**Compute**

**Amazon Elastic Container Service**

Q: What is Amazon Elastic Container Service?

Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure. With simple API calls, you can launch and stop container-enabled applications, query the complete state of your cluster, and access many familiar features like security groups, Elastic Load Balancing, EBS volumes and IAM roles. You can use Amazon ECS to schedule the placement of containers across your cluster based on your resource needs and availability requirements. You can also integrate your own scheduler or third-party schedulers to meet business or application specific requirements.

Q: Why should I use Amazon ECS?

Amazon ECS makes it easy to use containers as a building block for your applications by eliminating the need for you to install, operate, and scale your own cluster management infrastructure. Amazon ECS lets you schedule long-running applications, services, and batch processes using Docker containers. Amazon ECS maintains application availability and allows you to scale your containers up or down to meet your application's capacity requirements. Amazon ECS is integrated with familiar features like Elastic Load Balancing, EBS volumes, VPC, and IAM. Simple APIs let you integrate and use your own schedulers or connect Amazon ECS into your existing software delivery process.

Q: How is Amazon ECS different from AWS Elastic Beanstalk?

AWS Elastic Beanstalk is an application management platform that helps customers easily deploy and scale web applications and services. It keeps the provisioning of building blocks (e.g., EC2, RDS, Elastic Load Balancing, Auto Scaling, CloudWatch), deployment of applications, and health monitoring abstracted from the user so they can just focus on writing code. You simply specify which container images are to be deployed, the CPU and memory requirements, the port mappings, and the container links.

Elastic Beanstalk will automatically handle all the details such as provisioning an Amazon ECS cluster, balancing load, auto-scaling, monitoring, and placing your containers across your cluster. Elastic Beanstalk is ideal if you want to leverage the benefits of containers but just want the simplicity of deploying applications from development to production by uploading a container image. You can work with Amazon ECS directly if you want more fine-grained control for custom application architectures.

Q: How is Amazon ECS different from AWS Lambda?

Amazon ECS is a highly scalable Docker container management service that allows you to run and manage distributed applications that run in Docker containers. AWS Lambda is an event-driven task compute service that runs your code in response to “events” such as changes in data, website clicks, or messages from other AWS services without you having to manage any compute infrastructure.

**Amazon EC2**

Q: What is Amazon Elastic Compute Cloud (Amazon EC2)?

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in [the cloud](https://aws.amazon.com/what-is-cloud-computing/). It is designed to make web-scale computing easier for developers.

Q: What can I do with Amazon EC2?

Just as Amazon Simple Storage Service (Amazon S3) enables storage in the cloud, Amazon EC2 enables “compute” in the cloud. Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.

Q: What can developers now do that they could not before?

Until now, small developers did not have the capital to acquire massive compute resources and ensure they had the capacity they needed to handle unexpected spikes in load. Amazon EC2 enables any developer to leverage Amazon’s own benefits of massive scale with no up-front investment or performance compromises. Developers are now free to innovate knowing that no matter how successful their businesses become, it will be inexpensive and simple to ensure they have the compute capacity they need to meet their business requirements.

The “Elastic” nature of the service allows developers to instantly scale to meet spikes in traffic or demand. When computing requirements unexpectedly change (up or down), Amazon EC2 can instantly respond, meaning that developers have the ability to control how many resources are in use at any given point in time. In contrast, traditional hosting services generally provide a fixed number of resources for a fixed amount of time, meaning that users have a limited ability to easily respond when their usage is rapidly changing, unpredictable, or is known to experience large peaks at various intervals.

Q: What is the difference between using the local instance store and Amazon Elastic Block Store (Amazon EBS) for the root device?

When you launch your Amazon EC2 instances you have the ability to store your root device data on Amazon EBS or the local instance store. By using Amazon EBS, data on the root device will persist independently from the lifetime of the instance. This enables you to stop and restart the instance at a subsequent time, which is similar to shutting down your laptop and restarting it when you need it again.

Alternatively, the local instance store only persists during the life of the instance. This is an inexpensive way to launch instances where data is not stored to the root device. For example, some customers use this option to run large web sites where each instance is a clone to handle web traffic.

Q: How quickly will systems be running?

It typically takes less than 10 minutes from the issue of the RunInstances call to the point where all requested instances begin their boot sequences. This time depends on a number of factors including: the size of your AMI, the number of instances you are launching, and how recently you have launched that AMI. Images launched for the first time may take slightly longer to boot.

Q: How do I load and store my systems with Amazon EC2?

Amazon EC2 allows you to set up and configure everything about your instances from your operating system up to your applications. An Amazon Machine Image (AMI) is simply a packaged-up environment that includes all the necessary bits to set up and boot your instance. Your AMIs are your unit of deployment. You might have just one AMI or you might compose your system out of several building block AMIs (e.g., webservers, appservers, and databases). Amazon EC2 provides a number of tools to make creating an AMI easy. Once you create a custom AMI, you will need to bundle it. If you are bundling an image with a root device backed by Amazon EBS, you can simply use the bundle command in the AWS Management Console. If you are bundling an image with a boot partition on the instance store, then you will need to use the AMI Tools to upload it to Amazon S3. Amazon EC2 uses Amazon EBS and Amazon S3 to provide reliable, scalable storage of your AMIs so that we can boot them when you ask us to do so.

Or, if you want, you don’t have to set up your own AMI from scratch. You can choose from a number of globally available AMIs that provide useful instances. For example, if you just want a simple Linux server, you can choose one of the standard Linux distribution AMIs.

Q: Is Amazon EC2 used in conjunction with Amazon S3?

Yes, Amazon EC2 is used jointly with Amazon S3 for instances with root devices backed by local instance storage. By using Amazon S3, developers have access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. In order to execute systems in the Amazon EC2 environment, developers use the tools provided to load their AMIs into Amazon S3 and to move them between Amazon S3 and Amazon EC2.

We expect developers to find the combination of Amazon EC2 and Amazon S3 to be very useful. Amazon EC2 provides cheap, scalable compute in the cloud while Amazon S3 allows users to store their data reliably.

Q: How is this service different than a plain hosting service?

Traditional hosting services generally provide a pre-configured resource for a fixed amount of time and at a predetermined cost. Amazon EC2 differs fundamentally in the flexibility, control and significant cost savings it offers developers, allowing them to treat Amazon EC2 as their own personal data center with the benefit of Amazon.com’s robust infrastructure.

When computing requirements unexpectedly change (up or down), Amazon EC2 can instantly respond, meaning that developers have the ability to control how many resources are in use at any given point in time. In contrast, traditional hosting services generally provide a fixed number of resources for a fixed amount of time, meaning that users have a limited ability to easily respond when their usage is rapidly changing, unpredictable, or is known to experience large peaks at various intervals.

