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

You just learned more about AWS and how almost all of 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 fulfills it by returning the information to the client.

**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 DEPLOYMENTON-PREMISES DEPLOYMENTHYBRID DEPLOYMENT**](https://assets.skillbuilder.aws/files/a/w/aws_prod1_docebosaas_com/1651125600/OIdgeDX5xQW5bKQVMOODFQ/tincan/31d9c0cca79c54bdceaf3e938fd424e97c98c7e8/index.html?enhanced_signature=WXmcXgs-RcnYk05UJNFXJDTvEAWXNLmAxKoyJp2yyVk&endpoint=https%3A%2F%2Fassets.skillbuilder.aws%2Flrs-api%2F&auth=Basic%20LzRjMDhlZGJjLTBjZGItNDU3Yy1iYWI5LWVkZDQ3ZDVmMTZjNTokMnkkMTMkQmlQaWRjVEZzMUl6anUxRjdwOUJBZVlSNkJ0YU83a29JNlp4Q1RiMGNaRmswQ0xiVHZGVmE%3D&actor=%7B%22mbox%22%3A%22mailto%3Amounika.rapelly%40capgemini.com%22%2C%22name%22%3A%22mounika+rapelly%22%7D&registration=045803a5-a821-4e91-bd09-009132e65ab8&activity_id=http%3A%2F%2FeMs3fVuk5fRrdzRW22BTIyGPEJ5VTs6k_rise&Accept-Language=en&course_id=134&content_token=045803a5-a821-4e91-bd09-009132e65ab8&session_context=lms&host=explore.skillbuilder.aws&course_code=DIG-BF-100-CECPEB-10-EN&course_id=134&username=4c08edbc-0cdb-457c-bab9-edd47d5f16c5&user_id=1621345&hash=991d7667a27fba61c75fadc325b90c3641c8b6eb3a57d18d0fe6bafdecc36fba)

* 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.

**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 centers, 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.

**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:

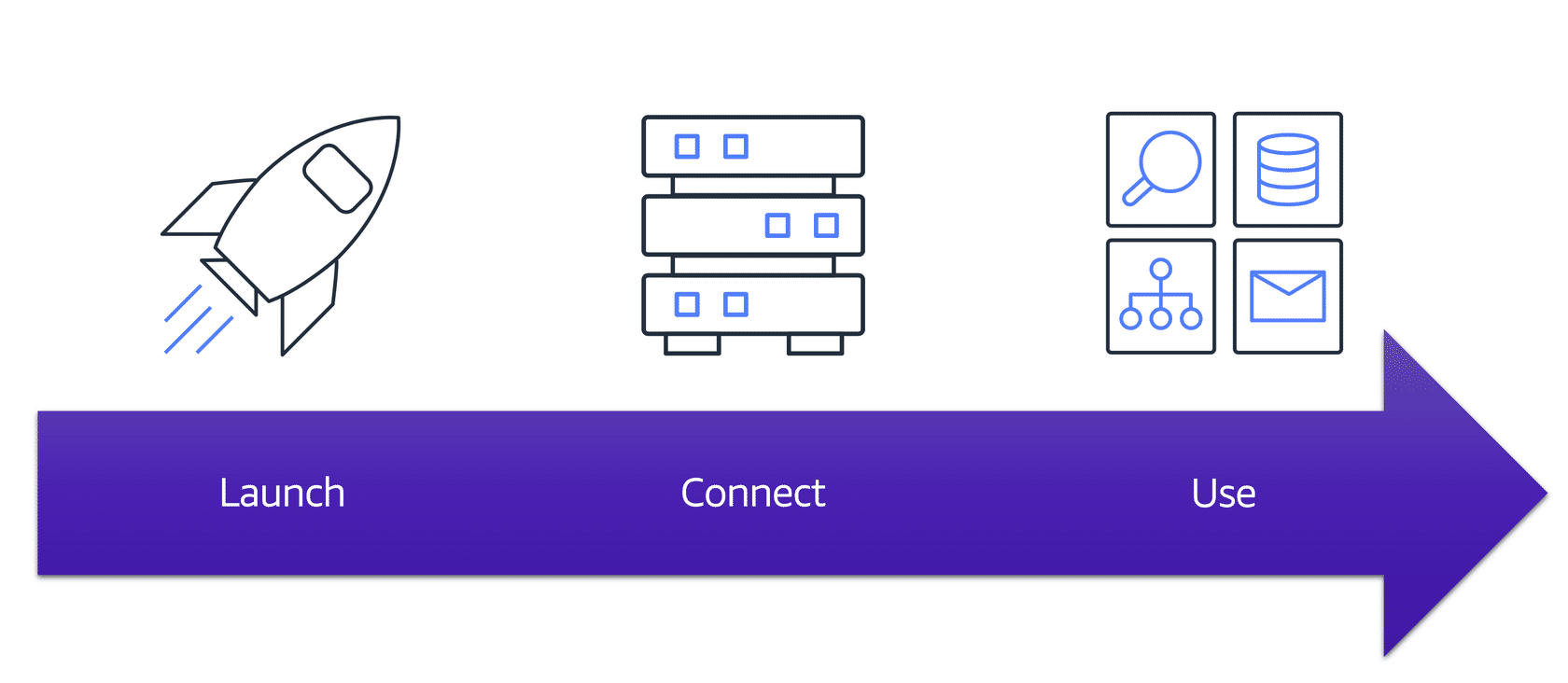
* Spend money upfront to purchase hardware.
* Wait for the servers to be delivered to you.
* Install the servers in your physical data center.
* Make all the necessary configurations.

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](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.

