**Module 1 introduction**

**Learning objectives**

In this module, you will learn how to:

* Summarize the benefits of AWS.
* Describe differences between on-demand delivery and cloud deployments.
* Summarize the pay-as-you-go pricing model.

**What is a client-server model?**

You just learned more about AWS and how almost all modern computing uses a basic client-server model. Let’s recap what a client-server model is.



In computing, a**client** can be a web browser or desktop application that a person interacts with to make requests to computer servers. A **server** can be services such as Amazon Elastic Compute Cloud (Amazon EC2), a type of virtual server.

For example, suppose that a client makes a request for a news article, the score in an online game, or a funny video. The server evaluates the details of this request and fulfils it by returning the information to the client.

# Cloud computing

**Deployment models for cloud computing**

When selecting a cloud strategy, a company must consider factors such as required cloud application components, preferred resource management tools, and any legacy IT infrastructure requirements.

The three cloud computing deployment models are cloud-based, on-premises, and hybrid.

Select each tab to learn about each category.

Cloud based deployment:

* Run all parts of the application in the cloud.
* Migrate existing applications to the cloud.
* Design and build new applications in the cloud.

In a **cloud-based deployment** model, you can migrate existing applications to the cloud, or you can design and build new applications in the cloud. You can build those applications on low-level infrastructure that requires your IT staff to manage them. Alternatively, you can build them using higher-level services that reduce the management, architecting, and scaling requirements of the core infrastructure.

For example, a company might create an application consisting of virtual servers, databases, and networking components that are fully based in the cloud.

On-Premises Deployment:

* Deploy resources by using virtualization and resource management tools.
* Increase resource utilization by using application management and virtualization technologies.

**On-premises deployment**is also known as a *private cloud* deployment. In this model, resources are deployed on premises by using virtualization and resource management tools.

For example, you might have applications that run on technology that is fully kept in your on-premises data centre. Though this model is much like legacy IT infrastructure, its incorporation of application management and virtualization technologies helps to increase resource utilization.

**Hybrid Deployment:**

* Connect cloud-based resources to on-premises infrastructure.
* Integrate cloud-based resources with legacy IT applications.

In a **hybrid deployment**, cloud-based resources are connected to on-premises infrastructure. You might want to use this approach in a number of situations. For example, you have legacy applications that are better maintained on premises, or government regulations require your business to keep certain records on premises.

For example, suppose that a company wants to use cloud services that can automate batch data processing and analytics. However, the company has several legacy applications that are more suitable on premises and will not be migrated to the cloud. With a hybrid deployment, the company would be able to keep the legacy applications on premises while benefiting from the data and analytics services that run in the cloud.

**Benefits of cloud computing**

Consider why a company might choose to take a particular cloud computing approach when addressing business needs.

**Trade upfront expense for variable expense:**

Upfront expense refers to data centres, physical servers, and other resources that you would need to invest in before using them. Variable expense means you only pay for computing resources you consume instead of investing heavily in data centers and servers before you know how you’re going to use them.

By taking a cloud computing approach that offers the benefit of variable expense, companies can implement innovative solutions while saving on costs.

**Stop spending money to run and maintain data centers:**

Computing in data centers often requires you to spend more money and time managing infrastructure and servers.

A benefit of cloud computing is the ability to focus less on these tasks and more on your applications and customers.

**Stop guessing capacity:**

With cloud computing, you don’t have to predict how much infrastructure capacity you will need before deploying an application.

For example, you can launch Amazon EC2 instances when needed, and pay only for the compute time you use. Instead of paying for unused resources or having to deal with limited capacity, you can access only the capacity that you need. You can also scale in or scale out in response to demand.

**Benefit from massive economies of scale:**

By using cloud computing, you can achieve a lower variable cost than you can get on your own.

Because usage from hundreds of thousands of customers can aggregate in the cloud, providers, such as AWS, can achieve higher economies of scale. The economy of scale translates into lower pay-as-you-go prices.

**Increase speed and agility**

The flexibility of cloud computing makes it easier for you to develop and deploy applications.

This flexibility provides you with more time to experiment and innovate. When computing in data centers, it may take weeks to obtain new resources that you need. By comparison, cloud computing enables you to access new resources within minutes.

**Go global in minutes**

The global footprint of the AWS Cloud enables you to deploy applications to customers around the world quickly, while providing them with low latency. This means that even if you are located in a different part of the world than your customers, customers are able to access your applications with minimal delays.

Later in this course, you will explore the AWS global infrastructure in greater detail. You will examine some of the services that you can use to deliver content to customers around the world.

# Module 1 quiz

Test your knowledge of some of the key concepts from this module by answering the questions in this quiz.

After answering each question, review the detailed answer explanations to reinforce your understanding of the concepts.

What is cloud computing?

* Backing up files that are stored on desktop and mobile devices to prevent data loss
* Deploying applications connected to on-premises infrastructure
* Running code without needing to manage or provision servers
* **On-demand delivery of IT resources and applications through the internet with pay-as-you-go pricing**

What is another name for on-premises deployment?

* **Private cloud deployment**
* Cloud-based application
* Hybrid deployment
* AWS Cloud

How does the scale of cloud computing help you to save costs?

* You do not have to invest in technology resources before using them.
* **The aggregated cloud usage from a large number of customers results in lower pay-as-you-go prices.**
* Accessing services on-demand helps to prevent excess or limited capacity.
* You can quickly deploy applications to customers and provide them with low latency.

# Module 2 introduction(Compute in the cloud)

**Learning objectives**

In this module, you will learn how to:

* Describe the benefits of Amazon EC2 at a basic level.
* Identify the different Amazon EC2 instance types.
* Differentiate between the various billing options for Amazon EC2.
* Summarize the benefits of Amazon EC2 Auto Scaling.
* Summarize the benefits of Elastic Load Balancing.
* Give an example of the uses for Elastic Load Balancing.
* Summarize the differences between Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).
* Summarize additional AWS compute options.

**Amazon Elastic Compute Cloud (Amazon EC2)**

[Amazon Elastic Compute Cloud (Amazon EC2)](https://aws.amazon.com/ec2/) provides secure, resizable compute capacity in the cloud as Amazon EC2 instances.

Imagine you are responsible for the architecture of your company's resources and need to support new websites. With traditional on-premises resources, you have to do the following:

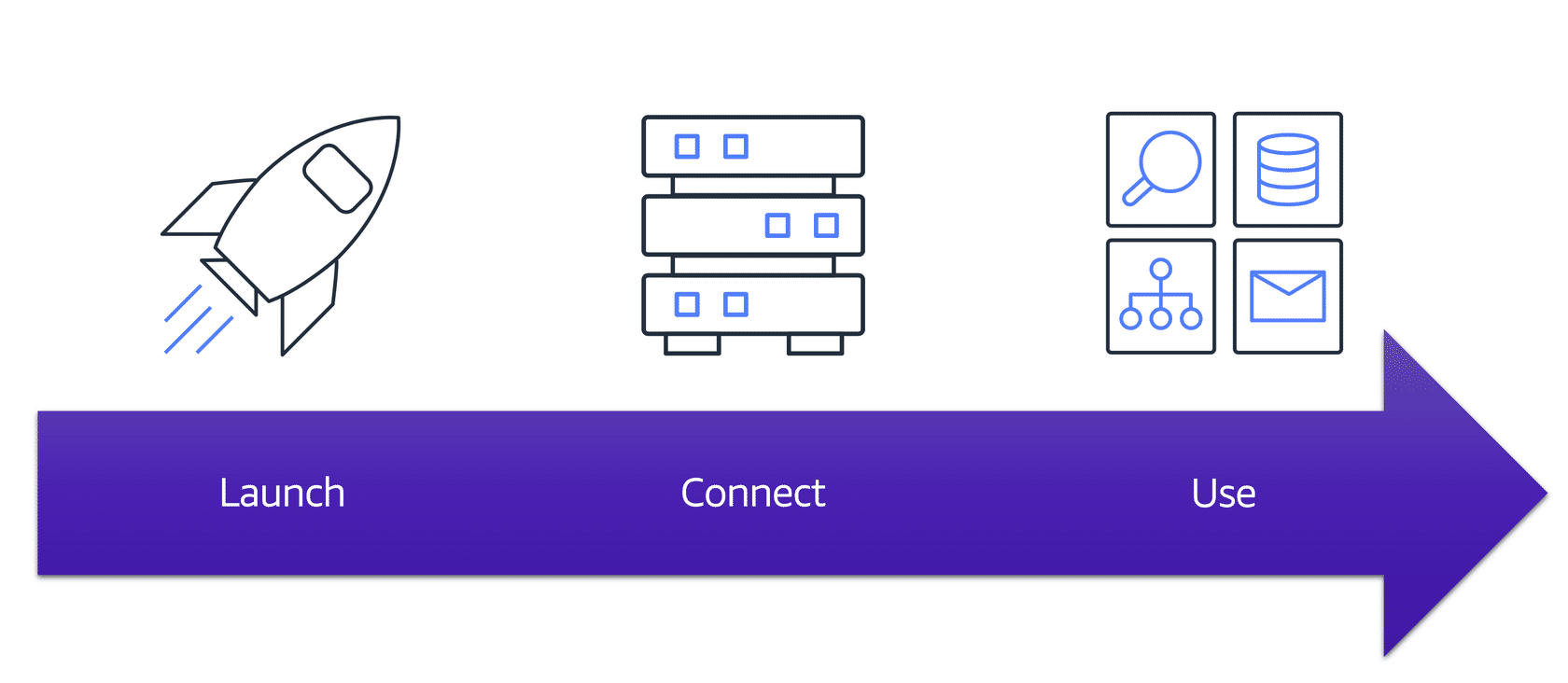
* Spend money upfront to purchase hardware.
* Wait for the servers to be delivered to you.
* Install the servers in your physical data center.
* Make all the necessary configurations.