Secondly, many hosting services don’t provide full control over the compute resources being provided. Using Amazon EC2, developers can choose not only to initiate or shut down instances at any time, they can completely customize the configuration of their instances to suit their needs – and change it at any time. Most hosting services cater more towards groups of users with similar system requirements, and so offer limited ability to change these.

Finally, with Amazon EC2 developers enjoy the benefit of paying only for their actual resource consumption – and at very low rates. Most hosting services require users to pay a fixed, up-front fee irrespective of their actual computing power used, and so users risk overbuying resources to compensate for the inability to quickly scale up resources within a short time frame.

**Instance Types**

**Accelerated Computing instances**

Q: What are Accelerated Computing instances?

Accelerated Computing instance family is a family of instances which use hardware accelerators, or co-processors, to perform some functions, such as floating-point number calculation and graphics processing, more efficiently than is possible in software running on CPUs. Amazon EC2 provides three types of Accelerated Computing instances – GPU compute instances for general-purpose computing, GPU graphics instances for graphics intensive applications, and FPGA programmable hardware compute instances for advanced scientific workloads.

Q. When should I use GPU Graphics and Compute instances?

GPU instances work best for applications with massive parallelism such as workloads using thousands of threads. Graphics processing is an example with huge computational requirements, where each of the tasks is relatively small, the set of operations performed form a pipeline, and the throughput of this pipeline is more important than the latency of the individual operations. To be able build applications that exploit this level of parallelism, one needs GPU device specific knowledge by understanding how to program against various graphics APIs (DirectX, OpenGL) or GPU compute programming models (CUDA, OpenCL).

Q: How are P3 instances different from G3 instances?

P3 instances are the next-generation of EC2 general-purpose GPU computing instances, powered by up to 8 of the latest-generation NVIDIA Tesla V100 GPUs. These new instances significantly improve performance and scalability, and add many new features, including new Streaming Multiprocessor (SM) architecture for machine learning (ML)/deep learning (DL) performance optimization, second-generation NVIDIA NVLink high-speed GPU interconnect, and highly tuned HBM2 memory for higher-efficiency.

G3 instances use NVIDIA Tesla M60 GPUs and provide a high-performance platform for graphics applications using DirectX or OpenGL. NVIDIA Tesla M60 GPUs support NVIDIA GRID Virtual Workstation features, and H.265 (HEVC) hardware encoding. Each M60 GPU in G3 instances supports 4 monitors with resolutions up to 4096x2160, and is licensed to use NVIDIA GRID Virtual Workstation for one Concurrent Connected User. Example applications of G3 instances include 3D visualizations, graphics-intensive remote workstation, 3D rendering, application streaming, video encoding, and other server-side graphics workloads.

Q. What is Amazon EC2 F1?

Amazon EC2 F1 is a compute instance with programmable hardware you can use for application acceleration. The new F1 instance type provides a high performance, easy to access FPGA for developing and deploying custom hardware accelerations.

Q. What are FPGAs and why do I need them?

FPGAs are programmable integrated circuits that you can configure using software. By using FPGAs you can accelerate your applications up to 30x when compared with servers that use CPUs alone. And, FPGAs are reprogrammable, so you get the flexibility to update and optimize your hardware acceleration without having to redesign the hardware.

Q. How does F1 compare with traditional FPGA solutions?

F1 is an AWS instance with programmable hardware for application acceleration. With F1, you have access to FPGA hardware in a few simple clicks, reducing the time and cost of full-cycle FPGA development and scale deployment from months or years to days. While FPGA technology has been available for decades, adoption of application acceleration has struggled to be successful in both the development of accelerators and the business model of selling custom hardware for traditional enterprises, due to time and cost in development infrastructure, hardware design, and at-scale deployment. With this offering, customers avoid the undifferentiated heavy lifting associated with developing FPGAs in on-premises data centers.

**Compute Optimized instances**

Q. When should I use Compute Optimized instances?

Compute Optimized instances are designed for applications that benefit from high compute power. These applications include compute-intensive applications like high-performance web servers, high-performance computing (HPC), scientific modelling, distributed analytics and machine learning inference.

Q. Can I launch C4 instances as Amazon EBS-optimized instances?

Each C4 instance type is EBS-optimized by default. C4 instances 500 Mbps to 4,000 Mbps to EBS above and beyond the general-purpose network throughput provided to the instance. Since this feature is always enabled on C4 instances, launching a C4 instance explicitly as EBS-optimized will not affect the instance's behavior.

**General purpose instances**

Q: What are Amazon EC2 A1 instances?

Amazon EC2 A1 instances are new general purpose instances powered by the AWS Graviton Processors that are custom designed by AWS.

Q: When should I use A1 instances?

A1 instances deliver significant cost savings for customer workloads that are supported by the extensive Arm ecosystem and can fit within the available memory footprint. A1 instances are ideal for scale-out applications such as web servers, containerized microservices, caching fleets, and distributed data stores. These instances will also appeal to developers, enthusiasts, and educators across the Arm developer community. Most applications that make use of open source software like Apache HTTP Server, Perl, PHP, Ruby, Python, NodeJS, and Java easily run on multiple processor architectures due to the support of Linux based operating systems. We encourage customers running such applications to give A1 instances a try.

Applications that require higher compute and network performance, require higher memory, or have dependencies on x86 architecture will be better suited for existing instances like the M5, C5, or R5 instances. Applications with variable CPU usage that experience occasional spikes in demand will get the most cost savings from the burstable performance T3 instances.

Q: What are the key use cases for Amazon EC2 M5 Instances?

M5 instances offer a good choice for running development and test environments, web, mobile and gaming applications, analytics applications, and business critical applications including ERP, HR, CRM, and collaboration apps. Customers who are interested in running their data intensive workloads (e.g. HPC, or SOLR clusters) on instances with a higher memory footprint will also find M5 to be a good fit. Workloads that heavily use single and double precision floating point operations and vector processing such as video processing workloads and need higher memory can benefit substantially from the AVX-512 instructions that M5 supports.

Q: When should I choose a Burstable Performance Instance, such as T2?

T2 instances provide a cost-effective platform for a broad range of general purpose production workloads. T2 Unlimited instances can sustain high CPU performance for as long as required. If your workloads consistently require CPU usage much higher than the baseline, consider a dedicated CPU instance family such as the M or C.

**High Memory instances**

Q. What are EC2 High Memory instances?

Amazon EC2 High Memory instances offer 6 TB, 9 TB, or 12 TB of memory in a single instance. These instances are designed to run large in-memory databases, including production installations of SAP HANA, in the cloud. EC2 High Memory instances are the first Amazon EC2 instances powered by an 8-socket platform with latest generation Intel® Xeon® Platinum 8176M (Skylake) processors that are optimized for mission-critical enterprise workloads. EC2 High Memory instances deliver high networking throughput and low-latency with 25 Gbps of aggregate network bandwidth using Amazon Elastic Network Adapter (ENA)-based Enhanced Networking. EC2 High Memory instances are EBS-Optimized by default, and support encrypted and unencrypted EBS volumes.

