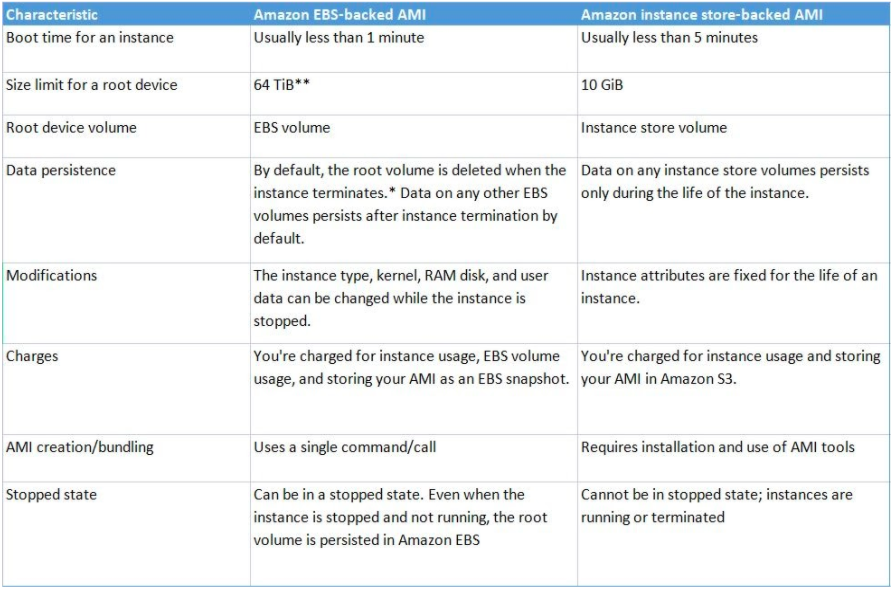
Amazon EC2

* currently supports a variety of operating systems including: Amazon Linux, Ubuntu, Windows Server, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Fedora, Debian, CentOS, Gentoo Linux, Oracle Linux, and FreeBSD.
* With EC2 you have full control at the operating system layer.
* Key pairs are used to securely connect to EC2 instances:
  + A key pair consists of a public key that AWS stores, and a private key file that you store.
  + For Windows AMIs, the private key file is required to obtain the password used to log into your instance.
  + For Linux AMIs, the private key file allows you to securely SSH (secure shell) into your instance.
* You are limited to running up to a total of 20 On-Demand instances across the instance family, purchasing 20 Reserved Instances, and requesting Spot Instances per your dynamic spot limit per region **(by default).**
* The **AWS Nitro** System is the underlying platform of the next generation of EC2 instances.
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
  + t-type and m-type for general purpose
  + c-type for compute optimized
  + r-type, x-type and z-type for memory optimized
  + d-type, h-type and i-type for storage optimized
  + f-type, g-type and p-type for accelerated computing
* Storage volumes for temporary data that are deleted when you STOP or TERMINATE your instance, known as instance store volumes. Take note that you can stop an EBS-backed instance but not an Instance Store-backed instance. You can only either start or terminate an Instance Store-backed instance.
* Metadata, known as tags, that you can create and assign to your EC2 resources
* **Host Recovery** for Amazon EC2 automatically restarts your instances on a new host in the event of an unexpected hardware failure on a Dedicated Host.
* **EC2 Hibernation** is available for On-Demand and Reserved Instances running on freshly launched M3, M4, M5, C3, C4, C5, R3, R4, and R5 instances running Amazon Linux and Ubuntu 18.04 LTS. You can enable hibernation for your EBS-backed instances at launch. You can then hibernate and resume your instances through the AWS Management Console, or though the AWS SDK and CLI using the existing stop-instances and start-instances commands. Hibernation requires an EC2 instance to be an encrypted EBS-backed instance.
* **Hibernating an instance saves the contents of RAM to the Amazon EBS root volume. When the instance restarts, the RAM contents are reloaded.**
* AWS imposes a small hourly charge if an Elastic IP address is not associated with a running instance, or if it is associated with a stopped instance or an unattached network interface.
* You are charged for any additional Elastic IP addresses associated with an instance.
* If data is transferred between these two instances, it is charged at “Data Transfer Out from EC2 to Another AWS Region” for the first instance and at “Data Transfer In from Another AWS Region” for the second instance.
* An Elastic IP address is for use in a specific region only.
* By default, all AWS accounts **are limited to five (5) Elastic IP addresses per region,** because public (IPv4) internet addresses are a scarce public resource.
* **Instance states:** Start, Stop, Hibernate and Terminate

Root Device Volumes

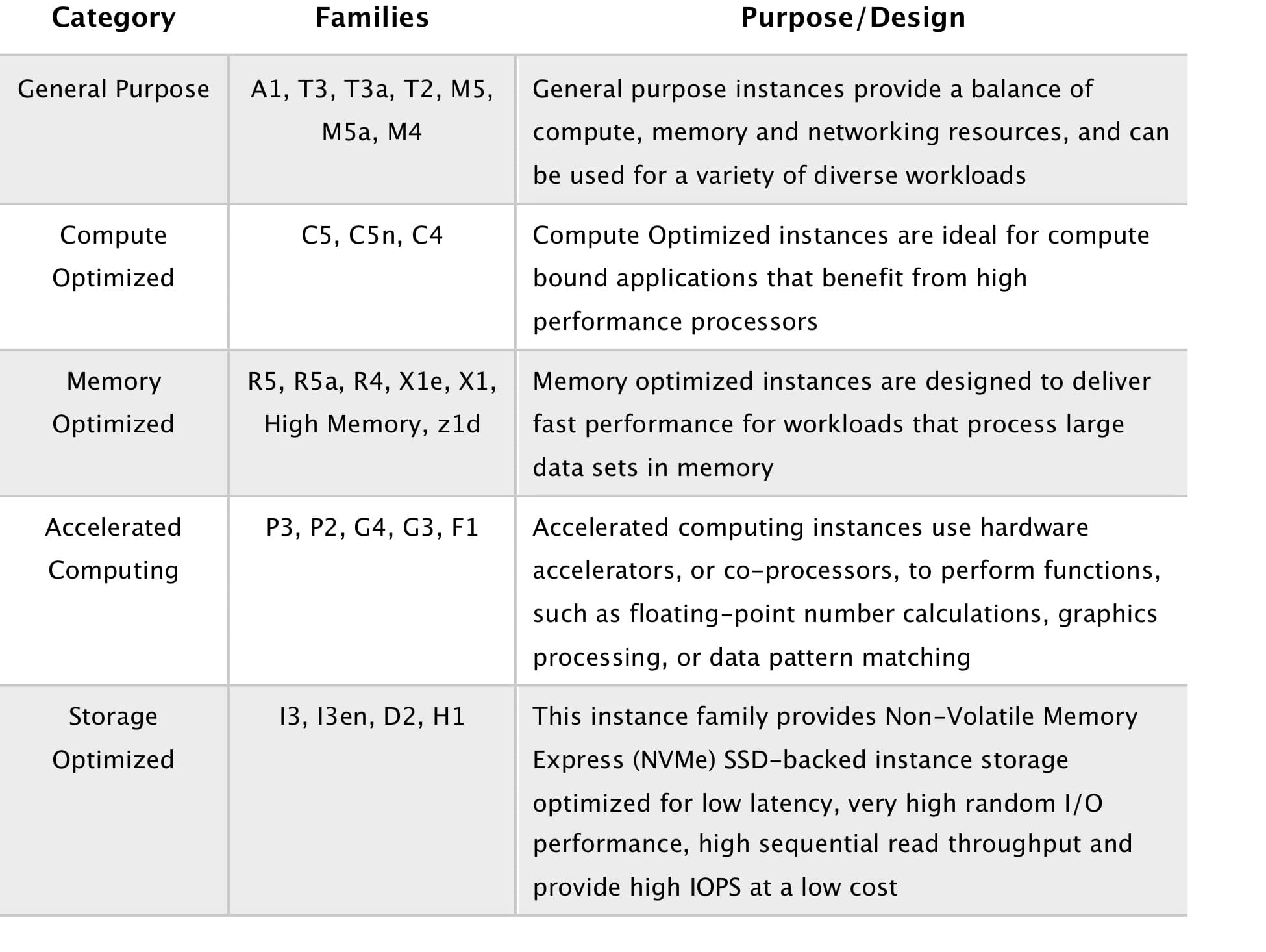
* The root device volume contains the image used to boot the instance.
* Instance Store-backed Instances
  + Any data on the instance store volumes is deleted when the instance is terminated (**instance store-backed instances do not support the Stop action**) or if it fails (such as if an underlying drive has issues).
  + You should also back up critical data from your instance store volumes to persistent storage on a regular basis.
* Amazon EBS-backed Instances
  + An Amazon EBS-backed instance can be stopped and later restarted without affecting data stored in the attached volumes.
  + When in a stopped state, you can modify the properties of the instance, change its size, or update the kernel it is using, or **you can attach your root volume to a different running instance for debugging or any other purpose**.
  + By default, the **root device volume** for an AMI **backed by Amazon EBS** is deleted when the instance terminates.
  + Previously, to launch an encrypted EBS-backed EC2 instance from an unencrypted AMI, you would first need to create an encrypted copy of the AMI and use that to launch the EC2 instance. **Now, you can launch encrypted EBS-backed EC2 instances from unencrypted AMIs directly8999**.