By comparison, with an Amazon EC2 instance you can use a virtual server to run applications in the AWS Cloud.

* You can provision and launch an Amazon EC2 instance within minutes.
* You can stop using it when you have finished running a workload.
* You pay only for the compute time you use when an instance is running, not when it is stopped or terminated.
* You can save costs by paying only for server capacity that you need or want.

**How Amazon EC2 works**

To learn more, select each marker.



# Amazon EC2 instance types

**Amazon EC2 instance types**

[Amazon EC2 instance types](https://aws.amazon.com/ec2/instance-types/) are optimized for different tasks. When selecting an instance type, consider the specific needs of your workloads and applications. This might include requirements for compute, memory, or storage capabilities.

To learn more, select the **+** symbol next to each category.

**General purpose instances**

**General purpose instances** provide a balance of compute, memory, and networking resources. You can use them for a variety of workloads, such as:

* application servers
* gaming servers
* backend servers for enterprise applications
* small and medium databases

Suppose that you have an application in which the resource needs for compute, memory, and networking are roughly equivalent. You might consider running it on a general purpose instance because the application does not require optimization in any single resource area.

# 

**Compute optimized instances**

**Compute optimized instances** are ideal for compute-bound applications that benefit from high-performance processors. Like general purpose instances, you can use compute optimized instances for workloads such as web, application, and gaming servers.

However, the difference is compute optimized applications are ideal for high-performance web servers, compute-intensive applications servers, and dedicated gaming servers. You can also use compute optimized instances for batch processing workloads that require processing many transactions in a single group.

**Memory optimized instances**

**Memory optimized instances** are designed to deliver fast performance for workloads that process large datasets in memory. In computing, memory is a temporary storage area. It holds all the data and instructions that a central processing unit (CPU) needs to be able to complete actions. Before a computer program or application is able to run, it is loaded from storage into memory. This preloading process gives the CPU direct access to the computer program.

Suppose that you have a workload that requires large amounts of data to be preloaded before running an application. This scenario might be a high-performance database or a workload that involves performing real-time processing of a large amount of unstructured data. In these types of use cases, consider using a memory optimized instance. Memory optimized instances enable you to run workloads with high memory needs and receive great performance.

**Accelerated computing instances**

–

**Accelerated computing instances** use hardware accelerators, or coprocessors, to perform some functions more efficiently than is possible in software running on CPUs. Examples of these functions include floating-point number calculations, graphics processing, and data pattern matching.

In computing, a hardware accelerator is a component that can expedite data processing. Accelerated computing instances are ideal for workloads such as graphics applications, game streaming, and application streaming.

**Storage optimized instances**

–

**Storage optimized instances** are designed for workloads that require high, sequential read and write access to large datasets on local storage. Examples of workloads suitable for storage optimized instances include distributed file systems, data warehousing applications, and high-frequency online transaction processing (OLTP) systems.

In computing, the term input/output operations per second (IOPS) is a metric that measures the performance of a storage device. It indicates how many different input or output operations a device can perform in one second. Storage optimized instances are designed to deliver tens of thousands of low-latency, random IOPS to applications.

You can think of input operations as data put into a system, such as records entered into a database. An output operation is data generated by a server. An example of output might be the analytics performed on the records in a database. If you have an application that has a high IOPS requirement, a storage optimized instance can provide better performance over other instance types not optimized for this kind of use case.

# Amazon EC2 pricing

**Amazon EC2 pricing**

With Amazon EC2, you pay only for the compute time that you use. Amazon EC2 offers a variety of pricing options for different use cases. For example, if your use case can withstand interruptions, you can save with Spot Instances. You can also save by committing early and locking in a minimum level of use with Reserved Instances.

**On-Demand**

–

**On-Demand Instances** are ideal for short-term, irregular workloads that cannot be interrupted. No upfront costs or minimum contracts apply. The instances run continuously until you stop them, and you pay for only the compute time you use.  
  
Sample use cases for On-Demand Instances include developing and testing applications and running applications that have unpredictable usage patterns. On-Demand Instances are not recommended for workloads that last a year or longer because these workloads can experience greater cost savings using Reserved Instances.

**Amazon EC2 Savings Plans**

–

AWS offers Savings Plans for several compute services, including Amazon EC2. **Amazon EC2 Savings Plans** enable you to reduce your compute costs by committing to a consistent amount of compute usage for a 1-year or 3-year term. This term commitment results in savings of up to 66% over On-Demand costs.

Any usage up to the commitment is charged at the discounted plan rate (for example, $10 an hour). Any usage beyond the commitment is charged at regular On-Demand rates.

Later in this course, you will review AWS Cost Explorer, a tool that enables you to visualize, understand, and manage your AWS costs and usage over time. If you are considering your options for Savings Plans, AWS Cost Explorer can analyze your Amazon EC2 usage over the past 7, 30, or 60 days. AWS Cost Explorer also provides customized recommendations for Savings Plans. These recommendations estimate how much you could save on your monthly Amazon EC2 costs, based on previous Amazon EC2 usage and the hourly commitment amount in a 1-year or 3-year plan.

**Reserved Instances**

–

**Reserved Instances** are a billing discount applied to the use of On-Demand Instances in your account. You can purchase Standard Reserved and Convertible Reserved Instances for a 1-year or 3-year term, and Scheduled Reserved Instances for a 1-year term. You realize greater cost savings with the 3-year option.

At the end of a Reserved Instance term, you can continue using the Amazon EC2 instance without interruption. However, you are charged On-Demand rates until you do one of the following:

* Terminate the instance.
* Purchase a new Reserved Instance that matches the instance attributes (instance type, Region, tenancy, and platform).

**Spot Instances**

**Spot Instances** are ideal for workloads with flexible start and end times, or that can withstand interruptions. Spot Instances use unused Amazon EC2 computing capacity and offer you cost savings at up to 90% off of On-Demand prices.  
  
Suppose that you have a background processing job that can start and stop as needed (such as the data processing job for a customer survey). You want to start and stop the processing job without affecting the overall operations of your business. If you make a Spot request and Amazon EC2 capacity is available, your Spot Instance launches. However, if you make a Spot request and Amazon EC2 capacity is unavailable, the request is not successful until capacity becomes available. The unavailable capacity might delay the launch of your background processing job.  
  
After you have launched a Spot Instance, if capacity is no longer available or demand for Spot Instances increases, your instance may be interrupted. This might not pose any issues for your background processing job. However, in the earlier example of developing and testing applications, you would most likely want to avoid unexpected interruptions. Therefore, choose a different EC2 instance type that is ideal for those tasks.

**Dedicated Hosts**

**Dedicated Hosts**are physical servers with Amazon EC2 instance capacity that is fully dedicated to your use.

You can use your existing per-socket, per-core, or per-VM software licenses to help maintain license compliance. You can purchase On-Demand Dedicated Hosts and Dedicated Hosts Reservations. Of all the Amazon EC2 options that were covered, Dedicated Hosts are the most expensive.

