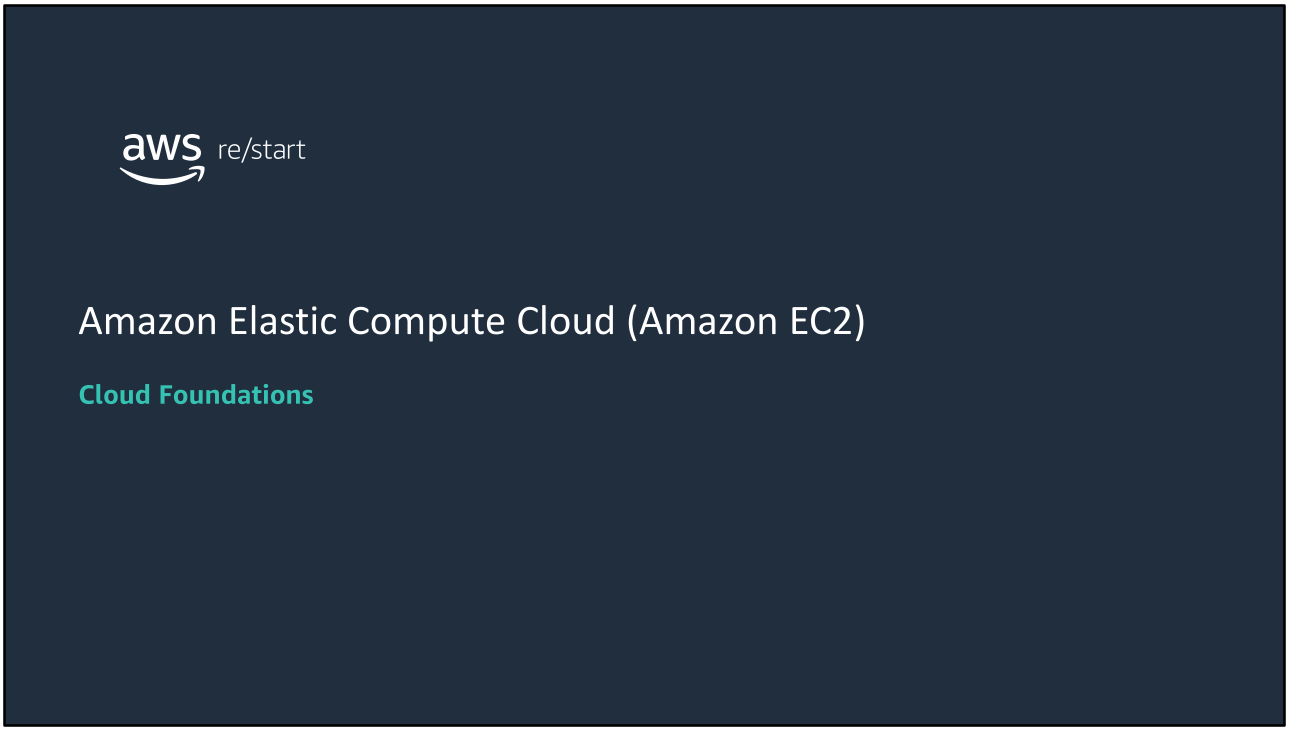
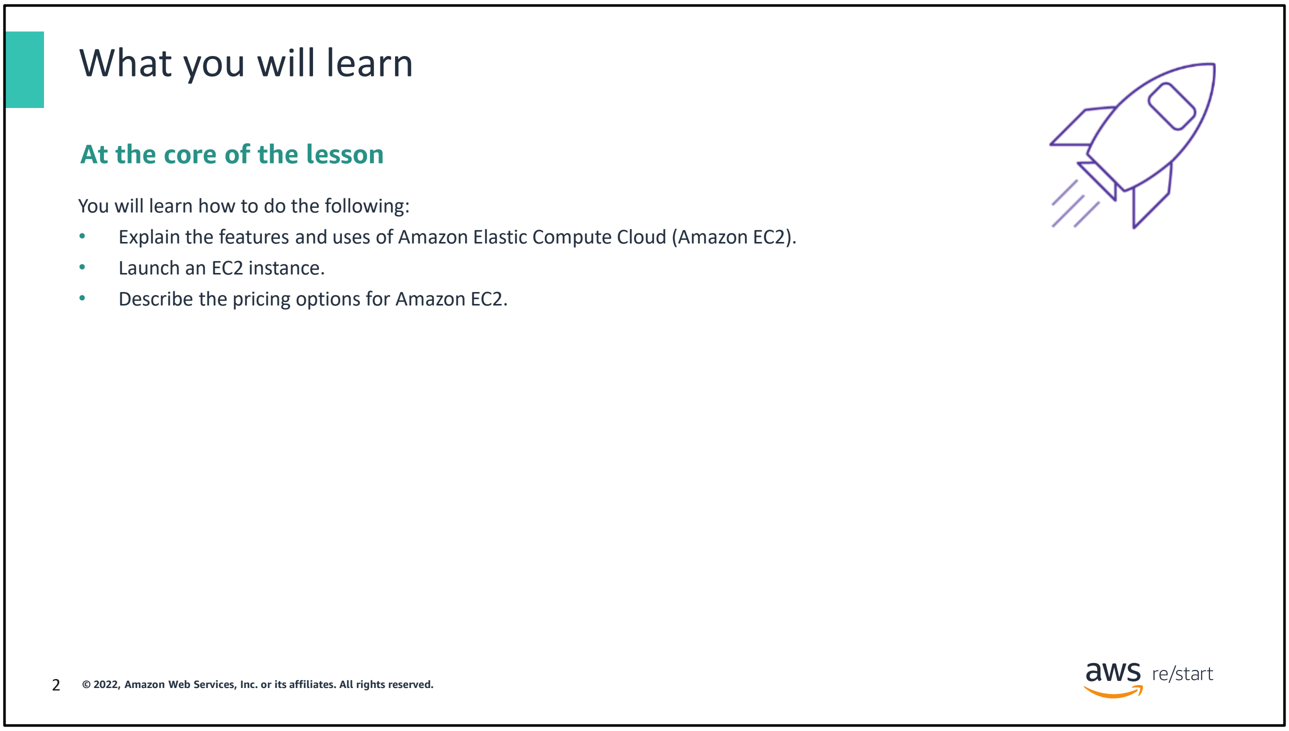


Amazon Elastic Compute Cloud (Amazon EC2)

Cloud Foundations

Welcome to Amazon Elastic Compute Cloud (Amazon EC2).



What you will learn

At the core of the lesson

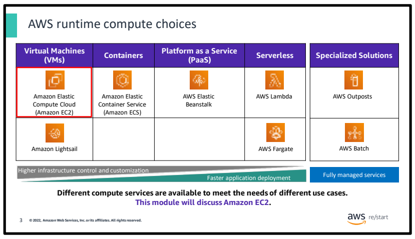
You will learn how to do the following:

- Explain the features and uses of Amazon Elastic Compute Cloud (Amazon EC2).
- Launch an EC2 instance.
- Describe the pricing options for Amazon EC2.

Amazon Web Services (AWS) provides multiple services to build a solution. Some of those services provide the foundation to all solutions, which are also known as the **core services**. This module provides insight into the offerings of each service category and looks at the first group of services, Compute.

You will first get an overview of Compute services. Then, you will learn about Amazon Elastic Compute Cloud (or Amazon EC2).

Whether you want to build mobile apps or run massive clusters to sequence the human genome, building and running your business starts with compute. AWS has a broad catalog of Compute services. It offers everything from application services to flexible virtual servers and even serverless computing.



AWS runtime compute choices

Virtual Machines (VMs)	Containers	Platform as a Service (PaaS)	Serverless	Specialized Solutions
Amazon Elastic Compute Cloud (Amazon EC2)	Amazon Elastic Container Service (Amazon ECS)	AWS Elastic Beanstalk	AWS Lambda	AWS Outposts
Amazon Lightsail			AWS Fargate	AWS Batch

Higher infrastructure control and customization
Faster application deployment
Fully managed services

Different compute services are available to meet the needs of different use cases.
This module will discuss Amazon EC2.

3

AWS offers several compute options to meet different needs. When you consider the service to use for a given type of workload, it is important that you understand the available compute options. As the diagram shows, the key runtime compute choices can be grouped into four categories of cloud computing models:

- Virtual machines (VMs)
- Containers
- Platform as a service (also known as PaaS)
- Serverless

In addition, you can use specialized solutions to address specific compute use cases.

In the virtual machines category, AWS offers two core services. The first service is Amazon EC2. It provides secure and resizable virtual servers in the cloud. The second service is Amazon Lightsail. It provides virtual private servers to run workloads in a cost-effective way.

In the containers category, AWS offers Amazon Elastic Container Service (Amazon ECS). With this service you can run Docker container applications on AWS.

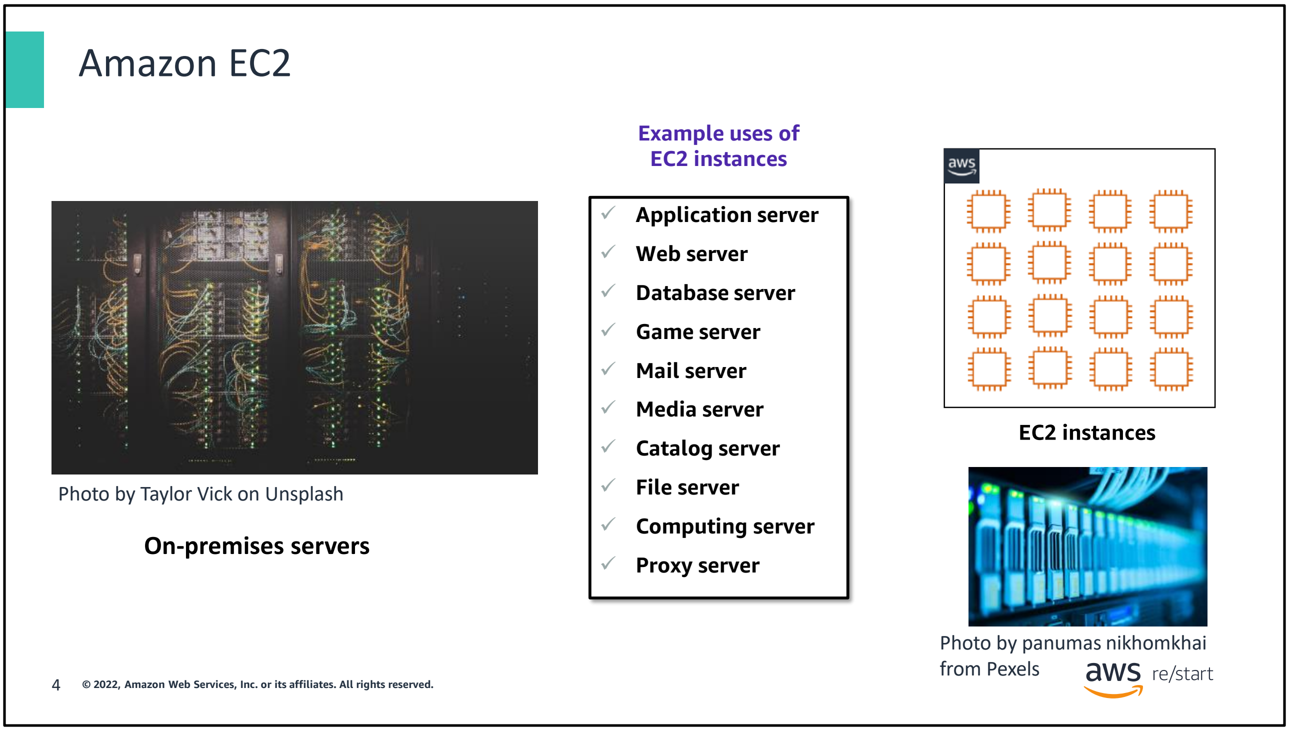
The platform as a service (PaaS) category includes AWS Elastic Beanstalk. It is a solution that runs web applications and services that are developed in languages such as Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker.

The serverless category includes AWS Lambda, which is a serverless compute solution that runs Java, Go, PowerShell, Node.js, C#, Python, or Ruby code. This category also includes AWS Fargate, which provides a serverless compute platform for containers.

For specialized solutions, AWS Outposts provides a way to run AWS infrastructure and services on premises. AWS Batch is a service that runs batch jobs at any scale.

When you select an AWS compute runtime for your workload, consider that virtual machines and container-based services provide more control over your infrastructure. They also allow for higher degrees of customization. PaaS and serverless services help you focus more on your application and less on infrastructure. They also facilitate quick deployment. The services in the specialized solutions category address specific types of workloads, or hybrid cloud and batch. These specialized services work well for these use cases because they are also fully managed by AWS.

Amazon EC2 is one of the core AWS services and is the focus of this module.