**Scaling Amazon EC2**

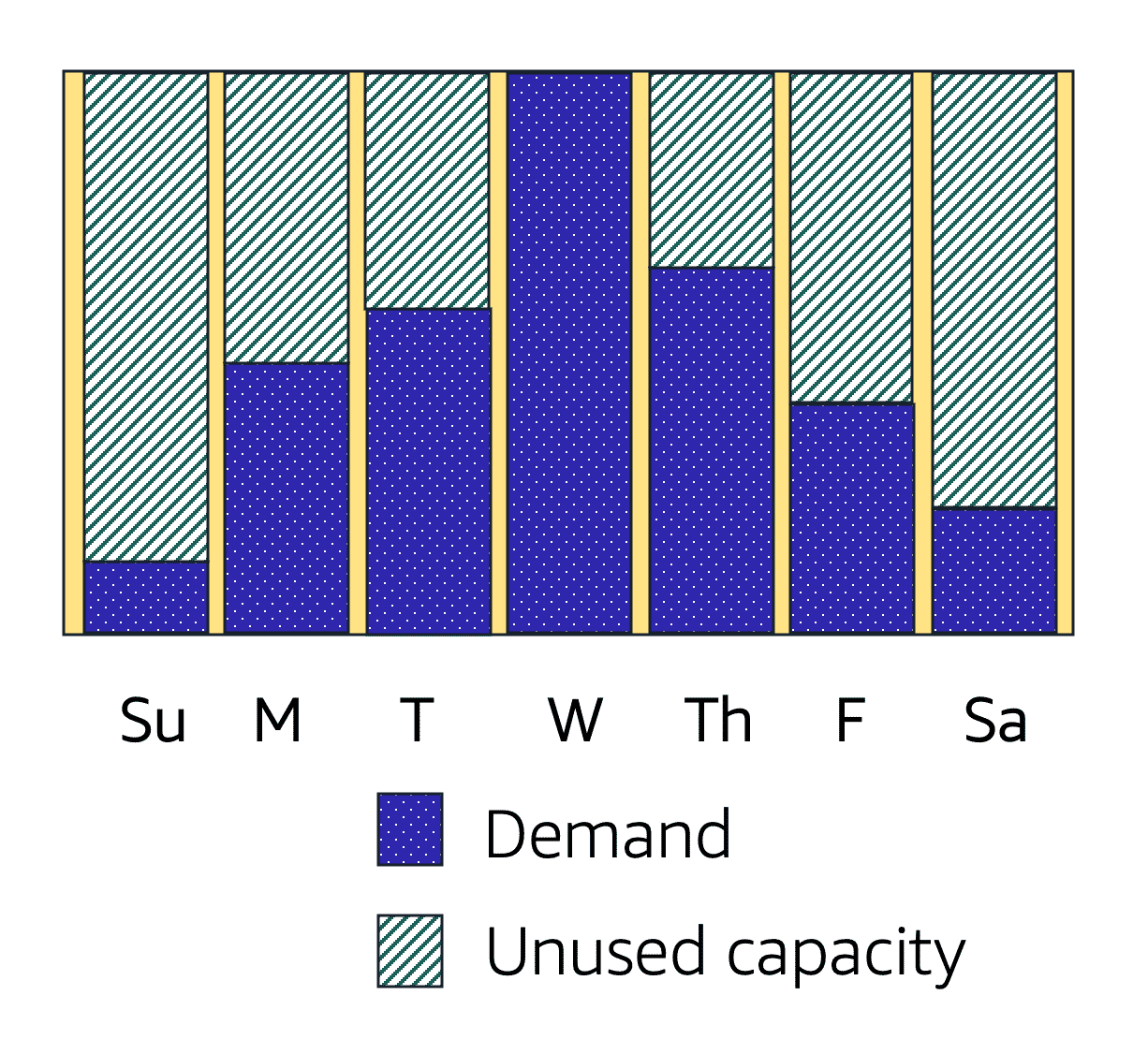
**Scalability**

**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.

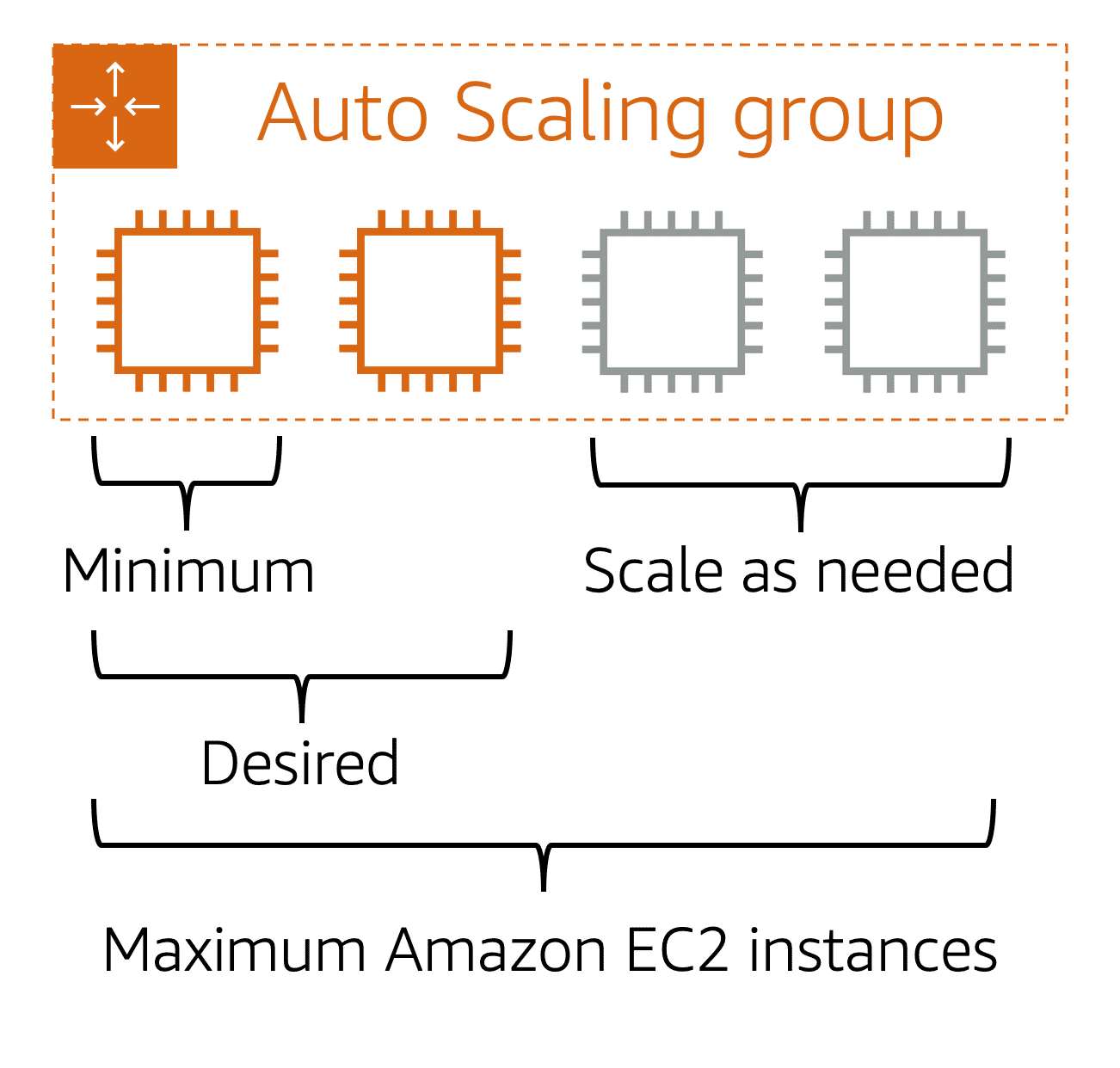
To scale faster, you can use dynamic scaling and predictive scaling together.