**Memory-optimized instances**

Q. When should I use Memory-optimized instances?

Memory-optimized instances offer large memory size for memory intensive applications including in-memory applications, in-memory databases, in-memory analytics solutions, High Performance Computing (HPC), scientific computing, and other memory-intensive applications.

**Storage Optimized Instances**

Q. What is a Dense-storage Instance?

Dense-storage instances are designed for workloads that require high sequential read and write access to very large data sets, such as Hadoop distributed computing, massively parallel processing data warehousing, and log processing applications. The Dense-storage instances offer the best price/GB-storage and price/disk-throughput across other EC2 instances.

Q. How do Dense-storage and HDD-storage instances compare to High I/O instances?

High I/O instances (I2) are targeted at workloads that demand low latency and high random I/O in addition to moderate storage density and provide the best price/IOPS across other EC2 instance types. Dense-storage instances (D2) and HDD-storage instances (H1) are optimized for applications that require high sequential read/write access and low cost storage for very large data sets and provide the best price/GB-storage and price/disk-throughput across other EC2 instances.

Q. When should I use X1 instances?

X1 instances are ideal for running in-memory databases like SAP HANA, big data processing engines like Apache Spark or Presto, and high performance computing (HPC) applications. X1 instances are certified by SAP to run production environments of the next-generation Business Suite S/4HANA, Business Suite on HANA (SoH), Business Warehouse on HANA (BW), and Data Mart Solutions on HANA on the AWS cloud.

Q. How do Dense-storage and HDD-storage instances differ from Amazon EBS?

Amazon EBS offers simple, elastic, reliable (replicated), and persistent block level storage for Amazon EC2 while abstracting the details of the underlying storage media in use. Amazon EC2 instance storage provides directly attached non-persistent, high performance storage building blocks that can be used for a variety of storage applications. Dense-storage instances are specifically targeted at customers who want high sequential read/write access to large data sets on local storage, e.g. for Hadoop distributed computing and massively parallel processing data warehousing.

Q. AWS has other database and Big Data offerings. When or why should I use High I/O instances?

High I/O instances are ideal for applications that require access to millions of low latency IOPS, and can leverage data stores and architectures that manage data redundancy and availability. Example applications are:

* NoSQL databases like Cassandra and MongoDB
* In-memory databases like Aerospike
* Elasticsearch and analytics workloads
* OLTP systems

**Amazon EBS volume**

Q: What kind of performance can I expect from Amazon EBS volumes?

Amazon EBS provides four current generation volume types and are divided into two major categories: SSD-backed storage for transactional workloads and HDD-backed storage for throughput intensive workloads. These volume types differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your applications.

Q: What happens to my data when a system terminates?

The data stored on a local instance store will persist only as long as that instance is alive. However, data that is stored on an Amazon EBS volume will persist independently of the life of the instance. Therefore, we recommend that you use the local instance store for temporary data and, for data requiring a higher level of durability, we recommend using Amazon EBS volumes or backing up the data to Amazon S3. If you are using an Amazon EBS volume as a root partition, you will need to set the Delete On Terminate flag to "N" if you want your Amazon EBS volume to persist outside the life of the instance.

Q: Which volume type should I choose?

Amazon EBS includes two major categories of storage: SSD-backed storage for transactional workloads (performance depends primarily on IOPS) and HDD-backed storage for throughput workloads (performance depends primarily on throughput, measured in MB/s). SSD-backed volumes are designed for transactional, IOPS-intensive database workloads, boot volumes, and workloads that require high IOPS. SSD-backed volumes include Provisioned IOPS SSD (io1) and General Purpose SSD (gp2). HDD-backed volumes are designed for throughput-intensive and big-data workloads, large I/O sizes, and sequential I/O patterns. HDD-backed volumes include Throughput Optimized HDD (st1) and Cold HDD (sc1).

ST1 volumes are backed by hard disk drives (HDDs) and are ideal for frequently accessed, throughput intensive workloads with large datasets and large I/O sizes, such as MapReduce, Kafka, log processing, data warehouse, and ETL workloads.

SC1 volumes are backed by hard disk drives (HDDs) and provides the lowest cost per GB of all EBS volume types. It is ideal for less frequently accessed workloads with large, cold datasets.

Q: Do you support multiple instances accessing a single volume?

While you are able to attach multiple volumes to a single instance, attaching multiple instances to one volume is not supported at this time.

Q: Do volumes need to be un-mounted in order to take a snapshot? Does the snapshot need to complete before the volume can be used again?

No, snapshots can be done in real time while the volume is attached and in use. However, snapshots only capture data that has been written to your Amazon EBS volume, which might exclude any data that has been locally cached by your application or OS. In order to ensure consistent snapshots on volumes attached to an instance, we recommend cleanly detaching the volume, issuing the snapshot command, and then reattaching the volume. For Amazon EBS volumes that serve as root devices, we recommend shutting down the machine to take a clean snapshot.

**Amazon Elastic File System (EFS)**

Q. What is Amazon Elastic File System?

Amazon EFS is a fully-managed service that makes it easy to set up and scale file storage in the Amazon Cloud. With a few clicks in the AWS Management Console, you can create file systems that are accessible to Amazon EC2 instances via a file system interface (using standard operating system file I/O APIs) and supports full file system access semantics (such as strong consistency and file locking).

Amazon EFS file systems can automatically scale from gigabytes to petabytes of data without needing to provision storage. Tens, hundreds, or even thousands of Amazon EC2 instances can access an Amazon EFS file system at the same time, and Amazon EFS provides consistent performance to each Amazon EC2 instance. Amazon EFS is designed to be highly durable and highly available. With Amazon EFS, there is no minimum fee or setup costs, and you pay only for the storage you use.

Q. What use cases does Amazon EFS support?

Amazon EFS is designed to provide performance for a broad spectrum of workloads and applications, including Big Data and analytics, media processing workflows, content management, web serving, and home directories.

Q. When should I use Amazon EFS vs. Amazon S3 vs. Amazon Elastic Block Store (EBS)?

Amazon Web Services (AWS) offers cloud storage services to support a wide range of storage workloads.

Amazon EFS is a [file storage service](https://aws.amazon.com/what-is-cloud-file-storage/) for use with Amazon EC2. Amazon EFS provides a file system interface, file system access semantics (such as strong consistency and file locking), and concurrently-accessible storage for up to thousands of Amazon EC2 instances.

[Amazon EBS](https://aws.amazon.com/ebs/) is a block level storage service for use with Amazon EC2. Amazon EBS can deliver performance for workloads that require the lowest-latency access to data from a single EC2 instance.

[Amazon S3](https://aws.amazon.com/s3/) is an object storage service. Amazon S3 makes data available through an Internet API that can be accessed anywhere.