Amazon Machine Image (AMI)

* Package OS and additional installations in a reusable template called Amazon Machine Images.
* **A template for the root volume** for the instance (OS, application server, and applications)
* **Launch permissions** that control which AWS accounts can use the AMI to launch instances
* **A block device mapping** that specifies the volumes to attach to the instance when it’s launched
* You can copy AMIs to different regions using aws CLI.
* There is a **data transfer charge when** copying AMI from one region to another
* Image Hardening: After you apply security hardening or some customization to an ec2 instance from Amazon Linux Image, you can create a hardend image of it and then launch all the new EC2s based on that image so you don't have to apply security hardening individually.

Instance Types

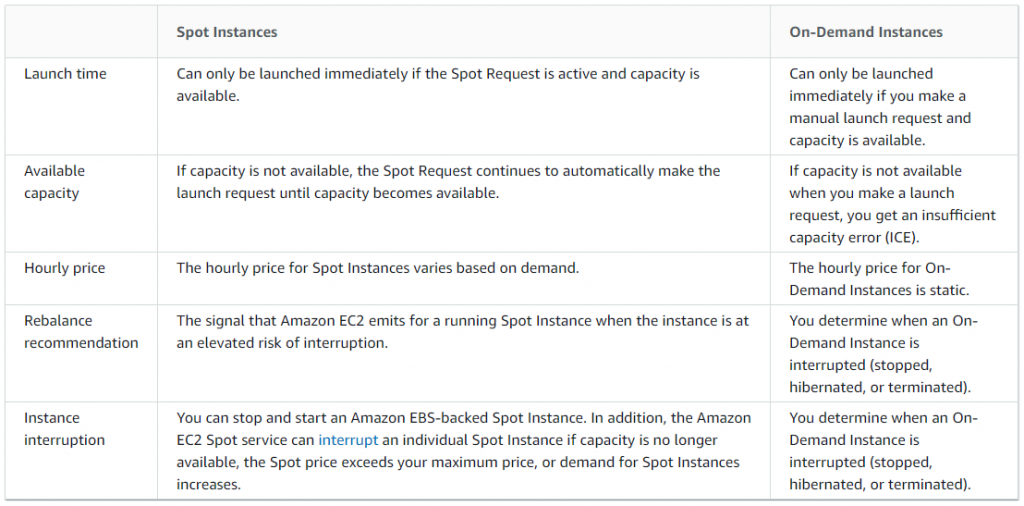


**Pricing**

* On-Demand  – pay for the instances that you use by the second, with no long-term commitments or upfront payments.
  + Pay for hours used with no commitment.
  + Low cost and flexibility with no upfront cost.
  + Ideal for auto scaling groups and unpredictable workloads.
  + Good for dev/test.
* Reserved – make a low, one-time, up-front payment for an instance, reserve it for a *one*– or *three*-year term, and pay a significantly lower hourly rate for these instances**, up to 70% off.**
* Can **switch AZ within the same region**.
* **Instance type modifications are supported for Linux only.**
* Cannot change the instance size of Windows RIs.
* Billed whether running or not.
  + The Standard class provides the **most significant discount** but you can only modify some of its attributes during the term.
    - It can also be sold in the Reserved Instance Marketplace.
    - Can change the instance size **within the same instance type.**
    - Can be used in Auto Scaling Groups.
    - Can be used in Placement Groups.
    - Can be shared across multiple accounts within Consolidated Billing.
  + The Convertible class provides a lower discount than Standard Reserved Instances, but can be exchanged for another Convertible Reserved Instance with different instance attributes. However, this one cannot be sold in the Reserved Instance Marketplace.

|  |  |  |
| --- | --- | --- |
|  | **Standard RI** | **Convertible RI** |
| **Terms**  (average discount off On-Demand) | 1 year (40%)  3 years (60%) | 1 year (31%)  3 years (54%) |
| Change Availability Zone, Instance size (for Linux OS), Networking type | Yes | Yes |
| Change instance families, operating system, tenancy, and payment option |  | Yes |
| Benefit from Price Reductions |  | Yes |

* Scheduled Reserved Instances = reserved for specific periods of time, accrue charges hourly, billed in monthly increments over the term (1 year). Scheduled RIs match your capacity reservation to a predictable recurring schedule.
  + After you complete your purchase, the **instances are available to launch during the time windows that you specified**.
  + When you require for a fraction of day / week / month
* Spot – request unused EC2 instances, which can lower your costs significantly. Spot Instances are available at up to a 90% discount compared to On-Demand prices.
* You can only cancel Spot Instance requests that are open, active, or disabled. Cancelling a Spot Request does not terminate instances You must first cancel a Spot Request, and then terminate the associated Spot Instances
  + Spot Instances with a **defined duration** (**also known as Spot blocks) are designed not to be interrupted and will run continuously for the duration you select**. Ideal for jobs that take a finite time to complete, such as batch processing, encoding and rendering, modeling and analysis, and continuous integration.
  + **A Spot Fleet** **is a collection of Spot Instances and optionally On-Demand Instances**. The service attempts to launch the number of Spot Instances and On-Demand Instances to meet your specified target capacity. The request for Spot Instances is fulfilled if there is available capacity and the maximum price you specified in the request exceeds the current Spot price. **The Spot Fleet also attempts to maintain its target capacity fleet if your Spot Instances are interrupted.**
  + **A Spot Instance pool** is a set of unused EC2 instances with the same instance type, operating system, Availability Zone, and network platform.
  + You can start and stop your Spot Instances backed by Amazon EBS at Will.
* You can modify instance types and weights for a running EC2 Fleet or Spot Fleet without having to recreate it.
* Allocation strategy for Spot Instances
  + **LowestPrice**– The Spot Instances come from the pool with the lowest price. **This is the default strategy**.
  + **Diversified** – The Spot Instances **are distributed across all pools.**
  + **CapacityOptimized** – The Spot **Instances come from the pool with optimal capacity** for the number of instances that are launching.
  + **InstancePoolsToUseCount** – The Spot Instances are **distributed across the number of Spot pools that you specify**. This parameter is valid only when used in combination with the lowest Price.