# Scaling Amazon EC2

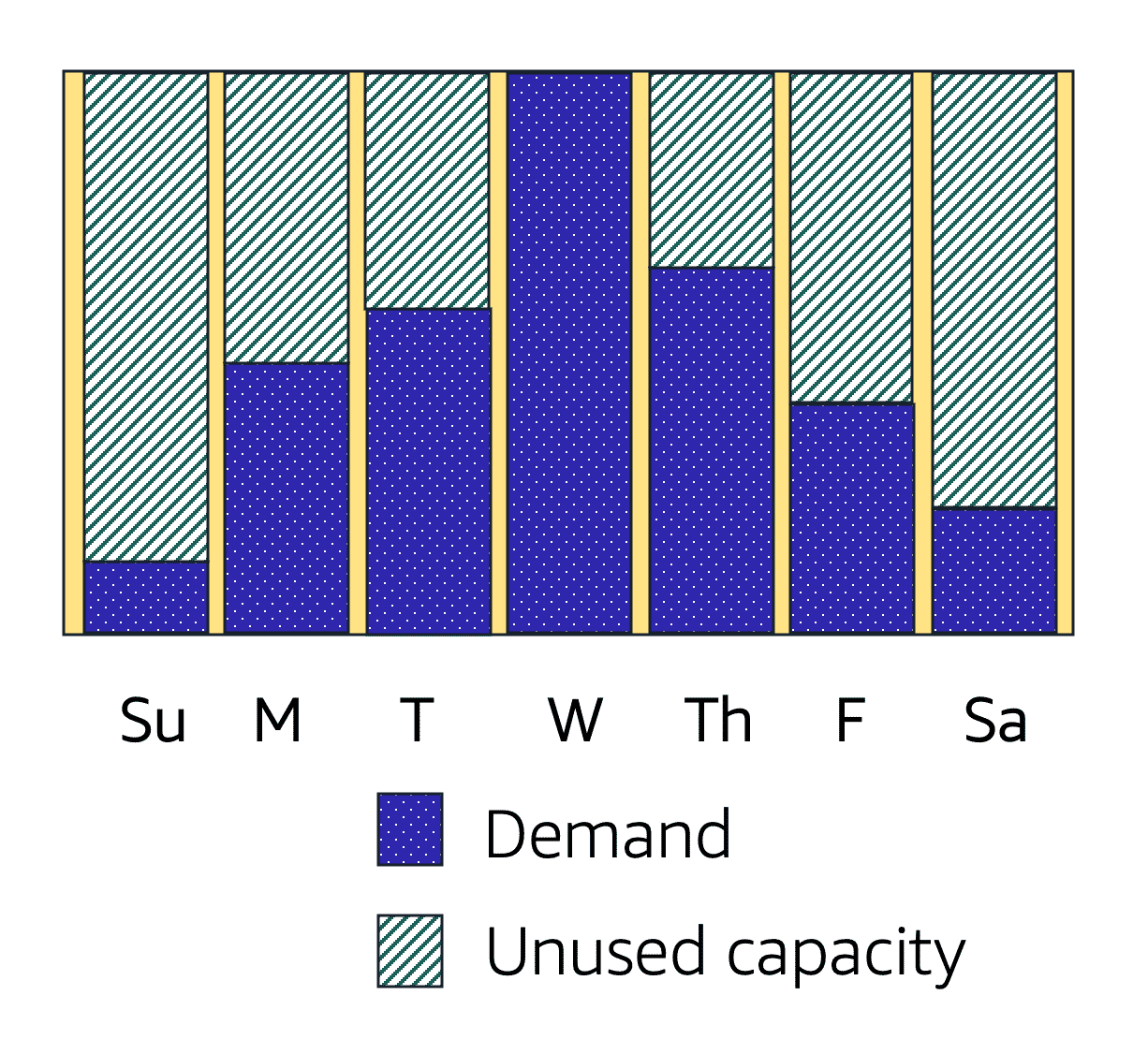
**Scalability**

**Scalability** involves beginning with only the resources you need and designing your architecture to automatically respond to changing demand by scaling out or in. As a result, you pay for only the resources you use. You don’t have to worry about a lack of computing capacity to meet your customers’ needs.

If you wanted the scaling process to happen automatically, which AWS service would you use? The AWS service that provides this functionality for Amazon EC2 instances is **Amazon EC2 Auto Scaling**.

**Amazon EC2 Auto Scaling**

If you’ve tried to access a website that wouldn’t load and frequently timed out, the website might have received more requests than it was able to handle. This situation is similar to waiting in a long line at a coffee shop, when there is only one barista present to take orders from customers.



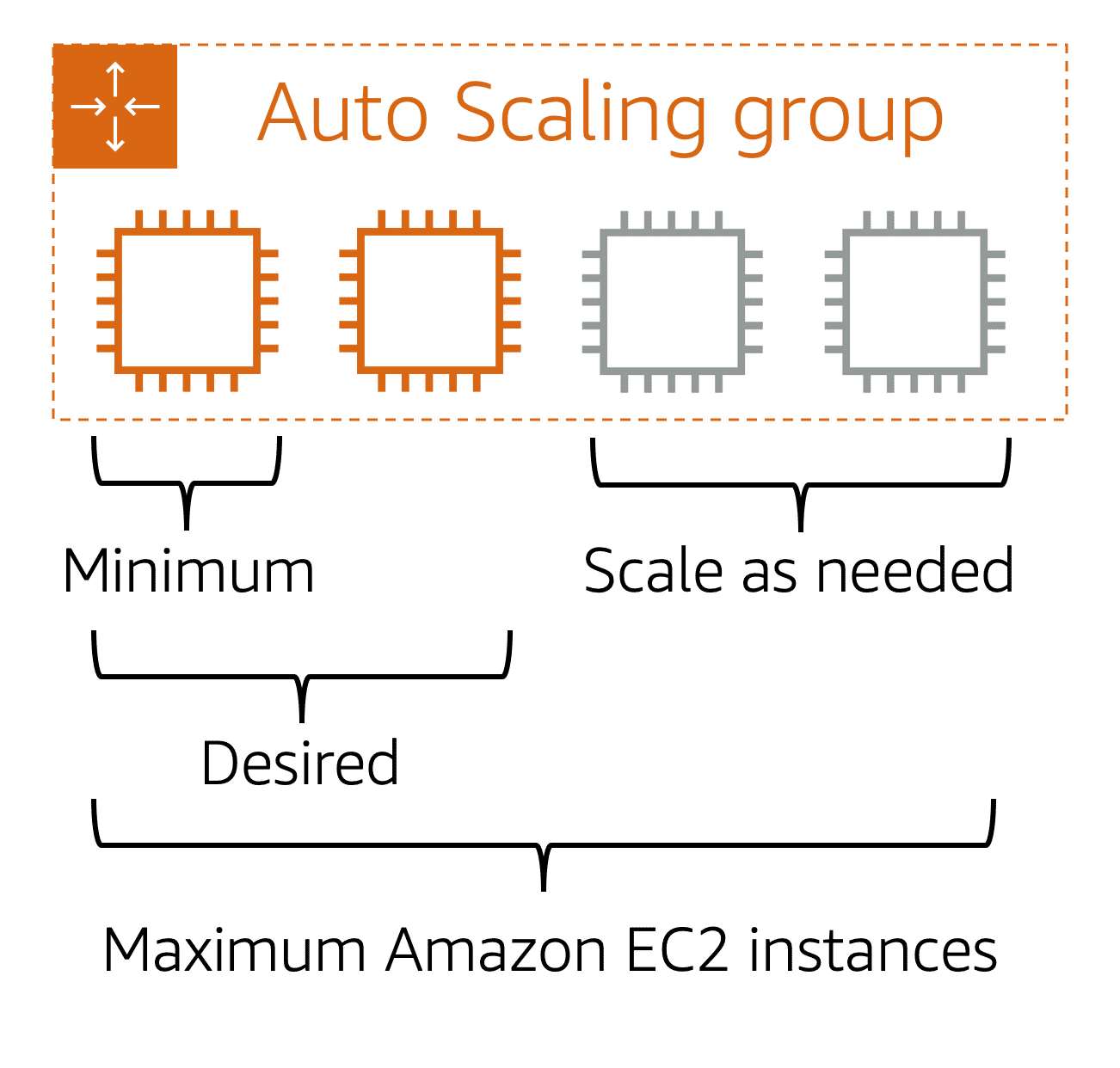
Amazon EC2 Auto Scaling enables you to automatically add or remove Amazon EC2 instances in response to changing application demand. By automatically scaling your instances in and out as needed, you are able to maintain a greater sense of application availability.

Within Amazon EC2 Auto Scaling, you can use two approaches: dynamic scaling and predictive scaling.

* *Dynamic scaling* responds to changing demand.
* *Predictive scaling*automatically schedules the right number of Amazon EC2 instances based on predicted demand.

**Example: Amazon EC2 Auto Scaling**

* In the cloud, computing power is a programmatic resource, so you can take a more flexible approach to the issue of scaling. By adding Amazon EC2 Auto Scaling to an application, you can add new instances to the application when necessary and terminate them when no longer needed.
* Suppose that you are preparing to launch an application on Amazon EC2 instances. When configuring the size of your Auto Scaling group, you might set the minimum number of Amazon EC2 instances at one. This means that at all times, there must be at least one Amazon EC2 instance running.



When you create an Auto Scaling group, you can set the minimum number of Amazon EC2 instances. The **minimum capacity** is the number of Amazon EC2 instances that launch immediately after you have created the Auto Scaling group. In this example, the Auto Scaling group has a minimum capacity of one Amazon EC2 instance.

Next, you can set the **desired capacity** at two Amazon EC2 instances even though your application needs a minimum of a single Amazon EC2 instance to run.

**If you do not specify the desired number of Amazon EC2 instances in an Auto Scaling group, the desired capacity defaults to your minimum capacity.**

The third configuration that you can set in an Auto Scaling group is the **maximum capacity**. For example, you might configure the Auto Scaling group to scale out in response to increased demand, but only to a maximum of four Amazon EC2 instances.

Because Amazon EC2 Auto Scaling uses Amazon EC2 instances, you pay for only the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while reducing expenses.