Amazon EC2

Photo by Taylor Vick on Unsplash

On-premises servers

Example uses of EC2 instances

- ✓ **Application server**
- ✓ **Web server**
- ✓ **Database server**
- ✓ **Game server**
- ✓ **Mail server**
- ✓ **Media server**
- ✓ **Catalog server**
- ✓ **File server**
- ✓ **Computing server**
- ✓ **Proxy server**

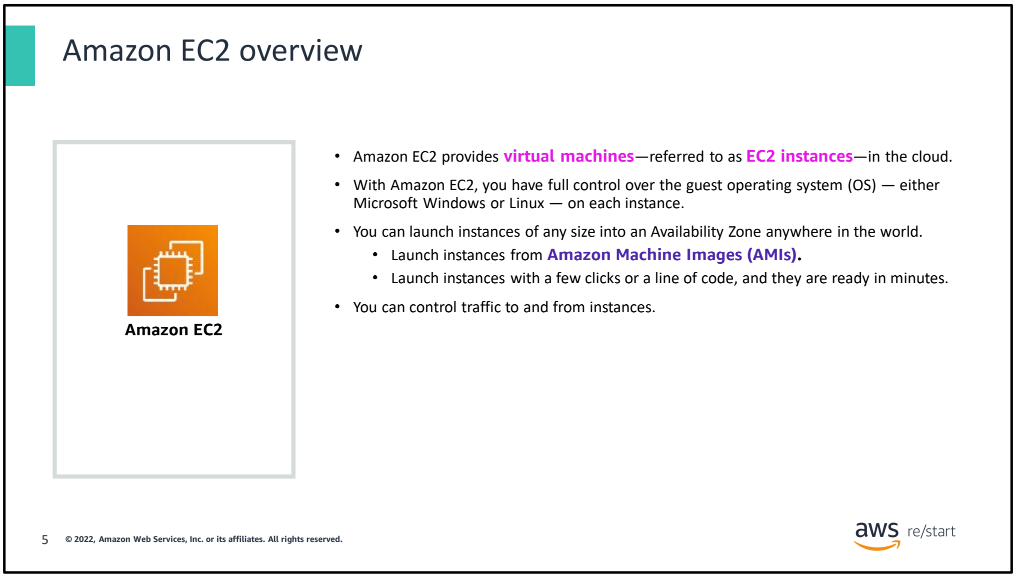
EC2 instances

Photo by panumas nikhomkhai from Pexels

Running on-premises servers is an expensive undertaking. Hardware must be procured, and this procurement can be based on project plans instead of on the reality of how the servers are used. Data centers are expensive to build, staff, and maintain. Organizations also need to permanently provision a sufficient amount of hardware to handle traffic spikes and peak workloads. After traditional on-premises deployments are built, server capacity might be unused and idle for a significant portion of the time that the servers are running, which is wasteful.

Amazon EC2 provides virtual machines where you can host the same kinds of applications that you might run on a traditional on-premises server. It provides secure, resizable compute capacity in the cloud. EC2 instances can support a variety of workloads. Common uses for EC2 instances include the following:

- Application servers
- Web servers
- Database servers
- Game servers
- Mail servers
- Media servers
- Catalog servers
- File servers
- Computing servers
- Proxy servers



Amazon EC2 overview

Amazon EC2

- Amazon EC2 provides **virtual machines**—referred to as **EC2 instances**—in the cloud.
- With Amazon EC2, you have full control over the guest operating system (OS) — either Microsoft Windows or Linux — on each instance.
- You can launch instances of any size into an Availability Zone anywhere in the world.
  - Launch instances from **Amazon Machine Images (AMIs).**
  - Launch instances with a few clicks or a line of code, and they are ready in minutes.
- You can control traffic to and from instances.

The EC2 in Amazon EC2 stands for Elastic Compute Cloud:

- *Elastic* refers to the fact that you can automatically increase or decrease the number of servers that you run to support an application. You can also increase or decrease the size of existing servers.
- *Compute* refers to reason why most users run servers: to host running applications or process data. These actions require compute resources, including processing power (central processing unit, or CPU) and memory (random access memory, or RAM).
- *Cloud* refers to the fact that the EC2 instances that you run are hosted in the cloud.

Amazon EC2 provides virtual machines in the cloud. This service gives you full administrative control over the Microsoft Windows or Linux operating system (OS) that runs on the instance. Most server OSs are supported, including Windows 2008, 2012, 2016, and 2019; Red Hat; SUSE; Ubuntu; and Amazon Linux.

An OS that runs on a virtual machine is often called a guest OS to distinguish it from the host OS. The host OS is directly installed on any server hardware that hosts one or more virtual machines.

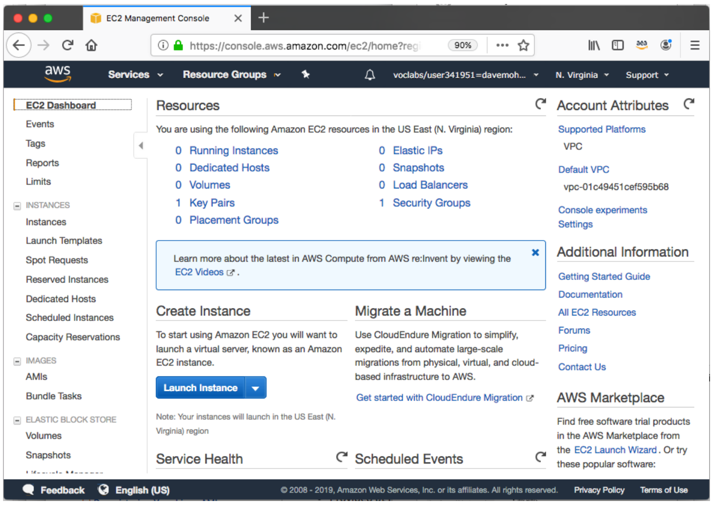
With Amazon EC2, you can launch any number of instances of any size into any Availability Zone anywhere in the world in minutes. Instances launch from Amazon Machine Images (AMIs), which are effectively virtual machine templates. This module discusses AMIs in more detail later.

You can control traffic to and from instances by using security groups. Also, because the servers run in the AWS Cloud, you can build solutions that use multiple AWS services.

Launching an EC2 instance

This module walks through **nine key decisions** to make when you create an EC2 instance by using the AWS Management Console **Launch Instance Wizard.**

- ➤ Along the way, this module will explore essential Amazon EC2 concepts.



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0	Running Instances	0	Elastic IPs
0	Dedicated Hosts	0	Snapshots
0	Volumes	0	Load Balancers
1	Key Pairs	1	Security Groups
0	Placement Groups		

Learn more about the latest in AWS Compute from AWS re:Invent by viewing the EC2 Videos.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (N. Virginia) region

Migrate a Machine

Use CloudEndure Migration to simplify, expedite, and automate large-scale migrations from physical, virtual, and cloud-based infrastructure to AWS.

Get started with CloudEndure Migration

Account Attributes

Supported Platforms: VPC

Default VPC: vpc-01c49451cef595b68

AWS Marketplace

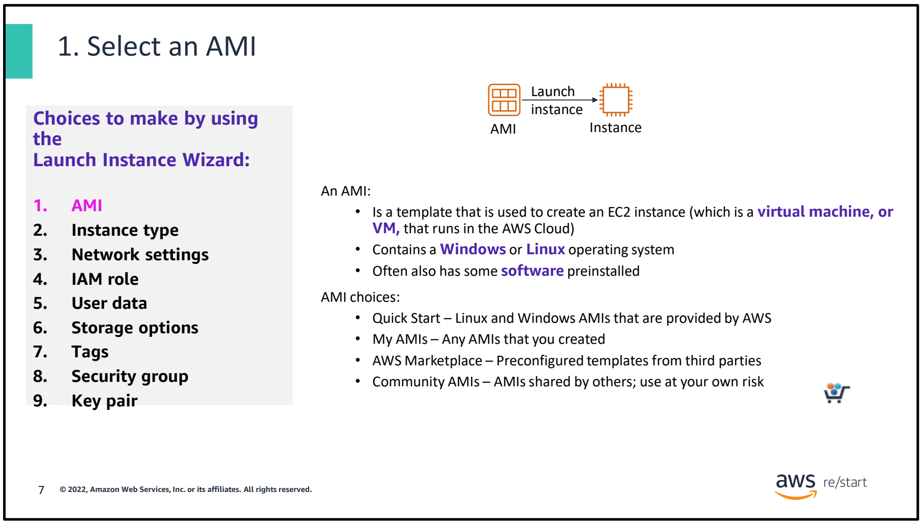
Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard. Or try these popular software:

The first time that you launch an EC2 instance, you will likely use the AWS Management Console Launch Instance Wizard.

The Launch Instance Wizard simplifies launching an instance. For example, if you choose to accept all the default settings, you can skip most of the steps that are provided by the wizard and launch an EC2 instance in a few clicks.

However, for most deployments, you will want to modify the default settings so that the servers you launch are deployed in a way that matches your specific needs.

The next slides introduce you to the essential choices that you must make when you launch an instance. The slides cover essential concepts that are good to know when you make these choices. You learn about these concepts to help you understand the options that are available and the effects of the decisions that you make.



1. Select an AMI

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

An AMI:

- Is a template that is used to create an EC2 instance (which is a **virtual machine, or VM,** that runs in the AWS Cloud)
- Contains a **Windows** or **Linux** operating system
- Often also has some **software** preinstalled

AMI choices:

- Quick Start – Linux and Windows AMIs that are provided by AWS
- My AMIs – Any AMIs that you created
- AWS Marketplace – Preconfigured templates from third parties
- Community AMIs – AMIs shared by others; use at your own risk

aws re/start

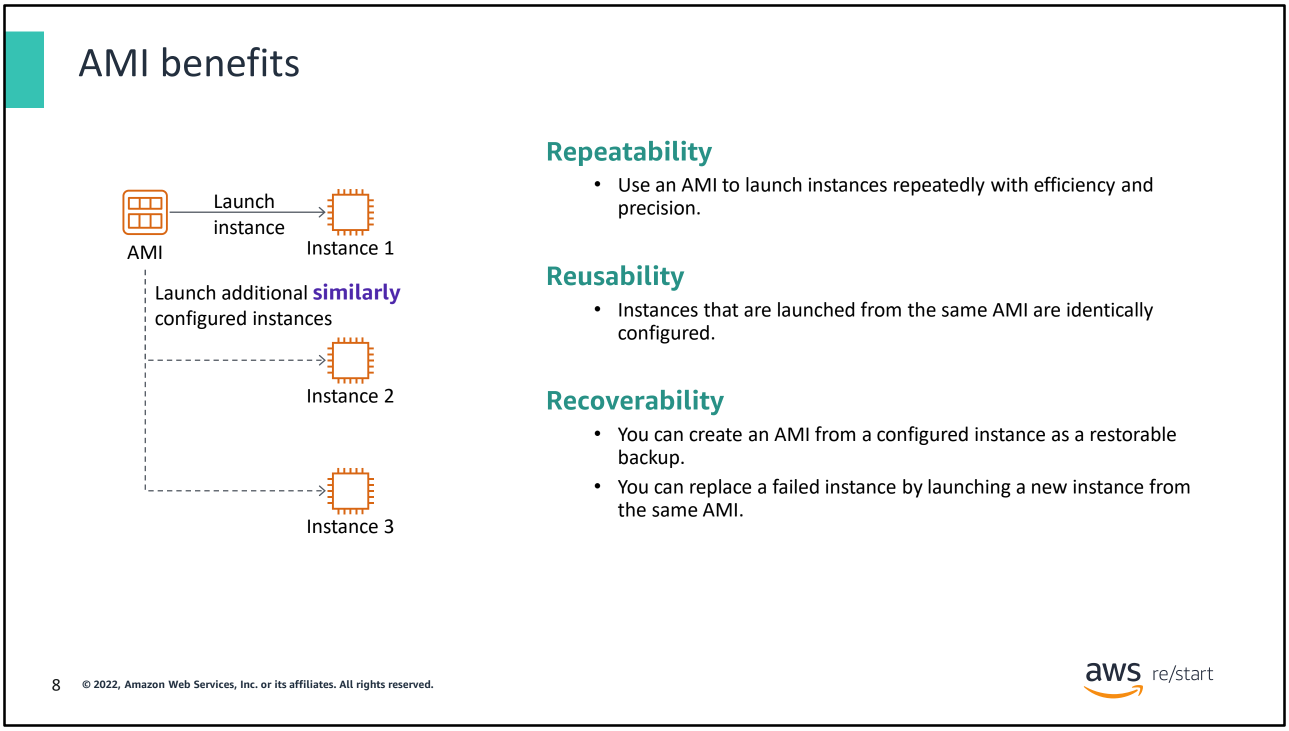
An AMI provides information that is needed to launch an EC2 instance. You must specify a source AMI when you launch an instance. You can use different AMIs to launch different types of instances. For example, you can choose one AMI to launch an instance that will become a web server and another AMI to deploy an instance that will host an application server. You can also launch multiple instances from a single AMI.