Dedicated Hosts – pay for a **physical host that is fully dedicated to running your instances,** and bring your existing per-socket, per-core, or per-VM software licenses to reduce costs.

* You have control over which instances are deployed on that host.
* **Available as On-Demand or with Dedicated Host Reservation**.
* Useful if you have server-bound software licences that use metrics like per-core, per-socket, or per-VM.
* Each dedicated host can only run one EC2 instance size and type.
* Predictable performance.
* Complete isolation.
* Most expensive option.
* Billing is per host.

Dedicated Instances – pay, by the hour, for instances that run on single-tenant hardware.

* Virtualized **instances on hardware just for you**.
* Also uses physically dedicated EC2 servers.
* Does not provide the additional visibility and controls of dedicated hosts (e.g. how instance are placed on a server).
* **Billing is per instance.**
* May share hardware with other non-dedicated instances in the same account.
* **Available as On-Demand, Reserved Instances, and Spot Instances.**
* Cost additional $2 per hour per region.

On-Demand Capacity Reservations **– reserve capacit**y for your Amazon EC2 instances in a specific Availability Zone **for any duration.**

* Unlike Reserved instances, you **don’t need to have one-year or three-year term commitment.**
* When you create a Capacity Reservation, you specify:
  + The Availability Zone in which to reserve the capacity
  + The number of instances for which to reserve capacity
  + The instance attributes, including the instance type, tenancy, and platform/OS
* Your Savings Plans and regional Reserved Instances can be applied with your capacity reservations to receive discounts. Without these, your **capacity reservations do not have billing discounts.**
* Capacity Reservations **can’t be created in placement groups**
* Capacity Reservations **can’t be used with Dedicated Hosts**
* Your capacity reservation usage metrics can be monitored in Amazon Cloudwatch.

[Zonal Reserved Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/reserved-instances-scope.html) provide a capacity reservation for EC2 instances in a specific Availability Zone. The capacity reservation can be used by instances in a placement group. However, it is not possible to explicitly reserve capacity for a placement group.

EC2 Placement Groups

* For fast performance, the placement group tries to place the serves in the same host or same rack or as close as possible to have minimal latency.
* Usually, if you launch the same type of servers in single AZs they may be running on the same host, but it can become risky and more prone to failure. Hence it is recommended to have a multi-AZ setup.
* For single AZ, with the help of a placement group you can specify that you don't want them to be part of the same host or same rack.
* NOTE: It is recommended to launch all instances together. if you launch later it can lead to capacity error in placement groups. To resolve this, you should stop and start all the instances in a placement group.
* Recommended to keep instance types homogenous within a placement group.

Types of Placement Groups:

There are three types of placement groups available:

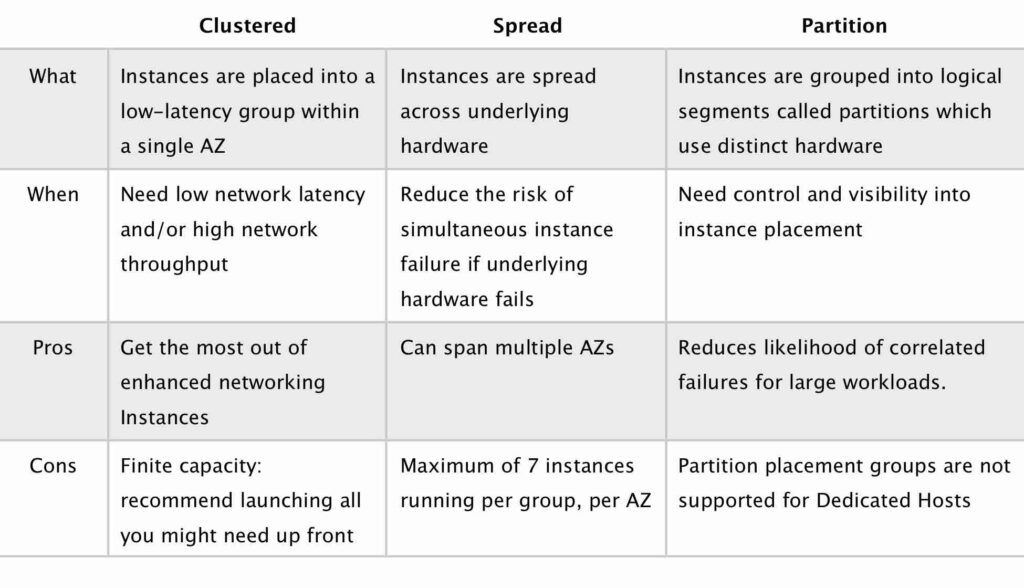
1 Cluster: Packs instances close to each other in an Availability Zone.

2 Partition: Spreads instances in logical partition such that group of instances in one partition do not share underlying hardware. Say 9 ec2 then 3 ec2 in 3 partition(rack)

3 Spread: Strictly places group of instances across distinct hardware to reduce failures. All ec2 in diff racks.

**Rules**

* The **name you specify for a placement group must be unique** within your AWS account for the region.
* You **can’t merge placement groups.**
* An instance can be launched in one placement group at a time; it cannot span multiple placement groups.
* Instances with a tenancy of host cannot be launched in placement groups.



Enhanced Networking –Elastic Network Adapter (ENA)

It provides higher bandwidth, higher packet per second (PPS) performance, and consistent lower inter-instance latencies, which is being **used in Placement Groups**. It uses single root I/O virtualization (SR-IOV) to provide high-performance networking capabilities. SR-IOV is a method of device virtualization that provides higher I/O performance and lower CPU utilization when compared to traditional virtualized network interfaces.

Elastic Fabric Adapter (EFA) –

This is a **network devic**e that you can attach to your EC2 instance to significantly accelerate machine learning applications and High Performance Computing (HPC). It empowers your computing resources to achieve the application performance of an on-premises HPC cluster, with the elasticity and scalability provided by AWS. **Compared with a TCP transpor**t that is traditionally used in cloud-based HPC systems, **EFA provides lower and more consistent latency and higher throughput** as it enhances the performance of inter-instance communication.

ENI vs ENA vs EFA

**When to use ENI (Elastic Network Interface)**

* This is the basic adapter type for when you don’t have any high performance requirements.
* Can use with all instance types.

**When to use ENA:**

* Good for use cases that require higher bandwidth and lower inter-instance latency.
* Supported for limited instance types (HVM only).

**When to use EFA:**

* High Performance Computing.
* MPI and ML use cases.
* Tightly coupled applications.
* Can use with all instance types

**Instance Metadata and User Data**

* **Instance metadata** is data about your instance that you can use to configure or manage the running instance.
* Instance metadata and user data **are not protected by cryptographic methods.**
* View all categories of **instance metadata** from within a running instance at [**http://169.254.169.254/latest/meta-data/**](http://169.254.169.254/latest/meta-data/)
* You can pass two types of user data to EC2: shell scripts and cloud-init directives.
* **User data is limited to 16 KB.**
* If you stop an instance, modify its user data, and start the instance, the updated user data is not executed when you start the instance.
* Retrieve **user data** from within a running instance at [**http://169.254.169.254/latest/user-data**](http://169.254.169.254/latest/user-data)

Parallel Cluster

* An AWS-supported open source cluster management tool for deploying and managing High Performance Computing (HPC) clusters on AWS. **ParallelCluster uses a simple text file to model and provision all the resources** needed for your HPC applications in an automated and secure manner.
* AWS ParallelCluster provisions a master instance for build and control, a cluster of compute instances, a shared filesystem, and a batch scheduler. You can also extend and customize your use cases using custom pre-install and post-install bootstrap actions.