Play Video

**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.

**Directing traffic with Elastic Load Balancing**

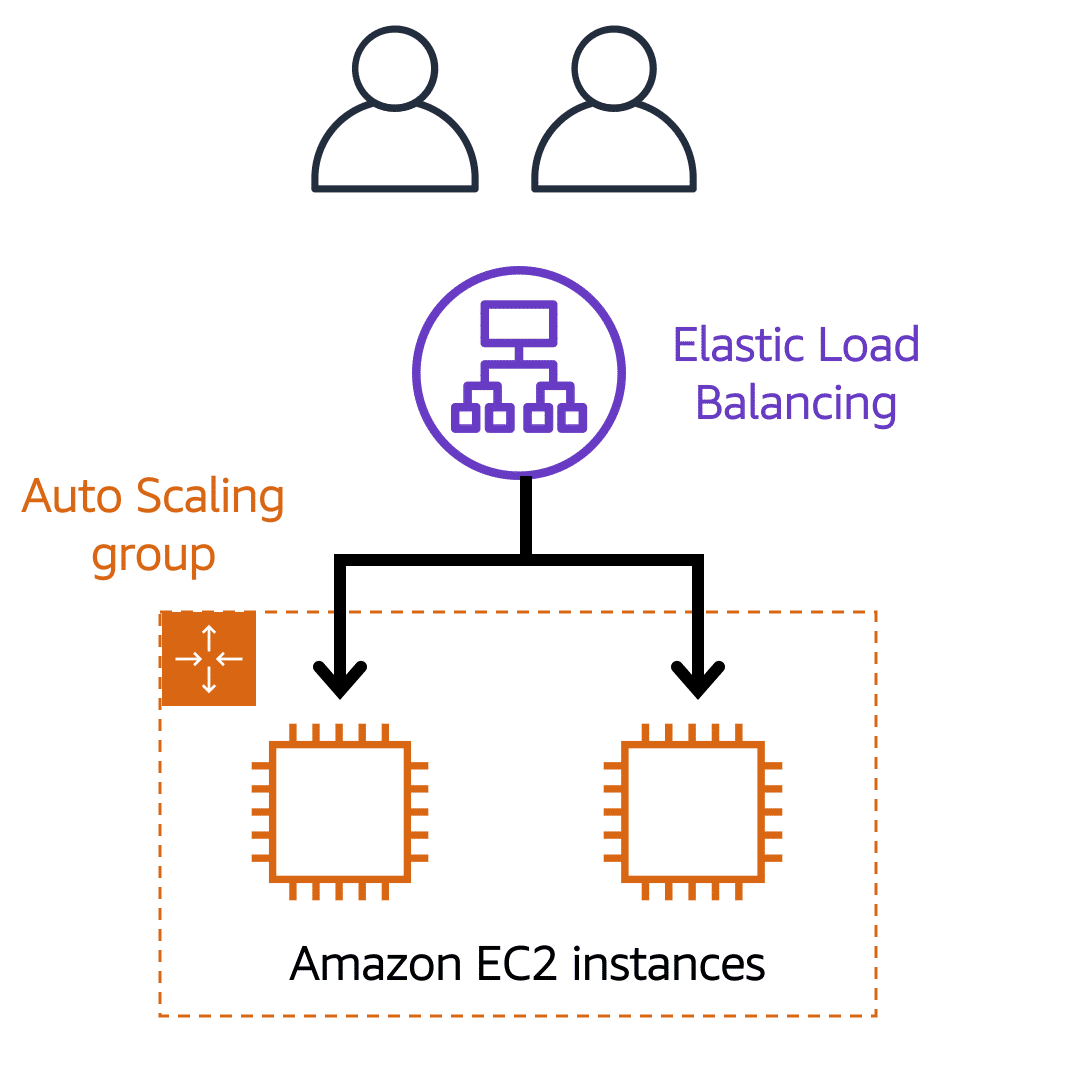
**Elastic Load Balancing**

**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.

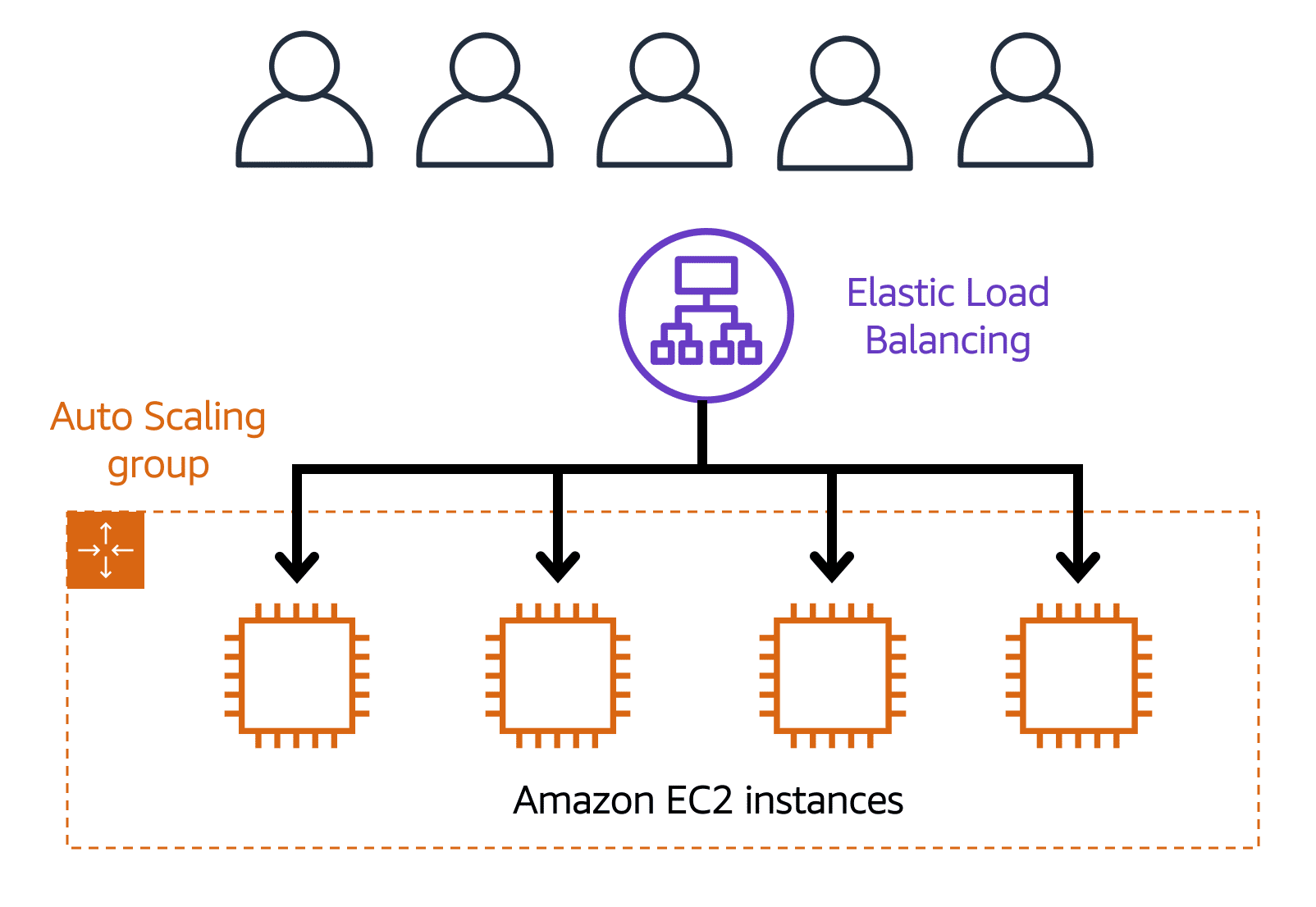
**Example: Elastic Load Balancing**



**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.



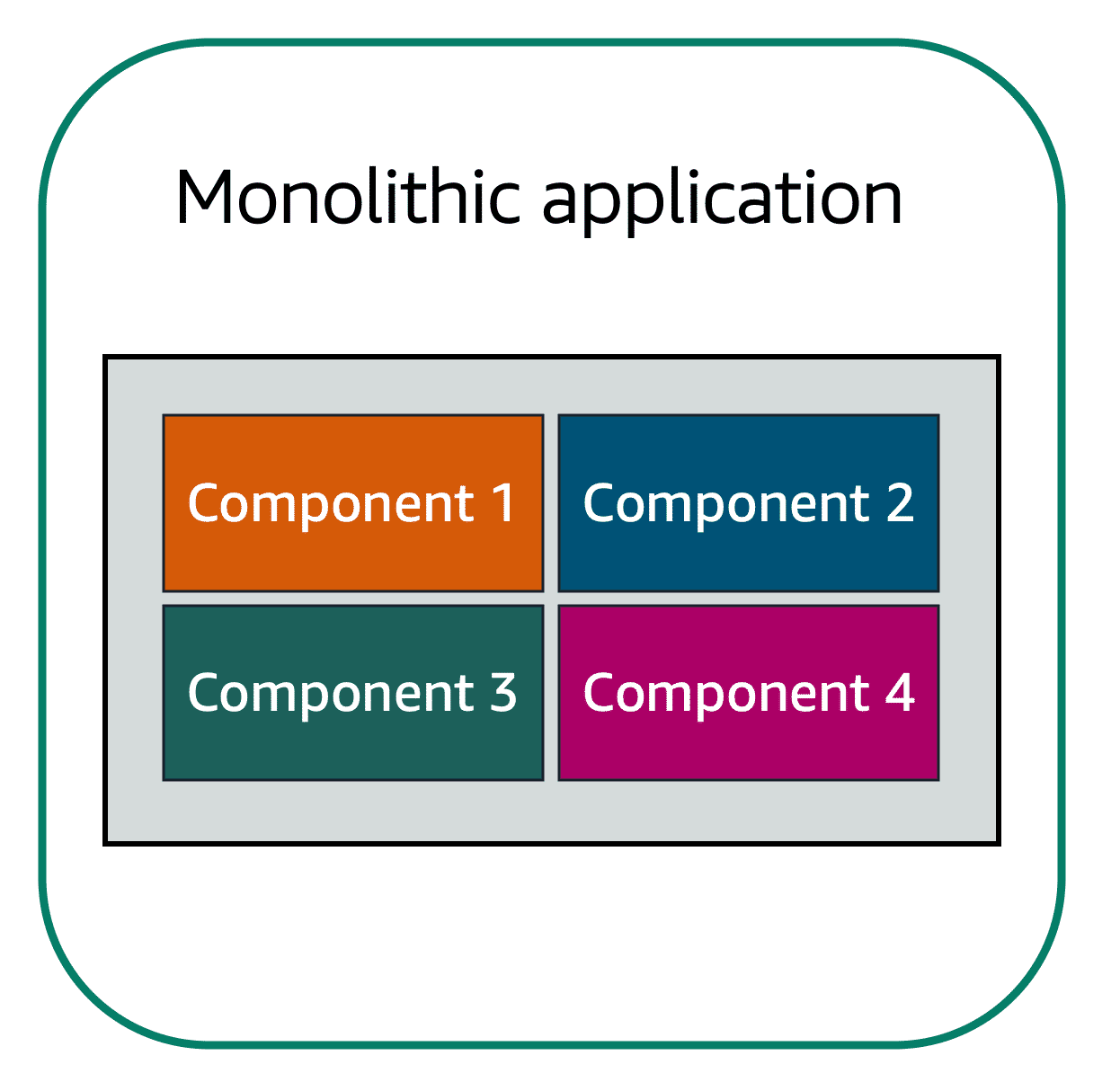
**High-demand period**

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**

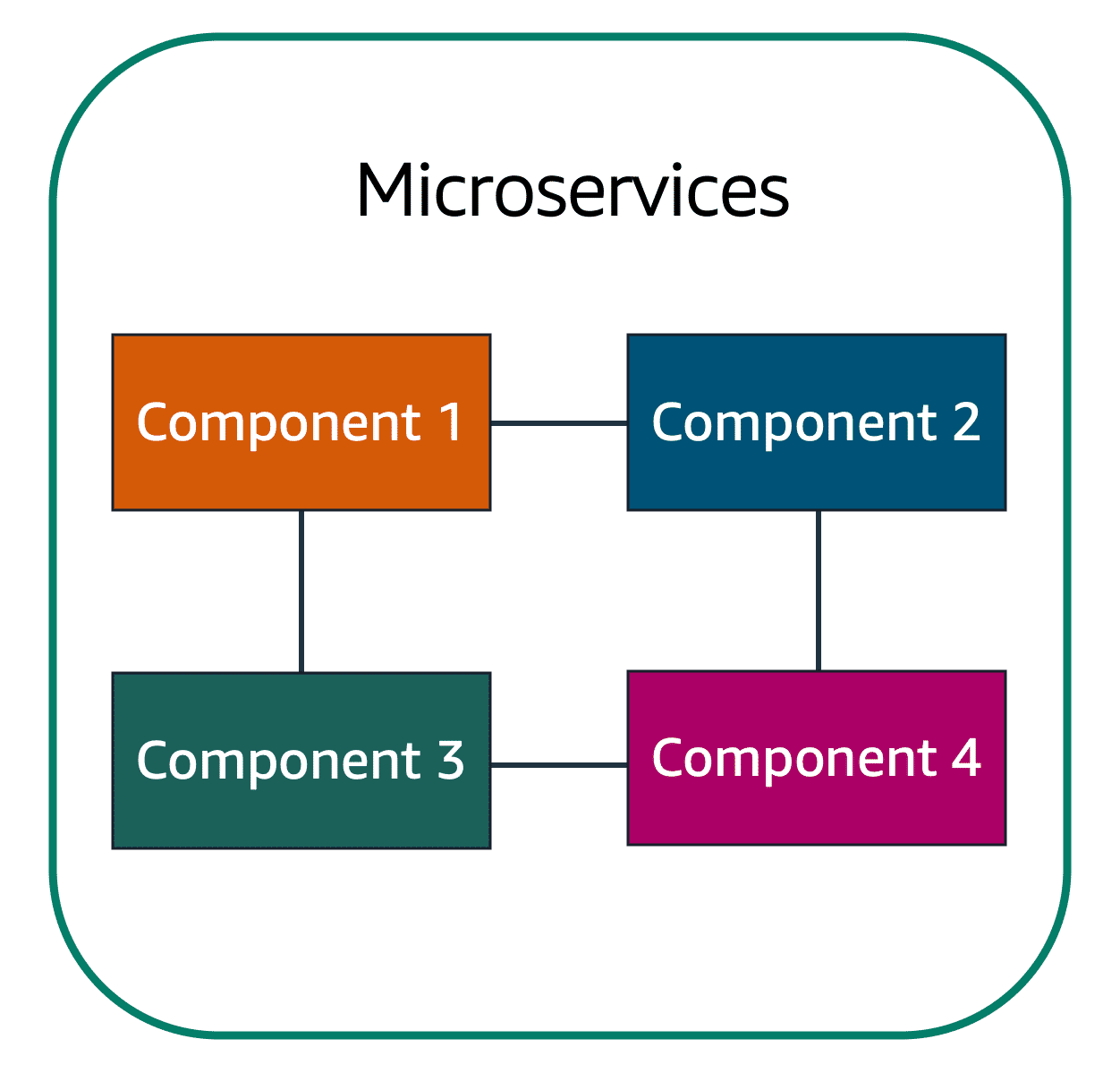


Applications are made of multiple components. The components communicate with each other to transmit data, fulfill requests, and keep the application running.

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.

In the next lesson, you will learn more about AWS Lambda.

