

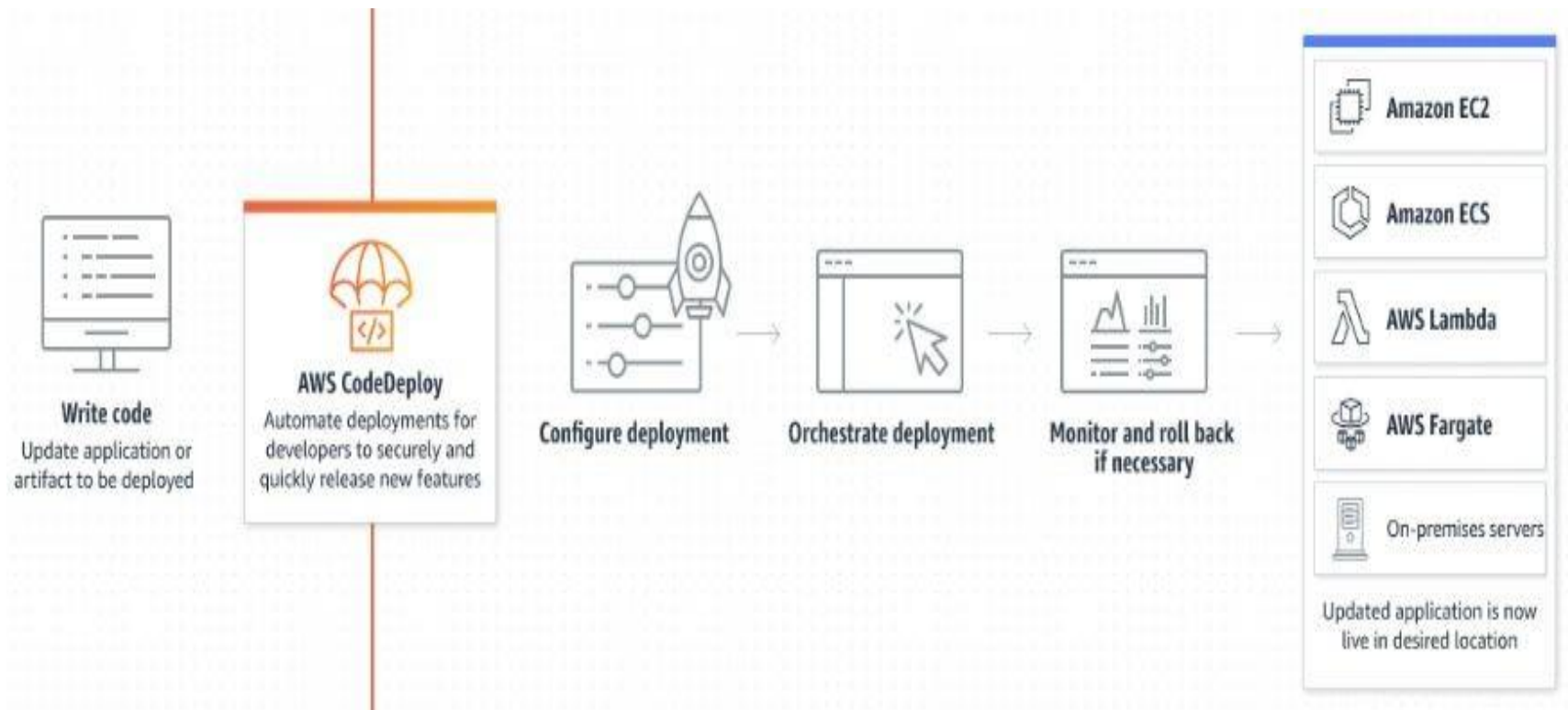
Service Deployment

AWS CodeDeploy

- is a fully managed deployment service by Amazon Web Services (AWS) that automates the process of deploying applications to various computing environments.
- It helps developers release new software versions efficiently while minimizing downtime and reducing manual errors.

Key Features of AWS CodeDeploy:

- ✓ **Automated Deployments –**
- ✓ **Supports Multiple Environments –**
- ✓ **Minimizes Downtime –**
- ✓ **Scalability –**
- ✓ **Rollback Capabilities –**
- ✓ **Monitoring & Logging –**



AWS Deployment Strategies

AWS provides different ways to **update and deploy applications** to make sure they run smoothly without downtime. Here are some **common strategies** you can use:

1. Bootstrapping an Instance

- When a new **Amazon EC2 instance** (a virtual server) starts, it installs all the required software, application code, and settings.
- **Pros:** Flexible, can be customized as needed.

2. Prebaking with Amazon Machine Images (AMI)

- Instead of installing software every time, you create a **pre-built image** (AMI) with everything already installed.
- **Pros:** Faster launches and deployments.