AWS high performance computing services

* Amazon EC2
* Elastic Fabric Adapter
* AWS ParallelCluster
* Amazon FSx for Lustre
* AWS Batch

Scale hundreds of thousands of computing jobs across all AWS compute services and features with AWS Batch, a cloud-native batch scheduler.

* NICE DCV

Deliver a high performance remote desktop and 3D application graphics with NICE DCV, a bandwidth-efficient and high performance streaming protocol.

RTO and RPO:

* **RTO:** recovery time objective, how long it will take to recover from a disaster
* **RPO:** Recovery Point Objective: What is last point of recovery before backup, like when was your last backup before a disaster occur. If RPO is 5 hrs then u have to take backups every 5 hrs. It means that losing 5 hrs of data is acceptable

RTO vs RPO

RTO is a more broader scope and covers the whole business and systems involved while

RPO is more directly related to the interval of backup to take to avoid data loss.

Elastic Load Balancers (ELB)

* Elastic Load Balancer is fully managed service allows us to distribute traffic to multiple EC2 instances thus avoiding a single point of failure.
* Components:
  + A load balancer serves as the single point of contact for clients.
  + A listener checks for connection requests from clients. You must define a default rule for each listener that specifies a target group, condition, and priority.
    - Each **ALB** needs at least one listener and can have up to 10.
    - Listeners define the port and protocol to listen on.
    - Cannot have the same port in multiple listeners.
  + Target group routes requests to one or more registered targets. You can register a target with multiple target groups, and configure health checks on a per target group basis.
    - **Target groups are a regional construct.**
    - **A target group can only be associated with one load balancer**.
    - The following attributes can be defined:
      * **Deregistration delay** – the amount of time for Elastic Load Balancing to wait before deregistering a target.
      * **Slow start duration** – the time period, in seconds, during which the load balancer sends a newly registered target a linearly increasing share of the traffic to the target group.
      * **Stickiness** – indicates whether sticky sessions are enabled.
    - **Auto Scaling groups can scale each target group individually**.
    - You can only use Auto Scaling with the load balancer if **using instance IDs** in your target group.
    - Health checks are defined **per target group.**
    - ALB can route to multiple target groups.
    - You can register the **same EC2 instance or IP address with the same target group multiple times using different ports** (used for routing requests to micro-services).
    - If you **register by instance ID** the traffic is routed using the primary private IP address of **the primary network interface.**
    - If you **register by IP address** you can route traffic to an instance using any private address **from one or more network interfaces.**
    - You cannot mix different types within a target group (EC2, ECS, IP).

AWS currently offers 3 major types of Load Balancers:

1. Classic Load Balancers.

* support back-end server authentication
* support EC2-Classic
* CLB’s do not have pre-defined IPv4 addresses but are resolved using a DNS name.
* Does not support Elastic IPs.
* Supports IPv4 and IPv6.
* Within a VPC only IPv4 is supported.
* Provides SSL termination and processing.
* Support Sticky Sessions
* Cross zone load balancing enable by default, when created through console and not by CLI or API.

2. Network Load Balancers.

Network Load Balancer supports features including:

* WebSockets.
* TLS termination.
* Does not support sticky sessions
* Preserves the source IP of the clients.
* Provides stable IP support and Zonal isolation.
* Long-running connections that are very useful for WebSocket type applications.
* High throughput – designed to handle traffic as it grows and can load balance millions of requests/second.
* Extremely low latencies for latency-sensitive applications.
* **Cross-zone load balancing is disabled by default.**
* Offers multi-protocol listeners, allowing you to run applications such as DNS that rely on both TCP and UDP protocols on the same port behind a Network Load Balancer.
* **You CANNOT** enable or disable Availability Zones for a Network Load Balancer after you create it.

3. Application Load Balancers

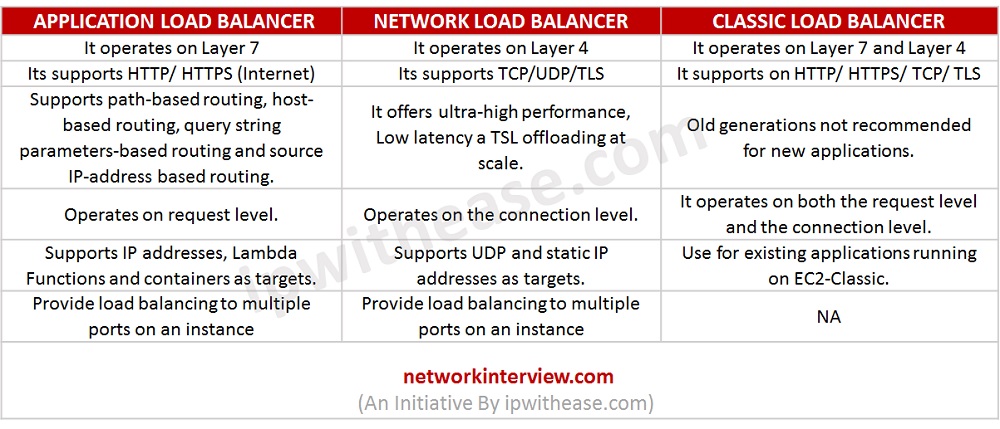
* Need at least 2 availability zones and you can distribute incoming traffic across your targets in multiple Availability Zones.
* **Cross-zone load balancing** is always enabled and cannot be disabled.
* **Automatically scales its request handling capacity** in response to incoming application traffic.
* Can configure an Application Load Balancer to be Internet facing or create a load balancer without public IP addresses to serve as an internal (non-Internet-facing) load balancer.
* Native IPv6 support.
* Internal only ALB only supports IPv4.
* Support sticky session: Session stickiness uses cookies and ensures a client is bound to an individual back-end instance for the duration of the cookie lifetime.

**Gateway Load Balancer ( GWLB )**

* Primarily used for deploying, scaling, and running third-party virtual appliances.
* The virtual appliances can be your custom firewalls, deep packet inspection systems, or intrusion detection and prevention systems in AWS
* Uses the Internet Protocol (IP) to pass the OSI Layer 3 traffic to its registered targets.

