COMPUTE SERVICES IN AWS

EC2, ECS and Lambda

Compute services in the cloud are used for the provision of processing power/compute capacity required to run a workload. So, the virtual machines that are going to be running your applications on the cloud for example.

AWS has three main compute services, namely, EC2, ECS and Lambda. These are discussed further below:

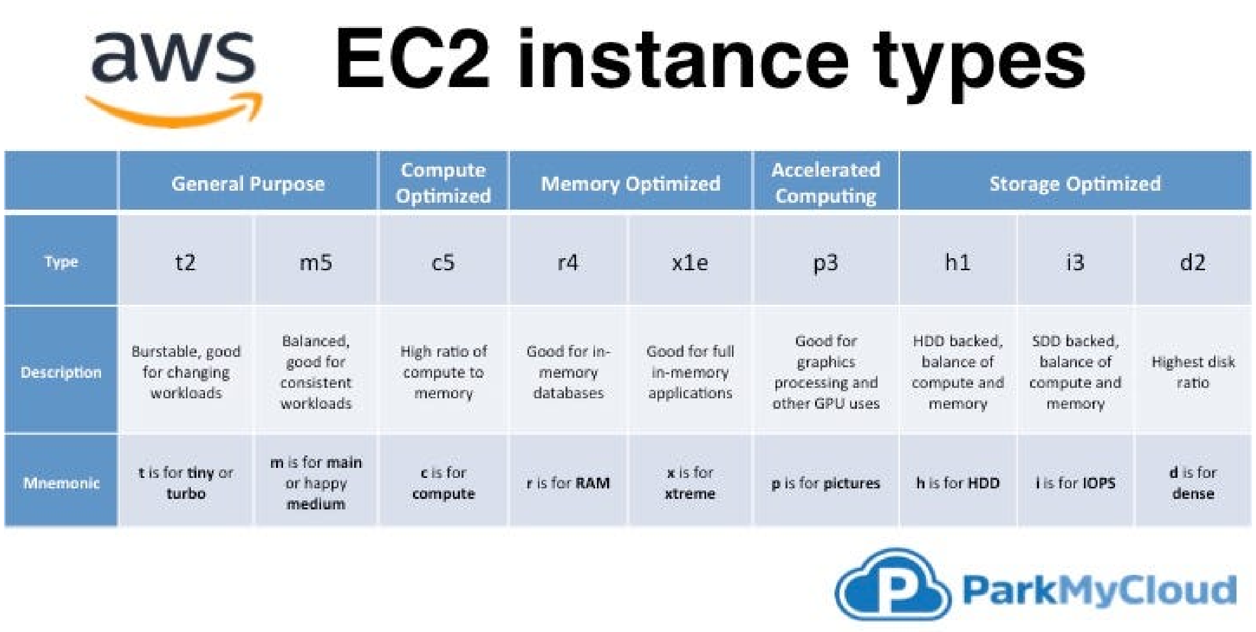
EC2 (Elastic Compute Cloud): EC2 provides virtual machines in the cloud, known as instances, offering a wide range of flexibility. These instances can be optimized for various use cases, including memory, compute, or storage. EC2 instances are highly customizable, allowing users to configure CPU, memory, storage, and networking capacity according to their needs. They also come with security features like security groups and IAM roles for access control.

ECS (Elastic Container Service): ECS is a managed container orchestration service designed for running containerized applications. While it supports Docker containers, it's not limited to them; it also accommodates containers built to the Open Container Initiative (OCI) image format. Users can manage ECS clusters themselves on EC2 instances or opt for AWS Fargate, a serverless option that abstracts away infrastructure management. With Fargate, users focus solely on deploying and managing containers, without dealing with server provisioning.

Lambda: Lambda is a serverless compute service that allows users to run code without managing servers. Users upload their code, and Lambda handles scaling and infrastructure management automatically. When a Lambda function is invoked, users receive an ARN (Amazon Resource Name) that uniquely identifies the function. Lambda is commonly used with API Gateway to create serverless REST APIs, enabling infinitely scalable endpoints. Lambda prioritizes simplicity, offering a streamlined development experience without the burden of managing infrastructure.