An AMI includes the following components:

- A template for the root volume of the instance – A root volume typically contains an OS and everything that was installed in that OS (applications, libraries, and so on). Amazon EC2 copies the template to the root volume of a new EC2 instance, and then starts it.
- Launch permissions that control which AWS accounts can use the AMI
- A block device mapping that specifies the volumes to attach to the instance (if any) when it is launched

You can choose many AMIs:

- Quick Start – AWS offers several prebuilt AMIs for launching your instances. These AMIs include many Linux and Windows options.
- My AMIs – These AMIs are AMIs that you created.
- AWS Marketplace – The AWS Marketplace offers a digital catalog that lists thousands of software solutions. These AMIs can offer specific use cases to help you get started quickly.
- Community AMIs – These AMIs are created by people all around the world. AWS does not check these AMIs, so use them at your own risk. Community AMIs can offer many different solutions to various problems, but use them with care. Avoid using them in any production or corporate environment.


AMI
Launch instance
Instance 1
Launch additional **similarly** configured instances
Instance 2
Instance 3
aws re/start

AMI benefits

Repeatability

- Use an AMI to launch instances repeatedly with efficiency and precision.

Reusability

- Instances that are launched from the same AMI are identically configured.

Recoverability

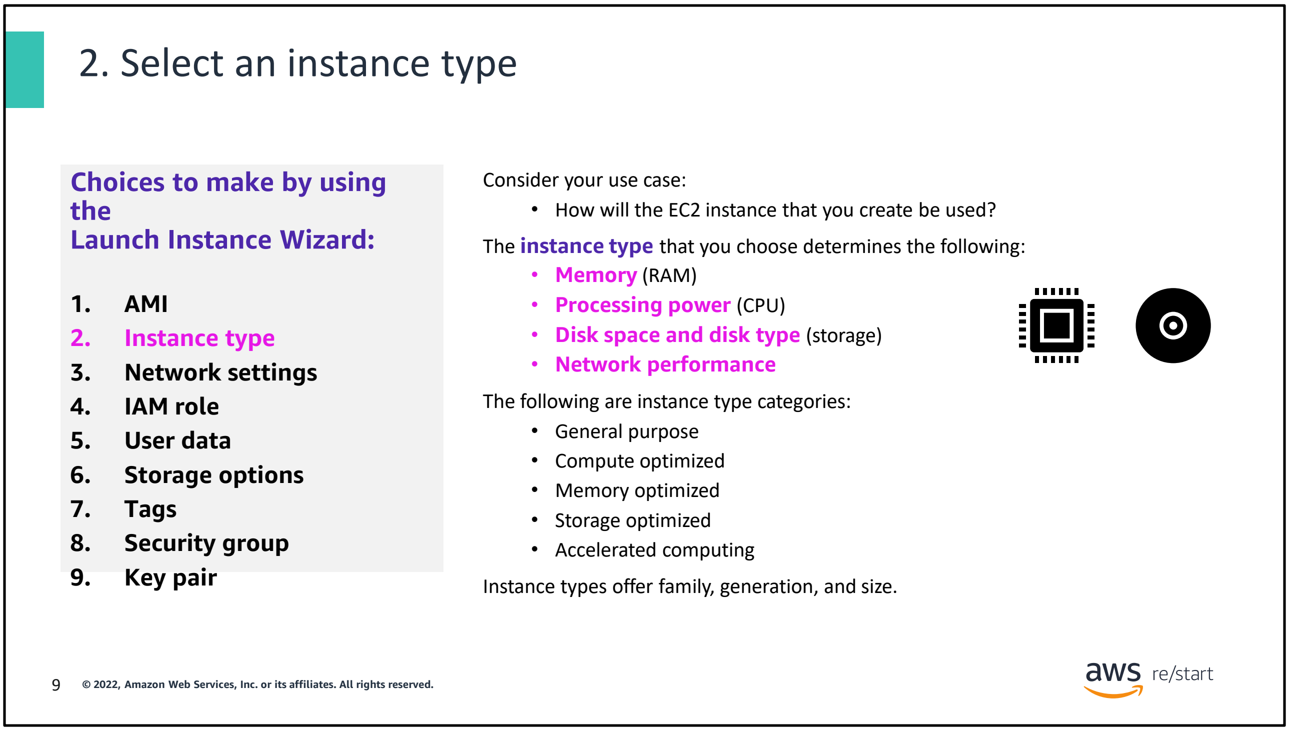
- You can create an AMI from a configured instance as a restorable backup.
- You can replace a failed instance by launching a new instance from the same AMI.

An AMI provides the information that is needed to launch an instance. The benefits of using an AMI include repeatability, reusability, and recoverability.

AMIs provide repeatability because an AMI packages the full configuration and content of an EC2 instance. As such, you can use it repeatedly to launch multiple instances with efficiency and precision.

AMIs promote reusability because instances that are launched from the same AMI are exact replicas of each other. This design facilitates building clusters of similar instances or recreate compute environments.

AMIs also facilitate recoverability. If an instance fails, you can replace it by launching a new instance from the same AMI that you used to launch the original instance. In addition, AMIs provide a way to back up a complete EC2 instance configuration, which you can use to launch a replacement instance if there is a failure.



2. Select an instance type

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

Consider your use case:

- How will the EC2 instance that you create be used?

The **instance type** that you choose determines the following:

- **Memory** (RAM)
- **Processing power** (CPU)
- **Disk space and disk type** (storage)
- **Network performance**

The following are instance type categories:

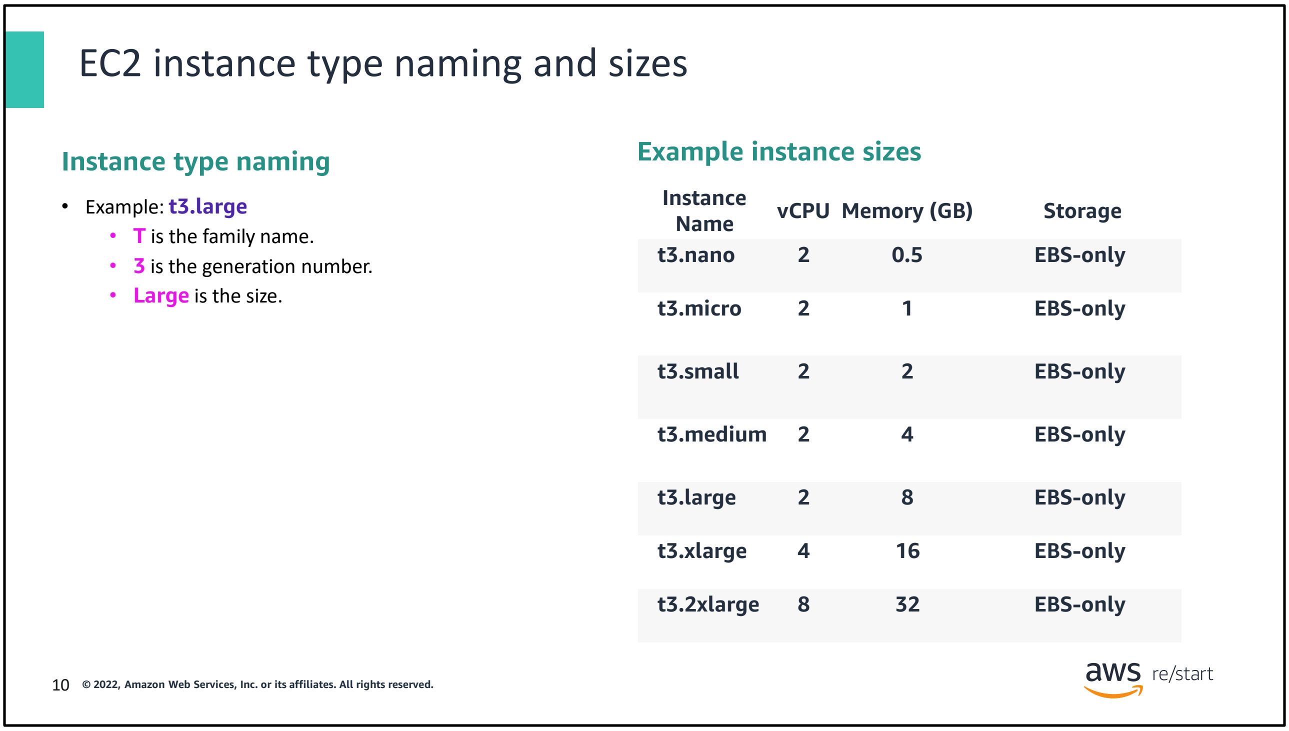
- General purpose
- Compute optimized
- Memory optimized
- Storage optimized
- Accelerated computing

Instance types offer family, generation, and size.

After you choose the AMI for launching the instance, you must choose on an instance type.

Amazon EC2 provides a selection of instance types that are optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity. The different instance types give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, which help you scale your resources to the requirements of your target workload.

Instance type categories include general-purpose, compute-optimized, memory-optimized, storage-optimized, and accelerated computing instances. Each instance type category offers many instance types to choose from.



EC2 instance type naming and sizes

Instance type naming

- Example: **t3.large**
  - **T** is the family name.
  - **3** is the generation number.
  - **Large** is the size.

Example instance sizes

Instance Name	vCPU	Memory (GB)	Storage
t3.nano	2	0.5	EBS-only
t3.micro	2	1	EBS-only
t3.small	2	2	EBS-only
t3.medium	2	4	EBS-only
t3.large	2	8	EBS-only
t3.xlarge	4	16	EBS-only
t3.2xlarge	8	32	EBS-only

The name of an EC2 instance type has several parts. For example, consider the T type.

T is the family name, which is then followed by a number. Here, that number is 3.

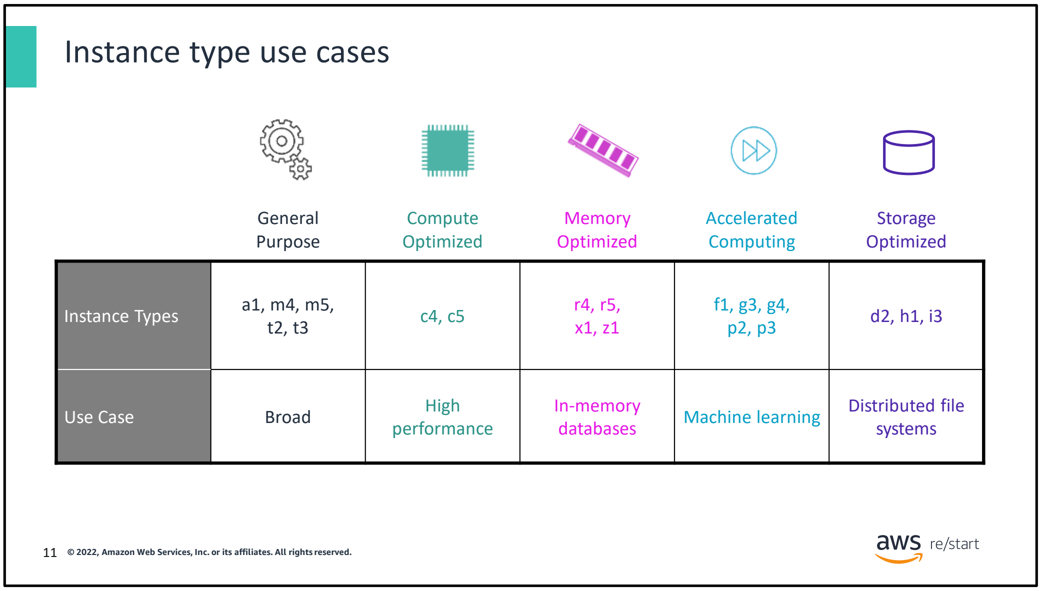
The number is the generation number of that type. For example, a T3 instance is the third generation of the T family. In general, instance types in a higher-number generation are more powerful and provide a better value for the price.

The next part of the name is the size portion of the instance. When you compare sizes, it's important to note the coefficient portion of the size category.

For example, a t3.2xlarge has twice the virtual CPU (vCPU) and memory of a t3.xlarge. The t3.xlarge has, in turn, twice the vCPU and memory of a t3.large.

It is also important to note that network bandwidth is also tied to the size of the EC2 instance. If you run jobs that are network intensive, you might need to increase the instance specifications to meet your needs.

In this slide, EBS refers to Amazon Elastic Block Store (Amazon EBS), which this module discusses later.