### Pricing

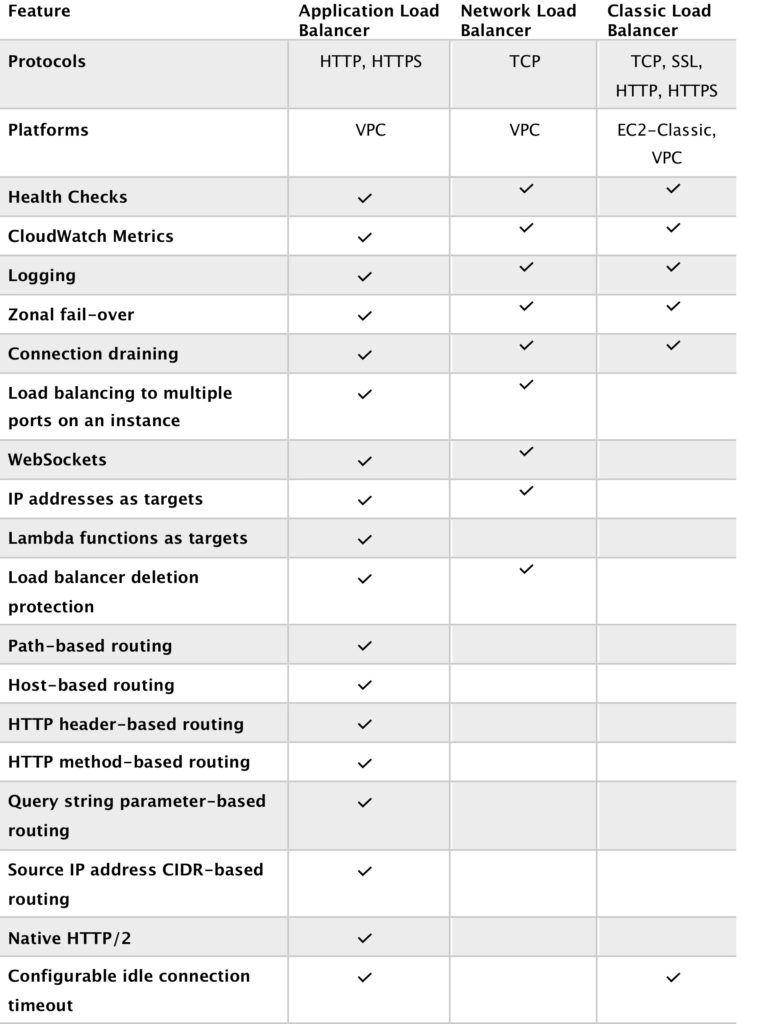
* You are charged for each hour or partial hour that an AL/NL/CL Balancer is running and the number of Load Balancer Capacity Units (LCU) used per hour.

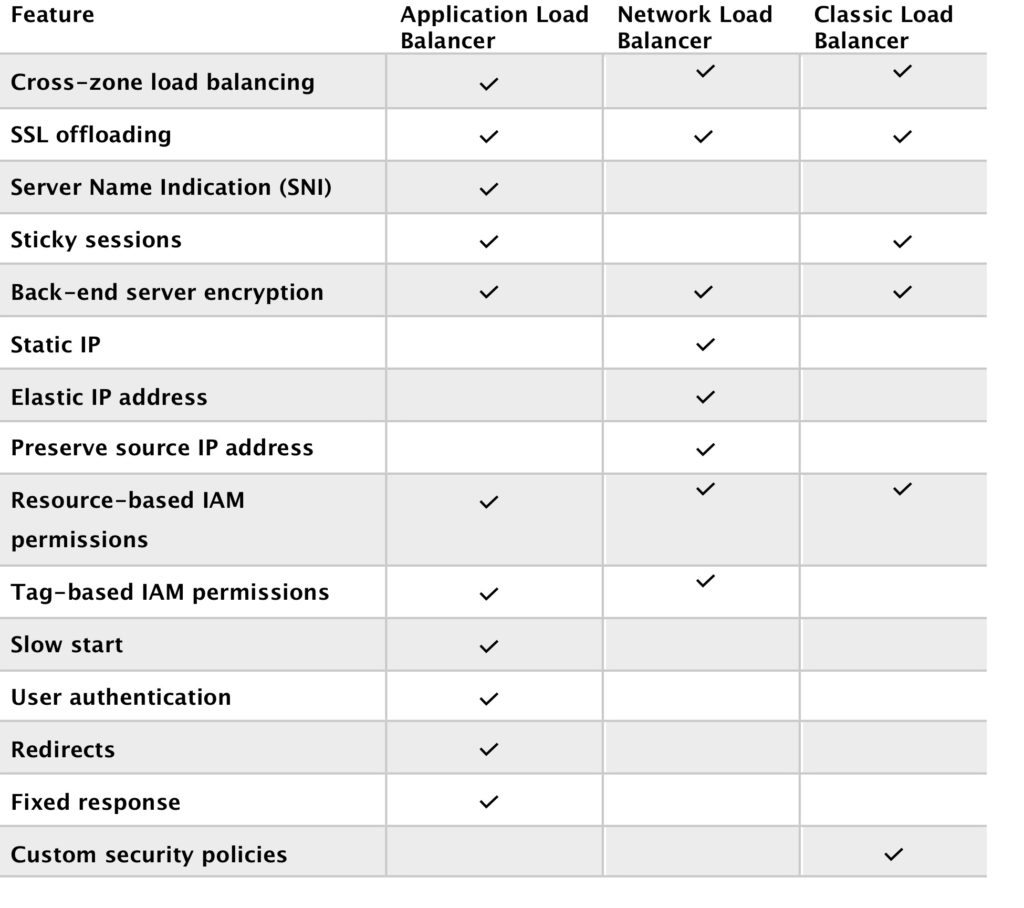


* Only 1 subnet per AZ can be enabled for each ELB.
* Route 53 can be used for region load balancing with ELB instances configured in each region.

ELBs can be Internet facing or internal-only.

* Internet facing ELB:
* ELB nodes have public IPs.
* Routes traffic to the private IP addresses of the EC2 instances.
* **Need one public subnet in each AZ where the ELB is defined.**
* ELB DNS name format: <name>-<id-number>.<region>.elb.amazonaws.com.
* Internal only ELB:
* ELB nodes have private IPs.
* Routes traffic to the private IP addresses of the EC2 instances.
* ELB DNS name format: internal-<name>-<id-number>.<region>.elb.amazonaws.com.
* Internal-only load balancers **do not need an Internet gateway.**
* EC2 instances and containers can be registered against an ELB.
* ELB nodes use IP addresses within your subnets, ensure at least a /27 subnet and make sure there are at least 8 IP addresses available in order for the ELB to scale.
* An ELB forwards traffic to eth0 (primary IP address).
* For ALB at least 2 subnets must be specified.
* For NLB only one subnet must be specified (recommended to add at least 2).
* For CLB you don’t need to specify any subnets unless you have “Enable advanced VPC configuration” enabled in which case you must specify two.
* **ELB uses a DNS record TTL of 60 seconds to ensure new ELB node IP addresses are used to service clients.**
* **By default the ELB has an idle connection timeout of 60 seconds, set the idle timeout for applications to at least 60 seconds.**
* **Perfect Forward Secrecy (PFS)** provides additional safeguards against the eavesdropping of encrypted data, through the use of a unique random session key.
* **Server Order Preference** lets you configure the load balancer to enforce cipher ordering, providing more control over the level of security used by clients to connect with your load balancer.
* ELB does not support client certificate authentication (API Gateway does support this).





ELB Health Checks

ELB will continuously poll the server to check if the app is running

If it gets 200 response from the server, it will consider server UP, otherwise it will declare instance unhealthy and stop sending it requests.

ELB - Nodes & Availability Zones

* When you enable an Availability Zone for your load balancer, Elastic Load Balancing creates a

load balancer node in the Availability Zone.

* If you register targets in an Availability Zone but do not enable the Availability Zone, these registered targets do not receive traffic.
* Your load balancer is most effective when you ensure that each enabled Availability Zone has at

least one registered target.

* It is recommended that you enable multiple Availability Zones. This configuration helps ensure

that the load balancer can continue to route traffic.

Cross Zone Load Balancing

* With cross-zone load balancing, each load balancer node for your Classic Load Balancer distributes requests evenly across the registered instances in all enabled Availability Zones.
* If cross-zone load balancing is disabled, each load balancer node distributes requests evenly across the registered instances in its Availability Zone only & if Az fails system goes down

## **Bastion/Jump Hosts**

* You can configure EC2 instances as bastion hosts (aka jump boxes) in order to access your VPC instances for management.
* Can **use the SSH or RDP protocols** to connect to your bastion host.
* Need to configure a security group with the relevant permissions.
* Can use auto-assigned public IPs or Elastic IPs.
* Can use security groups to restrict the IP addresses/CIDRs that can access the bastion host.
* Use auto-scaling groups for HA (set to 1 instance to just replace if it fails).
* Best practice is to deploy Linux bastion hosts in two AZs, use auto-scaling and Elastic IP addresses.