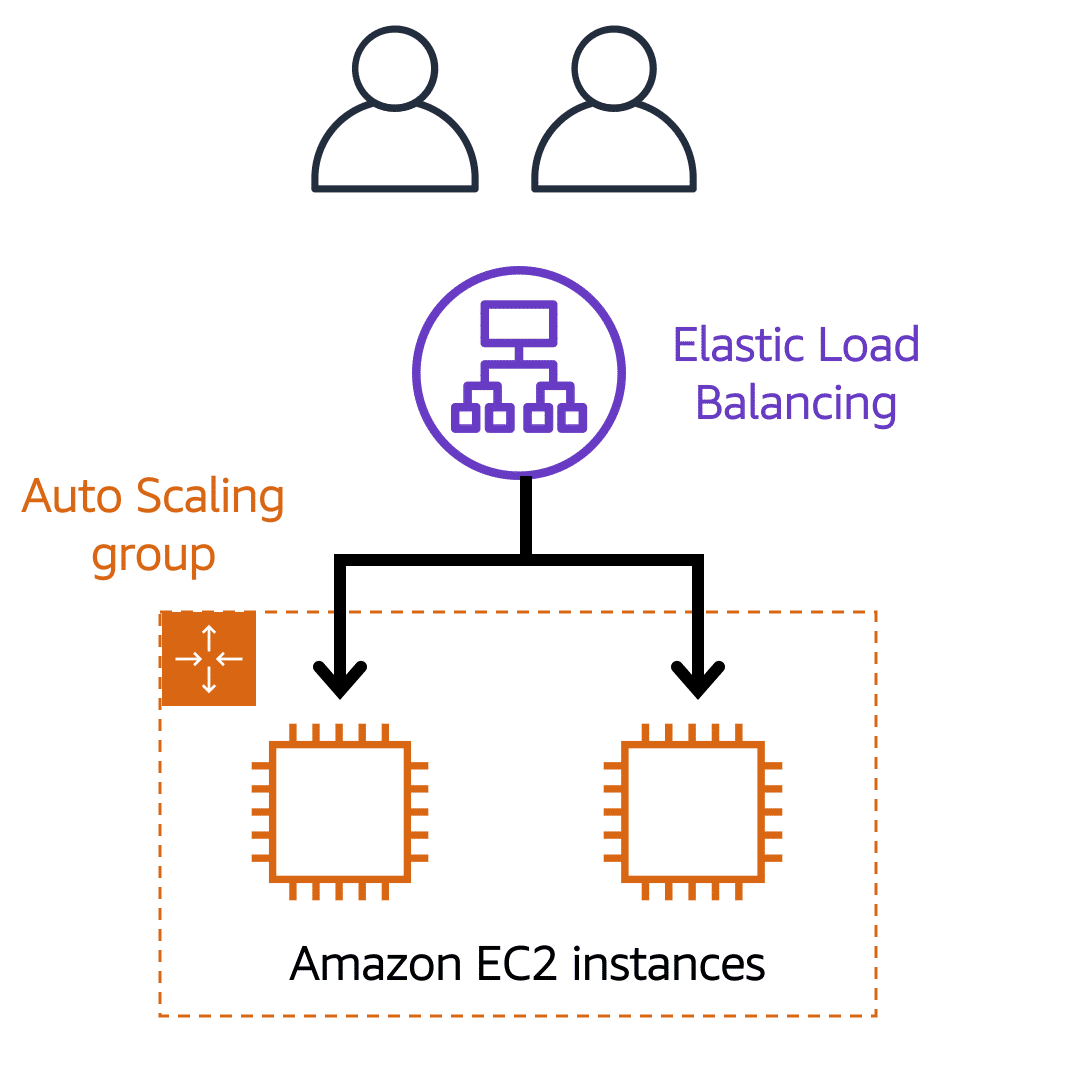
**Elastic Load Balancing**

**Elastic Load Balancing** is the AWS service that automatically distributes incoming application traffic across multiple resources, such as Amazon EC2 instances.

A load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group. This means that as you add or remove Amazon EC2 instances in response to the amount of incoming traffic, these requests route to the load balancer first. Then, the requests spread across multiple resources that will handle them. For example, if you have multiple Amazon EC2 instances, Elastic Load Balancing distributes the workload across the multiple instances so that no single instance has to carry the bulk of it.

Although Elastic Load Balancing and Amazon EC2 Auto Scaling are separate services, they work together to help ensure that applications running in Amazon EC2 can provide high performance and availability.

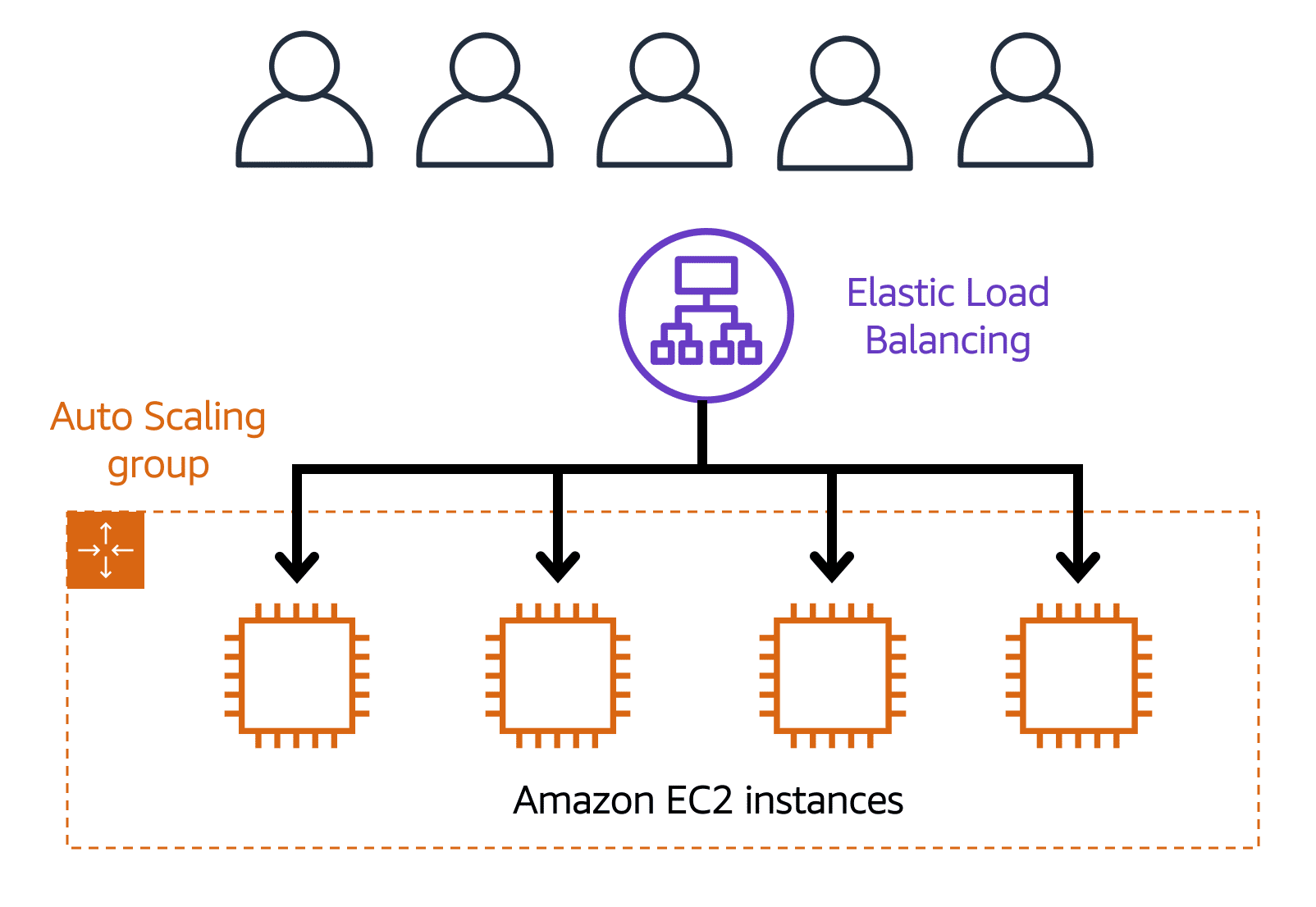
**Example: Elastic Load Balancing**



**Low-demand period**

Here’s an example of how Elastic Load Balancing works. Suppose that a few customers have come to the coffee shop and are ready to place their orders.

If only a few registers are open, this matches the demand of customers who need service. The coffee shop is less likely to have open registers with no customers. In this example, you can think of the registers as Amazon EC2 instances.