Instance type use cases

	General Purpose	Compute Optimized	Memory Optimized	Accelerated Computing	Storage Optimized
Instance Types	a1, m4, m5, t2, t3	c4, c5	r4, r5, x1, z1	f1, g3, g4, p2, p3	d2, h1, i3
Use Case	Broad	High performance	In-memory databases	Machine learning	Distributed file systems

Instance types vary in several ways, including CPU type, CPU or core count, storage type, storage amount, memory amount, and network performance. The chart provides a high-level view of the different instance categories and which instance type families and generation numbers fit into each category type. Consider a few of the instance types in more detail:

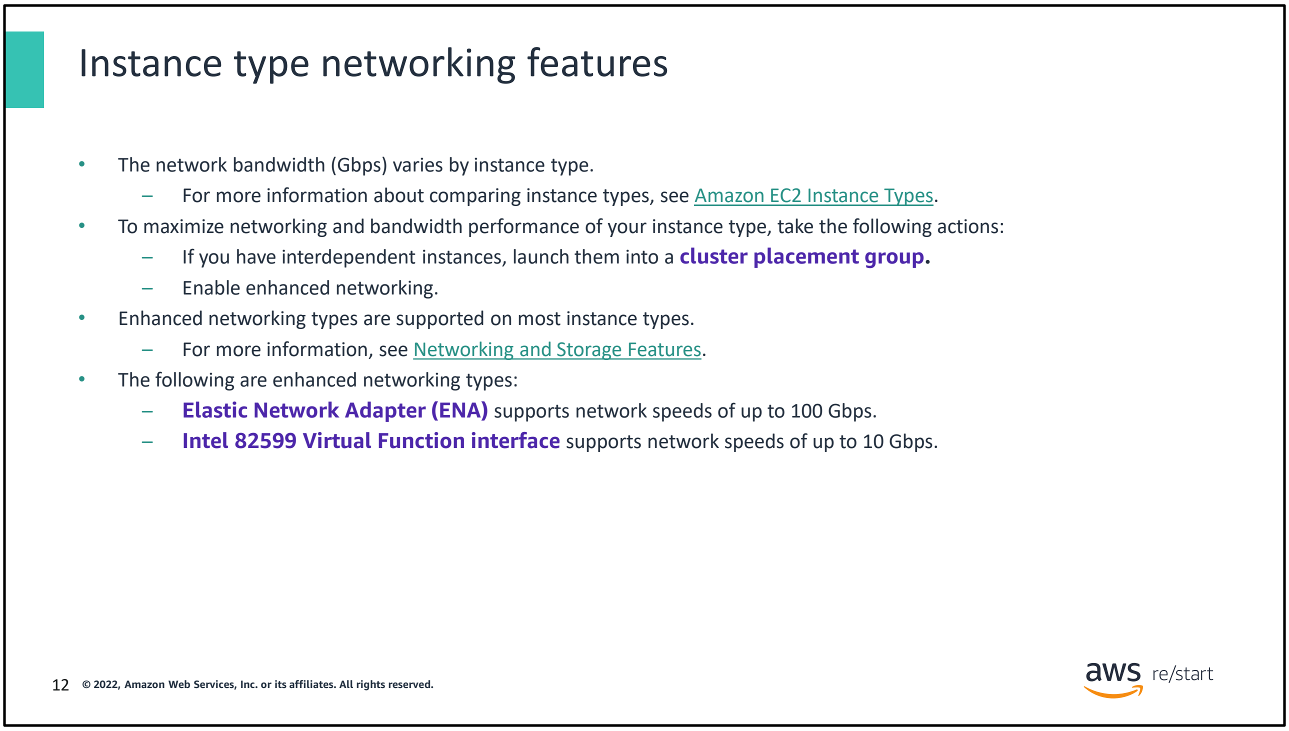
- T3 instances provide burstable performance general-purpose instances that provide a baseline level of CPU performance, with the ability to burst above the baseline. Use cases for this type of instance include the following:
  - Websites and web applications
  - Development environments
  - Build servers
  - Code repositories
  - Microservices
  - Test and staging environments
  - Line-of-business applications

- C5 instances are optimized for compute-intensive workloads. They deliver cost-effective high performance at a low price per compute ratio. Use cases include the following:
  - Scientific modeling
  - Batch processing
  - Ad serving
  - Highly scalable multiplayer gaming
  - Video encoding

- R5 instances are optimized for memory-intensive applications. Use cases include the following:

- High-performance databases
- Data mining and analysis
- In-memory databases
- Distributed web-scale in-memory caches
- Applications that perform real-time processing of unstructured big data,
- Apache Hadoop or Apache Spark clusters
- Other enterprise applications

For more information about each instance type, see *Amazon EC2 Instance Types* at https://aws.amazon.com/ec2/instance-types/.



Instance type networking features

- The network bandwidth (Gbps) varies by instance type.
  - For more information about comparing instance types, see Amazon EC2 Instance Types.
- To maximize networking and bandwidth performance of your instance type, take the following actions:
  - If you have interdependent instances, launch them into a **cluster placement group.**
  - Enable enhanced networking.
- Enhanced networking types are supported on most instance types.
  - For more information, see Networking and Storage Features.
- The following are enhanced networking types:
  - **Elastic Network Adapter (ENA)** supports network speeds of up to 100 Gbps.
  - **Intel 82599 Virtual Function interface** supports network speeds of up to 10 Gbps.

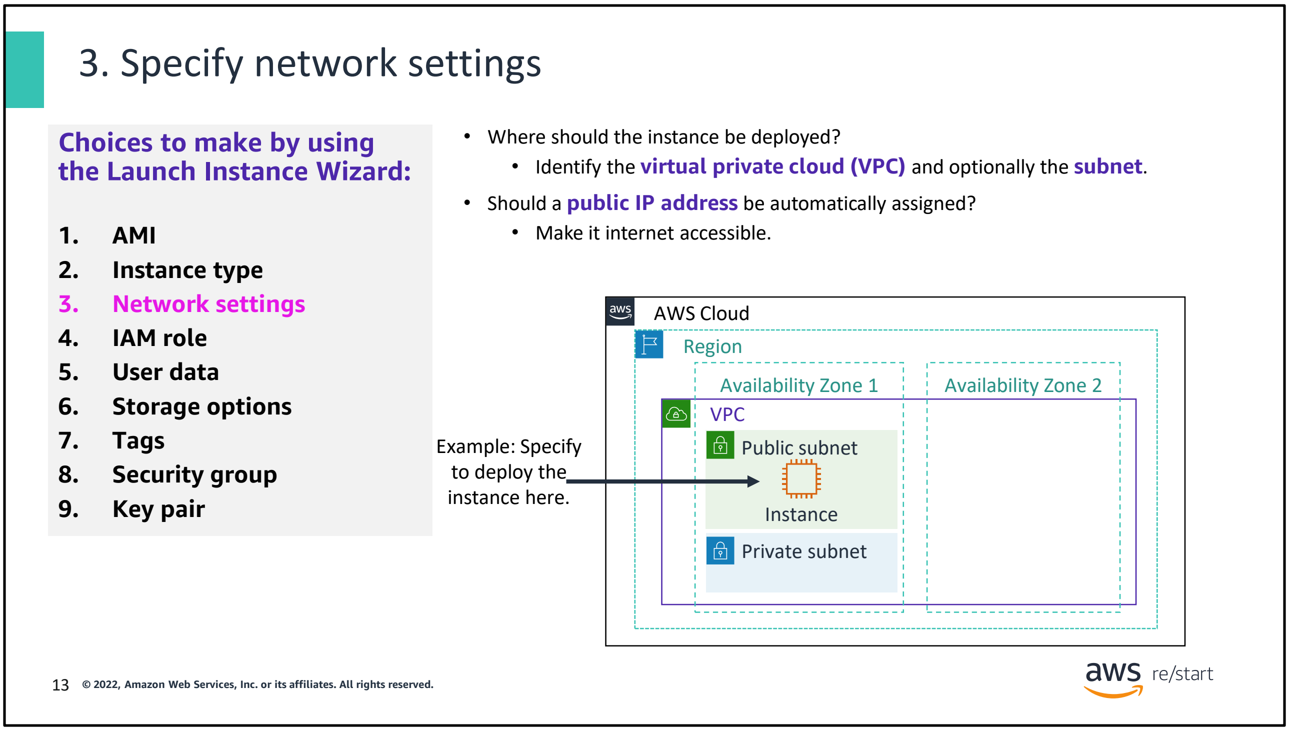
12

In addition to considering the CPU, RAM, and storage needs of your workloads, it is also important to consider your network bandwidth requirements.

Each instance type provides a documented network performance level. For example, an a1.medium instance will provide up to 10 Gbps, but a p3dn.24xlarge instance provides up to 100 Gbps. Choose an instance type that meets your requirements.

When you launch multiple new EC2 instances, Amazon EC2 attempts to place the instances so that they are spread out across the underlying hardware by default. It does so to minimize correlated failures. However, if you want to specify placement criteria, you can use placement groups to influence the placement of a group of interdependent instances to meet the needs of your workload. For example, you might specify that three instances should all be deployed in the same Availability Zone to help ensure lower network latency and higher network throughput between instances. For more information, see *Placement groups* at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html.

Many instance types also give you the ability to configure enhanced networking to get significantly higher packet per second (PPS) performance, lower delay variation in the arrival of packets over the network (network jitter), and lower latencies. For more information, see *Enable enhanced networking with the Elastic Network Adapter (ENA) on Linux instances* at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-ena.html.



3. Specify network settings

Choices to make by using the Launch Instance Wizard:

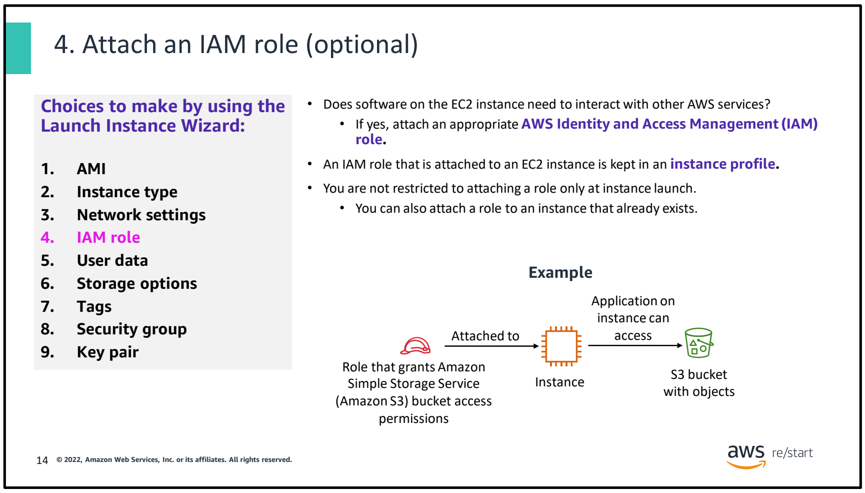
1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- Where should the instance be deployed?
  - Identify the **virtual private cloud (VPC)** and optionally the **subnet**.
- Should a **public IP address** be automatically assigned?
  - Make it internet accessible.

Example: Specify to deploy the instance here.

After you choose an AMI and an instance type, you must specify the network location where the EC2 instance will be deployed. You must choose the Region before you start the Launch Instance Wizard. Verify that you are in the correct Region page of the Amazon EC2 console before you choose **Launch Instance.**

When you launch an instance in a default virtual private cloud (VPC), AWS will assign it a public IP address by default. When you launch an instance into a nondefault VPC, the subnet has an attribute that determines whether instances that are launched into that subnet receive a public IP address from the public IPv4 address pool. By default, AWS will not assign a public IP address to instances that are launched in a nondefault subnet. You can control whether your instance receives a public IP address in two ways. First, you can modify the public IP addressing attribute of your subnet. Second, you can also enable or disable the public IP addressing feature during launch (which overrides the subnet's public IP addressing attribute).



4. Attach an IAM role (optional)

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- Does software on the EC2 instance need to interact with other AWS services?
  - If yes, attach an appropriate **AWS Identity and Access Management (IAM) role.**
- An IAM role that is attached to an EC2 instance is kept in an **instance profile.**
- You are not restricted to attaching a role only at instance launch.
  - You can also attach a role to an instance that already exists.

Example

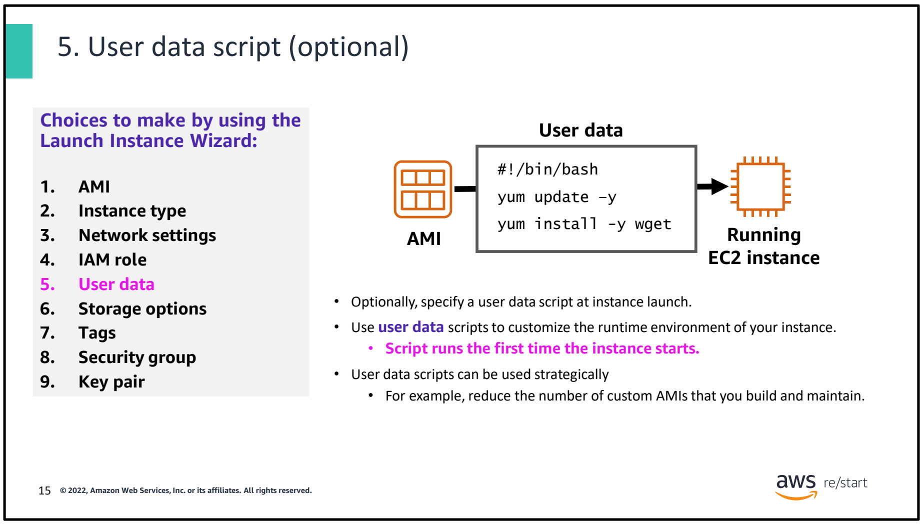
Attached to
Application on instance can access
Role that grants Amazon Simple Storage Service (Amazon S3) bucket access permissions
Instance
S3 bucket with objects

It is common to use EC2 instances to run an application that must make secure application programming interface (API) calls to other AWS services. To support these use cases, AWS gives you the ability to attach an AWS Identity and Access Management (IAM) role to an EC2 instance. Without this feature, you might be tempted to place AWS credentials on an EC2 instance for an application on that instance to use. However, you should never store AWS credentials on an EC2 instance. This practice is highly insecure. Instead, attach an IAM role to the EC2 instance. The IAM role then grants permissions to make API requests to the applications that run on the EC2 instance.