Q. How do I access a file system from an Amazon EC2 instance?

To access your file system, you mount the file system on an Amazon EC2 Linux-based instance using the standard Linux mount command and the file system’s DNS name. Once mounted, you can work with the files and directories in your file system just like you would with a local file system.

Amazon EFS uses the NFSv4.1 protocol.

Q. How do I load data into a file system?

There are a number of methods for loading existing file system data into Amazon EFS, whether your existing file system data is located in AWS or in your on-premises servers.

Amazon EFS file systems can be mounted on an Amazon EC2 instance, so any data that is accessible to an Amazon EC2 instance can also be read and written to Amazon EFS. To load file data that is not currently stored in AWS, you can use [AWS DataSync](https://aws.amazon.com/datasync/) to copy data directly to Amazon EFS.

For on-premises file systems, DataSync provides a fast and simple way to securely sync existing file systems into Amazon EFS. DataSync works over any network connection, including with [AWS Direct Connect](https://aws.amazon.com/directconnect/) or [AWS VPN](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_VPN.html). AWS Direct Connect provides a high bandwidth and lower latency dedicated network connection, over which you can mount your EFS file systems. You can also use standard Linux copy tools to move data files to Amazon EFS.

Q. How does Amazon EFS performance compare to that of other storage solutions?

Amazon EFS file systems are distributed across an unconstrained number of storage servers, enabling file systems to grow elastically to petabyte-scale and allowing massively parallel access from Amazon EC2 instances to your data. Amazon EFS’s distributed design avoids the bottlenecks and constraints inherent to traditional file servers.

This distributed data storage design means that multi-threaded applications and applications that concurrently access data from multiple Amazon EC2 instances can drive substantial levels of aggregate throughput and IOPS. Big Data and analytics workloads, media processing workflows, content management and web serving are examples of these applications.

Q. How do I control which Amazon EC2 instances can access my file system?

When you create a file system, you create endpoints in your VPC called “mount targets.” When mounting from an EC2 instance, your file system’s DNS name, which you provide in your mount command, resolves to a mount target’s IP address. Only resources that can access a mount target can access your file system. You can control the network traffic to and from your file system mount targets using VPC [security groups](https://docs.aws.amazon.com/efs/latest/ug/security-considerations.html#network-access).

Q: Can I access my Amazon EFS file system concurrently from my on-premises datacenter servers as well as Amazon EC2 instances?

Yes, you can access your Amazon EFS file system concurrently from servers in your on-premises datacenter as well as Amazon EC2 instances in your Amazon VPC. Amazon EFS provides the same file system access semantics, such as strong data consistency and file locking, across all EC2 instances and on-premises servers accessing a file system.

Q: How do I copy existing data from on-premises file storage to Amazon EFS?

There are a number of methods to copy existing on-premises data into Amazon EFS. AWS DataSync provides a fast and simple way to securely sync existing file systems into Amazon EFS, and works over any network, including AWS Direct Connect

[AWS Direct Connect](https://aws.amazon.com/directconnect/) provides a high bandwidth and lower latency dedicated network connection over which you can mount your Amazon EFS file systems. Once mounted, you can use DataSync to copy data into Amazon EFS up to 5x faster than standard Linux copy tools.

Q. What is AWS DataSync?

AWS DataSync is an online data transfer service that makes it faster and simpler to move data between on-premises storage and Amazon EFS. DataSync uses a purpose-built protocol to accelerate and secure transfer over the Internet or AWS Direct Connect, at speeds up to 10 times faster than open-source tools. Using DataSync you can perform one-time data migrations, transfer on-premises data for timely in-cloud analysis, and automate replication to AWS for data protection and recovery.

Q. What interoperability and compatibility is there between existing AWS services and Amazon EFS?

Amazon EFS is integrated with a number of other AWS services, including Amazon CloudWatch, AWS CloudFormation, AWS CloudTrail, AWS IAM, and AWS Tagging services.

Amazon CloudWatch allows you to monitor file system activity using metrics. AWS CloudFormation allows you to create and manage file systems using templates.

AWS CloudTrail allows you to record all Amazon EFS API calls in log files.

AWS Identity and Access Management (IAM) allows you to control who can administer your file system. AWS Tagging services allows you to label your file systems with metadata that you define.

**Elastic Fabric Adapter (EFA)**

Q: Why should I use EFA?

EFA brings the scalability, flexibility, and elasticity of cloud to tightly-coupled HPC applications. With EFA, tightly-coupled HPC applications have access to lower and more consistent latency and higher throughput than traditional TCP channels, enabling them to scale better. EFA support can be enabled dynamically, on-demand on any supported EC2 instance without pre-reservation, giving you the flexibility to respond to changing business/workload priorities.

Q: What types of applications can benefit from using EFA?

High Performance Computing (HPC) applications distribute computational workloads across a cluster of instances for parallel processing. Examples of HPC applications include computational fluid dynamics (CFD), crash simulations, and weather simulations. HPC applications are generally written using the Message Passing Interface (MPI) and impose stringent requirements for inter-instance communication in terms of both latency and bandwidth. Applications using MPI and other HPC middleware which supports the libfabric communication stack can benefit from EFA.

Q: What is the minimum time interval granularity for the data that Amazon CloudWatch receives and aggregates?

Metrics are received and aggregated at 1 minute intervals.

Q: Will I lose the metrics data if I disable monitoring for an Amazon EC2 instance?

You can retrieve metrics data for any Amazon EC2 instance up to 2 weeks from the time you started to monitor it. After 2 weeks, metrics data for an Amazon EC2 instance will not be available if monitoring was disabled for that Amazon EC2 instance. If you want to archive metrics beyond 2 weeks you can do so by calling mon-get-stats command from the command line and storing the results in Amazon S3 or Amazon SimpleDB.

Q: Can I access the metrics data for a terminated Amazon EC2 instance or a deleted Elastic Load Balancer?

Yes. Amazon CloudWatch stores metrics for terminated Amazon EC2 instances or deleted Elastic Load Balancers for 2 weeks.

Q: Why does the graphing of the same time window look different when I view in 5 minute and 1 minute periods?

If you view the same time window in a 5 minute period versus a 1 minute period, you may see that data points are displayed in different places on the graph. For the period you specify in your graph, Amazon CloudWatch will find all the available data points and calculates a single, aggregate point to represent the entire period. In the case of a 5 minute period, the single data point is placed at the beginning of the 5 minute time window. In the case of a 1 minute period, the single data point is placed at the 1 minute mark. We recommend using a 1 minute period for troubleshooting and other activities that require the most precise graphing of time periods.

**Elastic IP**

Q: Why am I limited to 5 Elastic IP addresses per region?

Public (IPV4) internet addresses are a scarce resource. There is only a limited amount of public IP space available, and Amazon EC2 is committed to helping use that space efficiently.