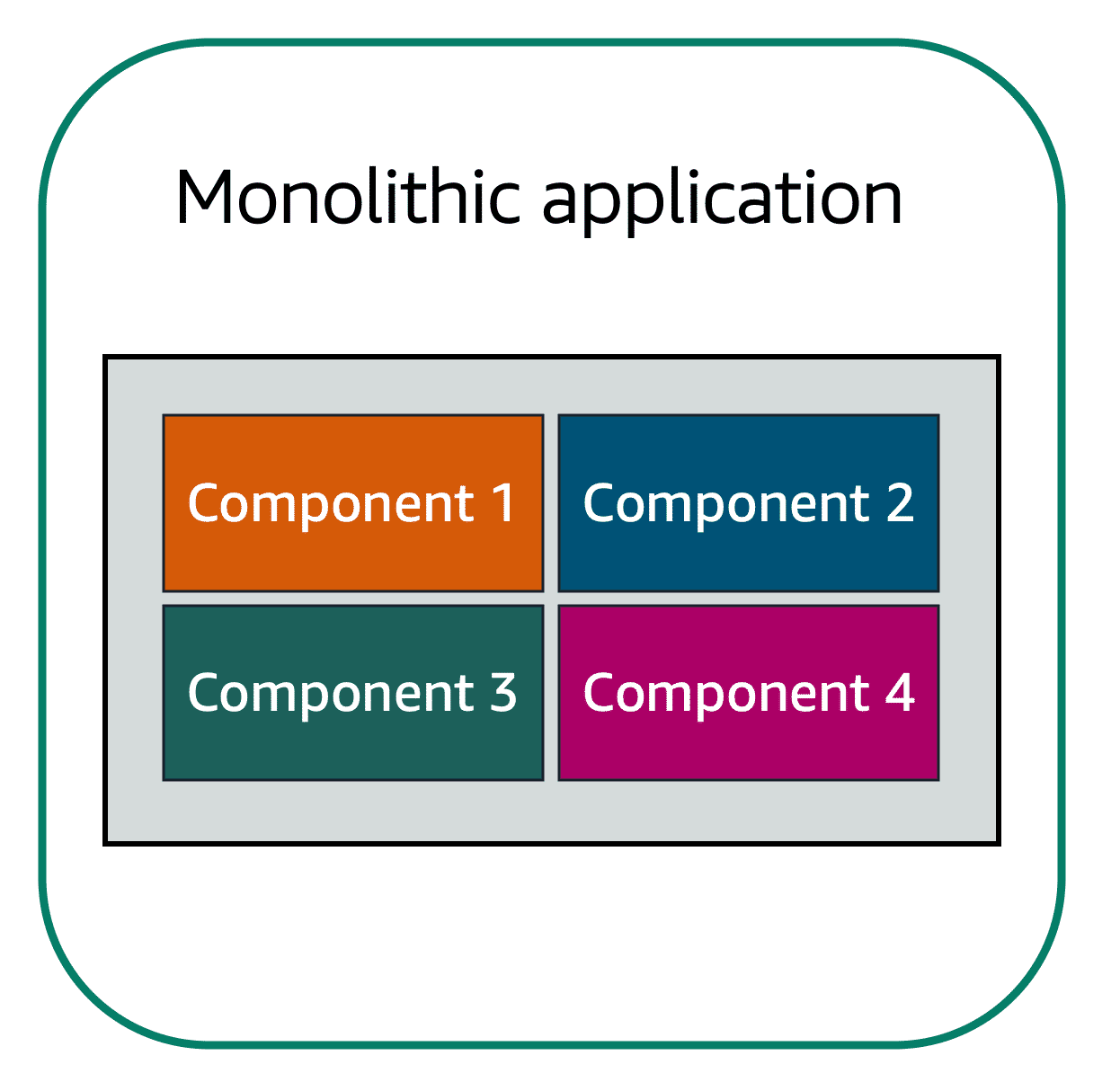
**High-demand period**

Throughout the day, as the number of customers increases, the coffee shop opens more registers to accommodate them. In the diagram, the Auto Scaling group represents this.

Additionally, a coffee shop employee directs customers to the most appropriate register so that the number of requests can evenly distribute across the open registers. You can think of this coffee shop employee as a load balancer.

# Messaging and queuing

**Monolithic applications and microservices**

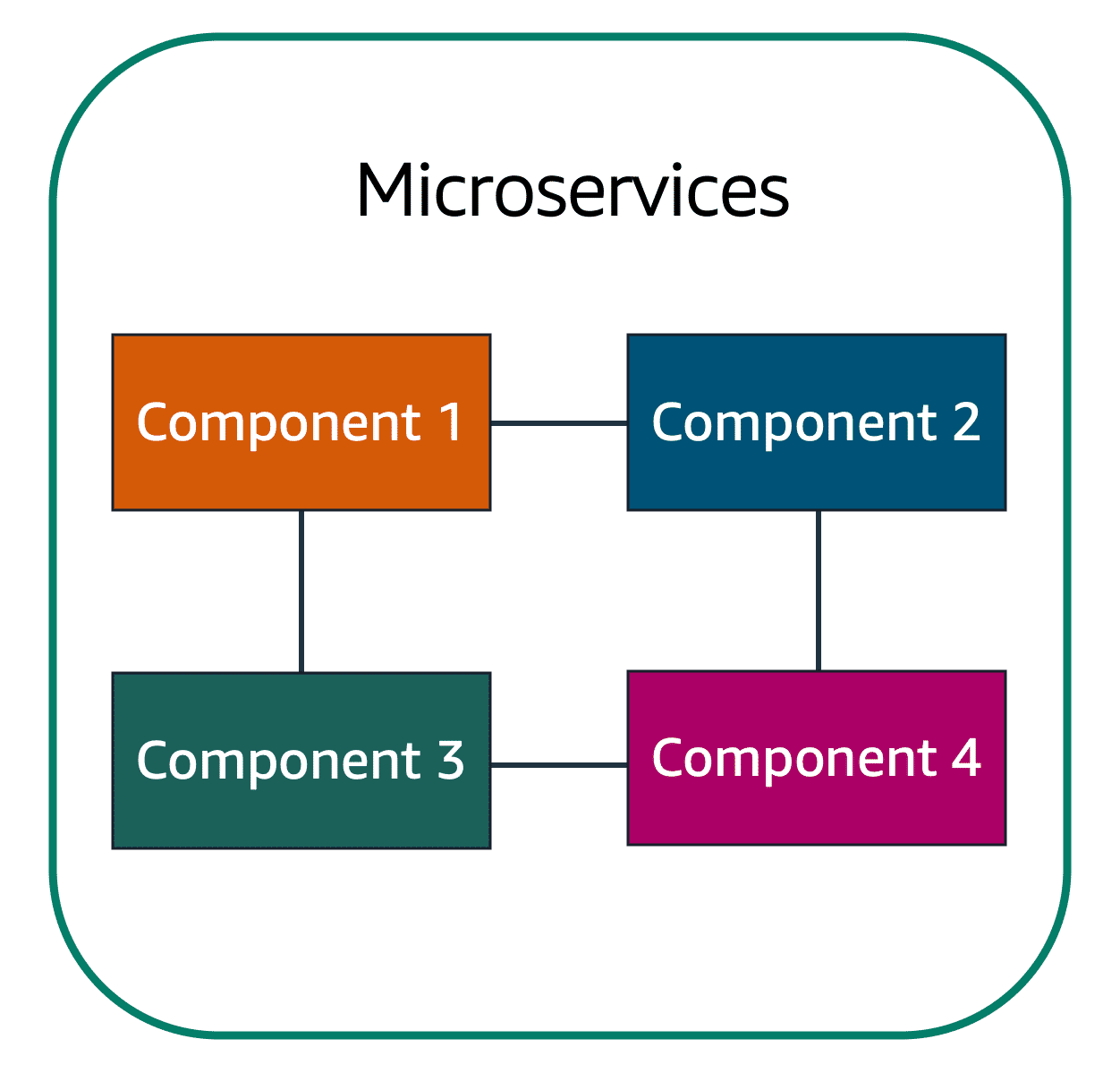


Applications are made of multiple components. The components communicate with each other to transmit data, fulfill requests, and keep the application running.

Suppose that you have an application with tightly coupled components. These components might include databases, servers, the user interface, business logic, and so on. This type of architecture can be considered a **monolithic application**.

In this approach to application architecture, if a single component fails, other components fail, and possibly the entire application fails.

To help maintain application availability when a single component fails, you can design your application through a **microservices** approach.



In a microservices approach, application components are loosely coupled. In this case, if a single component fails, the other components continue to work because they are communicating with each other. The loose coupling prevents the entire application from failing.

When designing applications on AWS, you can take a microservices approach with services and components that fulfill different functions. Two services facilitate application integration: Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).

**Amazon Simple Notification Service (Amazon SNS)**

**Amazon Simple Notification Service (Amazon SNS)**is a publish/subscribe service. Using Amazon SNS topics, a publisher publishes messages to subscribers. This is similar to the coffee shop; the cashier provides coffee orders to the barista who makes the drinks.