An instance profile is a container for an IAM role. If you use the AWS Management Console to create a role for Amazon EC2, the console automatically creates an instance profile and gives it the same name as the role. When you use the Amazon EC2 console to launch an instance with an IAM role, you can select a role to associate with the instance. In the console, the list that displays is actually a list of instance profile names.

In the example, an IAM role is used to grant permissions to an application that runs on an EC2 instance. The application must access a bucket in Amazon Simple Storage Service (Amazon S3).

You can attach an IAM role when you launch the instance, but you can also attach a role to an already-running EC2 instance. When you define a role that can be used by an EC2 instance, you define which accounts or AWS services can assume the role. You also define which API actions and resources the application can use after it assumes the role. If you change a role, the change is propagated to all instances that have the role attached to them.



5. User data script (optional)

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

User data

```
#!/bin/bash
yum update -y
yum install -y wget
```

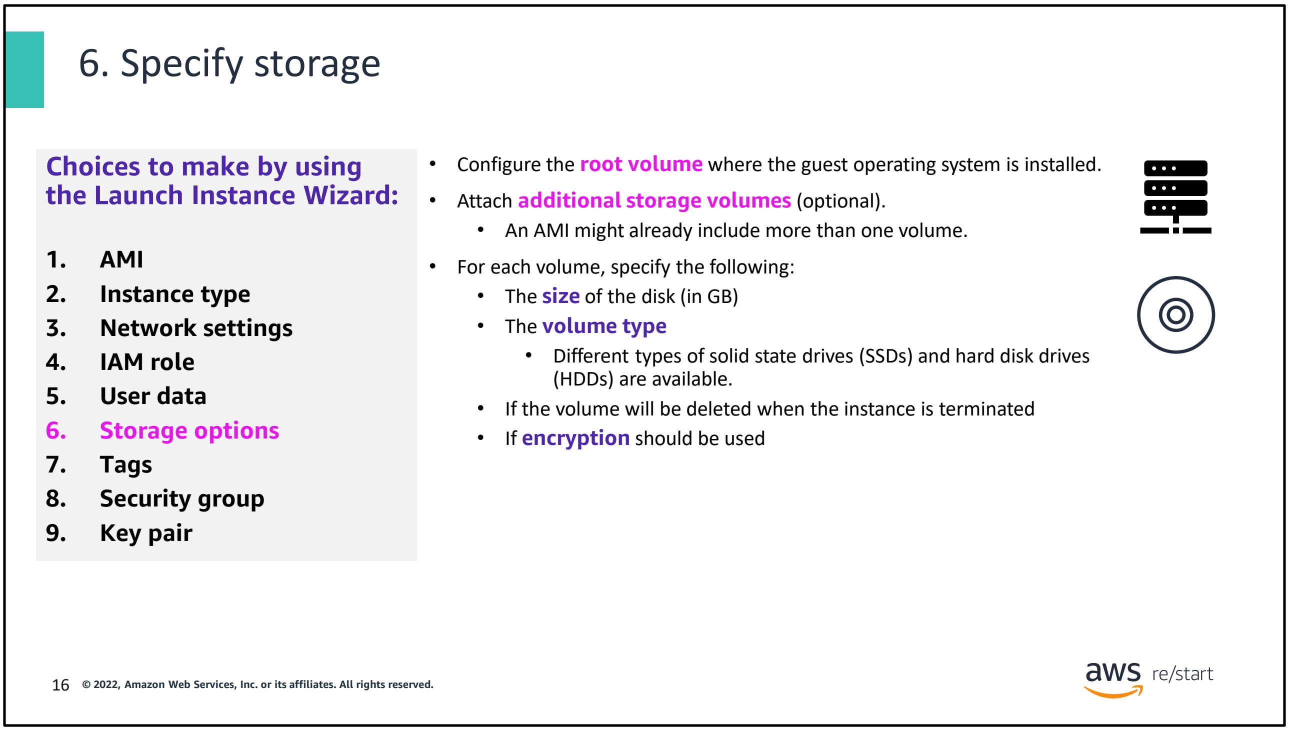
- Optionally, specify a user data script at instance launch.
- Use **user data** scripts to customize the runtime environment of your instance.
  - **Script runs the first time the instance starts.**
- User data scripts can be used strategically
  - For example, reduce the number of custom AMIs that you build and maintain.

When you create your EC2 instances, you have the option of passing user data to the instance. User data can automate the completion of installations and configurations at instance launch. For example, a user data script might patch and update the instance's operating system, fetch and install software license keys, or install additional software.

In the example user data script, you see a three-line Linux Bash shell script. The first line indicates that the script should be run by the Bash shell. The second line invokes the Yellowdog Updater, Modified (YUM) utility to retrieve software from an online repository and install it. A YUM utility is commonly used in many Linux distributions, such as Amazon Linux, CentOS, and Red Hat Linux. In line two of the example, that command tells YUM to update all installed packages to the latest versions that are known to the software repository that it's configured to access. Line three of the script indicates that the Wget utility should be installed. Wget is a common utility for downloading files from the web.

For a Microsoft Windows instance, the user data script should be written in a format that is compatible with a Command Prompt window (batch commands) or with Windows PowerShell. For more information, see *User data scripts* at https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ec2-windows-user-data.html#user-data-scripts.

When the EC2 instance is created, the user data script will run with AWS account root user privileges during the final phases of the boot process. On Linux instances, it is run by the cloud-init service. On Microsoft Windows instances, it is run by the EC2Config or EC2Launch utility. By default, user data runs only the first time that the instance starts. However, if you want your user data script to run every time the instance is booted, you can create a Multipurpose Internet Mail Extensions (MIME) multipart file user data script. (This process isn't common.) For more information, see *How can I utilize user data to automatically run a script with every restart of my Amazon EC2 Linux instance* at https://aws.amazon.com/premiumsupport/knowledge-center/execute-user-data-ec2/.



6. Specify storage

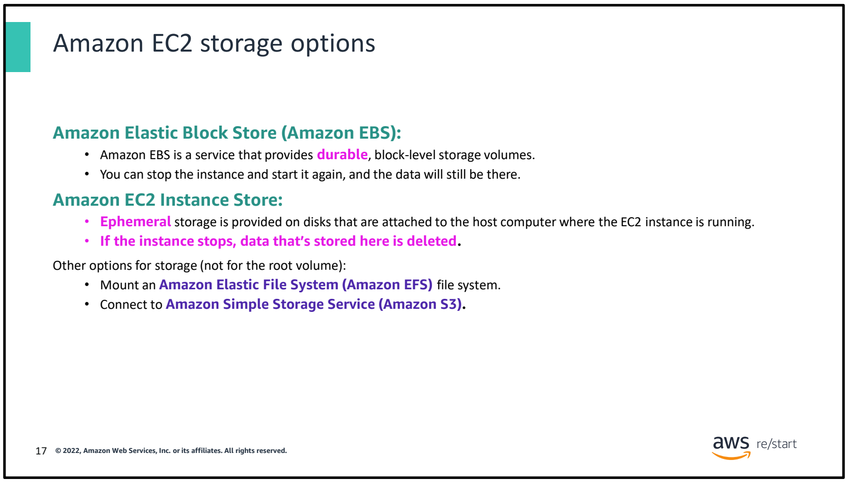
Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- Configure the **root volume** where the guest operating system is installed.
- Attach **additional storage volumes** (optional).
  - An AMI might already include more than one volume.
- For each volume, specify the following:
  - The **size** of the disk (in GB)
  - The **volume type**
    - Different types of solid state drives (SSDs) and hard disk drives (HDDs) are available.
  - If the volume will be deleted when the instance is terminated
  - If **encryption** should be used

When you launch an EC2 instance, you can configure storage options. For example, you can configure the size of the root volume where the guest OS is installed. You can also attach additional storage volumes when you launch the instance. Some AMIs are also configured to launch more than one storage volume by default to provide storage that is separate from the root volume.

For each volume that your instance will have, you can specify the size of the disks, the volume types, and whether the storage will be retained if the instance is terminated. You can also specify if encryption should be used.



Amazon EC2 storage options

Amazon Elastic Block Store (Amazon EBS):

- Amazon EBS is a service that provides **durable**, block-level storage volumes.
- You can stop the instance and start it again, and the data will still be there.

Amazon EC2 Instance Store:

- **Ephemeral** storage is provided on disks that are attached to the host computer where the EC2 instance is running.
- **If the instance stops, data that's stored here is deleted.**

Other options for storage (not for the root volume):

- Mount an **Amazon Elastic File System (Amazon EFS)** file system.
- Connect to **Amazon Simple Storage Service (Amazon S3).**

aws re/start

Amazon Elastic Block Store (Amazon EBS) is a high-performance durable block storage service that is designed to be used with Amazon EC2. It is used for both throughput-intensive and transaction-intensive workloads. With Amazon EBS, you can choose from four different volume types to balance the optimal price and performance. You can change volume types or increase volume size without disrupting your critical applications, so you can have cost-effective storage when you need it.

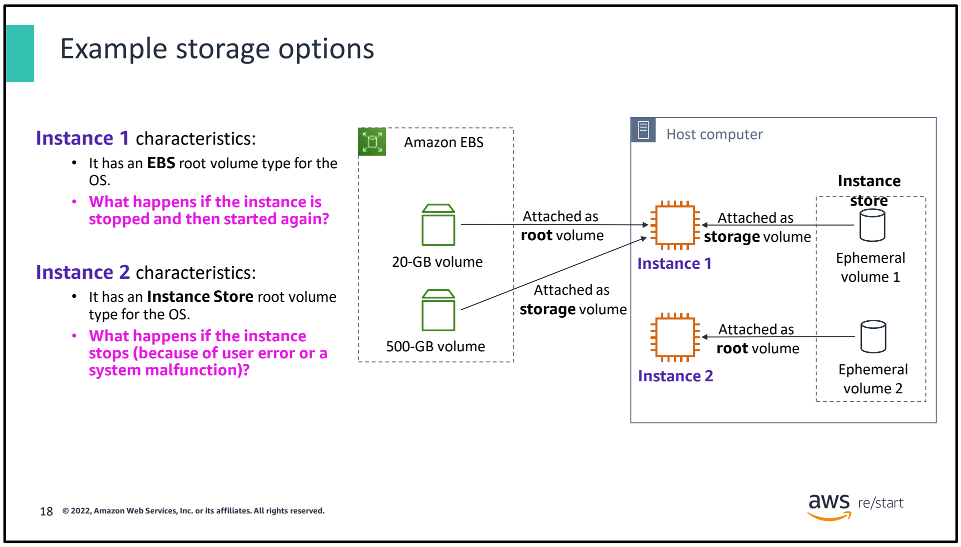
Amazon EC2 Instance Store provides ephemeral (or temporary) block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance Store works well when you must temporarily store information that changes frequently, such as buffers, caches, scratch data, and other temporary content. You can also use Instance Store for data that is replicated across a fleet of instances, such as a load balanced pool of web servers. If the instances are stopped—either because of user error or a malfunction—the data on the instance store will be deleted.

Amazon Elastic File System (Amazon EFS) provides a scalable, fully managed elastic Network File System (NFS) file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications. It grows and shrinks automatically as you add and remove files, which reduces the need to provision and manage capacity to accommodate growth.

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. You can store and protect any amount of data for a variety of use cases, such as the following:

- Websites
- Mobile apps
- Backup and restoration
- Archiving

- Enterprise applications
- Internet of Things (IoT) devices
- Big data analytics



Example storage options

Instance 1 characteristics:

- It has an **EBS** root volume type for the OS.
- **What happens if the instance is stopped and then started again?**

Instance 2 characteristics:

- It has an **Instance Store** root volume type for the OS.
- **What happens if the instance stops (because of user error or a system malfunction)?**

Amazon EBS

20-GB volume

500-GB volume

Attached as **root** volume

Attached as **storage** volume

Host computer

Instance 1

Attached as **storage** volume

Instance store

Ephemeral volume 1

Instance 2

Attached as **root** volume

Ephemeral volume 2

18

aws re/start

These two examples illustrate two different storage configurations for EC2 instances.