By default, all accounts are limited to 5 Elastic IP addresses per region. If you need more the 5 Elastic IP addresses, we ask that you apply for your limit to be raised. We will ask you to think through your use case and help us understand your need for additional addresses. Any increases will be specific to the region they have been requested for.

Q: Do I need one Elastic IP address for every instance that I have running?

No. You do not need an Elastic IP address for all your instances. By default, every instance comes with a private IP address and an internet routable public IP address. The private IP address remains associated with the network interface when the instance is stopped and restarted, and is released when the instance is terminated. The public address is associated exclusively with the instance until it is stopped, terminated or replaced with an Elastic IP address. These IP addresses should be adequate for many applications where you do not need a long lived internet routable end point. Compute clusters, web crawling, and backend services are all examples of applications that typically do not require Elastic IP addresses.

**Elastic Load Balancing**

Q: What load balancing options does the Elastic Load Balancing service offer?

Elastic Load Balancing offers two types of load balancers that both feature high availability, automatic scaling, and robust security. These include the [Classic Load Balancer](https://aws.amazon.com/elasticloadbalancing/classicloadbalancer/) that routes traffic based on either application or network level information, and the Application Load Balancer that routes traffic based on advanced application level information that includes the content of the request.

Q: When should I use the Classic Load Balancer and when should I use the Application Load Balancer?

The Classic Load Balancer is ideal for simple load balancing of traffic across multiple EC2 instances, while the Application Load Balancer is ideal for applications needing advanced routing capabilities, microservices, and container-based architectures.

**Security**

Q: How do I prevent other people from viewing my systems?

You have complete control over the visibility of your systems. The Amazon EC2 security systems allow you to place your running instances into arbitrary groups of your choice. Using the web services interface, you can then specify which groups may communicate with which other groups, and also which IP subnets on the Internet may talk to which groups. This allows you to control access to your instances in our highly dynamic environment. Of course, you should also secure your instance as you would any other server.

Q: Can I get a history of all EC2 API calls made on my account for security analysis and operational troubleshooting purposes?

Yes. To receive a history of all EC2 API calls (including VPC and EBS) made on your account, you simply turn on CloudTrail in the [AWS Management Console](https://console.aws.amazon.com/cloudtrail/home).

**Amazon EC2 Auto Scaling**

Q: Can I automatically scale my Amazon EC2 fleets?

Yes. [Amazon EC2 Auto Scaling](https://aws.amazon.com/ec2/autoscaling/) is a fully managed service designed to launch or terminate Amazon EC2 instances automatically to help ensure you have the correct number of Amazon EC2 instances available to handle the load for your application. EC2 Auto Scaling helps you maintain application availability through fleet management for EC2 instances, which detects and replaces unhealthy instances, and by scaling your Amazon EC2 capacity up or down automatically according to conditions you define. You can use EC2 Auto Scaling to automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

Q: What are the benefits of using Amazon EC2 Auto Scaling?

Amazon EC2 Auto Scaling helps to maintain your Amazon EC2 instance availability. Whether you are running one Amazon EC2 instance or thousands, you can use Amazon EC2 Auto Scaling to detect impaired Amazon EC2 instances, and replace the instances without intervention. This ensures that your application has the compute capacity that you expect. You can use Amazon EC2 Auto Scaling to automatically scale your Amazon EC2 fleet by following the demand curve for your applications, reducing the need to manually provision Amazon EC2 capacity in advance. For example, you can set a condition to add new Amazon EC2 instances in increments to the ASG when the average utilization of your Amazon EC2 fleet is high; and similarly, you can set a condition to remove instances in increments when CPU utilization is low. You can also use Amazon CloudWatch to send alarms to trigger scaling activities and Elastic Load Balancing (ELB) to distribute traffic to your instances within the ASG. If you have predictable load changes, you can set a schedule through Amazon EC2 Auto Scaling to plan your scaling activities. Amazon EC2 Auto Scaling enables you to run your Amazon EC2 fleet at optimal utilization.

Q: What is an EC2 Auto Scaling group (ASG)?

An Amazon EC2 Auto Scaling group (ASG) contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of fleet management and dynamic scaling. For example, if a single application operates across multiple instances, you might want to increase the number of instances in that group to improve the performance of the application, or decrease the number of instances to reduce costs when demand is low. Amazon EC2 Auto Scaling will automaticallly adjust the number of instances in the group to maintain a fixed number of instances even if a instance becomes unhealthy, or based on criteria that you specify.

Q: What happens to my Amazon EC2 instances if I delete my ASG?

If you have an EC2 Auto Scaling group (ASG) with running instances and you choose to delete the ASG, the instances will be terminated and the ASG will be deleted.

Q: How do I know when EC2 Auto Scaling is launching or terminating the EC2 instances in an EC2 Auto Scaling group?

When you use Amazon EC2 Auto Scaling to scale your applications automatically, it is useful to know when EC2 Auto Scaling is launching or terminating the EC2 instances in your EC2 Auto Scaling group. [Amazon SNS](https://aws.amazon.com/sns/) coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. You can configure EC2 Auto Scaling to send an SNS notification whenever your EC2 Auto Scaling group scales. Amazon SNS can deliver notifications as HTTP or HTTPS POST, email (SMTP, either plain-text or in JSON format), or as a message posted to an Amazon SQS queue. For example, if you configure your EC2 Auto Scaling group to use the autoscaling: EC2\_INSTANCE\_TERMINATE notification type, and your EC2 Auto Scaling group terminates an instance, it sends an email notification. This email contains the details of the terminated instance, such as the instance ID and the reason that the instance was terminated.

Q: What is a launch configuration?

A launch configuration is a template that an EC2 Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping. If you've launched an EC2 instance before, you specified the same information in order to launch the instance. When you create an EC2 Auto Scaling group, you must specify a launch configuration. You can specify your launch configuration with multiple EC2 Auto Scaling groups. However, you can only specify one launch configuration for an EC2 Auto Scaling group at a time, and you can't modify a launch configuration after you've created it. Therefore, if you want to change the launch configuration for your EC2 Auto Scaling group, you must create a launch configuration and then update your EC2 Auto Scaling group with the new launch configuration. When you change the launch configuration for your EC2 Auto Scaling group, any new instances are launched using the new configuration parameters, but existing instances are not affected.

Q: What are lifecycle hooks?

Lifecycle hooks let you take action before an instance goes into service or before it gets terminated. This can be especially useful if you are not baking your software environment into an Amazon Machine Image (AMI). For example, launch hooks can perform software configuration on an instance to ensure that it’s fully prepared to handle traffic before Amazon EC2 Auto Scaling proceeds to connect it to your load balancer. One way to do this is by connecting the launch hook to an AWS Lambda function that invokes RunCommand on the instance. Terminate hooks can be useful for collecting important data from an instance before it goes away. For example, you could use a terminate hook to preserve your fleet’s log files by copying them to an Amazon S3 bucket when instances go out of service.