## **EC2 Migration**

VM Import/Export

is a tool for migrating VMware, Microsoft, XEN VMs to the Cloud.

Can also be used to convert EC2 instances to VMware, Microsoft or XEN VMs.

Supported for:

* Windows and Linux.
* VMware ESX VMDKs and (OVA images for export only).
* Citrix XEN VHD.
* Microsoft Hyper-V VHD.

Can only be used via the API or CLI (not the console).

Stop the VM before generating VMDK or VHD images.

AWS has a VM connector plugin for vCenter:

* Allows migration of VMs to S3.
* Then converts into a EC2 AMI.
* Progress can be tracked in vCenter.

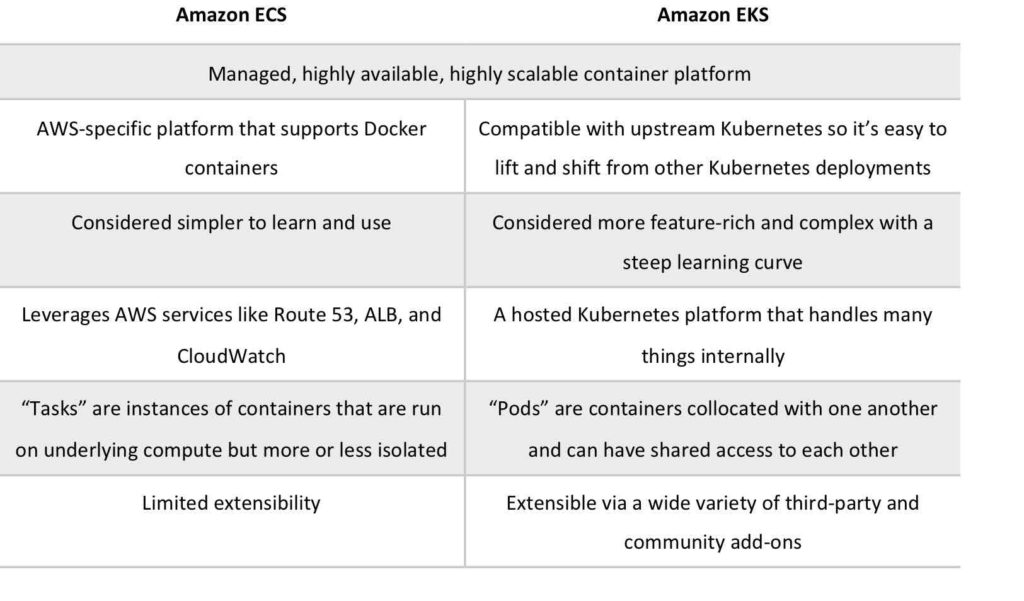
AWS Server Migration Service (SMS)

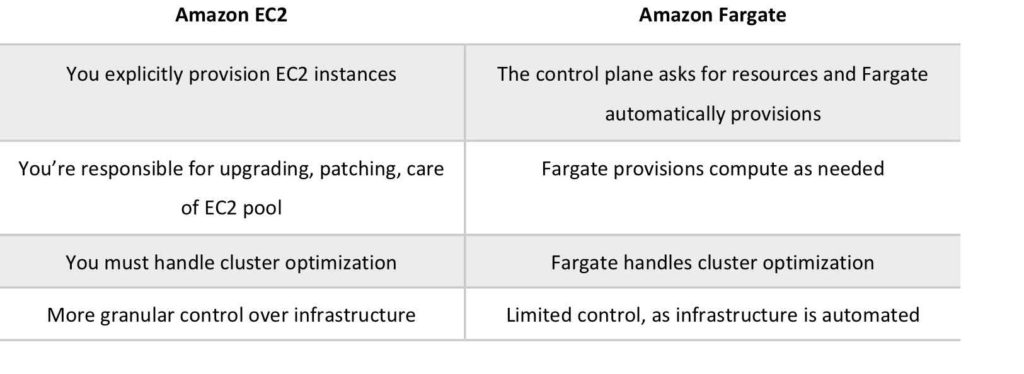
is an agent-less service which makes it easier and faster for you to migrate thousands of on-premises workloads to AWS.

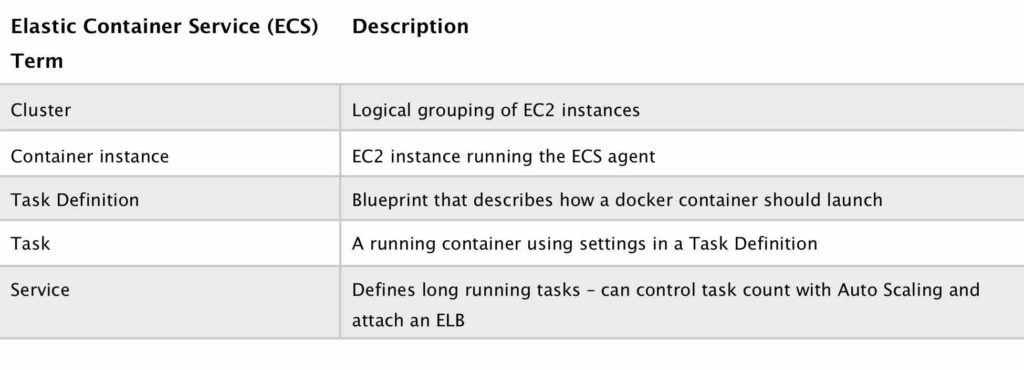
* AWS SMS allows you to automate, schedule, and track incremental replications of live server volumes, making it easier for you to coordinate large-scale server migrations.
* Automates migration of on-premises VMware vSphere or Microsoft Hyper-V/SCVMM virtual machines to AWS.
* Replicates VMs to AWS, syncing volumes and creating periodic AMIs.
* Minimizes cutover downtime by syncing VMs incrementally.
* **Supports Windows and Linux VMs only (just like AWS).**
* The Server Migration Connector is downloaded as a virtual appliance into your on-premises vSphere or Hyper-V environments.

Elastic Container Service (ECS)

* Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.
* Using API calls you can launch and stop container-enabled applications, query the complete state of clusters, and access many familiar features like security groups, Elastic Load Balancing, EBS volumes and IAM roles.
* ECS provides Blox, a collection of open source projects for container management and orchestration. Blox makes it easy to consume events from Amazon ECS, store the cluster state locally and query the local data store through APIs.







* **ecs task defination**are not deployed across all the ECS instances. Task are used for short running things. If task definition fails, ecs won't bring it up

VS

* For long running purpose we have to create **a ECS Servic**e , which runs across all the underlying instances. If ecs Service fails, Ecs will automatically bring it up.

## **Pricing**

EC2 Launch Type:

* No additional charge – you pay for the EC2 resources you launch including instances, EBS volumes and load balancers

Fargate:

* You pay for the vCPU and memory allocated to the containers you run.