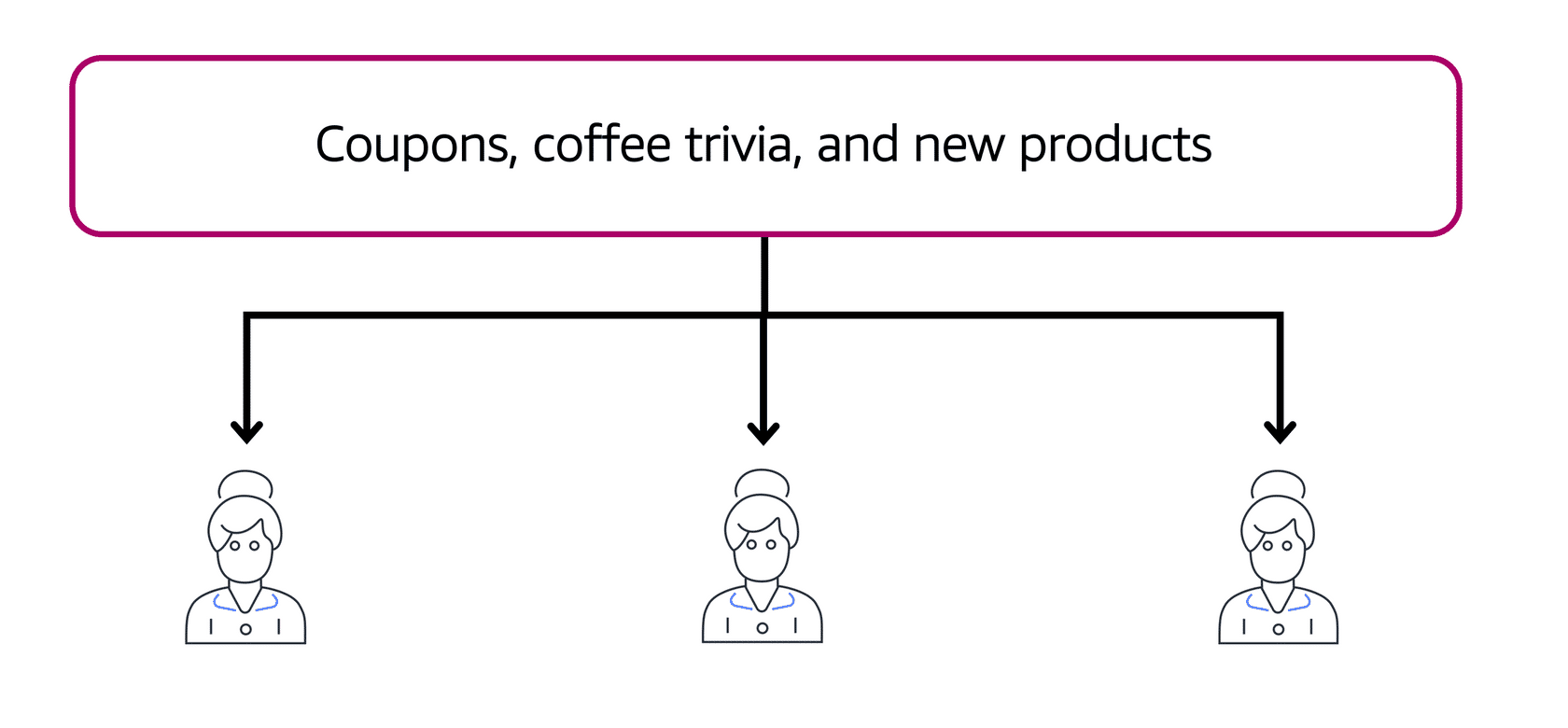
## Example: Amazon SNS

To review an example of how to use Amazon SNS, select **Start**.

START

**Step 1**

## Publishing updates from a single topic

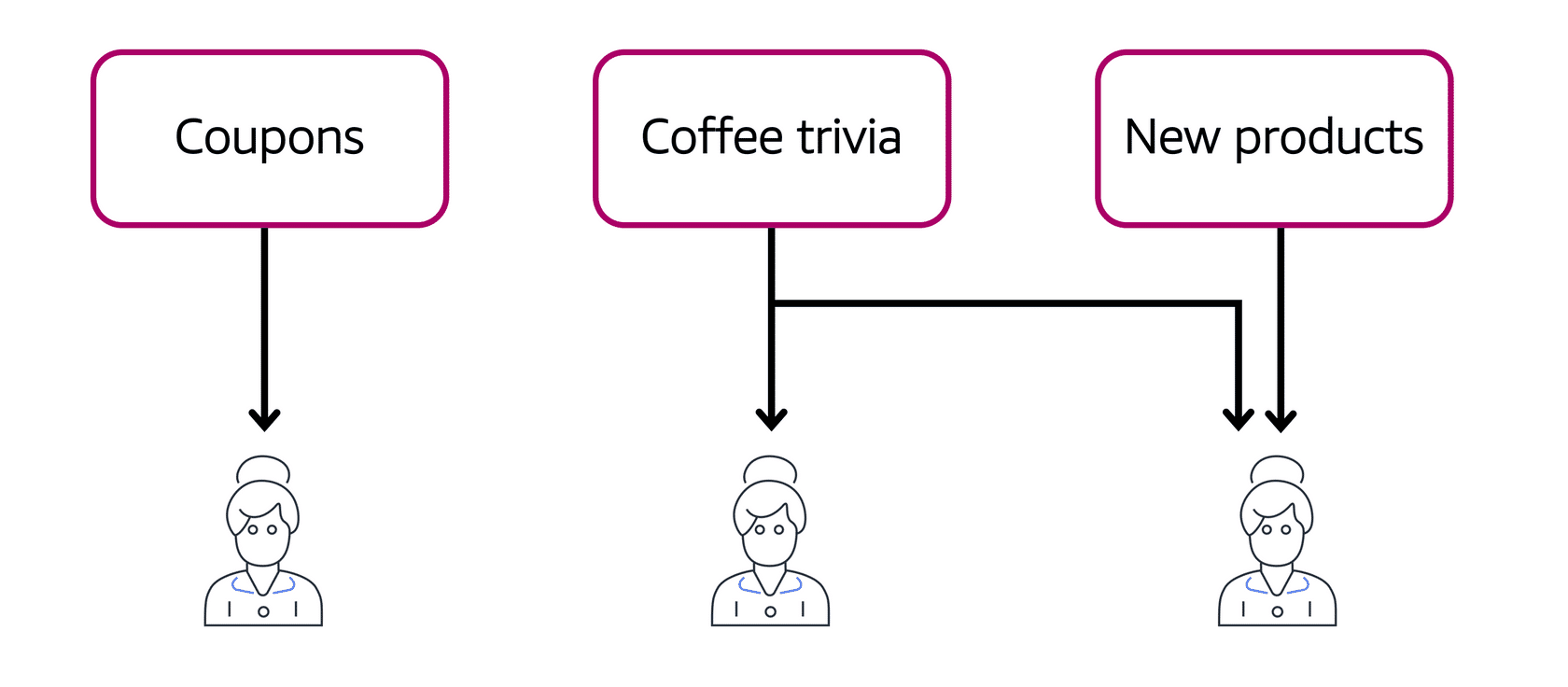


Suppose that the coffee shop has a single newsletter that includes updates from all areas of its business. It includes topics such as coupons, coffee trivia, and new products. All of these topics are grouped because this is a single newsletter. All customers who subscribe to the newsletter receive updates about coupons, coffee trivia, and new products.

After a while, some customers express that they would prefer to receive separate newsletters for only the specific topics that interest them. The coffee shop owners decide to try this approach.

**Step 2**

## Publishing updates from multiple topics



Now, instead of having a single newsletter for all topics, the coffee shop has broken it up into three separate newsletters. Each newsletter is devoted to a specific topic: coupons, coffee trivia, and new products.

Subscribers will now receive updates immediately for only the specific topics to which they have subscribed.

It is possible for subscribers to subscribe to a single topic or to multiple topics. For example, the first customer subscribes to only the coupons topic, and the second subscriber subscribes to only the coffee trivia topic. The third customer subscribes to both the coffee trivia and new products topics.

Although this example from the coffee shop involves subscribers who are people, 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.

