AWS Compute Services

What is Compute in AWS?

Compute in AWS refers to the ability to process data and run applications using cloud-based virtual servers, containers, or serverless functions — without needing physical hardware.

Why "Compute"?

"Compute" means computing power (like CPUs, memory, and networking) that allows you to run code, host websites, process data, or launch applications.

Key AWS Compute Services

Service	Type	Use Case
Amazon EC2	Virtual Server	Run Linux/Windows servers, host apps, custom control over OS & networking
AWS Lambda	Serverless	Run code without provisioning servers; great for automation or small functions
Amazon ECS / EKS	Containers	Run containerized apps (ECS = AWS-managed, EKS = Kubernetes-based)
AWS Elastic Beanstalk	PaaS	Deploy web apps easily (auto-handles scaling, patching, etc.)
AWS Lightsail	Simplified VM	Pre-configured VMs for small apps, websites, or development environments
AWS Batch	Job Processing	Run batch computing workloads at any scale
AWS Outposts	On-Prem Compute	Brings AWS compute to your own data center (hybrid solution)

Compute Models in AWS

- AWS provides different compute models based on your use case:
- IaaS (EC2): Infrastructure as a Service – complete control over the server.
- PaaS (Elastic Beanstalk): Platform as a Service – just deploy your code.
- FaaS (Lambda): Function as a Service – run code on demand, no server to manage.

Example Scenarios

Host a website or backend API → Use EC2 or Elastic Beanstalk

Run a scheduled report every night → Use AWS Lambda with EventBridge

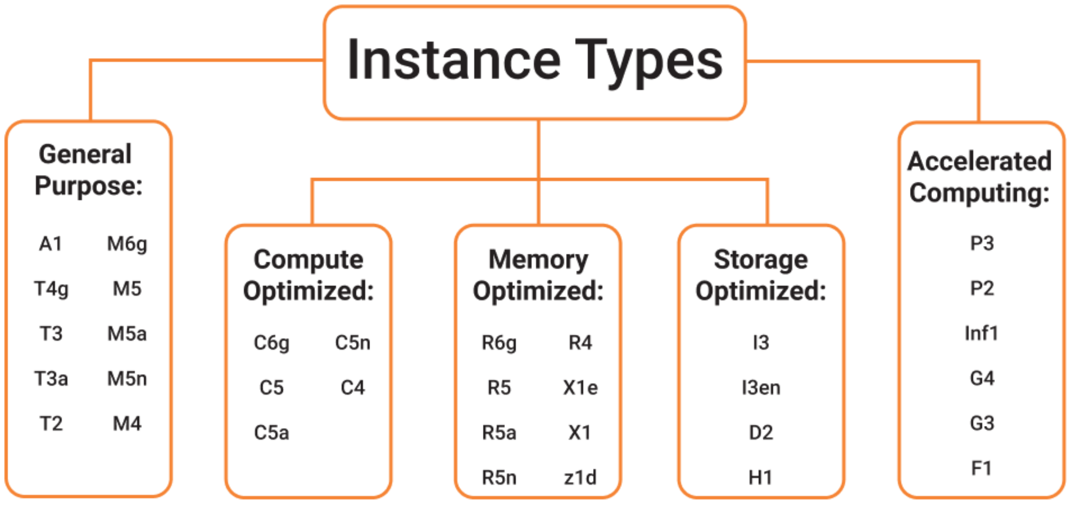
Deploy Docker containers → Use ECS or EKS

Build a low-cost blog → Use AWS Lightsail

Benefits of AWS Compute

- Scalability: Auto scale up/down based on traffic
- Flexibility: Choose OS, CPU, memory, instance type
- Cost-effective: Pay-as-you-go pricing model
- Global reach: Deploy in multiple AWS regions

Amazon EC2



1. EC2 Instance Types

AWS offers a wide array of instances organized into families optimized for different use-cases:

- **General Purpose** (e.g., T3, M6g): Balanced CPU, memory, and networking.
- **Compute-Optimized** (C7g, C8): High CPU performance for compute-heavy workloads.
- **Memory-Optimized** (R7, X3): For in-memory databases & analytics.

- **Storage-Optimized** (I5, D4): High-speed local NVMe storage.
- **Accelerated Computing** (P6/G6): GPU-backed instances for ML/AI.
- **Specialized**: ARM Graviton4 options (e.g., M8g) with substantial performance gains.
- Instance naming (e.g., r5d.xlarge) follows this pattern: Family → Generation → Size

2. EC2 Pricing Model

Multiple EC2 pricing options provide flexibility and cost optimization:

- **On-Demand**: Pay per hour/second usage—easy but expensive.
- **Reserved Instances / Savings Plans**: Commit 1–3 years for discounted rates.
- **Spot Instances**: Bid on unused capacity with steep discounts (70–90%) but subject to termination.
- **Dedicated Hosts / Savings**: For specific regulatory or licensing compliance.

Pricing varies by instance type, region, OS, and commitment type. For example, a Linux m5.large is roughly $0.096/hour in us-east-1.

3. EC2 Instance Life Cycle

EC2 instances go through these key states:

1. **Launch**: Instance is created based on AMI, instance type, config.
2. **Pending**: Initial setup & booting.
3. **Running**: Fully operational—billing starts.
4. **Stopping / Stopped**: Shuts down; root volumes persisted (EBS).
5. **Terminating / Terminated**: Instance deleted—ephemeral storage lost.

Transitioning stops and terminates can affect data, IP addresses, and networking.

4. Auto Scaling

Amazon EC2 Auto Scaling ensures the right number of instances are running:

- Define **Auto Scaling Groups (ASGs)** with min/max/desired size.
- Use **Launch Templates**, **Health Checks**, and **Scaling Policies** (based on CPU/memory/requests).
- Automatically replaces unhealthy instances and balances across AZs.
- Works seamlessly with Spot Instances, Reserved Instances, and On-Demand in mixed ASGs.
- **No extra charge** for Auto Scaling—only pay for underlying resources

5. Elastic Load Balancing (ELB)

Automatically distributes incoming traffic across healthy EC2 instances

- **Types**:
    - **ALB** (Layer 7): For HTTP(S) with host/path-based routing.
    - **NLB** (Layer 4): For ultra-low latency & TCP.

    - **GWLB**: For virtual appliances.
- **High availability** across AZs with health checks.
- **Pricing** based on hourly usage and capacity units (LCU/NLCU).
- **Integrates** with ASGs to register/deregister instances automatically.

🔧 Best Practices

1. Choose **instance type** such as m8g.large (Graviton4) for a web app.
2. Launch as part of an **Auto Scaling Group**, min=2, max=10.
3. Front with an **Application Load Balancer** distributing HTTP traffic.
4. Configure **Scaling Policies**: Add instances when CPU >70%; remove if <30%.
5. Optimize cost using Reserved + Spot instance mix.

📌 Summary Table

• **Feature**	**Description**
• **Instance Types**	Family, generation, and size optimized for specific workloads
• **Pricing Models**	On-Demand, Reserved, Spot, Dedicated
• **Life Cycle States**	Launch → Running → Stop/Terminate
• **Auto Scaling**	ASGs ensure performance, replace unhealthy instances, scale automatically
• **Elastic Load Balancing**	Distributes traffic, ensures high availability, work with ASGs