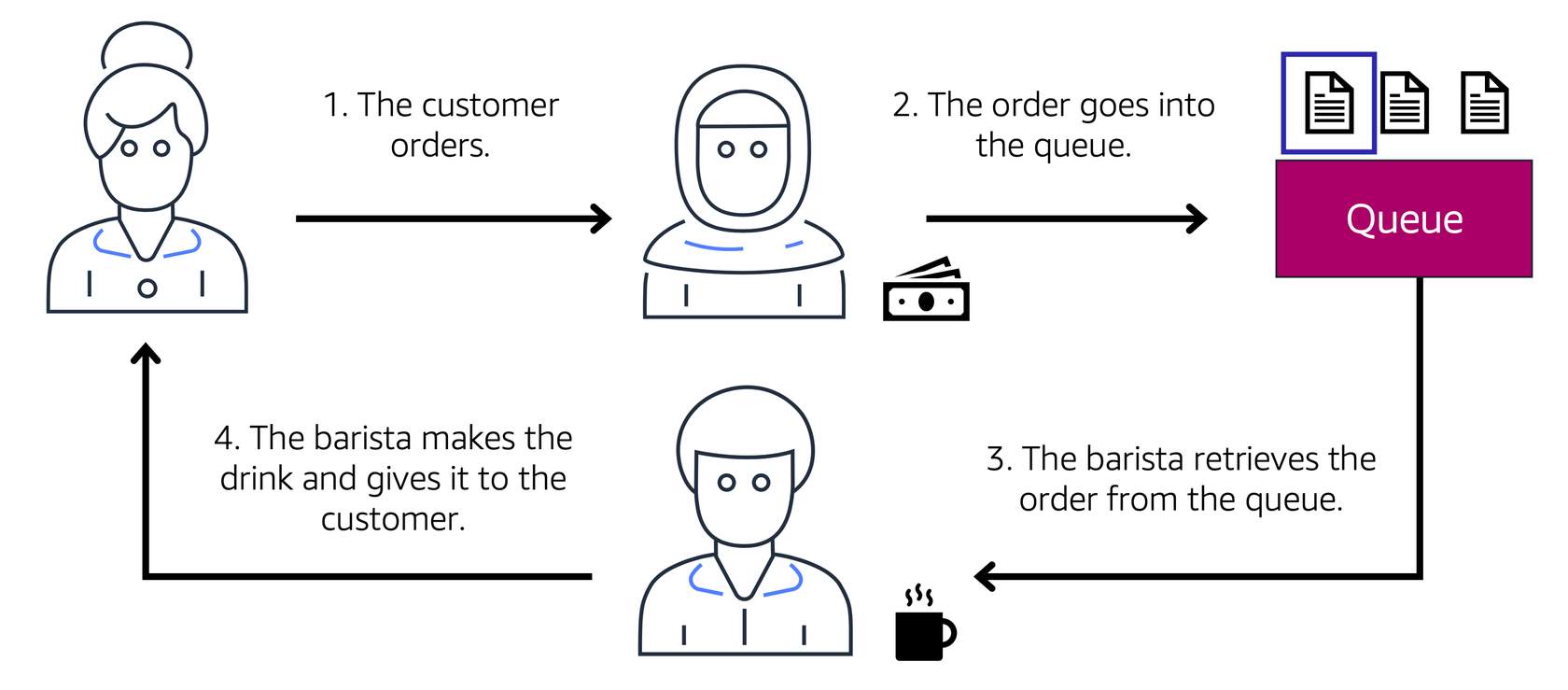
In Amazon SNS, subscribers can be web servers, email addresses, AWS Lambda functions, or several other options.

**Amazon Simple Queue Service (Amazon SQS)**

**Amazon Simple Queue Service (Amazon SQS)** is a message queuing service.

Using Amazon SQS, you can send, store, and receive messages between software components, without losing messages or requiring other services to be available. In Amazon SQS, an application sends messages into a queue. A user or service retrieves a message from the queue, processes it, and then deletes it from the queue.

## Example: Orders in a queue



Recall that the cashier and the barista are two separate components of an application. A message queuing service such as Amazon SQS enables messages between decoupled application complements.

In this example, the first step in the process remains the same as before: a customer places an order with the cashier.

The cashier puts the order into a queue. You can think of this as an order board that serves as a buffer between the cashier and the barista. Even if the barista is out on a break or busy with another order, the cashier can continue placing new orders into the queue.

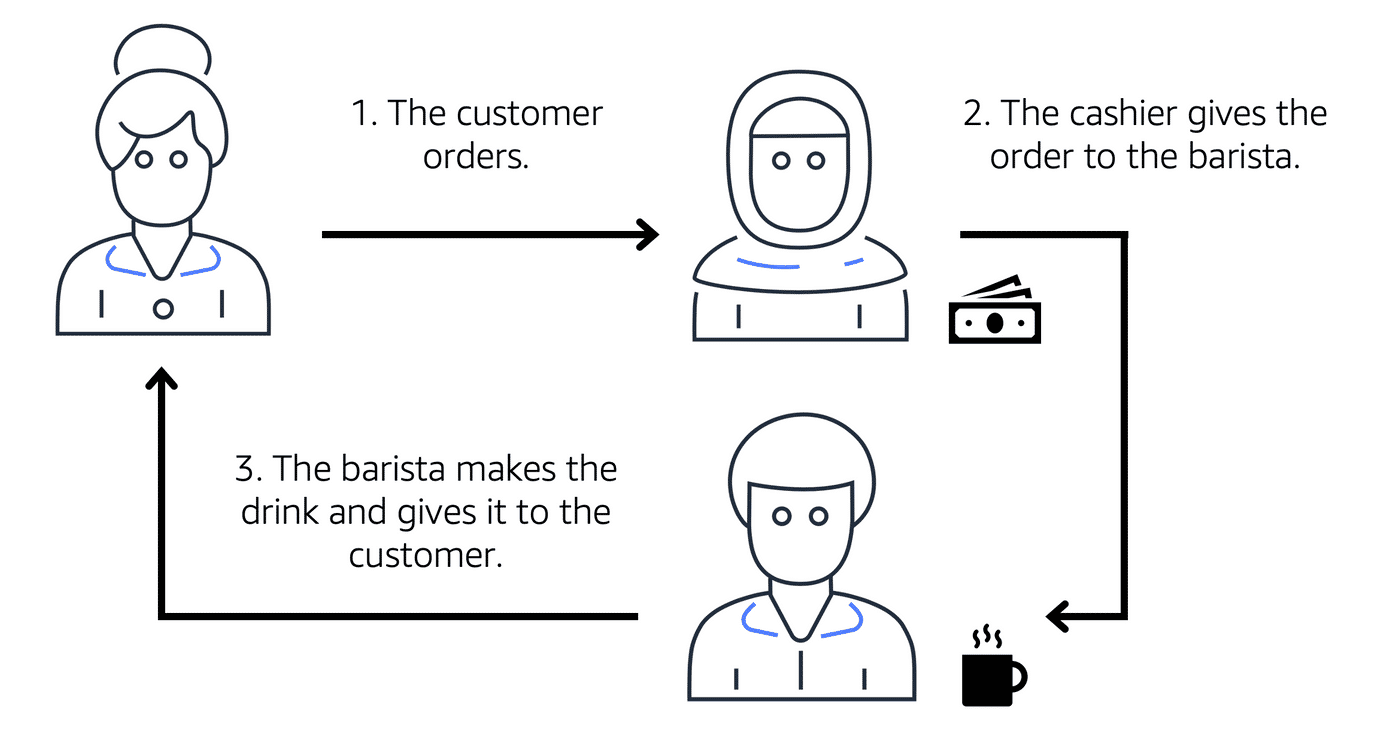
Next, the barista checks the queue and retrieves the order.

The barista prepares the drink and gives it to the customer.

The barista then removes the completed order from the queue.

While the barista is preparing the drink, the cashier is able to continue taking new orders and add them to the queue.

## Example: Fulfilling an order



Suppose that the coffee shop has an ordering process in which a cashier takes orders, and a barista makes the orders. Think of the cashier and the barista as two separate components of an application.

First, the cashier takes an order and writes it down on a piece of paper. Next, the cashier delivers the paper to the barista. Finally, the barista makes the drink and gives it to the customer.

When the next order comes in, the process repeats. This process runs smoothly as long as both the cashier and the barista are coordinated.

What might happen if the cashier took an order and went to deliver it to the barista, but the barista was out on a break or busy with another order? The cashier would need to wait until the barista is ready to accept the order. This would cause delays in the ordering process and require customers to wait longer to receive their orders.

As the coffee shop has become more popular and the ordering line is moving more slowly, the owners notice that the current ordering process is time consuming and inefficient. They decide to try a different approach that uses a queue.

Which AWS service is the best choice for publishing messages to subscribers?

Amazon Simple Queue Service (Amazon SQS)

Amazon EC2 Auto Scaling

Amazon Simple Notification Service (Amazon SNS)

# Additional compute services

**Serverless computing**

Earlier in this module, you learned about Amazon EC2, a service that lets you run virtual servers in the cloud. If you have applications that you want to run in Amazon EC2, you must do the following:

* 1

1

Provision instances (virtual servers).