The Instance 1 example shows that the root volume—which contains the OS and possibly other data—is stored on Amazon EBS. This instance also has two attached volumes. One volume is a 500-GB EBS storage volume, and the other volume is an Instance Store volume. If this instance was stopped and then started again, the OS would survive. Any data that was stored on either the 20-GB EBS volume or the 500-GB EBS volume would remain intact. However, any data that was stored on Ephemeral volume 1 would be permanently lost. Recall that the Instance Store works well for temporarily storing information that changes frequently (such as buffers, caches, scratch data, and other temporary content).

The Instance 2 example shows that the root volume is on an instance store (Ephemeral volume 2). An instance with an Instance Store root volume cannot be stopped by an Amazon EC2 API call. It can only be terminated. However, it could be stopped from within the instance's OS (for example, by issuing a shutdown command). It could also stop because of OS or disk failure, which would cause the instance to be terminated. If the instance is terminated, all the data that was stored on Ephemeral volume 2 would be lost, including the OS. You wouldn’t be able to start the instance again. Thus, don’t rely on Instance Store for valuable, long-term data. Instead, use more durable data storage, such as Amazon EBS, Amazon EFS, or Amazon S3.

If an instance reboots (intentionally or unintentionally), data on the instance store root volume does persist.

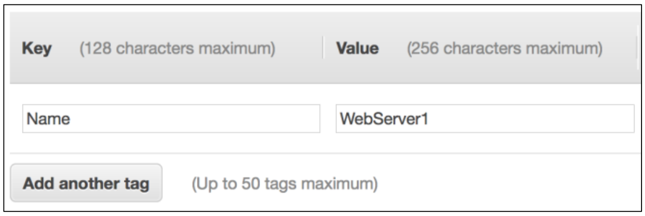
7. Add tags

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- A **tag** is a label that you can assign to an AWS resource.
  - It consists of a *key* and an optional *value*.
- Tagging is how you can attach **metadata** to an EC2 instance.
- Potential benefits of tagging include filtering, automation, cost allocation, and access control.

Example:



Key (128 characters maximum)	Value (256 characters maximum)
Name	WebServer1

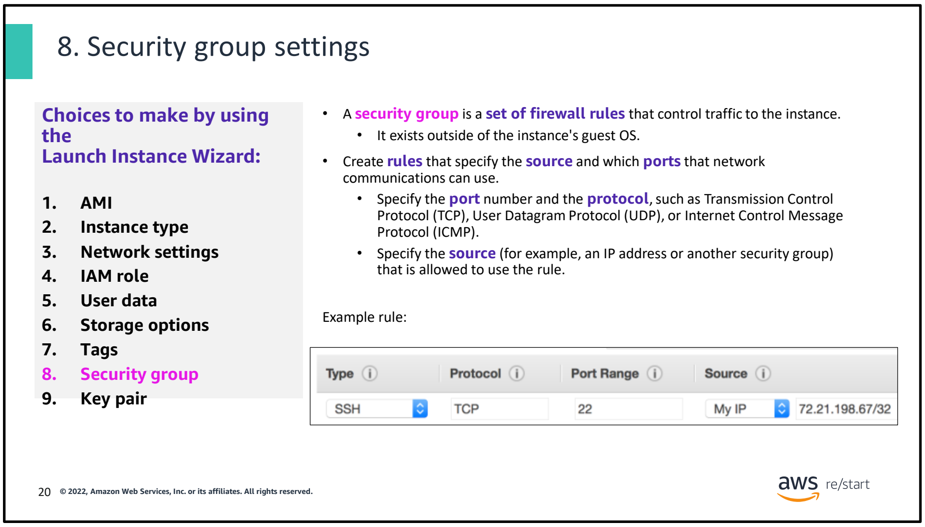
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value, both of which you define. With tags, you can categorize AWS resources (such as EC2 instances) in different ways. For example, you might tag instances by purpose, owner, or environment.

You can use tagging to attach metadata to an EC2 instance.

Tag keys and tag values are case sensitive. For example, a commonly used tag for EC2 instances is a tag key that's called Name and a tag value that describes the instance, such as My Web Server. The Name tag is exposed by default on the Amazon EC2 console Instances page. However, if you create a key that's called name (with lower-case n), it won't appear in the Name column for the list of instances. (However, it will still appear in the instance details panel in the Tags tab).

It's a best practice to develop tagging strategies. For more information, see *Tagging Best Practices* at https://d1.awsstatic.com/whitepapers/aws-tagging-best-practices.pdf.

A consistent set of tag keys facilitates effective management of resources. You can also search and filter the resources based on the tags that you add.



8. Security group settings

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- A **security group** is a **set of firewall rules** that control traffic to the instance.
  - It exists outside of the instance's guest OS.
- Create **rules** that specify the **source** and which **ports** that network communications can use.
  - Specify the **port** number and the **protocol**, such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or Internet Control Message Protocol (ICMP).
  - Specify the **source** (for example, an IP address or another security group) that is allowed to use the rule.

Example rule:

Type	Protocol	Port Range	Source	
SSH	TCP	22	My IP	72.21.198.67/32

20 © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

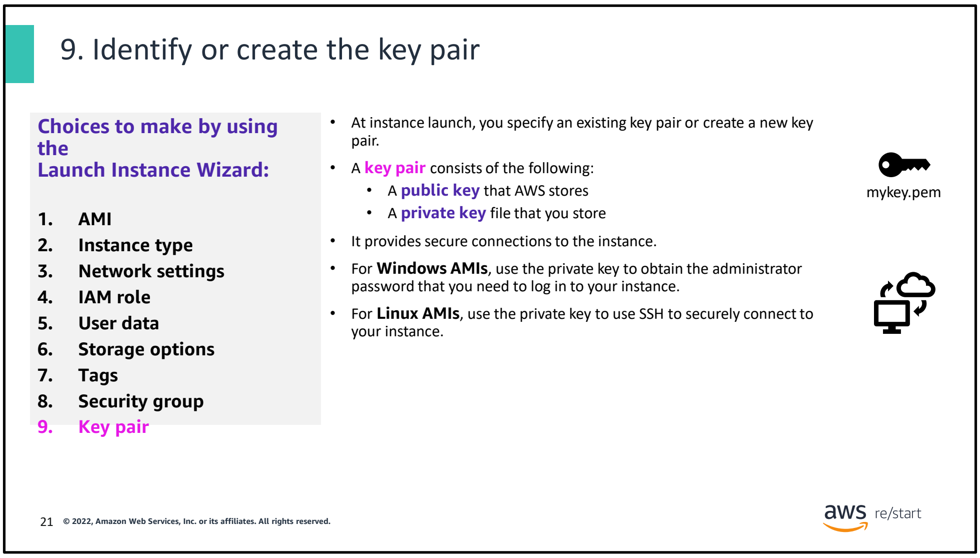
A security group acts as a virtual firewall that controls network traffic for one or more instances. When you launch an instance, you can specify one or more security groups. Otherwise, the default security group is used.

You can add rules to each security group. Rules allow traffic to or from the security group's associated instances. You can modify the rules for a security group at any time. The new rules will be automatically applied to all instances that are associated with the security group. When AWS decides whether to allow traffic to reach an instance, all the rules from all the security groups that are associated with the instance are evaluated. When you launch an instance in a VPC, you must either create a new security group or use one that already exists in that VPC. After you launch an instance, you can change its security groups.

When you define a rule, you can specify the allowable source of the network communication (inbound rules) or destination (outbound rules). The source can be an IP address, an IP address range, another security group, a gateway VPC endpoint, or anywhere (which means that all sources will be allowed). By default, a security group includes an outbound rule that allows all outbound traffic. You can remove the rule and add outbound rules that only allow specific outbound traffic. If your security group has no outbound rules, no outbound traffic that originates from your instance is allowed.

In the example rule, the rule allows Secure Shell (SSH) traffic over Transmission Control Protocol (TCP) port 22 if the source of the request is My IP. The My IP IP address is calculated when you define the rule by determining the IP address that you are using to connect to the AWS Cloud.

Network access control lists (network ACLs) can also be used as firewalls to protect subnets in a VPC.



9. Identify or create the key pair

Choices to make by using the Launch Instance Wizard:

1. **AMI**
2. **Instance type**
3. **Network settings**
4. **IAM role**
5. **User data**
6. **Storage options**
7. **Tags**
8. **Security group**
9. **Key pair**

- At instance launch, you specify an existing key pair or create a new key pair.
- A **key pair** consists of the following:
  - A **public key** that AWS stores
  - A **private key** file that you store
- It provides secure connections to the instance.
- For **Windows AMIs**, use the private key to obtain the administrator password that you need to log in to your instance.
- For **Linux AMIs**, use the private key to use SSH to securely connect to your instance.

mykey.pem

21

After you specify all the required configurations to launch an EC2 instance—and after you customize any optional EC2 launch wizard configuration settings—the Review Instance Launch window opens. If you then choose Launch, a dialog box appears. It asks you to choose an existing key pair, proceed without a key pair, or create a new key pair, before you can choose Launch Instances and create the EC2 instance.

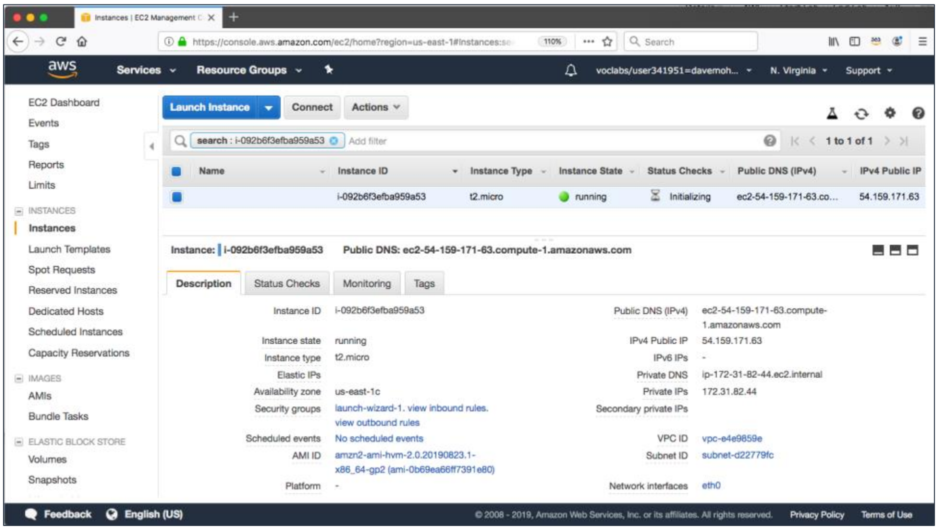
Amazon EC2 uses public key cryptography to encrypt and decrypt login information. The technology uses a public key to encrypt a piece of data. Then the recipient uses the private key to decrypt the data. The public and private keys are known as a key pair. Public key cryptography makes it possible for you to securely access your instances by using a private key instead of a password.

When you launch an instance, you specify a key pair. You can specify an existing key pair or a new key pair that you create at launch. If you create a new key pair, download it and save it in a safe location. This opportunity is the only chance that you get to save the private key file.

To connect to a Microsoft Windows instance, use the private key to obtain the administrator password. Then log in to the EC2 instance's Windows Desktop by using Remote Desktop Protocol (RDP). To establish an SSH connection from a Windows machine to an EC2 instance, you can use a tool such as PuTTY, which will require the same private key.

With Linux instances, the public key content is placed on the instance at boot time. An entry is created in `~/.ssh/authorized_keys`. To log in to your Linux instance (for example, by using SSH), you must provide the private key when you establish the connection.