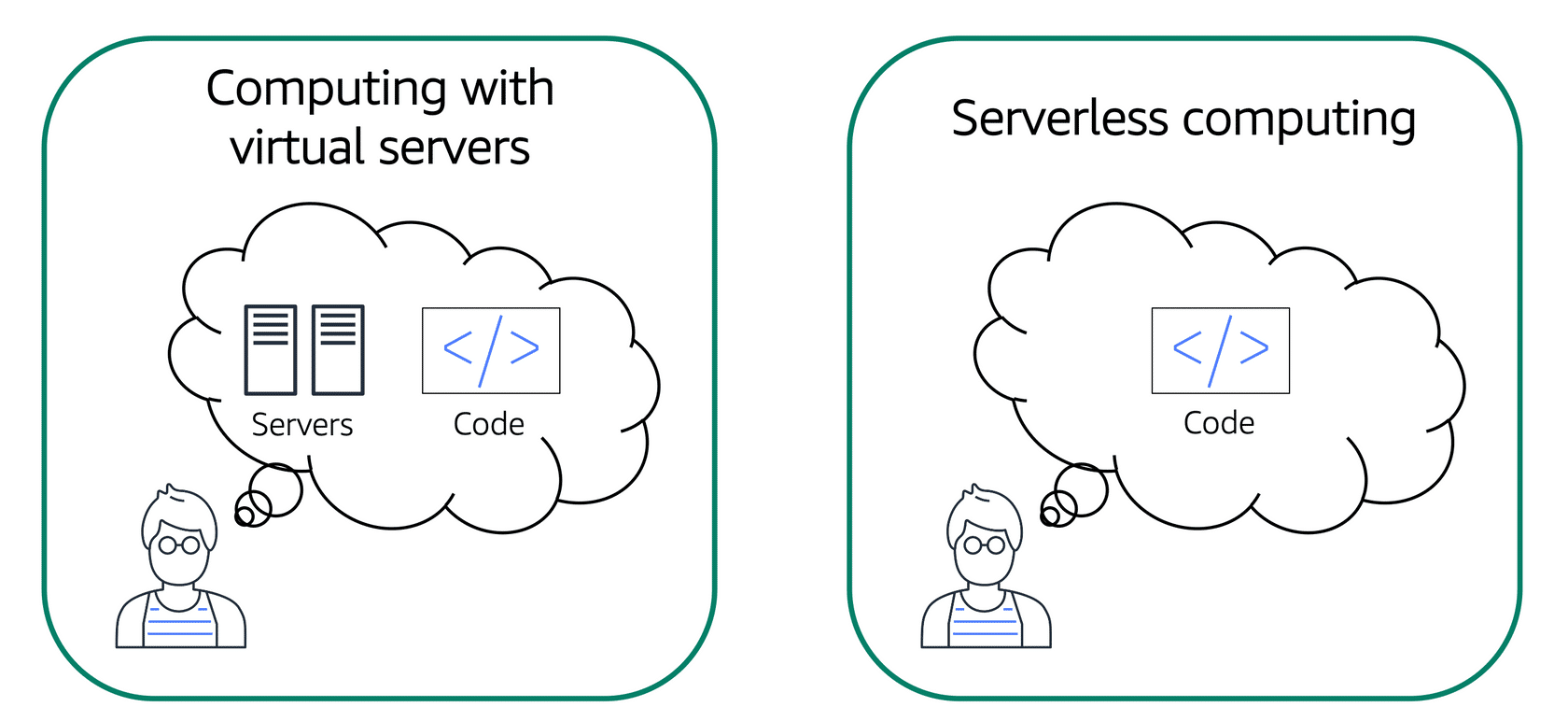
**Serverless computing**

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:

11.Provision instances (virtual servers).

2,Upload your code.

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**.

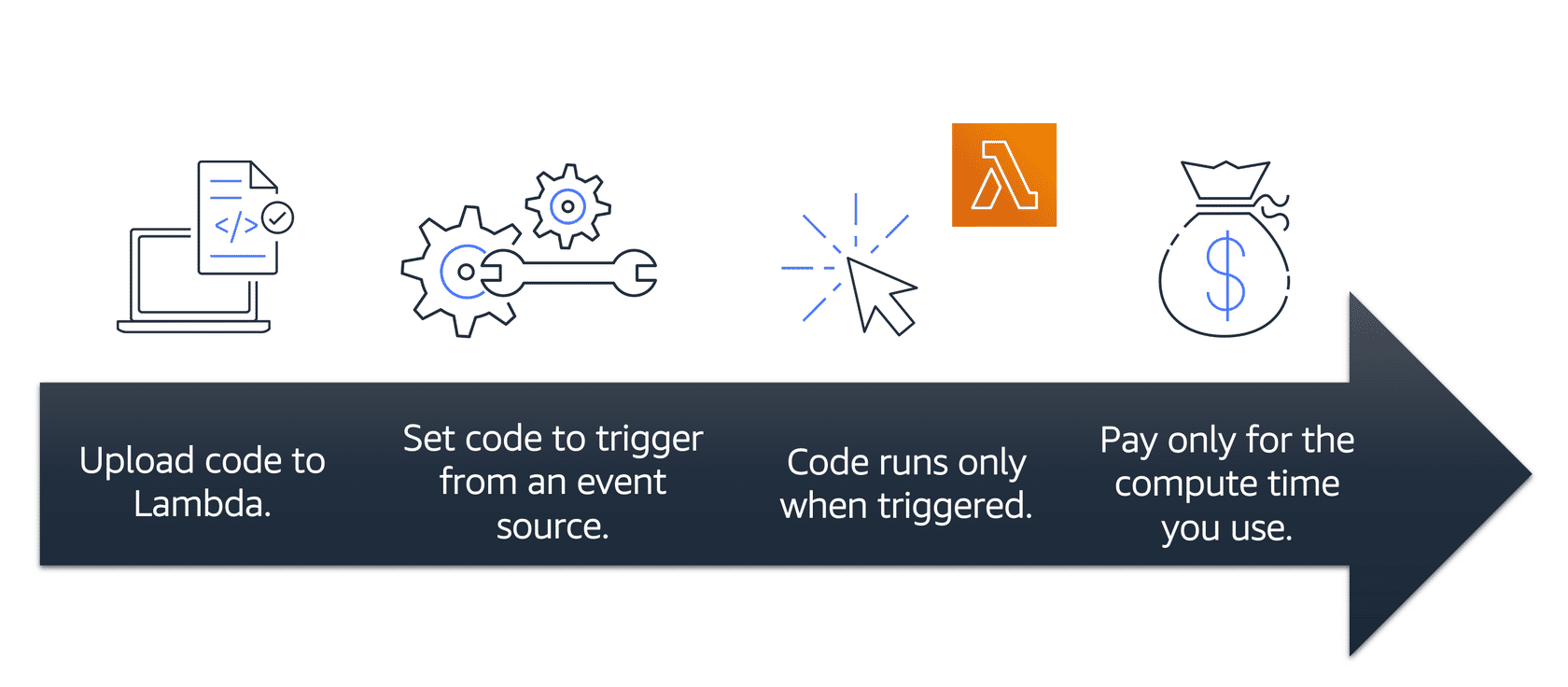
**AWS Lambda**

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

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.

**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.

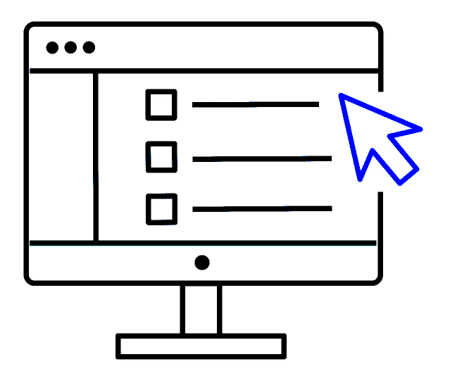
# How to provision AWS resources

**Ways to interact with AWS services**

To learn about each category, select each tab.

[**AWS MANAGEMENT CONSOLE**](https://assets.skillbuilder.aws/files/a/w/aws_prod1_docebosaas_com/1651125600/OIdgeDX5xQW5bKQVMOODFQ/tincan/31d9c0cca79c54bdceaf3e938fd424e97c98c7e8/index.html?enhanced_signature=WXmcXgs-RcnYk05UJNFXJDTvEAWXNLmAxKoyJp2yyVk&endpoint=https%3A%2F%2Fassets.skillbuilder.aws%2Flrs-api%2F&auth=Basic%20LzRjMDhlZGJjLTBjZGItNDU3Yy1iYWI5LWVkZDQ3ZDVmMTZjNTokMnkkMTMkQmlQaWRjVEZzMUl6anUxRjdwOUJBZVlSNkJ0YU83a29JNlp4Q1RiMGNaRmswQ0xiVHZGVmE%3D&actor=%7B%22mbox%22%3A%22mailto%3Amounika.rapelly%40capgemini.com%22%2C%22name%22%3A%22mounika+rapelly%22%7D&registration=045803a5-a821-4e91-bd09-009132e65ab8&activity_id=http%3A%2F%2FeMs3fVuk5fRrdzRW22BTIyGPEJ5VTs6k_rise&Accept-Language=en&course_id=134&content_token=045803a5-a821-4e91-bd09-009132e65ab8&session_context=lms&host=explore.skillbuilder.aws&course_code=DIG-BF-100-CECPEB-10-EN&course_id=134&username=4c08edbc-0cdb-457c-bab9-edd47d5f16c5&user_id=1621345&hash=991d7667a27fba61c75fadc325b90c3641c8b6eb3a57d18d0fe6bafdecc36fba)

The **AWS Management Console** is a web-based interface for accessing and managing AWS services. You can quickly access recently used services and search for other services by name, keyword, or acronym. The console includes wizards and automated workflows that can simplify the process of completing tasks.  
  
You can also use the AWS Console mobile application to perform tasks such as monitoring resources, viewing alarms, and accessing billing information. Multiple identities can stay logged into the AWS Console mobile app at the same time.



[**AWS COMMAND LINE INTERFACE**](https://assets.skillbuilder.aws/files/a/w/aws_prod1_docebosaas_com/1651125600/OIdgeDX5xQW5bKQVMOODFQ/tincan/31d9c0cca79c54bdceaf3e938fd424e97c98c7e8/index.html?enhanced_signature=WXmcXgs-RcnYk05UJNFXJDTvEAWXNLmAxKoyJp2yyVk&endpoint=https%3A%2F%2Fassets.skillbuilder.aws%2Flrs-api%2F&auth=Basic%20LzRjMDhlZGJjLTBjZGItNDU3Yy1iYWI5LWVkZDQ3ZDVmMTZjNTokMnkkMTMkQmlQaWRjVEZzMUl6anUxRjdwOUJBZVlSNkJ0YU83a29JNlp4Q1RiMGNaRmswQ0xiVHZGVmE%3D&actor=%7B%22mbox%22%3A%22mailto%3Amounika.rapelly%40capgemini.com%22%2C%22name%22%3A%22mounika+rapelly%22%7D&registration=045803a5-a821-4e91-bd09-009132e65ab8&activity_id=http%3A%2F%2FeMs3fVuk5fRrdzRW22BTIyGPEJ5VTs6k_rise&Accept-Language=en&course_id=134&content_token=045803a5-a821-4e91-bd09-009132e65ab8&session_context=lms&host=explore.skillbuilder.aws&course_code=DIG-BF-100-CECPEB-10-EN&course_id=134&username=4c08edbc-0cdb-457c-bab9-edd47d5f16c5&user_id=1621345&hash=991d7667a27fba61c75fadc325b90c3641c8b6eb3a57d18d0fe6bafdecc36fba)To save time when making API requests, you can use the **AWS Command Line Interface (AWS CLI)**. AWS CLI enables you to control multiple AWS services directly from the command line within one tool. AWS CLI is available for users on Windows, macOS, and Linux.