* 2

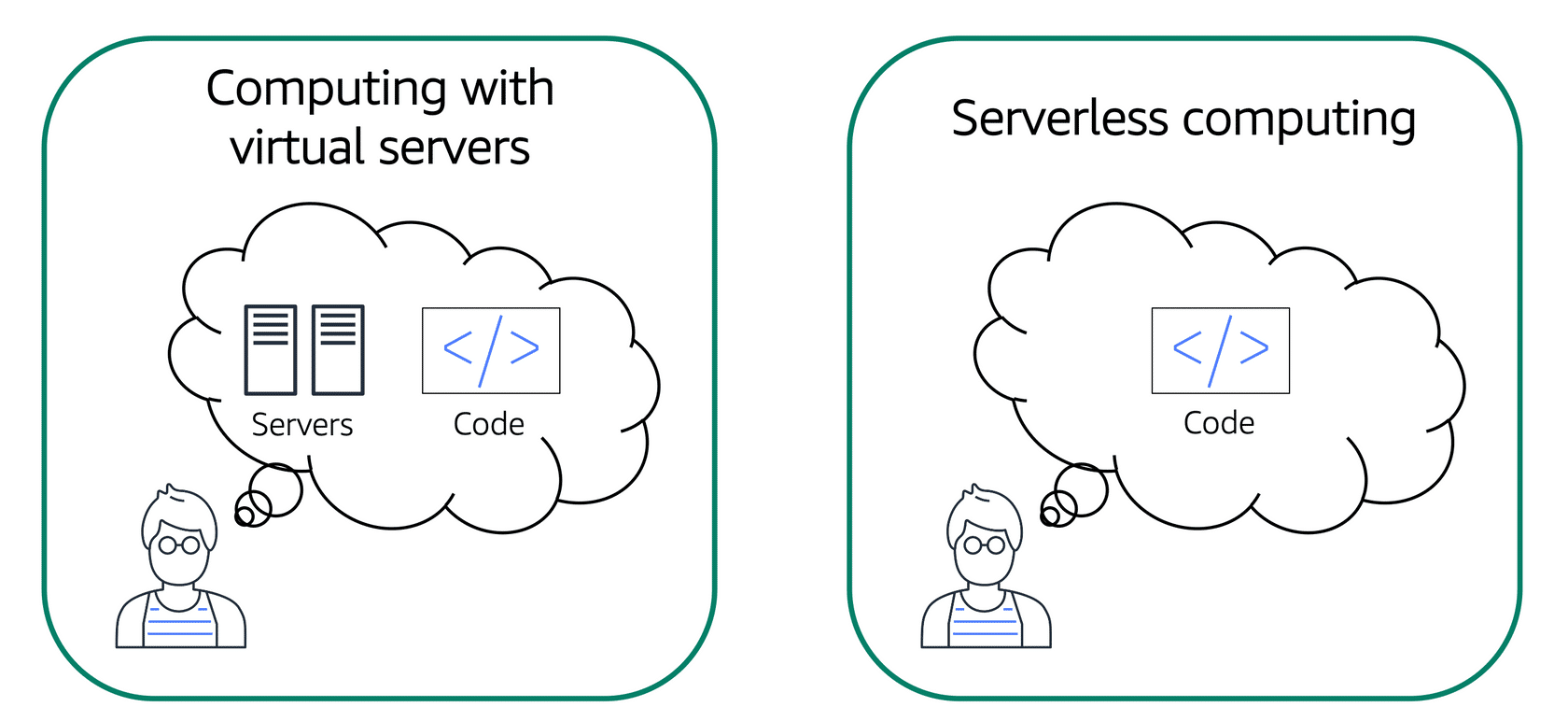
2

Upload your code.

* 3

3

Continue to manage the instances while your application is running.



The term “serverless” means that your code runs on servers, but you do not need to provision or manage these servers. With serverless computing, you can focus more on innovating new products and features instead of maintaining servers.

Another benefit of serverless computing is the flexibility to scale serverless applications automatically. Serverless computing can adjust the applications' capacity by modifying the units of consumptions, such as throughput and memory.

An AWS service for serverless computing is **AWS Lambda**.

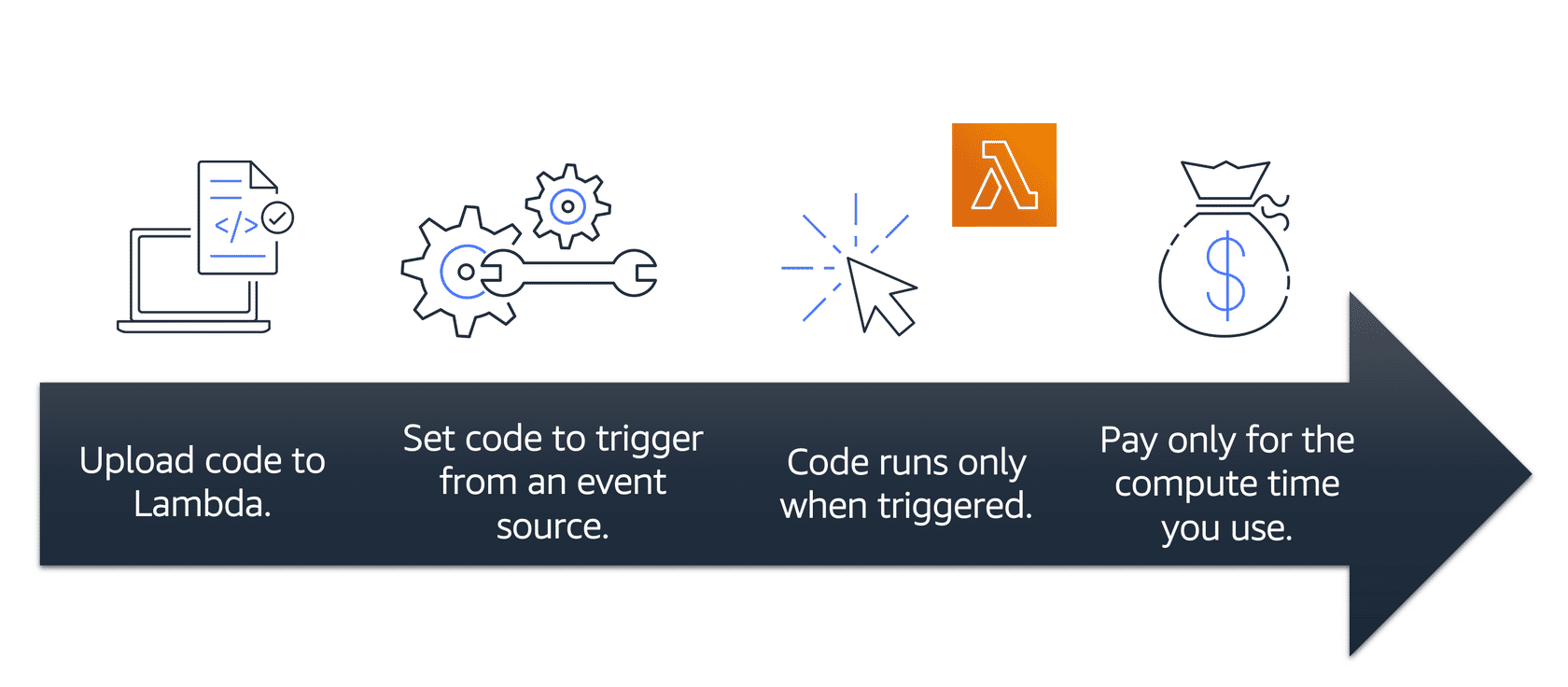
**AWS Lambda**

[**AWS Lambda**](https://aws.amazon.com/lambda) is a service that lets you run code without needing to provision or manage servers.

While using AWS Lambda, you pay only for the compute time that you consume. Charges apply only when your code is running. You can also run code for virtually any type of application or backend service, all with zero administration.

For example, a simple Lambda function might involve automatically resizing uploaded images to the AWS Cloud. In this case, the function triggers when uploading a new image.

**How AWS Lambda works**



* 1

1

You upload your code to Lambda.

* 2

2

You set your code to trigger from an event source, such as AWS services, mobile applications, or HTTP endpoints.

* 3

3

Lambda runs your code only when triggered.

* 4

4

You pay only for the compute time that you use. In the previous example of resizing images, you would pay only for the compute time that you use when uploading new images. Uploading the images triggers Lambda to run code for the image resizing function.

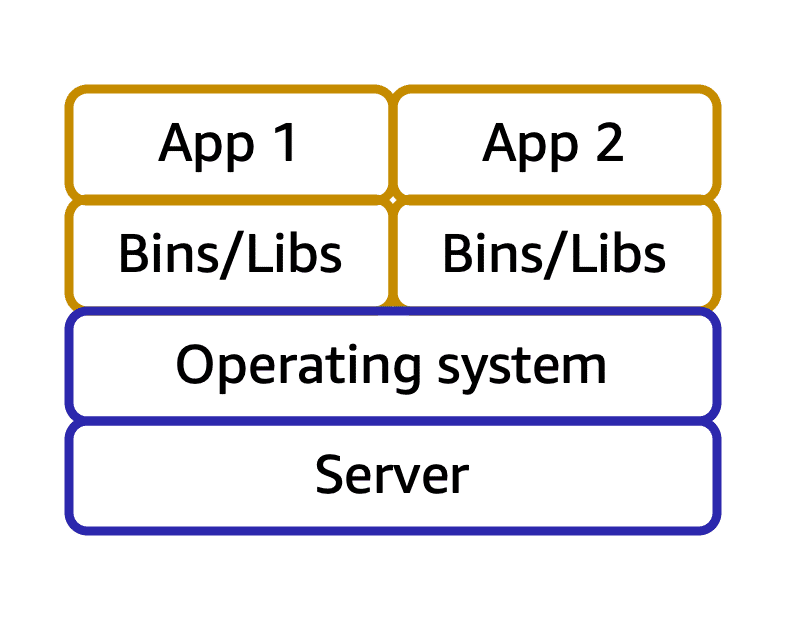
In AWS, you can also build and run **containerized** applications.

