

- Cloud SQL complies with a variety of security standards such as SOC 1, 2, 3, ISO/IEC 27001, and HIPAA, making it suitable for handling sensitive data.

10. Monitoring and Performance Optimization:

- It integrates with Google Cloud's Stackdriver Monitoring and Logging, providing insights into database performance and enabling you to set up alerts for database issues.

Benefits of Google Cloud SQL:

1. Ease of Use:

- It eliminates the need to manually manage database infrastructure, allowing developers to focus on writing code and optimizing applications.

2. Scalability:

- The service can scale to meet the needs of applications, whether it's a small-scale startup or a large enterprise.

3. High Availability and Reliability:

- With automated backups, failover, and recovery options, Cloud SQL provides a robust solution for ensuring data is always available.

4. Cost Efficiency:

- Cloud SQL uses a pay-as-you-go pricing model, allowing users to pay only for the resources they need, making it cost-efficient for businesses of all sizes.

5. Security:

- Google Cloud ensures the highest level of data security through encryption and integrations with IAM and VPCs.

Common Use Cases:

1. Web and Mobile Applications:

- Use Cloud SQL to store and manage relational data for web applications, mobile apps, and APIs.

2. Enterprise Applications:

- Cloud SQL can manage databases for enterprise resource planning (ERP), customer relationship management (CRM), and financial systems.

3. Data Warehousing:

- Although designed for transactional workloads, Cloud SQL can also be used in conjunction with BigQuery for analytics and reporting.

4. E-commerce:

- Cloud SQL is ideal for handling high-volume e-commerce applications that need reliable, scalable, and secure data storage.

Integration Example:

- You can connect Google Cloud SQL with App Engine or Compute Engine to store relational data for your applications. It's also easily integrated with Cloud Functions to process database events and BigQuery for large-scale data analysis.
- Google SQL is the relational database service from Google.
- Google Cloud SQL service allows you to host MySQL databases in the Google's cloud.
- Launching DB Instances
 - You can create new database instances from the console and manage existing instances. To create a new instance you select a region, database tier, billing plan and replication mode.
- Backups
 - You can schedule daily backups for your Google Cloud SQL instances, and also restore backed-up databases.
- Replication
 - Cloud SQL provides both synchronous or asynchronous geographic replication and the ability to import/ export databases.

The screenshot shows the Google Cloud Console interface for creating a new Cloud SQL instance. The breadcrumb navigation at the top indicates the path: Cloud > Cloud SQL > cloud:mydbinstance. On the left, there is a sidebar with a 'NEW INSTANCE' button and tabs for 'Summary' and 'Operations Log'. The main content area is titled 'Create a New Cloud SQL Instance' and is divided into several sections: 'Summary' (with fields for Instance ID 'mydbinstance' and Region 'United States'), 'Resources and Billing' (with fields for Tier 'D0 - 128M RAM' and Billing Plan 'Per Use'), and 'Options' (with fields for Backup Window '7:30 AM - 11:30 AM' and Replication 'Synchronous'). A right-hand summary panel provides a condensed view of these settings, including the instance ID, region, tier, billing plan, backup window, and replication mode. At the bottom right of this panel are 'Confirm' and 'Cancel' buttons.

Google Cloud Console

@gmail.com | Customer support | Sign out

Cloud > Cloud SQL > cloud:mydbinstance

NEW INSTANCE

Summary

Operations Log

Create a New Cloud SQL Instance

Instance ID mydbinstance

Region United States

Resources and Billing

Tier D0 - 128M RAM

Billing Plan Per Use

Usage will be charged by each hour that the database instance is accessed.

Package

Usage will be charged monthly by the number of days the database instance exists.

Options

Backup Window 7:30 AM - 11:30 AM

Replication Synchronous

Best reliability, slower writes.

Asynchronous

Faster writes, less reliability within a few seconds of updates.