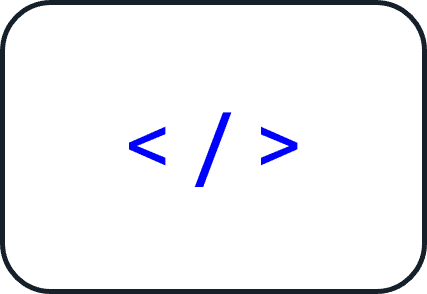
By using AWS CLI, you can automate the actions that your services and applications perform through scripts. For example, you can use commands to launch an Amazon EC2 instance, connect an Amazon EC2 instance to a specific Auto Scaling group, and more.



[**SOFTWARE DEVELOPMENT KITS**](https://assets.skillbuilder.aws/files/a/w/aws_prod1_docebosaas_com/1651125600/OIdgeDX5xQW5bKQVMOODFQ/tincan/31d9c0cca79c54bdceaf3e938fd424e97c98c7e8/index.html?enhanced_signature=WXmcXgs-RcnYk05UJNFXJDTvEAWXNLmAxKoyJp2yyVk&endpoint=https%3A%2F%2Fassets.skillbuilder.aws%2Flrs-api%2F&auth=Basic%20LzRjMDhlZGJjLTBjZGItNDU3Yy1iYWI5LWVkZDQ3ZDVmMTZjNTokMnkkMTMkQmlQaWRjVEZzMUl6anUxRjdwOUJBZVlSNkJ0YU83a29JNlp4Q1RiMGNaRmswQ0xiVHZGVmE%3D&actor=%7B%22mbox%22%3A%22mailto%3Amounika.rapelly%40capgemini.com%22%2C%22name%22%3A%22mounika+rapelly%22%7D&registration=045803a5-a821-4e91-bd09-009132e65ab8&activity_id=http%3A%2F%2FeMs3fVuk5fRrdzRW22BTIyGPEJ5VTs6k_rise&Accept-Language=en&course_id=134&content_token=045803a5-a821-4e91-bd09-009132e65ab8&session_context=lms&host=explore.skillbuilder.aws&course_code=DIG-BF-100-CECPEB-10-EN&course_id=134&username=4c08edbc-0cdb-457c-bab9-edd47d5f16c5&user_id=1621345&hash=991d7667a27fba61c75fadc325b90c3641c8b6eb3a57d18d0fe6bafdecc36fba)

Another option for accessing and managing AWS services is the **software development kits (SDKs)**. SDKs make it easier for you to use AWS services through an API designed for your programming language or platform. SDKs enable you to use AWS services with your existing applications or create entirely new applications that will run on AWS.

To help you get started with using SDKs, AWS provides documentation and sample code for each supported programming language. Supported programming languages include C++, Java, .NET, and more.



**AWS Elastic Beanstalk**

With **AWS Elastic Beanstalk**, you provide code and configuration settings, and Elastic Beanstalk deploys the resources necessary to perform the following tasks:

* Adjust capacity
* Load balancing
* Automatic scaling
* Application health monitoring

**AWS CloudFormation**

With **AWS CloudFormation**, you can treat your infrastructure as code. This means that you can build an environment by writing lines of code instead of using the AWS Management Console to individually provision resources.

AWS CloudFormation provisions your resources in a safe, repeatable manner, enabling you to frequently build your infrastructure and applications without having to perform manual actions. It determines the right operations to perform when managing your stack and rolls back changes automatically if it detects errors.

**Amazon Virtual Private Cloud (Amazon VPC)**

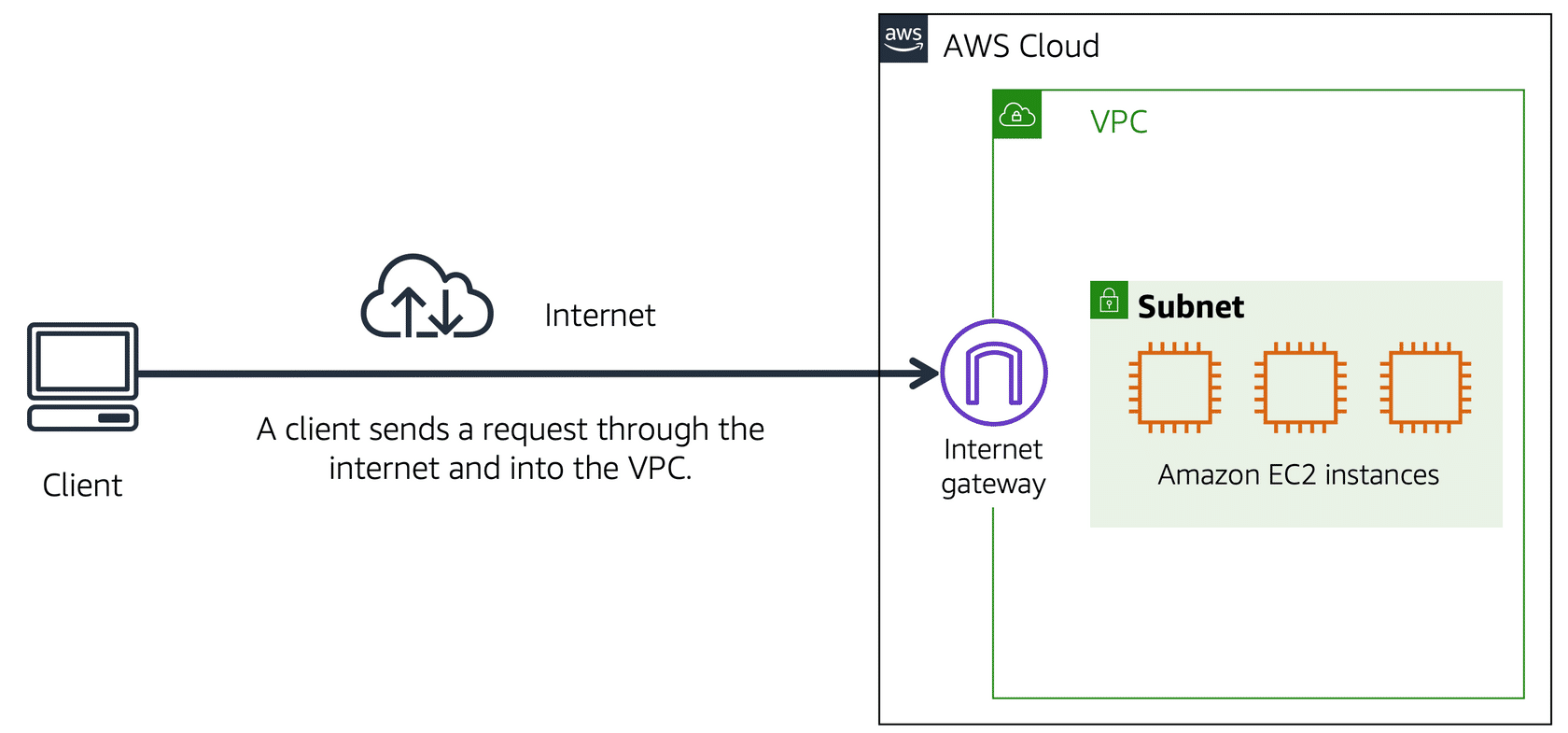
Imagine the millions of customers who use AWS services. Also, imagine the millions of resources that these customers have created, such as Amazon EC2 instances. Without boundaries around all of these resources, network traffic would be able to flow between them unrestricted.

A networking service that you can use to establish boundaries around your AWS resources is [**Amazon Virtual Private Cloud (Amazon VPC)**](https://aws.amazon.com/vpc/).

Amazon VPC enables you to provision an isolated section of the AWS Cloud. In this isolated section, you can launch resources in a virtual network that you define. Within a virtual private cloud (VPC), you can organize your resources into subnets. A **subnet** is a section of a VPC that can contain resources such as Amazon EC2 instances.

**Internet gateway**

To allow public traffic from the internet to access your VPC, you attach an **internet gateway** to the VPC.



An internet gateway is a connection between a VPC and the internet. You can think of an internet gateway as being similar to a doorway that customers use to enter the coffee shop. Without an internet gateway, no one can access the resources within your VPC.

What if you have a VPC that includes only private resources?