AWS Lambda

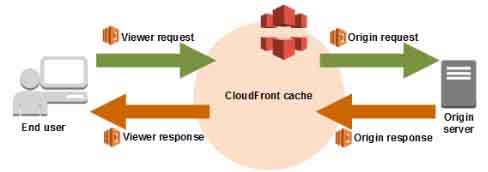
* A serverless compute service.
* Lambda executes your code only when needed and scales automatically.
* Lambda functions are stateless – no affinity to the underlying infrastructure.
* You choose the amount of memory you want to allocate to your functions and AWS Lambda allocates proportional CPU power, network bandwidth, and disk I/O.
* AWS Lambda is SOC, HIPAA, PCI, ISO compliant.
* For Lambda 100ms is the minimum duration u have to pay for. Say if your code ran only for 30 millisecond, but you will be charged for 100ms.
* A function **can use up to 5 layers at a time**. The **total unzipped size of the function and all layers can’t exceed** the unzipped deployment package size **limit of 250 MB.**
* Functions can access:
  + AWS services or non-AWS services.
  + AWS services running in VPCs (e.g. RedShift, Elasticache, RDS instances).
  + Non-AWS services running on EC2 instances in an AWS VPC.
* **To enable your Lambda function to access resources inside your private VPC, you must provide additional VPC-specific configuration information that includes VPC subnet IDs and security group IDs**.
* AWS Lambda uses this information to set up elastic network interfaces (ENIs) that enable your function.
* AWS Lambda supports code written in Node.js (JavaScript), Python, Java (Java 8 compatible), C# (.NET Core), Ruby, Go and PowerShell.
* AWS **Lambda stores code in Amazon S3** and encrypts it at rest.
* Continuous scaling – **scales out not up**.
* Lambda scales concurrently executing functions up to your default limit (1000).
* For non stream-based event sources each published event **is a unit of work, run in parallel up to your account limit (one Lambda function per event).**
* For stream-based event sources the **number of shards indicates the unit of concurrency (one function per shard).**

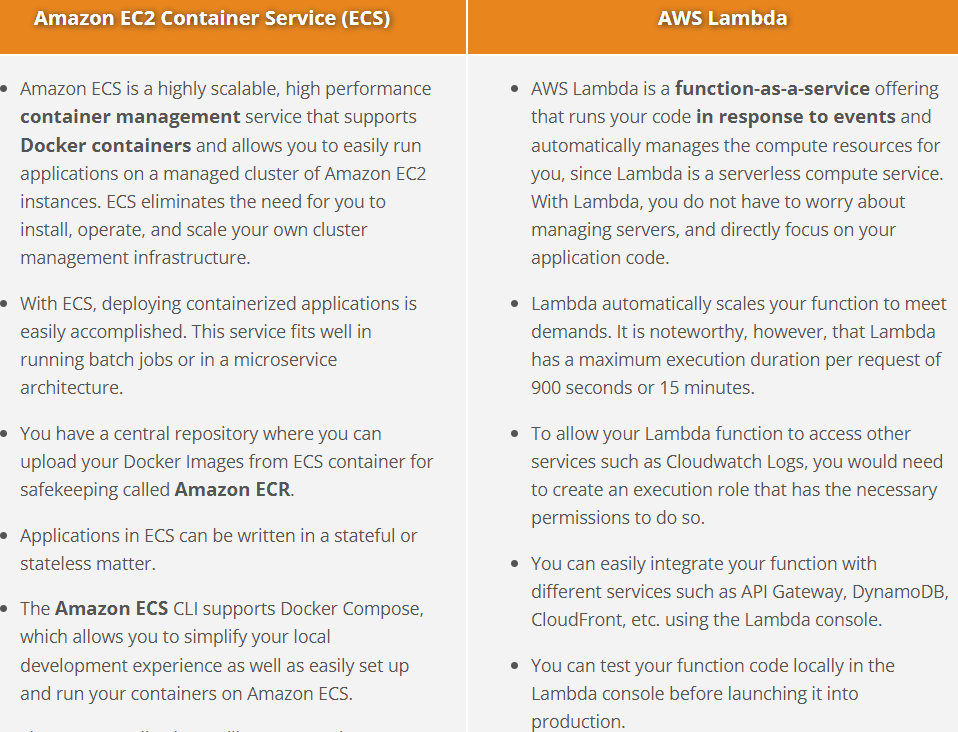
Lambda works globally.

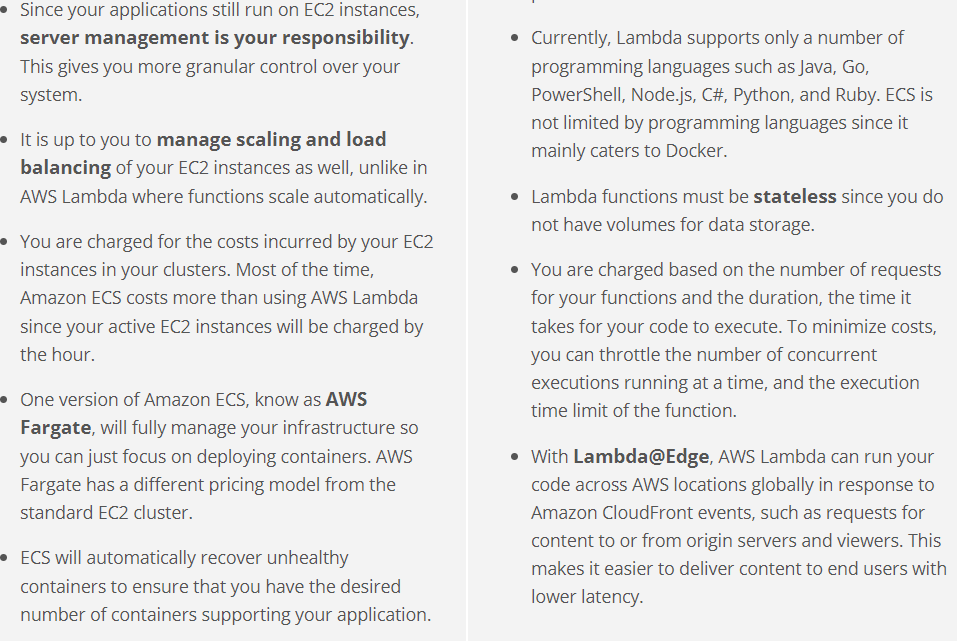
* **Event source mapping** maps an event source to a Lambda function. It enables automatic invocation of your Lambda function when events occur.
* Lambda provides event source mappings for the following services.
  + Amazon Kinesis
  + Amazon DynamoDB
  + Amazon Simple Queue Service
* Your functions’ concurrency **is the number of instances that serve requests at a given time**. When your function is invoked, Lambda allocates an instance of it to process the event. When the function code finishes running, it can handle another request. If the function is invoked again while a request is still being processed, another instance is allocated, which increases the function’s concurrency.
* reserved concurrency: When a function has reserved concurrency, **no other function can use that concurrency**. Reserved concurrency also limits the maximum concurrency for the function.
* provisioned concurrency: By allocating provisioned concurrency before an increase in invocations, you can **ensure that all requests are served by initialized instances** with very low latency.
* Priced based on:
  + Number of requests. First 1 million are free then $0.20 per 1 million.
  + Duration. Calculated from the time your code begins execution until it returns or terminates. Depends on the amount of memory allocated to a function.