Summary

mydbinstance

This is your instance id

United States

D0

\$0.025 per hour - 128M RAM

Per Use

Your will be charged by the hour.

Backups at 7:30 AM - 11:30 AM

Synchronous replication

Confirm Cancel

Database Services– Google Cloud Data store

- **Google Cloud Datastore** is a highly scalable, fully managed **NoSQL database** service provided by **Google Cloud**. It is designed for applications that require flexible, schema-less data storage and can handle massive amounts of unstructured or semi-structured data, making it ideal for web, mobile, and gaming applications.

Key Features of Google Cloud Datastore:

1. NoSQL Database:

- Cloud Datastore is a NoSQL database that stores data in the form of **entities** and **key-value pairs**. Entities are grouped into **kinds**, which are equivalent to tables in relational databases, and each entity can have different properties, offering flexibility.

2. Schemaless Data Storage:

- Datastore is schemaless, meaning you can store data without needing a predefined schema. This allows dynamic data models, where you can easily add, remove, or modify fields without affecting the whole database.

3. Fully Managed:

- Google Cloud Datastore is fully managed, meaning Google takes care of database provisioning, scaling, replication, backups, and infrastructure maintenance.

4. High Scalability:

- Cloud Datastore automatically scales to handle large datasets and high traffic loads. It's designed to support applications with thousands or millions of users.

5. ACID Transactions:

- Cloud Datastore provides **ACID (Atomicity, Consistency, Isolation, Durability)** transactions at the entity group level, ensuring strong consistency and reliability for critical operations.

6. Built-in Queries:

- You can perform powerful queries using **Google Cloud Datastore's query language**. It supports simple key-based lookups, composite queries (with multiple filters), and sorting based on property values.

7. Automatic Indexing:

- Cloud Datastore automatically indexes every field in the database, enabling fast queries. You can customize indexing to optimize performance or reduce storage costs.

8. Global Distribution and High Availability:

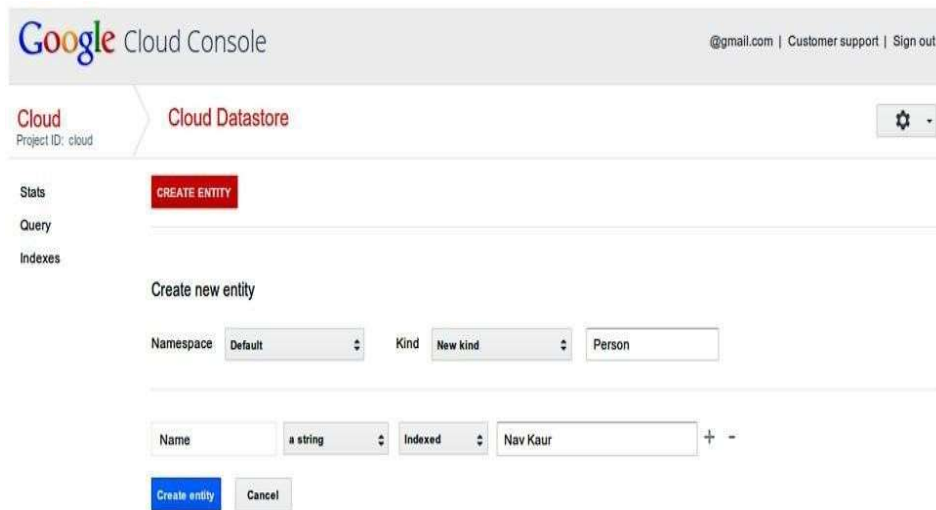
- Cloud Datastore is globally distributed and replicates data across multiple regions to ensure high availability and low-latency access. This makes it ideal for applications that serve a global user base.

9. Security and IAM Integration:

- Datastore integrates with **Google Cloud IAM (Identity and Access Management)**, allowing fine-grained access control to manage who can view, update, or delete data.
- It also supports **encryption at rest** and **in transit** for securing your data.