Types of EC2 Instances

Because EC2 Instances are used to power a wide variety of workloads with similarly varying resource needs, Amazon provides us with different types of EC2 instances. These are as follows:



EC2 instance types

	General Purpose		Compute Optimized	Memory Optimized		Accelerated Computing	Storage Optimized		
Type	t2	m5	c5	r4	x1e	p3	h1	i3	d2
Description	Burstable, good for changing workloads	Balanced, good for consistent workloads	High ratio of compute to memory	Good for in-memory databases	Good for full in-memory applications	Good for graphics processing and other GPU uses	HDD backed, balance of compute and memory	SDD backed, balance of compute and memory	Highest disk ratio
Mnemonic	t is for **tiny** or **turbo**	m is for **main** or happy **medium**	c is for **compute**	r is for **RAM**	**x** is for **xtreme**	p is for **pictures**	h is for **HDD**	i is for **IOPS**	**d** is for **dense**

ParkMyCloud

It might seem intimidating at first, but the designation names are there to help you, and allow you to make the best decision in regards to the type of EC2 instance that is best suited for your needs.

Memory Optimized Instances:

These instances are designed to deliver fast performance for workloads that process large data sets in memory. They typically have a higher amount of RAM compared to other instance types, making them suitable for memory-intensive applications like in-memory databases, real-time big data analytics, and high-performance computing.

Compute Optimized Instances:

These instances are optimized for compute-bound applications that require high-performance processors. They typically have a high ratio of vCPUs to memory, making them suitable for applications that require intensive computational processing such as gaming servers, scientific modeling, batch processing, and media transcoding.

Storage Optimized Instances:

These instances are designed to deliver high storage performance for workloads that require high I/O performance. They come with local instance storage optimized for high-speed, low-latency access, making them suitable for applications that require frequent and fast access to large data sets, such as NoSQL databases, data warehousing, and data processing.

Hibernation

Hibernation provides the convenience of pausing and resuming the instances, saves time by reducing the startup time taken by applications, and saves effort in setting up the environment or applications all over again. Instead of having to rebuild the memory footprint, hibernation allows applications to pick up exactly where they left off.

While the instance is in hibernation, you pay only for the EBS volumes and Elastic IP Addresses attached to it; there are no other hourly charges (just like any other stopped instance).

Note:
It is not possible to enable or disable hibernation for an instance after it has been launched.

Autoscaling Group and Spanning

Auto Scaling Groups (ASGs) in AWS provide dynamic scaling and management for EC2 instances, ensuring applications maintain performance and availability. ASGs automatically adjust the number of instances based on demand, scaling out to handle increased load and scaling in during low demand, optimizing cost and efficiency.

Key features of ASGs include health checks to automatically replace unhealthy instances, integration with Elastic Load Balancing (ELB) to distribute traffic evenly, and scheduled scaling to anticipate load changes. ASGs also support scaling policies driven by CloudWatch alarms, allowing fine-tuned responsiveness to real-time metrics. This automation ensures applications remain resilient, scalable, and cost-effective.

Note:

ASGs can span across multiple Availability Zones.

Autoscaling tip

When a scenario requires say, “2 as the minimum capacity”, it is always best to construct an architecture that gives more than 2 instances (i.e. at least 3 instances) at all times because if an AZ outage happened, ASG will launch a new instance on the unaffected AZ.

This provisioning does not happen instantly, which means that for a certain period of time, there will only be 1 running instance left.

For example:

If a company needs to deploy at least 2 EC2 instances to support the normal workloads of its application and automatically scale up to 6 EC2 instances to handle the peak load and the architecture is processing mission-critical workloads.

Then, as the Cloud Architect of the company, create an architecture that has at least 3 instances up and running at all times.

On-Demand Capacity Reservation

On-Demand Capacity Reservations provide AWS customers with the ability to reserve EC2 capacity without having to specify individual instance types or AZs in advance. This allows for greater flexibility in managing compute resources, especially for workloads that require precise control over instance placement and instance type selection.

Because Reserved Instances require a fixed one-year or three-year contract so for tasks that have a known capacity but are required for a shorter duration of time, it may be better to use On Demand Capacity Reservations compared to reserve instances.

Elastic Container Service (ECS)

One of the three compute options available in AWS, **Amazon Elastic Container Service (ECS)** is AWS's fully managed container orchestration service that simplifies the deployment and management of Docker containers.

It allows you to run containerized applications on a scalable cluster of EC2 instances or using AWS Fargate for serverless container execution.

Note:

Because Fargate is serverless, if you do not wish to worry about management of container workloads then, use Fargate.

Lambda

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it the fastest way to turn an idea into a modern, production, serverless applications. It is marketed as infinitely scalable, and rather cheap.

Note:

AWS Lambda can also run containerized images. This is often used in deployments which require external dependencies to be installed on the system.

ENV variables for Lambda functions

Quick Tip:

When you create or update Lambda functions that use environment variables, AWS Lambda encrypts them using the AWS Key Management Service. When your Lambda function is invoked, those values are decrypted and made available to the Lambda code.

So lambda uses KMS for Environment Variables. But not by default, you still have to do some shit.