3. Blue/Green Deployment

- You create **two identical environments**:
 - **Blue** (current version of the app).
 - **Green** (new version of the app).
- Once the green version is tested and ready, traffic is switched from blue to green.
- **Pros:** No downtime, easy rollback if something goes wrong.

4. Rolling Deployment

- Instead of replacing everything at once, the new version is gradually rolled out. Example: If you have 10 servers, update 2 at a time until all are updated.
- **Pros:** Avoids downtime, reduces risk of failure.

5. In-Place Deployment

- The old version of the app is stopped, the new version is installed, and then restarted on the same servers.
- **Pros:** No need for extra servers, simple to manage.

Which Deployment Strategy Should You Choose?

There's no single "best" method—it **depends on your needs**:

- ✓ **Need zero downtime?** → Use **Blue/Green Deployment**
- ✓ **Want faster updates?** → Use **Prebaked AMIs**
- ✓ **Want a gradual update?** → Use **Rolling Deployment**
- ✓ **Have a simple setup?** → Use **In-Place Deployment**

By **mixing these strategies**, businesses can **deploy, scale, and monitor applications** effectively on AWS.

What are the technologies used in cloud computing?

- There are four major cloud computing technologies-
- Virtualization
- Service Oriented Architecture
- Grid Computing
- Utility Computing

Grid Computing

- is a **distributed computing** technology where multiple computers (which may be located in different places) work together as a **single system** to solve large and complex problems.
- Instead of relying on a single powerful machine, **grid computing splits a task** into smaller parts and distributes them across different computers (nodes) in the grid.
- These computers can belong to different organizations or locations but work towards a common goal.

Key Features of Grid Computing:

- ✓ **Distributed Resources** – Computers in the grid are located in different physical locations.
- ✓ **High Efficiency** – Tasks are divided among multiple nodes, increasing speed and reducing workload on a single system.
- ✓ **Heterogeneous Network** – Different types of computers with different configurations can work together.
- ✓ **Fault Tolerance** – If one computer fails, others can take over the work.

Example of Grid Computing:





1. Folding@home (Scientific Research)

One of the most popular grid computing projects is **Folding@home**, which studies **protein folding and diseases** like Alzheimer's and cancer. It uses thousands of personal computers to process complex scientific calculations when they are not in use.

2. SETI@home (Space Research)

The **SETI@home** project searches for extraterrestrial life by analyzing signals from space using thousands of volunteered computers.

Where is Grid Computing Used?

-  Scientific Research – Climate modeling, genetics, space exploration.
-  Healthcare – Drug discovery, disease simulations.
-  E-commerce – Managing large-scale transactions.
-  Banking & Finance – Risk analysis, fraud detection.

Utility Computing

- is a **pay-per-use** model where computing resources like storage, processing power, and networking are provided **on demand**.
- Instead of investing in expensive hardware and software, businesses can **rent computing power and only pay for what they use**.
- This model helps companies **reduce costs** and **scale resources up or down** as needed.

Key Features of Utility Computing:

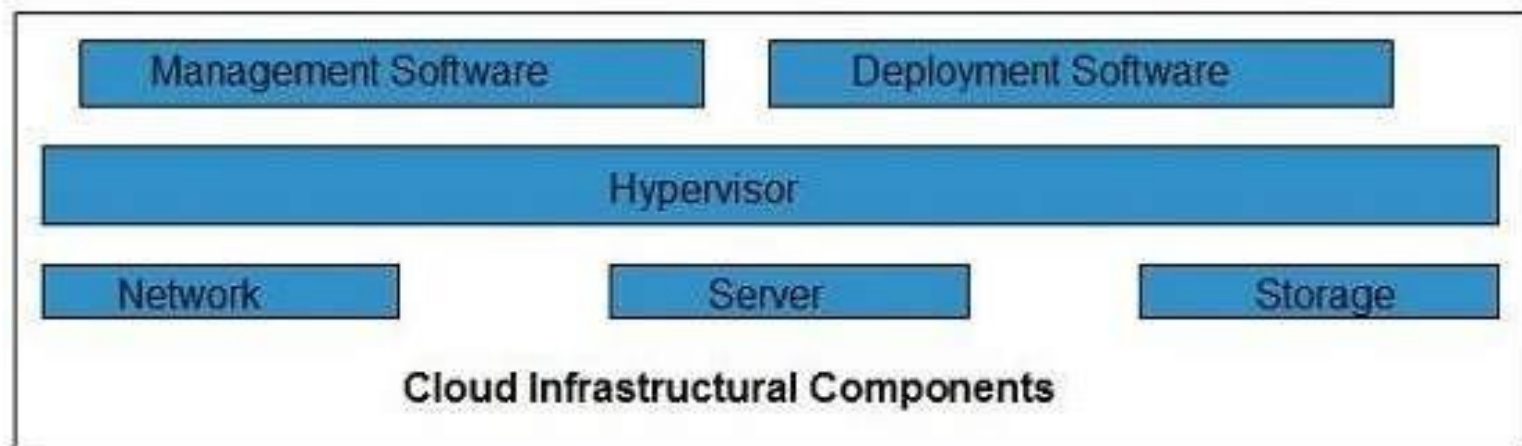
- ✓ **Pay-as-you-go Model**
- ✓ **Scalability**
- ✓ **Cost Savings**
- ✓ **Flexibility**

Example - Streaming Services

- **Netflix, YouTube, and Spotify** use cloud-based storage and processing.
- They only pay for the **server resources required to stream videos/music** to users in real time.