10. Integration with Google Cloud Services:

- Datastore integrates seamlessly with other Google Cloud services, such as **App Engine, Compute Engine, and Cloud Functions**, making it a versatile backend for various applications.



The screenshot shows the Google Cloud Console interface for Cloud Datastore. At the top, the 'Google Cloud Console' header is visible with a user email and links for 'Customer support' and 'Sign out'. Below the header, the 'Cloud Datastore' section is active, showing a 'Project ID: cloud' and a 'CREATE ENTITY' button. On the left, there are tabs for 'Stats', 'Query', and 'Indexes'. The main area is titled 'Create new entity' and contains a form with the following fields: 'Namespace' (set to 'Default'), 'Kind' (set to 'New kind'), and 'Person' (a text input). Below these, there is a 'Name' field (set to 'a string'), an 'Indexed' checkbox, and a text input for 'Nav Kaur'. At the bottom, there are 'Create entity' and 'Cancel' buttons.

Database Services– Windows Azure SQL DB

- Azure SQL Database is a fully managed relational database service provided by Microsoft Azure.
- It is built on SQL Server technology and offers a cloud-based platform to run, manage, and scale relational databases without having to worry about infrastructure, hardware, or software maintenance.
- Azure SQL Database is ideal for applications that need high availability, scalability, and security with minimal management overhead.

Key Features of Azure SQL Database:

1. Managed Service:

- Azure SQL Database is fully managed, meaning Microsoft takes care of database administration tasks such as patching, backups, and performance tuning, allowing developers to focus on building applications.

2. Built on SQL Server:

- The service is based on Microsoft SQL Server, offering the familiar T-SQL syntax and database engine, which allows for easy migration of existing on-premises SQL Server applications to the cloud.

3. Elastic Scaling:

- Azure SQL Database supports dynamic scaling to accommodate varying workloads. It can scale resources such as compute, memory, and storage automatically or manually, allowing for cost-efficient operations.

4. High Availability and Disaster Recovery:

- Azure SQL Database is built with high availability (HA) in mind. It provides built-in fault tolerance with automatic replication and failover to ensure minimal downtime.
- Geo-replication allows you to create database replicas in different Azure regions for disaster recovery and improved reliability.

5. Backup and Restore:

- Azure SQL Database provides automated backups and point-in-time restore functionality. You can restore your database to any point within a configurable retention period (up to 35 days), ensuring data protection.

6. Security and Compliance:

- It offers advanced security features, including data encryption at rest and in transit using TDE (Transparent Data Encryption) and SSL.
- Azure Active Directory (Azure AD) integration provides centralized user authentication and identity management.
- Features like firewall rules, auditing, threat detection, and advanced data security help protect databases from unauthorized access and attacks.
- It is compliant with industry standards such as GDPR, HIPAA, ISO, and SOC.

7. Intelligent Performance:

- Azure SQL Database includes built-in AI-driven performance optimization. Features like automatic indexing, query tuning, and adaptive query processing help improve database performance without manual intervention.

8. Multiple Deployment Options:

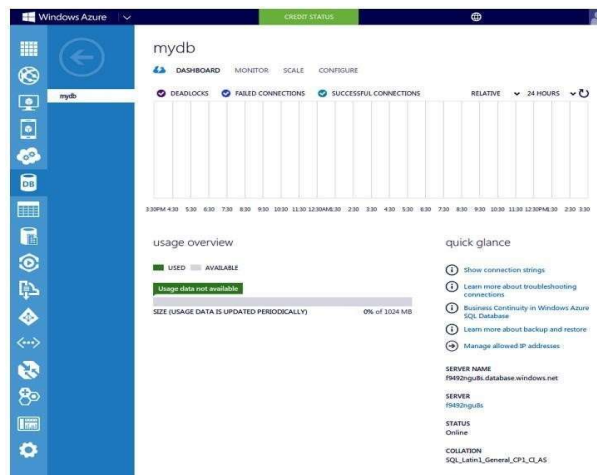
- Azure SQL Database can be deployed in different environments based on specific requirements:
 - Single Database: A dedicated, isolated database for applications with steady, predictable workloads.
 - Elastic Pool: Allows you to host multiple databases in a single pool, sharing resources to handle varying workload demands across databases efficiently.
 - Managed Instance: A near 100% compatible version of SQL Server for more complex database migrations and scenarios requiring greater control and SQL Server features.