Amazon EC2 console view of a running EC2 instance



Name	Instance ID	Instance Type	Instance State	Status Checks	Public DNS (IPv4)	IPv4 Public IP
	i-092b6f3efba959a53	t2.micro	● running	Initializing	ec2-54-159-171-63.co...	54.159.171.63

Instance: i-092b6f3efba959a53 **Public DNS:** ec2-54-159-171-63.compute-1.amazonaws.com

Instance ID	i-092b6f3efba959a53	Public DNS (IPv4)	ec2-54-159-171-63.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	54.159.171.63
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-82-44.ec2.internal
Availability zone	us-east-1c	Private IPs	172.31.82.44
Security groups	launch-wizard-1. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-e4e9859e
AMI ID	amzn2-ami-hvm-2.0.20190823.1-x86_64-gp2 (ami-0b69ea66ff7391e80)	Subnet ID	subnet-d22779fc
Platform	-	Network interfaces	eth0

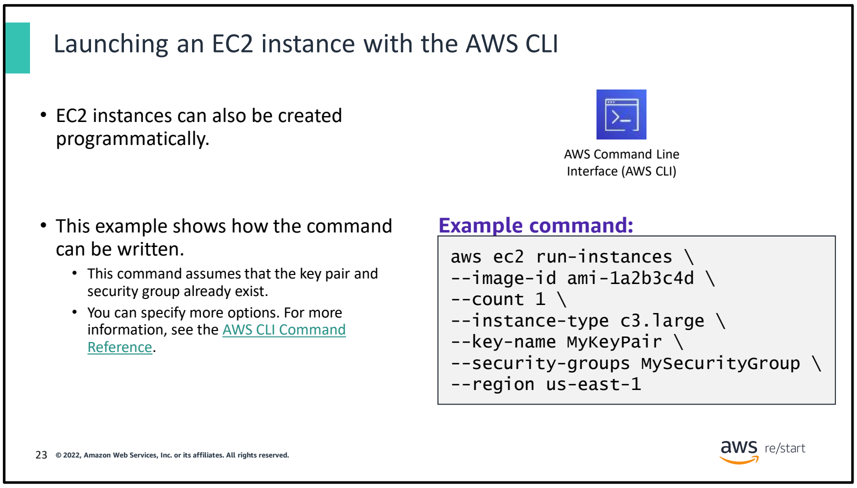
After you choose **Launch Instances** and then choose **View Instances**, a screen opens. It looks similar to the example.

Many of the settings that you specified during launch are in the **Description** panel.

Information about the available instance includes the following:

- IP address and Domain Name System (DNS) address information
- The instance type
- The unique instance ID that was assigned to the instance
- The AMI ID of the AMI that you used to launch the instance
- The VPC ID
- The subnet ID

Many of these details provide hyperlinks. You can choose them to learn more information about the instance’s resources.



Launching an EC2 instance with the AWS CLI

- EC2 instances can also be created programmatically.

AWS Command Line Interface (AWS CLI)

- This example shows how the command can be written.
  - This command assumes that the key pair and security group already exist.
  - You can specify more options. For more information, see the AWS CLI Command Reference.

Example command:

```
aws ec2 run-instances \
--image-id ami-1a2b3c4d \
--count 1 \
--instance-type c3.large \
--key-name MyKeyPair \
--security-groups MySecurityGroup \
--region us-east-1
```

23

aws re/start

You can also launch EC2 instances programmatically either by using the AWS Command Line Interface (AWS CLI) or one of the AWS software development kits (SDKs).

In the example AWS CLI command, you see a single command that specifies the minimal information that is needed to launch an instance. The command includes the following information:

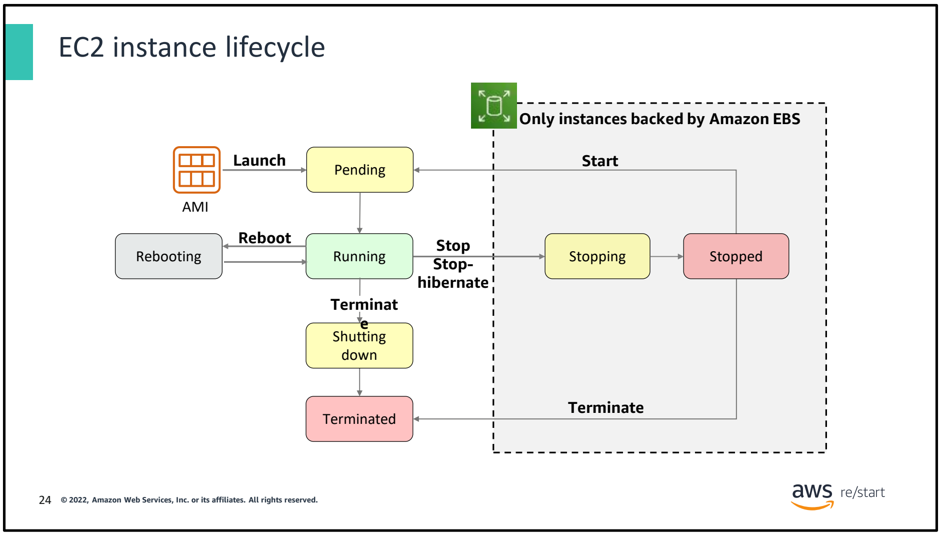
- `aws` – Specifies an invocation of the AWS CLI
- `ec2` – Specifies an invocation of the Amazon EC2 service command
- `run-instances` – The subcommand that is being invoked

The rest of the command specifies several parameters, including the following:

- `image-id` – This parameter is followed by an AMI ID. All AMIs have a unique AMI ID.
- `count` – You can specify more than one.
- `instance-type` – You can specify the instance type to create (for example a c3.large instance).
- `key-name` – In the example, assume that MyKeyPair already exists.
- `security-groups` – In this example, assume that MySecurityGroup already exists.
- `region` – AMIs exist in an AWS Region, so you must specify the Region where the AWS CLI will find the AMI and launch the EC2 instance.

The command should successfully create an EC2 instance if the command is properly formed, the resources that the command needs already exist, you have sufficient permissions to run the command, and you have sufficient capacity in the AWS account.

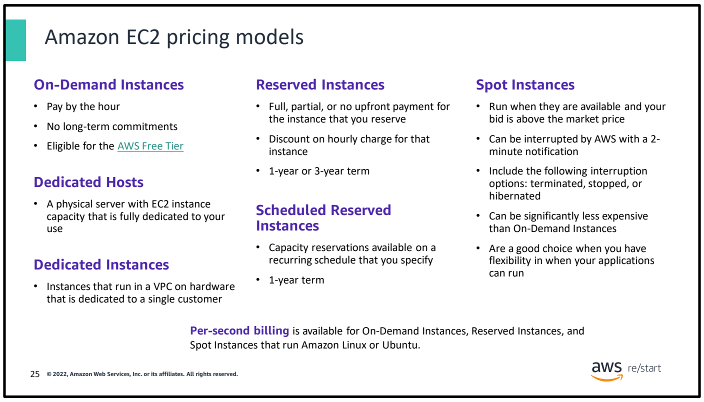
If the command is successful, the API responds to the command with the instance ID and other relevant data for your application to use in subsequent API requests.



EC2 instance lifecycle

This diagram illustrates the lifecycle of an instance. The arrows show actions that you can take. The boxes show the state the instance will enter after that action. An instance can be in one of the following states:

- Pending – When an instance is first launched from an AMI, or when you start a stopped instance, it enters the pending state when the instance is booted and deployed to a host computer. The instance type that you specified at launch determines the hardware of the host computer for your instance.
- Running – When the instance is fully booted and ready, it exits the pending state and enters the running state. You can connect over the internet to your running instance.
- Rebooting – Instead of invoking a reboot from the instance's guest OS, AWS recommends you reboot the instance by using the Amazon EC2 console, the AWS CLI, or AWS SDKs. A rebooted instance stays on the same physical host, and it maintains the same public DNS name and public IP address. If the instance has instance store volumes, it retains the data on those volumes.
- Shutting down – This state is an intermediate state between running and terminated.
- Terminated – A terminated instance remains in the Amazon EC2 console for some time before the virtual machine is deleted. However, you can't connect to or recover a terminated instance.
- Stopping – Instances that are backed by Amazon EBS can be stopped. They enter the stopping state before they attain the fully stopped state.
- Stopped – A stopped instance doesn't incur the same cost as a running instance. Starting a stopped instance puts it back into the pending state, which moves the instance to a new host machine.



Amazon EC2 pricing models

On-Demand Instances

- Pay by the hour
- No long-term commitments
- Eligible for the AWS Free Tier

Dedicated Hosts

- A physical server with EC2 instance capacity that is fully dedicated to your use

Dedicated Instances

- Instances that run in a VPC on hardware that is dedicated to a single customer

Reserved Instances

- Full, partial, or no upfront payment for the instance that you reserve
- Discount on hourly charge for that instance
- 1-year or 3-year term

Scheduled Reserved Instances

- Capacity reservations available on a recurring schedule that you specify
- 1-year term

Spot Instances

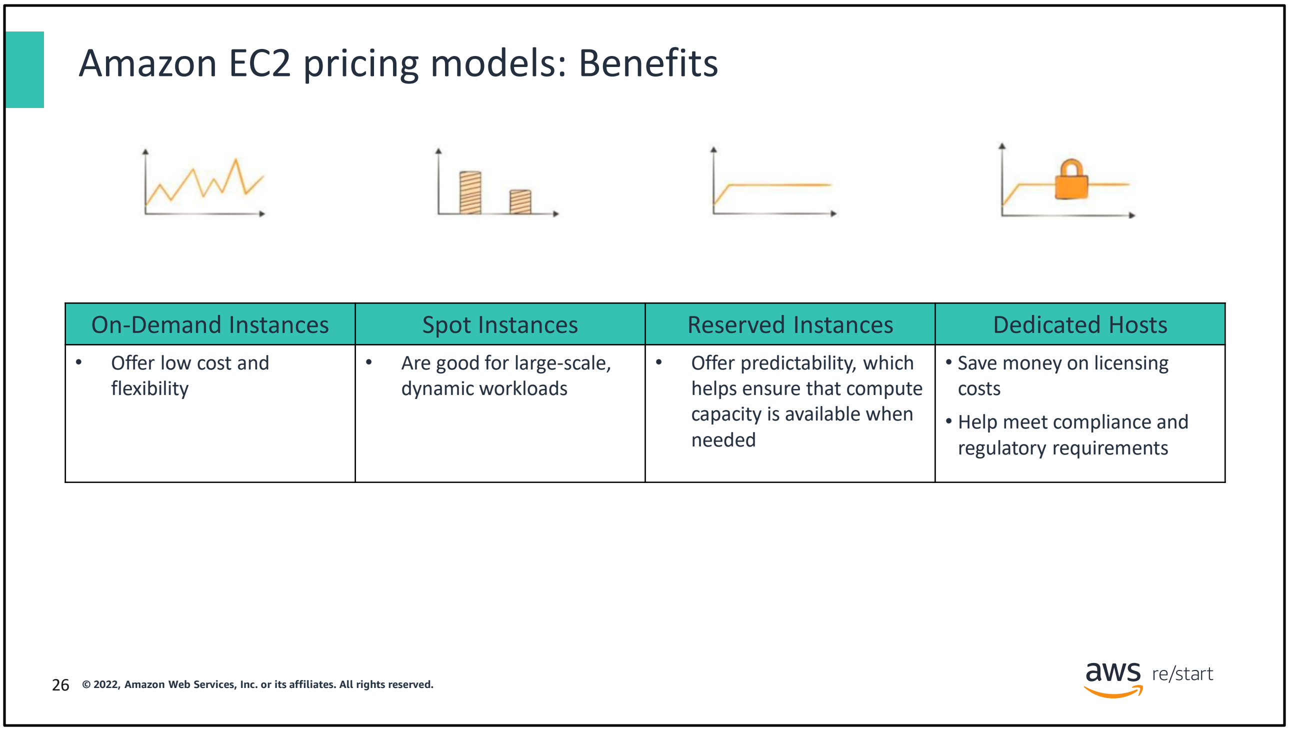
- Run when they are available and your bid is above the market price
- Can be interrupted by AWS with a 2-minute notification
- Include the following interruption options: terminated, stopped, or hibernated
- Can be significantly less expensive than On-Demand Instances
- Are a good choice when you have flexibility in when your applications can run

Per-second billing is available for On-Demand Instances, Reserved Instances, and Spot Instances that run Amazon Linux or Ubuntu.

25

Amazon offers different pricing models to choose from when you want to run EC2 instances:

- Per-second billing is available only for On-Demand Instances, Reserved Instances, and Spot Instances that run Amazon Linux or Ubuntu.
- On-Demand Instances are eligible for the AWS Free Tier. They have the lowest upfront cost and the most flexibility, with no upfront commitments or long-term contracts. They are a good choice for applications with short-term, spiky, or unpredictable workloads. For more information about the AWS Free Tier, see https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all.
- Dedicated Hosts are physical servers with instance capacity that's dedicated to your use. With a Dedicated Host, you can use your existing per-socket, per-core, or per-VM software licenses, such as for Microsoft Windows or Microsoft SQL Server.
- Dedicated Instances are instances that run in a VPC on hardware that's dedicated to a single customer. They are physically isolated from instances that belong to other AWS accounts at the level of the host hardware.
- Reserved Instances give you the ability to reserve computing capacity for a 1-year or 3-year term, with lower hourly running costs. The discounted usage price is fixed for as long as you own the Reserved Instance. If you expect consistent and heavy use, they can provide substantial savings compared to On-Demand Instances.
- Scheduled Reserved Instances give you the ability to purchase capacity reservations that recur on a daily, weekly, or monthly basis—with a specified duration—for a 1-year term. You pay for the time that the instances are scheduled even if you don't use them.
- Spot Instances give you the ability to bid on unused EC2 instances, which can lower your costs. The hourly price for a Spot Instance fluctuates depending on supply and demand. Your Spot Instance runs whenever your bid exceeds the current market price.