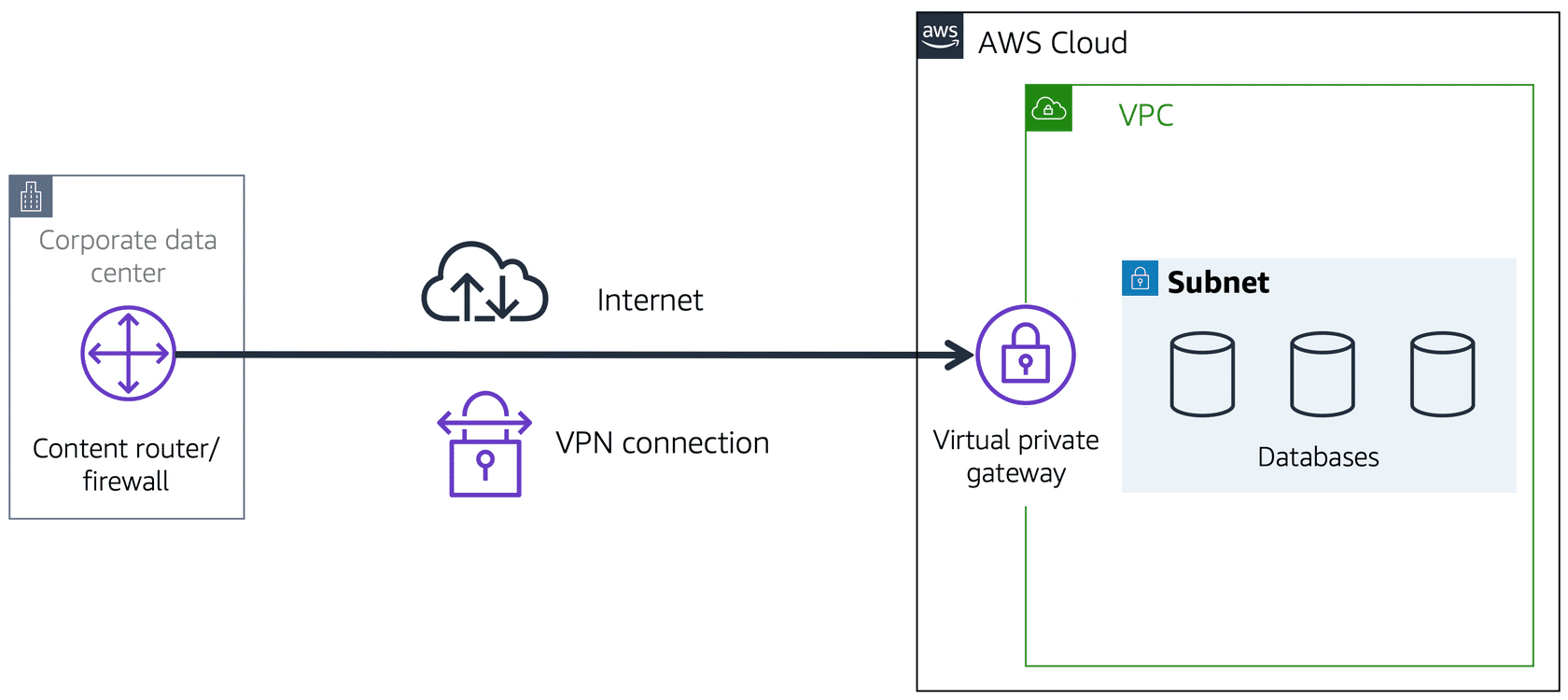
**Virtual private gateway**

To access private resources in a VPC, you can use a **virtual private gateway**.

Here’s an example of how a virtual private gateway works. You can think of the internet as the road between your home and the coffee shop. Suppose that you are traveling on this road with a bodyguard to protect you. You are still using the same road as other customers, but with an extra layer of protection.

The bodyguard is like a virtual private network (VPN) connection that encrypts (or protects) your internet traffic from all the other requests around it.

The virtual private gateway is the component that allows protected internet traffic to enter into the VPC. Even though your connection to the coffee shop has extra protection, traffic jams are possible because you’re using the same road as other customers.



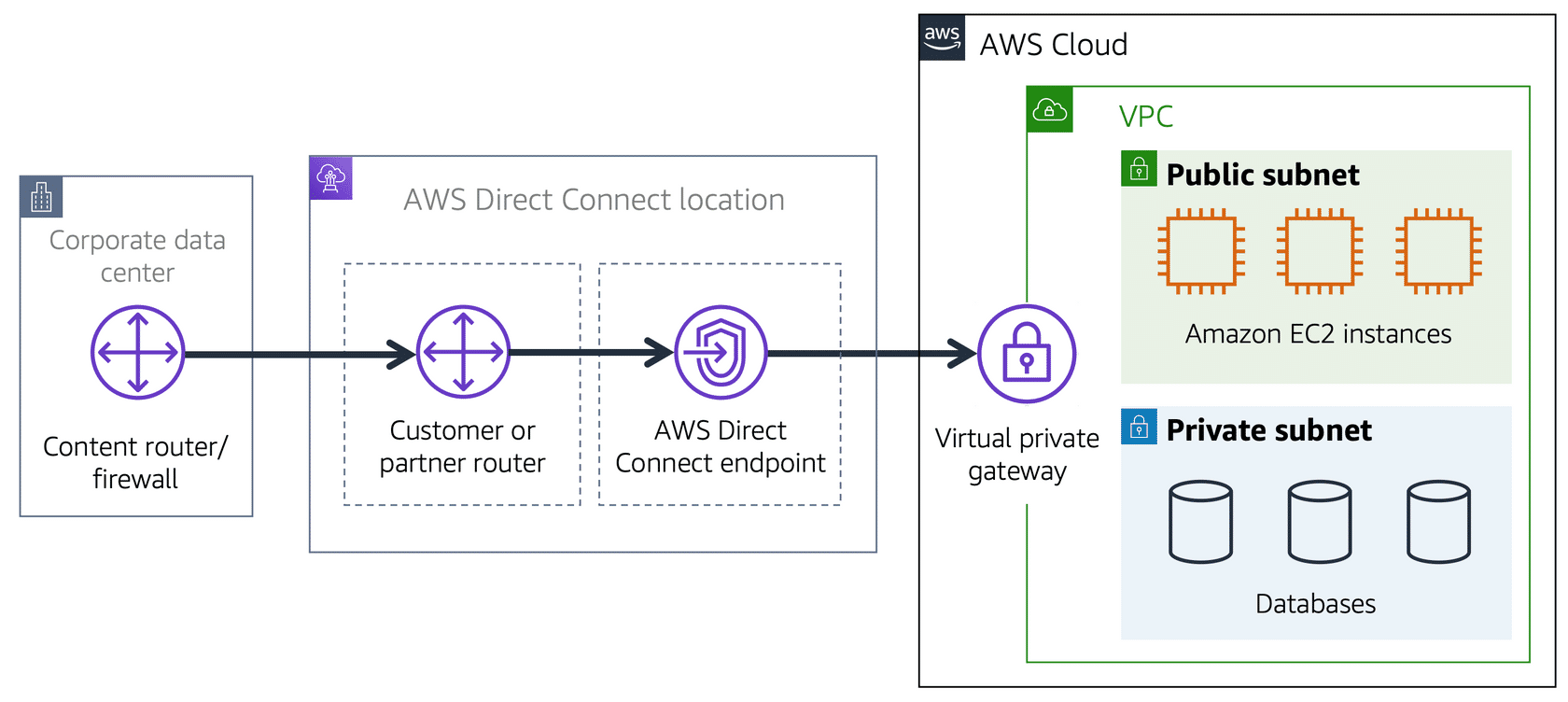
A virtual private gateway enables you to establish a virtual private network (VPN) connection between your VPC and a private network, such as an on-premises data center or internal corporate network. A virtual private gateway allows traffic into the VPC only if it is coming from an approved network.

**AWS Direct Connect**

[**AWS Direct Connect**](https://aws.amazon.com/directconnect/) is a service that enables you to establish a dedicated private connection between your data center and a VPC.

Suppose that there is an apartment building with a hallway directly linking the building to the coffee shop. Only the residents of the apartment building can travel through this hallway.

This private hallway provides the same type of dedicated connection as AWS Direct Connect. Residents are able to get into the coffee shop without needing to use the public road shared with other customers.



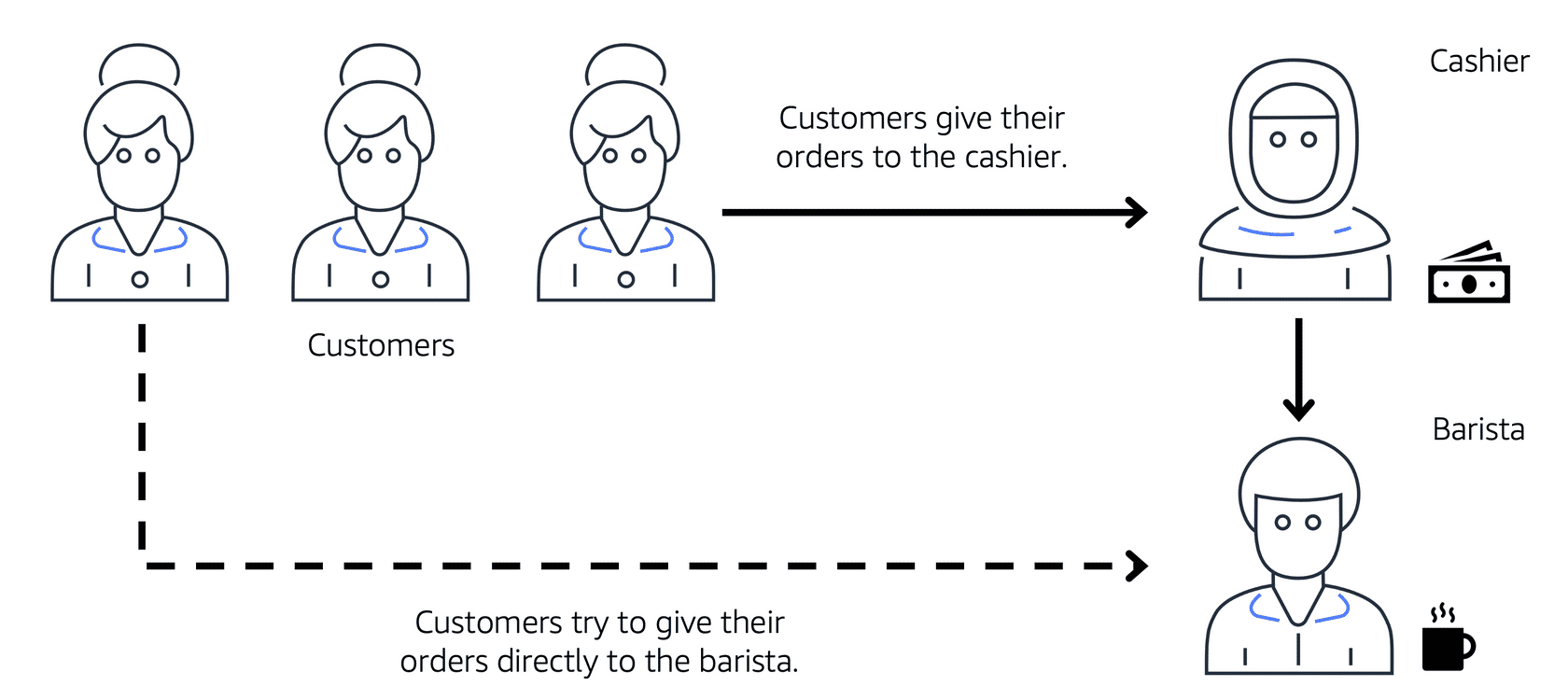
The private connection that AWS Direct Connect provides helps you to reduce network costs and increase the amount of bandwidth that can travel through your network.