9. Intelligent Insights and Analytics:

- The service provides Intelligent Insights, which automatically monitors and detects database performance issues and provides recommendations for optimization.
- Integration with Azure Monitor and Azure Log Analytics allows for detailed performance monitoring and logging.

10. Integration with Other Azure Services:

- Azure SQL Database integrates seamlessly with other Azure services such as Azure App Service, Azure Functions, Power BI, and Azure Machine Learning, enabling powerful analytics, reporting, and machine learning capabilities.



Database Services– Windows Azure Table Service

- Windows Azure Table Service is a non-relational (No-SQL) database service from Microsoft.
- Data Model
 - The Azure Table Service data model consists of tables having multiple entities.
 - Tables are divided into some number of partitions, each of which can be stored on a separate machine.
 - Each partition in a table holds a specified number of entities, each containing as many as 255 properties.
 - Each property can be one of the several supported data types such as integers and strings.
- No Fixed Schema
 - Tables do not have a fixed schema and different entities in a table can have different properties.

Microservices

Architecture Serverless Computing

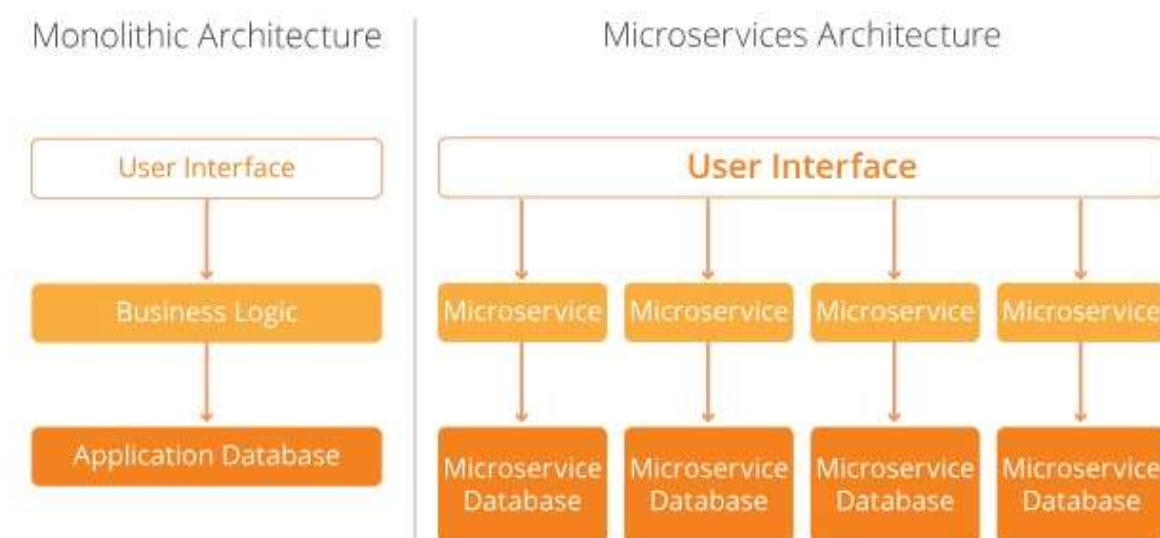
What is a microservice?