Cloud computing infrastructure for implementing cloud based services

- **Cloud infrastructure** consists of servers, storage devices, network, cloud management software, deployment software, and platform virtualization.



Cloud infrastructure is the foundation of cloud computing, enabling businesses to run applications efficiently, securely, and at scale.

1 A **hypervisor** creates virtual machines (VMs) that share server resources.

2 **Management software** configures and monitors cloud infrastructure.

3 **Deployment software** installs applications on the cloud.

4 **Networking** enables communication between cloud services.

5 **Servers** process application requests and distribute resources.

6 **Storage** ensures data is stored securely and replicated for reliability.

Key Components of Cloud Infrastructure

1. Hypervisor (Virtual Machine Manager)

- A **hypervisor** is software or firmware that enables **virtualization** by allowing multiple virtual machines (VMs) to share a **single physical server**.
- It helps in **efficient resource allocation** and improves scalability.
 - Examples: **VMware vSphere, Microsoft Hyper-V, KVM (Kernel-based Virtual Machine)**

2. Management Software

- Helps **configure, monitor, and maintain** cloud infrastructure.
- Provides automation, provisioning, and cloud resource monitoring.
- Examples: **AWS Management Console, Microsoft Azure Portal, Google Cloud Console**

3. Deployment Software

- Used to **deploy and integrate applications** into cloud environments.
- Automates application deployment for seamless scaling and updates.
 - Examples: **Docker, Kubernetes, AWS CodeDeploy**

4. Networking 🌐

- Cloud networking connects cloud services via the **Internet or private networks**.
- Allows businesses to **customize network routes, protocols, and security**.
 - Examples: **Virtual Private Cloud (VPC), Load Balancers, Content Delivery Network (CDN)**

5. Servers 🏠

- Cloud servers **compute, process, and allocate resources** to applications.
- Ensure **scalability, security, and high availability**.
- Examples: **Amazon EC2 , Google Compute Engine (GCE), Microsoft Azure Virtual Machines**

6. Storage 💾

- Cloud storage offers **reliable, scalable, and redundant** data storage.
- Multiple copies of data are stored to **prevent data loss** in case of failures.
- Examples: **Amazon S3 (Object Storage), Google Cloud Storage, Azure Blob Storage**

Key Components of Google App Engine's Environment

1. Container-Based Infrastructure

- Applications run in **pre-configured containers** on Google's cloud infrastructure.
- These containers are **optimized** for performance, security, and scalability.





2. Multiple Runtime Environments

- Google App Engine supports **multiple programming languages**, including:
 - Java 
 - Python 
 - PHP 
 - Go 

Developers can choose a runtime based on their preferred language and technology stack.

3. Automatic Scaling

- App Engine **automatically scales** applications based on incoming traffic.
- It **allocates or deallocates** computing resources dynamically.

- **Fully Managed Services** 
 - Developers don't need to worry about **server management, patching, or maintenance**.
 - Google manages **load balancing, traffic splitting, and application monitoring**.
- **Built-in Security** 
 - Google App Engine provides **firewall rules, authentication, and encryption** to protect applications.
 - It integrates with **Google Cloud Identity & Access Management (IAM)** for user control.
- **Support for Standard & Flexible Environments** 
 - **Standard Environment:** Uses predefined runtimes and scales automatically.
 - **Flexible Environment:** Runs applications in **Docker containers**, providing more customization.
- **Integration with Google Cloud Services** 
 - GAE seamlessly integrates with:
 - **Cloud Datastore** (Database)
 - **Cloud Storage** (File storage)
 - **Cloud Logging & Monitoring** (Application insights)

Setting Up Your Google Cloud Environment

To start using Google Cloud Platform (GCP), follow these steps:

1 Sign in to Google Cloud

2 Create a Google Cloud Project
A Google Cloud project organizes your resources.

3 Enable Billing 

4 Enable Required APIs
APIs must be enabled before using certain services.

5 Install & Set Up gcloud CLI (Command-Line Interface)

6 Set Up IAM (Identity and Access Management) Permissions

7 Configure Cloud Storage (Optional)

8 Deploy Your First App (Example: Cloud Run)