Q: Which health check type should I select?

If you are using Elastic Load Balancing (ELB) with your group, you should select an ELB health check. If you’re not using ELB with your group, you should select the EC2 health check.

**Hibernate**

Q: Why should I hibernate an instance?

You can hibernate an instance to get your instance and applications up and running quickly, if they take long time to bootstrap (e.g. load memory caches). You can start instances, bring them to a desired state and hibernate them. These “pre-warmed” instances can then be resumed to reduce the time it takes for an instance to return to service. Hibernation retains memory state across Stop/Start cycles.

Q: What happens when I hibernate my instance?

When you hibernate an instance, data from your EBS root volume and any attached EBS data volumes is persisted. Additionally, contents from the instance’s memory (RAM) are persisted to EBS root volume. When the instance is restarted, it returns to its previous state and reloads the RAM contents.

Q: What is the difference between hibernate and stop?

In the case of hibernate, your instance gets hibernated and the RAM data persisted. In the case of Stop, your instance gets shutdown and RAM is cleared.

In both the cases, data from your EBS root volume and any attached EBS data volumes is persisted. Your private IP address remains the same, as does your elastic IP address (if applicable). The network layer behavior will be similar to that of EC2 Stop-Start workflow. Stop and hibernate are available for Amazon EBS backed instances only. Local instance storage is not persisted.

**Reserved Instance**

Q: What is a Reserved Instance?

A Reserved Instance (RI) is an EC2 offering that provides you with a significant discount on EC2 usage when you commit to a one-year or three-year term.

**Spot instances**

Q. What is a Spot instance?

Spot instances are spare EC2 capacity that can save you up 90% off of On-Demand prices that AWS can interrupt with a 2-minute notification. Spot uses the same underlying EC2 instances as On-Demand and Reserved Instances, and is best suited for fault-tolerant, flexible workloads. Spot instances provides an additional option for obtaining compute capacity and can be used along with On-Demand and Reserved Instances.

Q. How is a Spot instance different than an On-Demand instance or Reserved Instance?

While running, Spot instances are exactly the same as On-Demand or Reserved instances. The main differences are that Spot instances typically offer a significant discount off the On-Demand prices, your instances can be interrupted by Amazon EC2 for capacity requirements with a 2-minute notification, and Spot prices adjust gradually based on long term supply and demand for spare EC2 capacity.

Q. What price will I pay for a Spot instance?

You pay the Spot price that’s in effect at the beginning of each instance-hour for your running instance. If Spot price changes after you launch the instance, the new price is charged against the instance usage for the subsequent hour.

**Availability zones**

Q: How isolated are Availability Zones from one another?

Each Availability Zone runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable. Common points of failures like generators and cooling equipment are not shared across Availability Zones. Additionally, they are physically separate, such that even extremely uncommon disasters such as fires, tornados or flooding would only affect a single Availability Zone.

Q: How can I make sure that I am in the same Availability Zone as another developer?

We do not currently support the ability to coordinate launches into the same Availability Zone across AWS developer accounts. One Availability Zone name (for example, us-east-1a) in two AWS customer accounts may relate to different physical Availability Zones.

**Cluster instances**

Q. What is a Cluster Compute Instance?

Cluster Compute Instances combine high compute resources with a high performance networking for High Performance Compute (HPC) applications and other demanding network-bound applications. Cluster Compute Instances provide similar functionality to other Amazon EC2 instances but have been specifically engineered to provide high performance networking.

Amazon EC2 cluster placement group functionality allows users to group Cluster Compute Instances in clusters – allowing applications to get the low-latency network performance necessary for tightly-coupled node-to-node communication typical of many HPC applications. Cluster Compute Instances also provide significantly increased network throughput both within the Amazon EC2 environment and to the Internet. As a result, these instances are also well suited for customer applications that need to perform network-intensive operations.

Q. What is a cluster placement group?

A cluster placement group is a logical entity that enables creating a cluster of instances by launching instances as part of a group. The cluster of instances then provides low latency connectivity between instances in the group. Cluster placement groups are created through the Amazon EC2 API or AWS Management Console.

Q: How do I select the right instance type?

Amazon EC2 instances are grouped into 5 families: General Purpose, Compute Optimized, Memory Optimized, Storage Optimized and Accelerated Computing instances. General Purpose Instances have memory to CPU ratios suitable for most general purpose applications and come with fixed performance (M5, M4) or burstable performance (T2); Compute Optimized instances (C5, C4) have proportionally more CPU resources than memory (RAM) and are well suited for scale out compute-intensive applications and High Performance Computing (HPC) workloads; Memory Optimized Instances (X1e, X1, R4) offer larger memory sizes for memory-intensive applications, including database and memory caching applications; Accelerating Computing instances (P3, P2, G3, F1) take advantage of the parallel processing capabilities of NVIDIA Tesla GPUs for high performance computing and machine/deep learning; GPU Graphics instances (G3) offer high-performance 3D graphics capabilities for applications using OpenGL and DirectX; F1 instances deliver Xilinx FPGA-based reconfigurable computing; Storage Optimized Instances (H1, I3, D2) that provide very high, low latency, I/O capacity using SSD-based local instance storage for I/O-intensive applications, with D2 or H1, the dense-storage and HDD-storage instances, provide local high storage density and sequential I/O performance for data warehousing, Hadoop and other data-intensive applications. When choosing instance types, you should consider the characteristics of your application with regards to resource utilization (i.e. CPU, Memory, Storage) and select the optimal instance family and instance size.

**AWS Elastic Beanstalk**

Q: What is AWS Elastic Beanstalk?

AWS Elastic Beanstalk makes it even easier for developers to quickly deploy and manage applications in the AWS Cloud. Developers simply upload their application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

Q: Who should use AWS Elastic Beanstalk?

Those who want to deploy and manage their applications within minutes in the AWS Cloud. You don’t need experience with [cloud computing](https://aws.amazon.com/what-is-cloud-computing/) to get started. AWS Elastic Beanstalk supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications.

Q: What can developers now do with AWS Elastic Beanstalk that they could not before?

AWS Elastic Beanstalk automates the details of capacity provisioning, load balancing, auto scaling, and application deployment, creating an environment that runs a version of your application. You can simply upload your deployable code (e.g., WAR file), and AWS Elastic Beanstalk does the rest. The AWS Toolkit for Visual Studio and the AWS Toolkit for Eclipse allow you to deploy your application to AWS Elastic Beanstalk and manage it without leaving your IDE. Once your application is running, Elastic Beanstalk automates management tasks–such as monitoring, application version deployment, a basic health check–and facilitates log file access. By using Elastic Beanstalk, developers can focus on developing their application and are freed from deployment-oriented tasks, such as provisioning servers, setting up load balancing, or managing scaling.

Q: How is AWS Elastic Beanstalk different from existing application containers or platform-as-a-service solutions?