Amazon EC2 pricing models: Benefits

On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Offer low cost and flexibility	• Are good for large-scale, dynamic workloads	• Offer predictability, which helps ensure that compute capacity is available when needed	• Save money on licensing costs • Help meet compliance and regulatory requirements

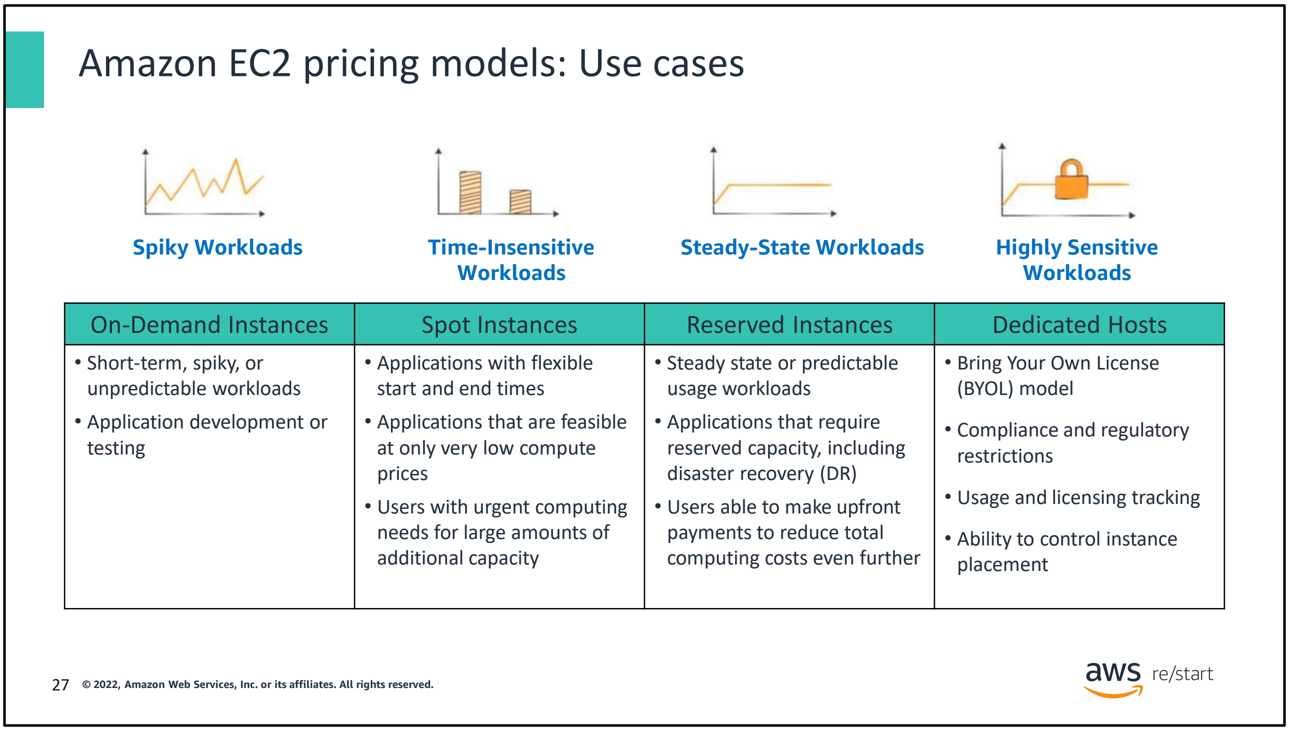
Each Amazon EC2 pricing model provides a different set of benefits.

On-Demand Instances offer the most flexibility, with no long-term contract and low rates.

Spot Instances provide large scale at a significantly discounted price.

Reserved Instances are a good choice if you have predictable or steady-state compute needs. An example would be an instance that you know you want to keep running most or all the time for months or years.

Dedicated Hosts are a good choice when you have licensing restrictions for the software that you want to run on Amazon EC2 or when you have specific compliance or regulatory requirements that prevent you from using the other deployment options.



Amazon EC2 pricing models: Use cases

Spiky Workloads	Time-Insensitive Workloads	Steady-State Workloads	Highly Sensitive Workloads
On-Demand Instances	**Spot Instances**	**Reserved Instances**	**Dedicated Hosts**
• Short-term, spiky, or unpredictable workloads • Application development or testing	• Applications with flexible start and end times • Applications that are feasible at only very low compute prices • Users with urgent computing needs for large amounts of additional capacity	• Steady state or predictable usage workloads • Applications that require reserved capacity, including disaster recovery (DR) • Users able to make upfront payments to reduce total computing costs even further	• Bring Your Own License (BYOL) model • Compliance and regulatory restrictions • Usage and licensing tracking • Ability to control instance placement

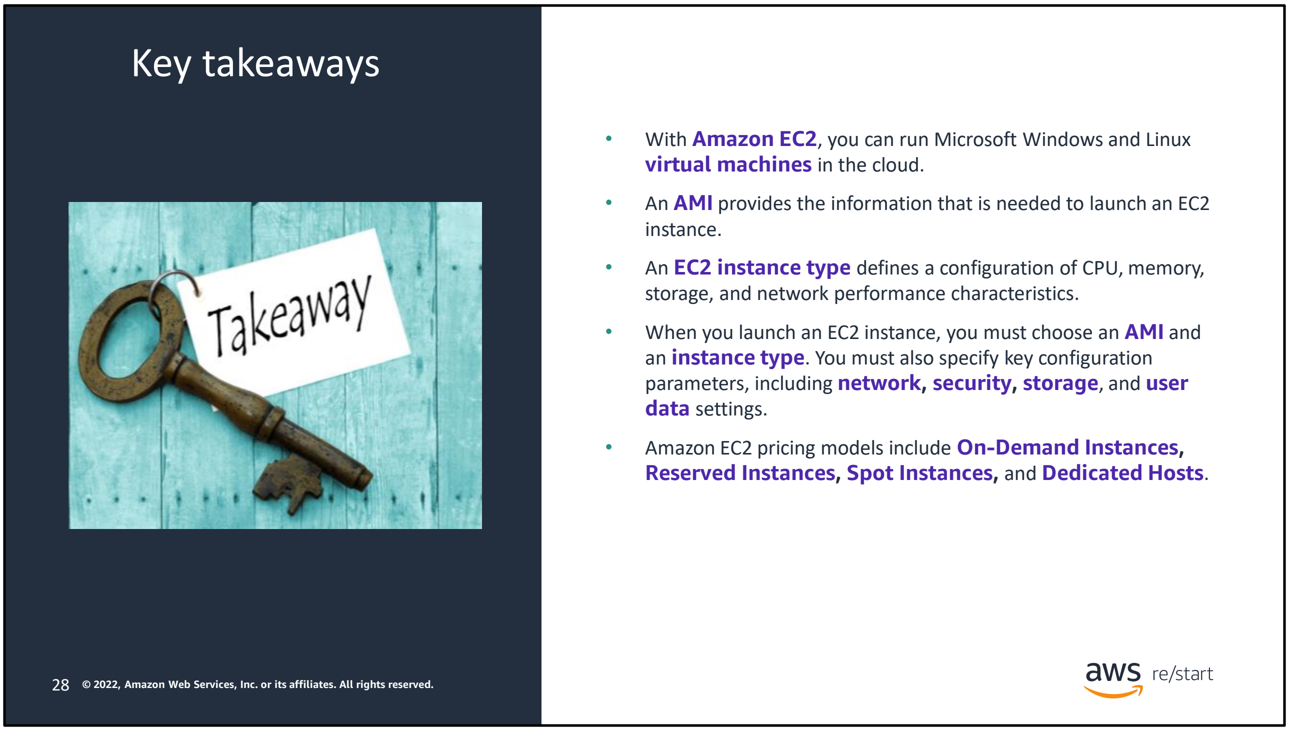
Here is a review of some use cases for the various pricing options.

On-Demand Instance pricing works well for spiky workloads, or if you need to test or run an application for only a short time (for example, during application development or testing). Sometimes, your workloads are unpredictable, and On-Demand Instances are a good choice for these cases.

Spot Instances are a good choice if your applications can tolerate interruption with a 2-minute warning notification. By default, instances are terminated, but you can configure them to stop or hibernate instead. Common use cases include fault-tolerant applications such as web servers, API backends, and big data processing. Workloads that constantly save data to persistent storage (such as Amazon S3) are also good candidates.

Reserved Instances are a good choice when you have long-term workloads with predictable usage patterns, such as servers that you want to run in a consistent way over many months.

Dedicated Hosts are a good choice when you have existing per-socket, per-core, or per-VM software licenses, or when you must address specific corporate compliance and regulatory requirements.

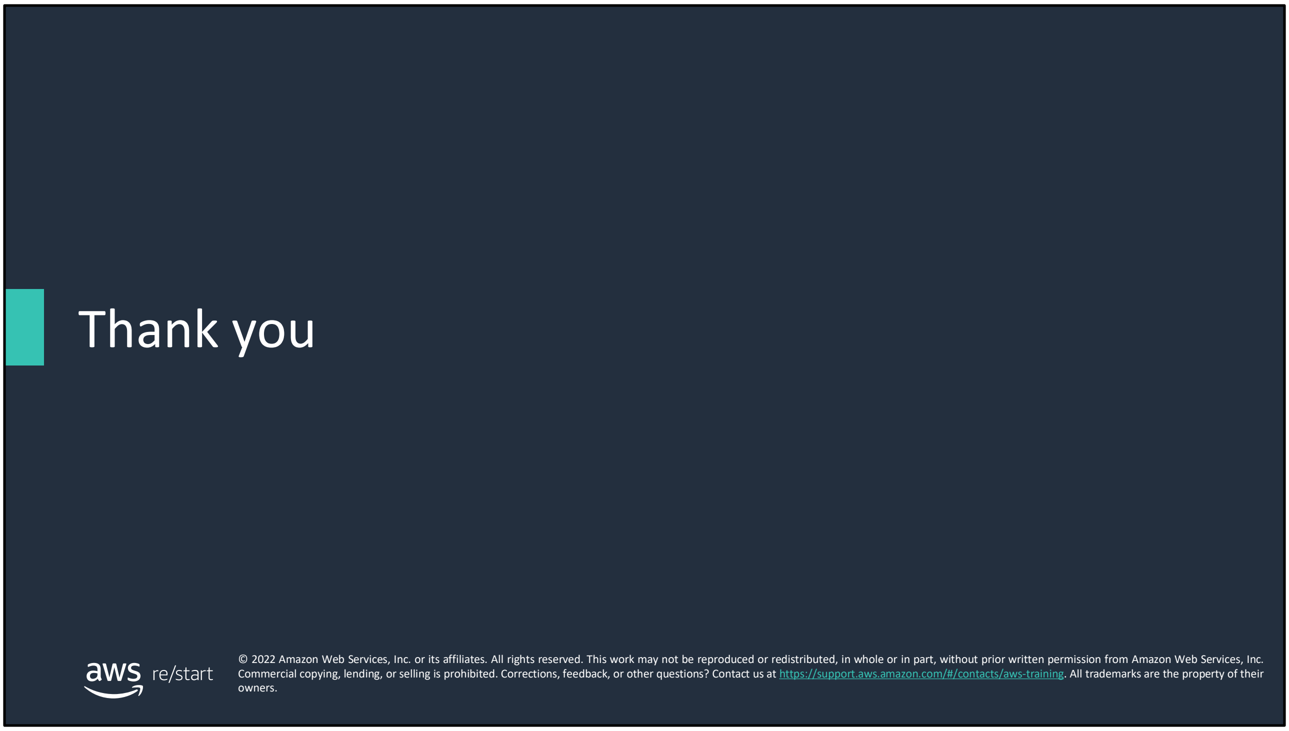


Key takeaways

- With **Amazon EC2**, you can run Microsoft Windows and Linux **virtual machines** in the cloud.
- An **AMI** provides the information that is needed to launch an EC2 instance.
- An **EC2 instance type** defines a configuration of CPU, memory, storage, and network performance characteristics.
- When you launch an EC2 instance, you must choose an **AMI** and an **instance type**. You must also specify key configuration parameters, including **network, security, storage**, and **user data** settings.
- Amazon EC2 pricing models include **On-Demand Instances, Reserved Instances, Spot Instances,** and **Dedicated Hosts**.

This module includes the following key takeaways:

- With Amazon EC2, you can run Microsoft Windows and Linux virtual machines in the cloud.
- An AMI provides the information that’s needed to launch an EC2 instance.
- An EC2 instance type defines a configuration of CPU, memory, storage, and network performance characteristics.
- When you launch an EC2 instance, you must choose an AMI and an instance type. You must also specify key configuration parameters, including network, security, storage, and user data settings.
- Amazon EC2 pricing models include On-Demand Instances, Reserved Instances, Spot Instances, and Dedicated Hosts.



Thank you

Thank you for completing this module.