### Lambda@Edge

* Lets you **run Lambda functions to customize content that CloudFront delivers, executing the functions in AWS locations closer to the viewer**. The functions run in response to CloudFront events, without provisioning or managing servers.
* You can use Lambda functions to change CloudFront requests and responses at the following points:
  + After CloudFront receives a request from a viewer (viewer request)
  + Before CloudFront forwards the request to the origin (origin request)
  + After CloudFront receives the response from the origin (origin response)
  + Before CloudFront forwards the response to the viewer (viewer response)







|  |  |  |
| --- | --- | --- |
| **Resource** | **Type** | **Description** |
|  |  |  |
| AWS account | Global | You can use the same AWS account in all regions. |
| Key pairs | Global or Regional | The key pairs that you create using EC2 are tied to the region where you created them. You can create your own RSA key pair and upload it to the region in which you want to use it; therefore, you can make your key pair globally available by uploading it to each region. |
| Amazon EC2 resource identifiers | Regional | Each resource identifier, such as an AMI ID, instance ID, EBS volume ID, or EBS snapshot ID, is tied to its region and can be used only in the region where you created the resource. |
| User-supplied resource names | Regional | Each resource name, such as a security group name or key pair name, is tied to its region and can be used only in the region where you created the resource. Although you can create resources with the same name in multiple regions, they aren’t related to each other. |
| AMIs | Regional | An AMI is tied to the region where its files are located within S3. You can copy an AMI from one region to another. |
| Elastic IP addresses | Regional | An Elastic IP address is tied to a region and can be associated only with an instance in the same region. |
| Security groups | Regional | A security group is tied to a region and can be assigned only to instances in the same region. You can’t enable an instance to communicate with an instance outside its region using security group rules. |
| EBS snapshots | Regional | An EBS snapshot is tied to its region and can only be used to create volumes in the same region. You can copy a snapshot from one region to another. |
| EBS volumes | Availability Zone | An EBS volume is tied to its Availability Zone and can be attached only to instances in the same Availability Zone. |
| Instances | Availability Zone | An instance is tied to the Availability Zones in which you launched it. However, its instance ID is tied to the region. |

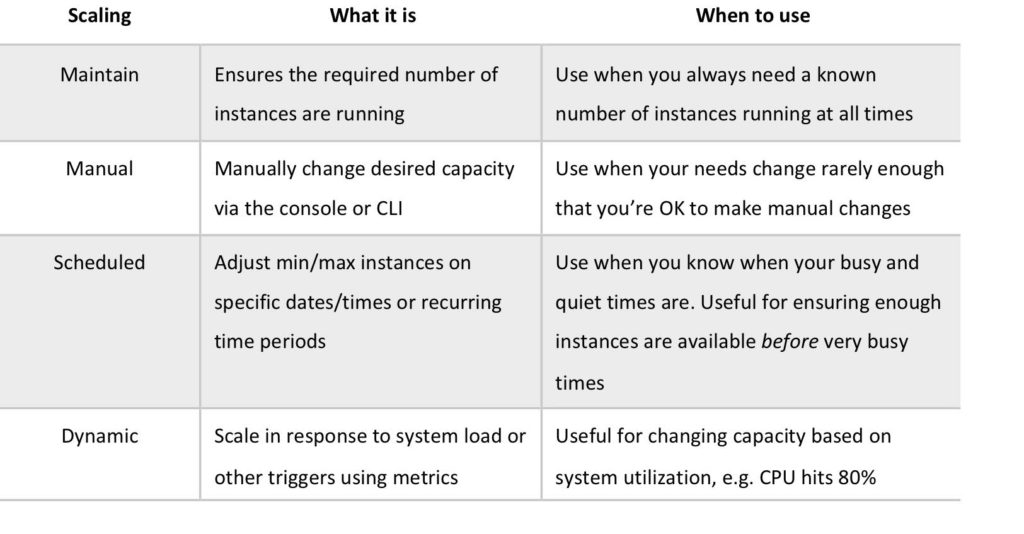
# AWS Elastic Beanstalk

* AWS Elastic Beanstalk can be used to quickly deploy and manage applications in the AWS Cloud.
* Developers upload applications and **Elastic Beanstalk handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.**
* AWS Elastic Beanstalk leverages Elastic Load Balancing and Auto Scaling to automatically scale your application in and out based on your application’s specific needs.
* In addition, multiple availability zones give you an option to improve application reliability and availability by running in more than one zone.
* Considered a **Platform as a Service (PaaS) solution**.
* Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications.

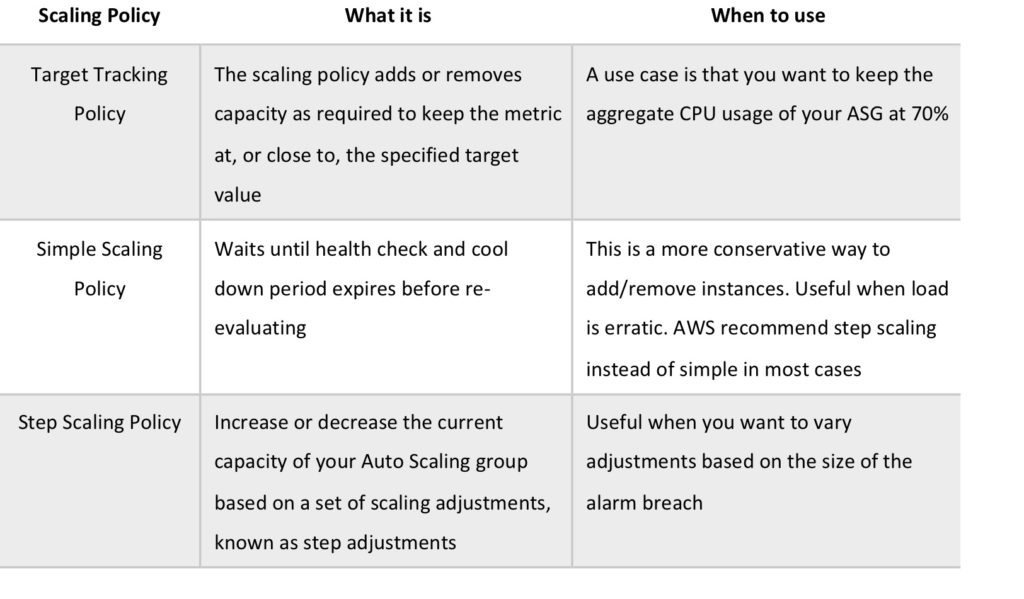
## **Auto Scaling**

* Configure automatic scaling for the AWS resources quickly through a scaling plan that uses **dynamic scaling** and **predictive scaling**.
* Optimize for availability, for cost, or a balance of both.
* Scaling in means decreasing the size of a group while scaling out means increasing the size of a group.
* Auto Scaling is a region specific service.
* Auto Scaling can span multiple AZs within the same AWS region
* Auto Scaling can be configured from the Console, CLI, SDKs and APIs.
* There is no additional cost for Auto Scaling, you just pay for the resources (EC2 instances) provisioned.
* Auto Scaling **works with ELB, CloudWatch and CloudTrail.**

There are four scaling options:



scaling policy types



**Scheduled scaling** —Scale a resource based on the date and time. The timezone can either be in UTC or in your local timezone.

The *scale out cooldown period* is the amount of time, in seconds, after a scale out activity completes before another scale out activity can start.

The *scale in cooldown period* is the amount of time, in seconds, after a scale in activity completes before another scale in activity can start.

Custom Termination Policies

* + *OldestInstance* – Terminate the oldest instance in the group.
  + *NewestInstance* – Terminate the newest instance in the group.
  + *OldestLaunchConfiguration* – Terminate instances that have the oldest launch configuration.
  + *ClosestToNextInstanceHour* – Terminate instances that are closest to the next billing hour.

You can only specify one launch configuration for an Auto Scaling group at a time, and you can’t modify a launch configuration after you’ve created it.

When you create a VPC, by default its tenancy attribute is set to *default*. You can launch instances with a tenancy value of *dedicated* so that they run as single-tenancy instances. Otherwise, they run as shared-tenancy instances by default.

**Amazon EC2-related Cheat Sheets:**

* [EC2 Instance Health Check vs ELB Health Check vs Auto Scaling and Custom Health Check](https://tutorialsdojo.com/ec2-instance-health-check-vs-elb-health-check-vs-auto-scaling-and-custom-health-check/)
* [EC2 Container Service (ECS) vs Lambda](https://tutorialsdojo.com/ec2-container-service-ecs-vs-lambda/)