Most existing application containers or platform-as-a-service solutions, while reducing the amount of programming required, significantly diminish developers’ flexibility and control. Developers are forced to live with all the decisions predetermined by the vendor–with little to no opportunity to take back control over various parts of their application’s infrastructure. However, with AWS Elastic Beanstalk, developers retain full control over the AWS resources powering their application. If developers decide they want to manage some (or all) of the elements of their infrastructure, they can do so seamlessly by using Elastic Beanstalk’s management capabilities.

Q: How do I make my application private?

By default, your application is available publicly at myapp.elasticbeanstalk.com for anyone to access. You can use Amazon VPC to provision a private, isolated section of your application in a virtual network that you define. This virtual network can be made private through specific security group rules, network ACLs, and custom route tables. You can also easily control what other incoming traffic, such as SSH, is delivered or not to your application servers by changing the EC2 security group settings.

Q: Can I restrict access to specific AWS Elastic Beanstalk resources?

Yes. You can allow or deny permissions to specific AWS Elastic Beanstalk resources, such as applications, application versions, and environments.

Q: Who has access to an AWS Elastic Beanstalk environment launched by an IAM user?

The root account has full access to all AWS Elastic Beanstalk environments launched by any IAM user under that account. If you use the Elastic Beanstalk template to grant read-only access to an IAM user, that user will be able to view all applications, application versions, environments, and any associated resources in that account. If you use the Elastic Beanstalk template to grant full access to an IAM user, that user will be able to create, modify, and terminate any Elastic Beanstalk resources under that account.

Q: How can I keep the underlying platform of the environment running my application automatically up-to-date?

You can opt-in to having your AWS Elastic Beanstalk environments automatically updated to the latest version of the underlying platform running your application during a specified maintenance window. Elastic Beanstalk regularly releases new versions of supported platforms (Java, PHP, Ruby, Node.js, Python, .NET, Go, and Docker) with operating system, web and application server, and language and framework updates.

**AWS Lambda**

Q: What is serverless computing?

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by AWS. At the core of serverless computing is AWS Lambda, which lets you run your code without provisioning or managing servers.

Q: What is AWS Lambda?

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Q: What events can trigger an AWS Lambda function?

**Services That Lambda Reads Events From -**

Amazon Kinesis

Amazon DynamoDB

Amazon Simple Queue Service

**Services That Invoke Lambda Functions Synchronously -**

Elastic Load Balancing (Application Load Balancer)

Amazon Cognito

Amazon Lex

Amazon Alexa

Amazon API Gateway

Amazon CloudFront (Lambda@Edge)

Amazon Kinesis Data Firehose

**Services That Invoke Lambda Functions Asynchronously -**

Amazon Simple Storage Service

Amazon Simple Notification Service

Amazon Simple Email Service

AWS CloudFormation

Amazon CloudWatch Logs

Amazon CloudWatch Events

AWS CodeCommit

AWS Config

Q: When should I use AWS Lambda versus Amazon EC2?

[Amazon EC2](https://aws.amazon.com/ec2/) offers flexibility, with a wide range of instance types and the option to customize the operating system, network and security settings, and the entire software stack, allowing you to easily move existing applications to the cloud. With Amazon EC2 you are responsible for provisioning capacity, monitoring fleet health and performance, and designing for fault tolerance and scalability. [AWS Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/) offers an easy-to-use service for deploying and scaling web applications in which you retain ownership and full control over the underlying EC2 instances. [Amazon EC2 Container Service](https://aws.amazon.com/ecs/) is a scalable management service that supports Docker containers and allows you to easily run distributed applications on a managed cluster of Amazon EC2 instances.

AWS Lambda makes it easy to execute code in response to events, such as changes to Amazon S3 buckets, updates to an Amazon DynamoDB table, or custom events generated by your applications or devices. With Lambda you do not have to provision your own instances; Lambda performs all the operational and administrative activities on your behalf, including capacity provisioning, monitoring fleet health, applying security patches to the underlying compute resources, deploying your code, running a web service front end, and monitoring and logging your code. AWS Lambda provides easy scaling and high availability to your code without additional effort on your part.

Q: What kind of code can run on AWS Lambda?

AWS Lambda offers an easy way to accomplish many activities in the cloud. For example, you can use AWS Lambda to build mobile back-ends that retrieve and transform data from Amazon DynamoDB, handlers that compress or transform objects as they are uploaded to Amazon S3, auditing and reporting of API calls made to any Amazon Web Service, and server-less processing of streaming data using Amazon Kinesis.

Q: What kind of code can run on AWS Lambda?

AWS Lambda offers an easy way to accomplish many activities in the cloud. For example, you can use AWS Lambda to build mobile back-ends that retrieve and transform data from Amazon DynamoDB, handlers that compress or transform objects as they are uploaded to Amazon S3, auditing and reporting of API calls made to any Amazon Web Service, and server-less processing of streaming data using Amazon Kinesis.

Q: Can I access the infrastructure that AWS Lambda runs on?

No. AWS Lambda operates the compute infrastructure on your behalf, allowing it to perform health checks, apply security patches, and do other routine maintenance.

Q: How does AWS Lambda isolate my code?

Each AWS Lambda function runs in its own isolated environment, with its own resources and file system view. AWS Lambda uses the same techniques as Amazon EC2 to provide security and separation at the infrastructure and execution levels.

Q: How does AWS Lambda secure my code?

AWS Lambda stores code in Amazon S3 and encrypts it at rest. AWS Lambda performs additional integrity checks while your code is in use.

Q: What is an AWS Lambda function?

The code you run on AWS Lambda is uploaded as a “Lambda function”. Each function has associated configuration information, such as its name, description, entry point, and resource requirements. The code must be written in a “stateless” style i.e. it should assume there is no affinity to the underlying compute infrastructure. Local file system access, child processes, and similar artifacts may not extend beyond the lifetime of the request, and any persistent state should be stored in Amazon S3, Amazon DynamoDB, or another Internet-available storage service. Lambda functions can include libraries, even native ones.

Q: Why must AWS Lambda functions be stateless?

Keeping functions stateless enables AWS Lambda to rapidly launch as many copies of the function as needed to scale to the rate of incoming events. While AWS Lambda’s programming model is stateless, your code can access stateful data by calling other web services, such as Amazon S3 or Amazon DynamoDB.

Q: How do I monitor an AWS Lambda function?

AWS Lambda automatically monitors Lambda functions on your behalf, reporting real-time metrics through Amazon CloudWatch, including total requests, account-level and function-level concurrency usage, latency, error rates, and throttled requests. You can view statistics for each of your Lambda functions via the Amazon CloudWatch console or through the AWS Lambda console. You can also call third-party monitoring APIs in your Lambda function.

Q: How do I scale an AWS Lambda function?