**Containers**

**Containers** provide you with a standard way to package your application's code and dependencies into a single object. You can also use containers for processes and workflows in which there are essential requirements for security, reliability, and scalability.

**Step 1**

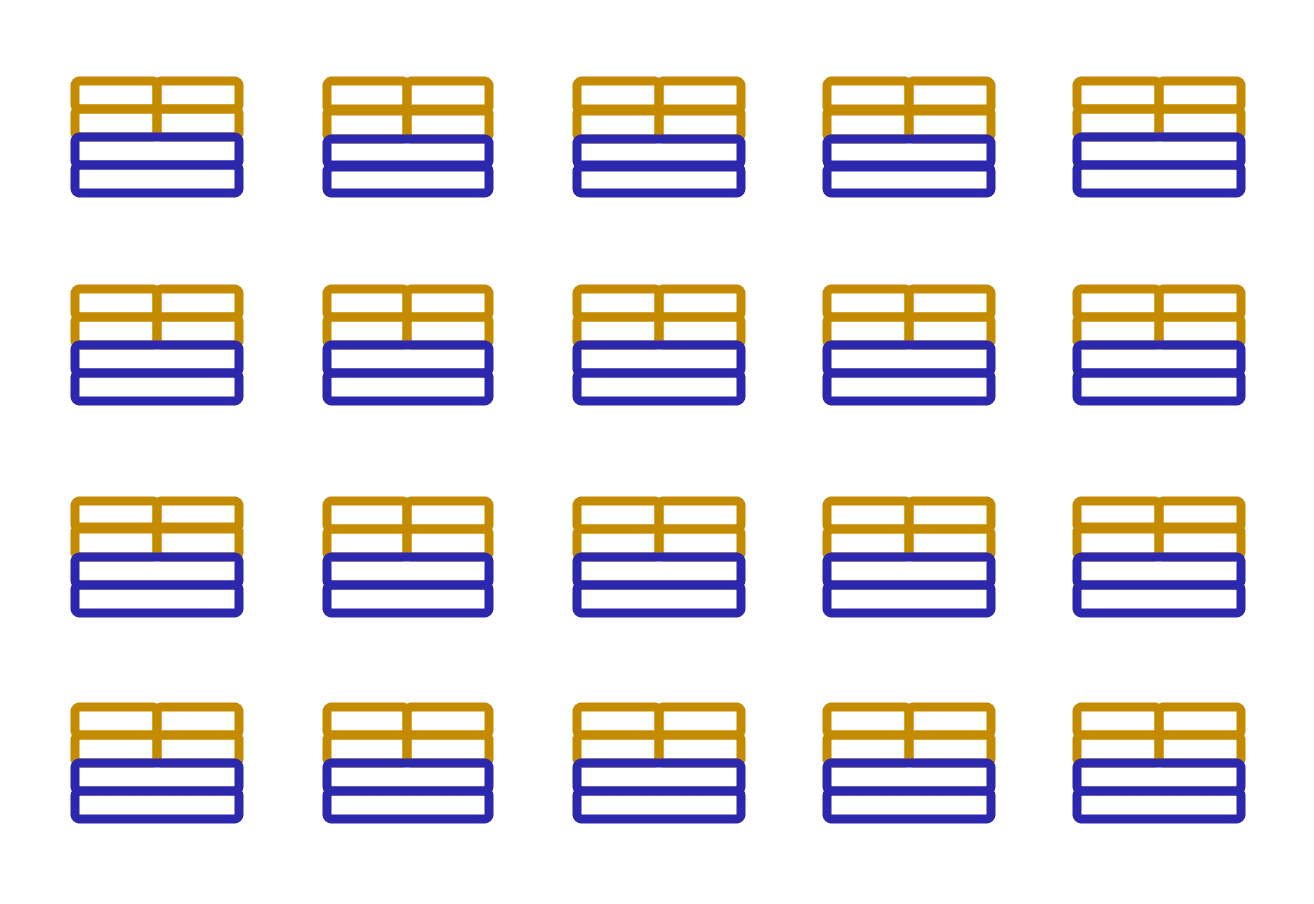
## One host with multiple containers



Suppose that a company’s application developer has an environment on their computer that is different from the environment on the computers used by the IT operations staff. The developer wants to ensure that the application’s environment remains consistent regardless of deployment, so they use a containerized approach. This helps to reduce time spent debugging applications and diagnosing differences in computing environments.

**Step 2**

## Tens of hosts with hundreds of containers



When running containerized applications, it’s important to consider scalability. Suppose that instead of a single host with multiple containers, you have to manage tens of hosts with hundreds of containers. Alternatively, you have to manage possibly hundreds of hosts with thousands of containers. At a large scale, imagine how much time it might take for you to monitor memory usage, security, logging, and so on.

Container orchestration services help you to deploy, manage, and scale your containerized applications. Next, you will learn about two services that provide container orchestration: Amazon Elastic Container Service and Amazon Elastic Kubernetes Service.

**Amazon Elastic Container Service (Amazon ECS)**

[**Amazon Elastic Container Service (Amazon ECS)**](https://aws.amazon.com/ecs/) is a highly scalable, high-performance container management system that enables you to run and scale containerized applications on AWS.

Amazon ECS supports Docker containers. [Docker](https://www.docker.com/) is a software platform that enables you to build, test, and deploy applications quickly. AWS supports the use of open-source Docker Community Edition and subscription-based Docker Enterprise Edition. With Amazon ECS, you can use API calls to launch and stop Docker-enabled applications.

**Amazon Elastic Kubernetes Service (Amazon EKS)**

[**Amazon Elastic Kubernetes Service (Amazon EKS)**](https://aws.amazon.com/eks/)is a fully managed service that you can use to run Kubernetes on AWS.

[Kubernetes](https://kubernetes.io/) is open-source software that enables you to deploy and manage containerized applications at scale. A large community of volunteers maintains Kubernetes, and AWS actively works together with the Kubernetes community. As new features and functionalities release for Kubernetes applications, you can easily apply these updates to your applications managed by Amazon EKS.

**AWS Fargate**

[**AWS Fargate**](https://aws.amazon.com/fargate/)is a serverless compute engine for containers. It works with both Amazon ECS and Amazon EKS.

When using AWS Fargate, you do not need to provision or manage servers. AWS Fargate manages your server infrastructure for you. You can focus more on innovating and developing your applications, and you pay only for the resources that are required to run your containers.

**Module 2 summary**

In Module 2, you learned about the following concepts:

* Amazon EC2 instance types and pricing options
* Amazon EC2 Auto Scaling
* Elastic Load Balancing
* AWS services for messaging, containers, and serverless computing

**Additional resources**

To learn more about the concepts that were explored in Module 2, review these resources.

* [Compute on AWS](https://aws.amazon.com/products/compute)
* [AWS Compute Blog](https://aws.amazon.com/blogs/compute/)
* [AWS Compute Services](https://docs.aws.amazon.com/whitepapers/latest/aws-overview/compute-services.html)
* [Hands-On Tutorials: Compute](https://aws.amazon.com/getting-started/hands-on/?awsf.getting-started-category=category%23compute&awsf.getting-started-content-type=content-type%23hands-on)
* [Category Deep Dive: Serverless](https://aws.amazon.com/getting-started/deep-dive-serverless/)
* [AWS Customer Stories: Serverless](https://aws.amazon.com/solutions/case-studies/?customer-references-cards.sort-by=item.additionalFields.publishedDate&customer-references-cards.sort-order=desc&awsf.customer-references-location=*all&awsf.customer-references-segment=*all&awsf.customer-references-product=product%23vpc%7Cproduct%23api-gateway%7Cproduct%23cloudfront%7Cproduct%23route53%7Cproduct%23directconnect%7Cproduct%23elb&awsf.customer-references-category=category%23serverless)

# Module 2 quiz

You want to use an Amazon EC2 instance for a batch processing workload. What would be the best Amazon EC2 instance type to use?

General purpose

Memory optimized

Compute optimized

Storage optimized

What are the contract length options for Amazon EC2 Reserved Instances? (Select TWO.)

1 year

3 yerar

You have a workload that will run for a total of 6 months and can withstand interruptions. What would be the most cost-efficient Amazon EC2 purchasing option?

Spot Instance

Which process is an example of Elastic Load Balancing?

Ensuring that no single Amazon EC2 instance has to carry the full workload on its own

You want to deploy and manage containerized applications. Which service should you use?

Amazon Elastic Kubernetes Service (Amazon EKS)