- Imagine taking an application, chopping it up into pieces, and running it as a collection of smaller parts instead of one monolithic whole. That's basically what a microservices architecture.
- Each piece of the application is called a 'microservice,' and it performs one service only, runs independently of the other parts of the application, operates in its own environment, and stores its own data.
- Despite the name, microservices do not have to be small. What makes them 'micro' is that they only handle one service and are part of a larger application.
- From the user's perspective, an application built with microservices has a single interface and should work just the same as an application designed as one stack.
- However, behind the scenes each microservice has its own database and runs separately from the rest of the application.
- In addition, microservices within the same application can be written in different languages and use different libraries.

Monolithic architecture vs. microservices architecture

- Microservices are often contrasted against monolithic architecture.
- Monolithic architecture is the classic way of building an application.
- A monolithic application is a single stack, with the user interface on top, the business logic in the middle, and the database on the bottom.
- Usually a monolithic application is hosted on a specific server or set of servers. One of the downsides of constructing an application in this way is that any small change to the application means the entire stack has to be updated. Another downside is that if one part of the application breaks, the whole application might fail.

Monolithic architecture vs. microservices architecture



Advantages of microservices

Resilience:

- The application is divided up, one part of the application breaking or crashing does not necessarily affect the rest of the application.

Selective scalability:

- Instead of scaling the entire application, only the microservices that receive a great deal of usage can be scaled

Easier to add or update features:

- Features can be rolled out or updated one at a time, instead of updating the entire application stack

Flexibility for developers:

- Microservices can be written in different languages and each have their own libraries.

Can microservices be part of a serverless architecture?

- Microservices can be deployed in a variety of ways, they can be part of a serverless architecture, hosted in containers, developed using PaaS, or, theoretically, used to build a locally hosted application.
- However, the advantages of building an application out of microservices are perhaps most apparent when the application is hosted in the cloud, either using containers or in a serverless architecture.

What are serverless microservices? How does a serverless microservices architecture work?

- Serverless microservices are deployed within a serverless vendor's infrastructure and only run when they are needed by the application.
- Depending on the size of a microservice, it may also be broken up into even smaller functions.

What is the difference between a microservice and a serverless function?

- This distinction is still being defined by the tech community, but typically, a microservice is larger and can do more than a function.
- A function is a relatively small bit of code that performs only one action in response to an event.
- Depending on how developers have divided up an application, a microservice may be equivalent to a function (meaning it performs only one action), or it may be made up of multiple functions.

Sample Questions

1. What is Platform as a Service (PaaS), and how does it differ from Infrastructure as a Service (IaaS) and Software as a Service (SaaS)?
2. What are the key benefits of using PaaS for application development and deployment?
3. In what scenarios is PaaS more advantageous than traditional on-premise infrastructure?
4. What are the primary features that differentiate Heroku from other PaaS providers?
5. How does Google App Engine simplify the process of deploying and scaling applications?

6. What are the pricing models of popular PaaS providers, and how do they affect application cost management?
7. What are the major considerations when developing applications for the cloud compared to on-premises infrastructure?
8. How can cloud-based applications be designed for auto-scaling, and what tools support this functionality?
9. What challenges do developers face when deploying applications in the cloud, and how can these be addressed?
10. What is Database as a Service (DBaaS), and how does it differ from self-managed database hosting?
11. What are the advantages of using DBaaS like Amazon RDS, Azure SQL Database, or Google Cloud SQL for enterprise applications?
12. How do DBaaS providers ensure high availability, scalability, and data security?
13. How does the microservices architecture approach differ from the traditional monolithic application model?
14. What are the advantages and challenges of breaking an application into microservices?
15. How can cloud services and tools like Kubernetes and Docker assist in deploying and managing microservices at scale?
16. What is serverless computing, and how does it eliminate the need for traditional server management?
17. In what situations is serverless architecture more efficient than traditional cloud-based or containerized applications?
18. How does the billing model in serverless computing (e.g., AWS Lambda, Azure Functions) differ from traditional cloud computing models, and how does it affect cost management?