You do not have to scale your Lambda functions – AWS Lambda scales them automatically on your behalf. Every time an event notification is received for your function, AWS Lambda quickly locates free capacity within its compute fleet and runs your code. Since your code is stateless, AWS Lambda can start as many copies of your function as needed without lengthy deployment and configuration delays. There are no fundamental limits to scaling a function. AWS Lambda will dynamically allocate capacity to match the rate of incoming events.

Q: How long can an AWS Lambda function execute?

AWS Lambda functions can be configured to run up to 15 minutes per execution. You can set the timeout to any value between 1 second and 15 minutes.

Q: Does AWS Lambda support versioning?

Yes. By default, each AWS Lambda function has a single, current version of the code. Clients of your Lambda function can call a specific version or get the latest implementation.

Q: How long after uploading my code will my AWS Lambda function be ready to call?

Deployment times may vary with the size of your code, but AWS Lambda functions are typically ready to call within seconds of upload.

**Using AWS Lambda to process AWS events**

Q: What is an event source?

An event source is an AWS service or developer-created application that produces events that trigger an AWS Lambda function to run. Some services publish these events to Lambda by invoking the cloud function directly (for example, Amazon S3). Lambda can also poll resources in other services that do not publish events to Lambda. For example, Lambda can pull records from an Amazon Kinesis stream or an Amazon SQS queue and execute a Lambda function for each fetched message.

Many other services, such as AWS CloudTrail, can act as event sources simply by logging to Amazon S3 and using S3 bucket notifications to trigger AWS Lambda functions.

Q: How are events represented in AWS Lambda?

Events are passed to a Lambda function as an event input parameter. For event sources where events arrive in batches, such as Amazon SQS, Amazon Kinesis, and Amazon DynamoDB Streams, the event parameter may contain multiple events in a single call, based on the batch size you request.

Q: How can my AWS Lambda function personalize their behavior based on the identity of the end user of an application?

When your app uses the Amazon Cognito identity, end users can authenticate themselves using a variety of public login providers such as Amazon, Facebook, Google, and other OpenID Connect-compatible services. User identity is then automatically and secured presented to your Lambda function in the form of an Amazon Cognito id, allowing it to access user data from Amazon Cognito, or as a key to store and retrieve data in Amazon DynamoDB or other web services.

**Using AWS Lambda to build applications**

Q: How do I deploy and manage a serverless application?

You can deploy and manage your serverless applications using the AWS Serverless Application Model (AWS SAM). AWS SAM is a specification that prescribes the rules for expressing serverless applications on AWS. This specification aligns with the syntax used by AWS CloudFormation today and is supported natively within AWS CloudFormation as a set of resource types (referred to as "serverless resources"). These resources make it easier for AWS customers to use CloudFormation to configure and deploy serverless applications, using existing CloudFormation APIs.

Q: How do I automate deployment for a serverless application?

You can automate your serverless application’s release process using AWS CodePipeline and AWS CodeDeploy. CodePipeline is a continuous delivery service that enables you to model, visualize and automate the steps required to release your serverless application. CodeDeploy provides a deployment automation engine for your Lambda-based applications. CodeDeploy lets you orchestrate deployments according to established best-practice methodologies such as canary and linear deployments, and helps you establish the necessary guardrails to verify that newly-deployed code is safe, stable, and ready to be fully released to production.

Q: How do I troubleshoot a serverless application?

You can enable your Lambda function for tracing with [AWS X-Ray](https://aws.amazon.com/xray/) by adding X-Ray permissions to your Lambda function’s execution role and changing your function’s “tracing mode” to “active. ” When X-Ray is enabled for your Lambda function, AWS Lambda will emit tracing information to X-Ray regarding the Lambda service overhead incurred when invoking your function. This will provide you with insights such as Lambda service overhead, function init time, and function execution time. In addition, you can include the X-Ray SDK in your Lambda deployment package to create your own trace segments, annotate your traces, or view trace segments for downstream calls made from your Lambda function. X-Ray SDKs are currently available for Node.js and Java.

Q: What is Lambda@Edge?

[Lambda@Edge](https://aws.amazon.com/lambda/edge/) allows you to run code across AWS locations globally without provisioning or managing servers, responding to end users at the lowest network latency. You just upload your Node.js code to AWS Lambda and configure your function to be triggered in response to [Amazon CloudFront](https://aws.amazon.com/cloudfront/) requests (i.e., when a viewer request lands, when a request is forwarded to or received back from the origin, and right before responding back to the end user). The code is then ready to execute across AWS locations globally when a request for content is received, and scales with the volume of CloudFront requests globally.

Q: How is AWS Lambda@Edge different from using AWS Lambda behind Amazon API Gateway?

The difference is that API Gateway and Lambda are regional services. Using [Lambda@Edge](https://aws.amazon.com/lambda/edge/) and [Amazon CloudFront](https://aws.amazon.com/cloudfront/) allows you to execute logic across multiple AWS locations based on where your end viewers are located.

Q: How do I coordinate calls between multiple Lambda functions?

You can use Amazon Step Functions to coordinate multiple invoking Lambda functions. You can invoke multiple Lambda functions serially, passing the output of one to the other, or in parallel.

Q: How do I allow my AWS Lambda function access to other AWS resources?

You grant permissions to your Lambda function to access other resources using an IAM role. AWS Lambda assumes the role while executing your Lambda function, so you always retain full, secure control of exactly which AWS resources it can use.

Q: Can I access resources behind Amazon VPC with my AWS Lambda function?

Yes. You can access resources behind Amazon VPC.

**Elastic Load Balancer**

Q: How do I decide which load balancer to select for my application?

A: Elastic Load Balancing supports three types of load balancers. You can select the appropriate load balancer based on your application needs. If you need flexible application management then we recommend you to use Application Load Balancer. If extreme performance and static IP is needed for your application then we recommend you to use Network Load Balancer. If your application is built within the EC2 Classic network then you should use Classic Load Balancer.

Q: Can I privately access Elastic Load Balancing APIs from my Amazon Virtual Private Cloud (VPC) without using public IPs?

A: Yes, you can privately access Elastic Load Balancing APIs from your Amazon Virtual Private Cloud (VPC) by creating [VPC endpoints](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-endpoints.html). With VPC endpoints, the routing between the VPC and Elastic Load Balancing APIs is handled by the AWS network without the need for an Internet gateway, NAT gateway, or VPN connection. The latest generation of VPC Endpoints used by Elastic Load Balancing are powered by AWS PrivateLink, an AWS technology enabling the private connectivity between AWS services using Elastic Network Interfaces (ENI) with private IPs in your VPCs.

Q. How can I protect my web applications behind a load balancer from web attacks?

A: You can integrate your Application Load Balancer with AWS WAF, a web application firewall that helps protect web applications from attacks by allowing you to configure rules based on IP addresses, HTTP headers, and custom URI strings. Using these rules, AWS WAF can block, allow, or monitor (count) web requests for your web application.