**Subnets and network access control lists**

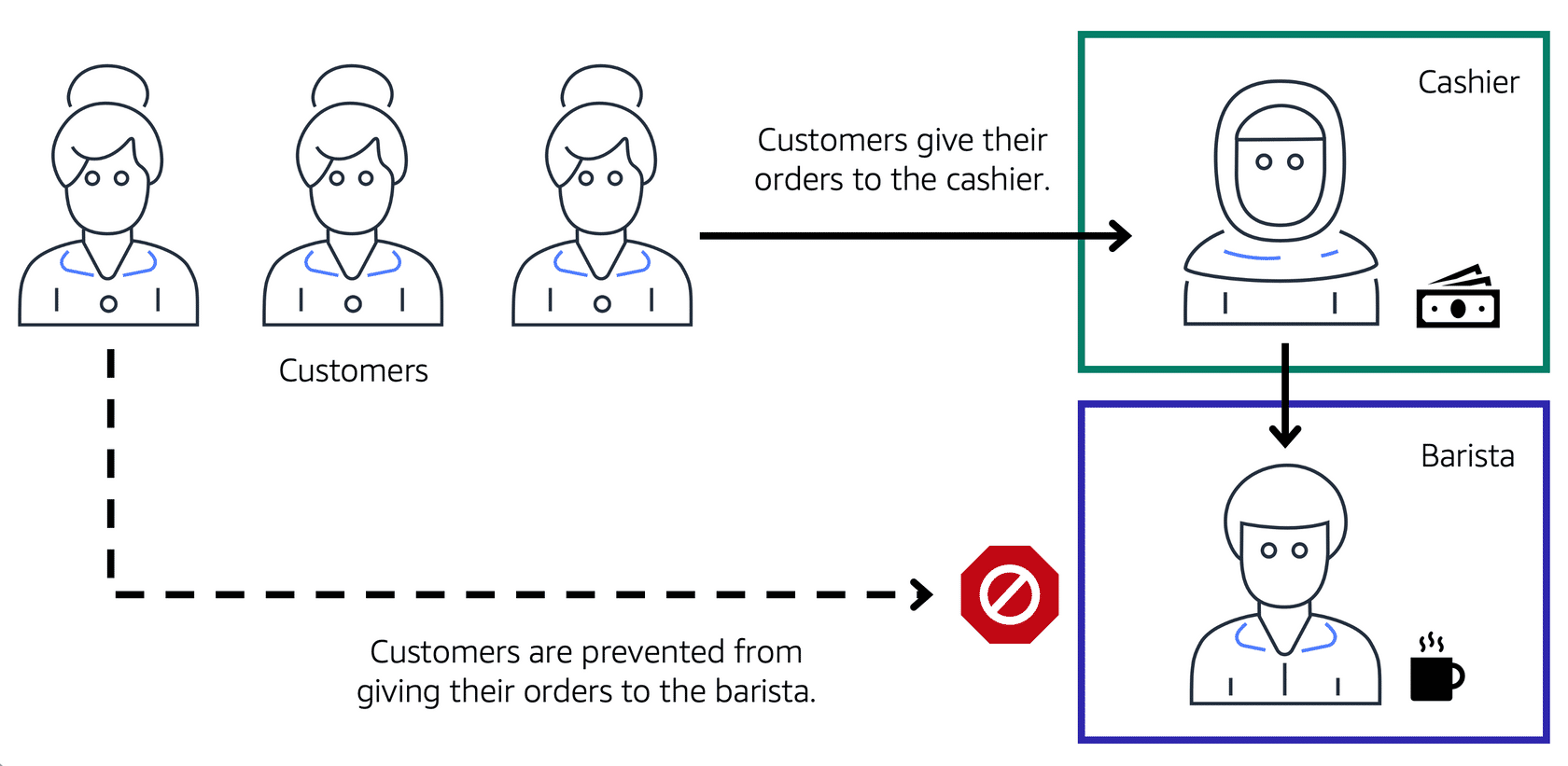
To learn more about the role of subnets within a VPC, review the following example from the coffee shop.

First, customers give their orders to the cashier. The cashier then passes the orders to the barista. This process allows the line to keep running smoothly as more customers come in.

Suppose that some customers try to skip the cashier line and give their orders directly to the barista. This disrupts the flow of traffic and results in customers accessing a part of the coffee shop that is restricted to them.



To fix this, the owners of the coffee shop divide the counter area by placing the cashier and the barista in separate workstations. The cashier’s workstation is public facing and designed to receive customers. The barista’s area is private. The barista can still receive orders from the cashier but not directly from customers.

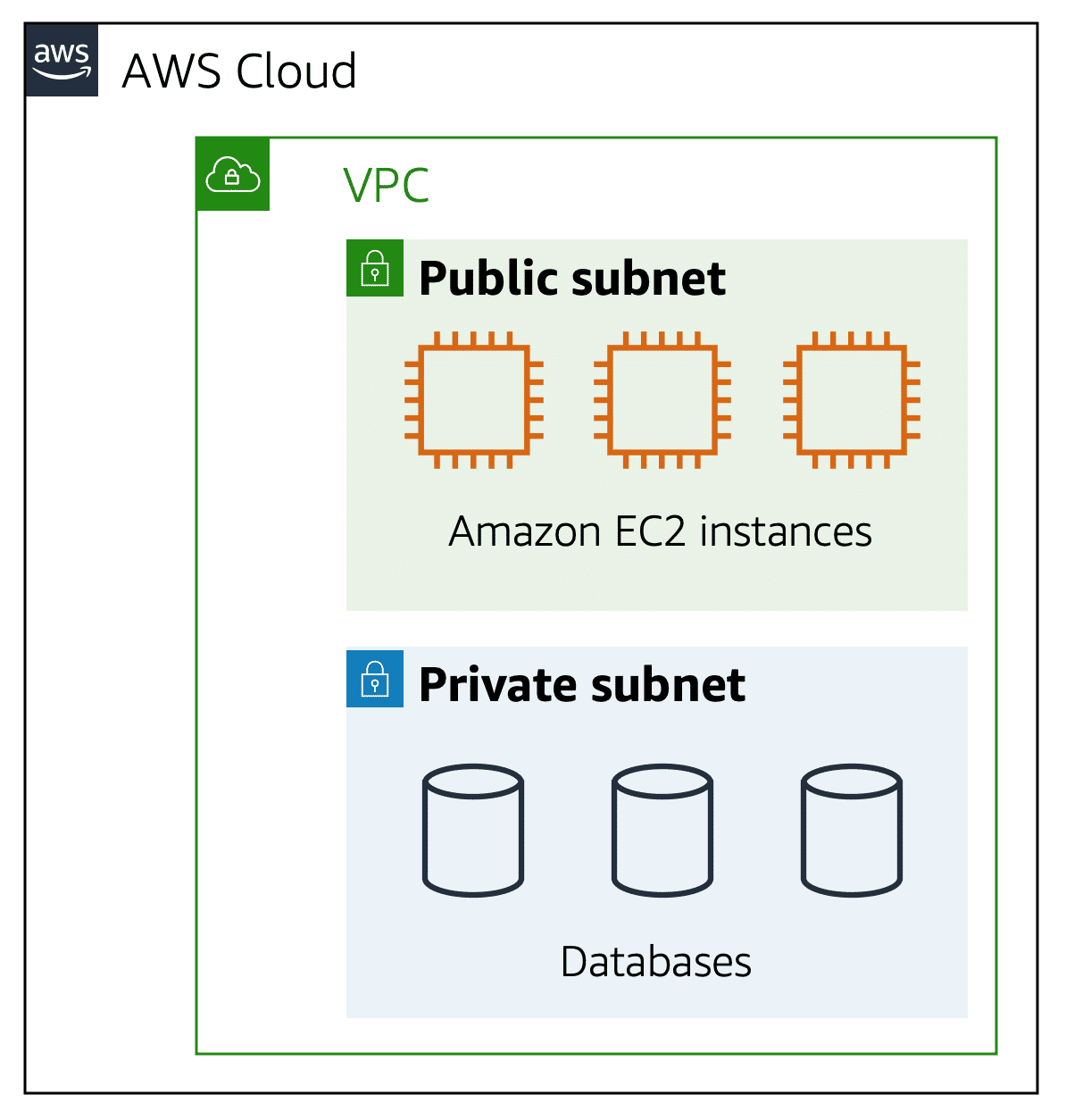


This is similar to how you can use AWS networking services to isolate resources and determine exactly how network traffic flows.

In the coffee shop, you can think of the counter area as a VPC. The counter area divides into two separate areas for the cashier’s workstation and the barista’s workstation. In a VPC, **subnets** are separate areas that are used to group together resources.

**Subnets**

A subnet is a section of a VPC in which you can group resources based on security or operational needs. Subnets can be public or private.



**Public subnets** contain resources that need to be accessible by the public, such as an online store’s website.

**Private subnets** contain resources that should be accessible only through your private network, such as a database that contains customers’ personal information and order histories.

In a VPC, subnets can communicate with each other. For example, you might have an application that involves Amazon EC2 instances in a public subnet communicating with databases that are located in a private subnet.

**Network traffic in a VPC**

When a customer requests data from an application hosted in the AWS Cloud, this request is sent as a packet. A **packet** is a unit of data sent over the internet or a network.

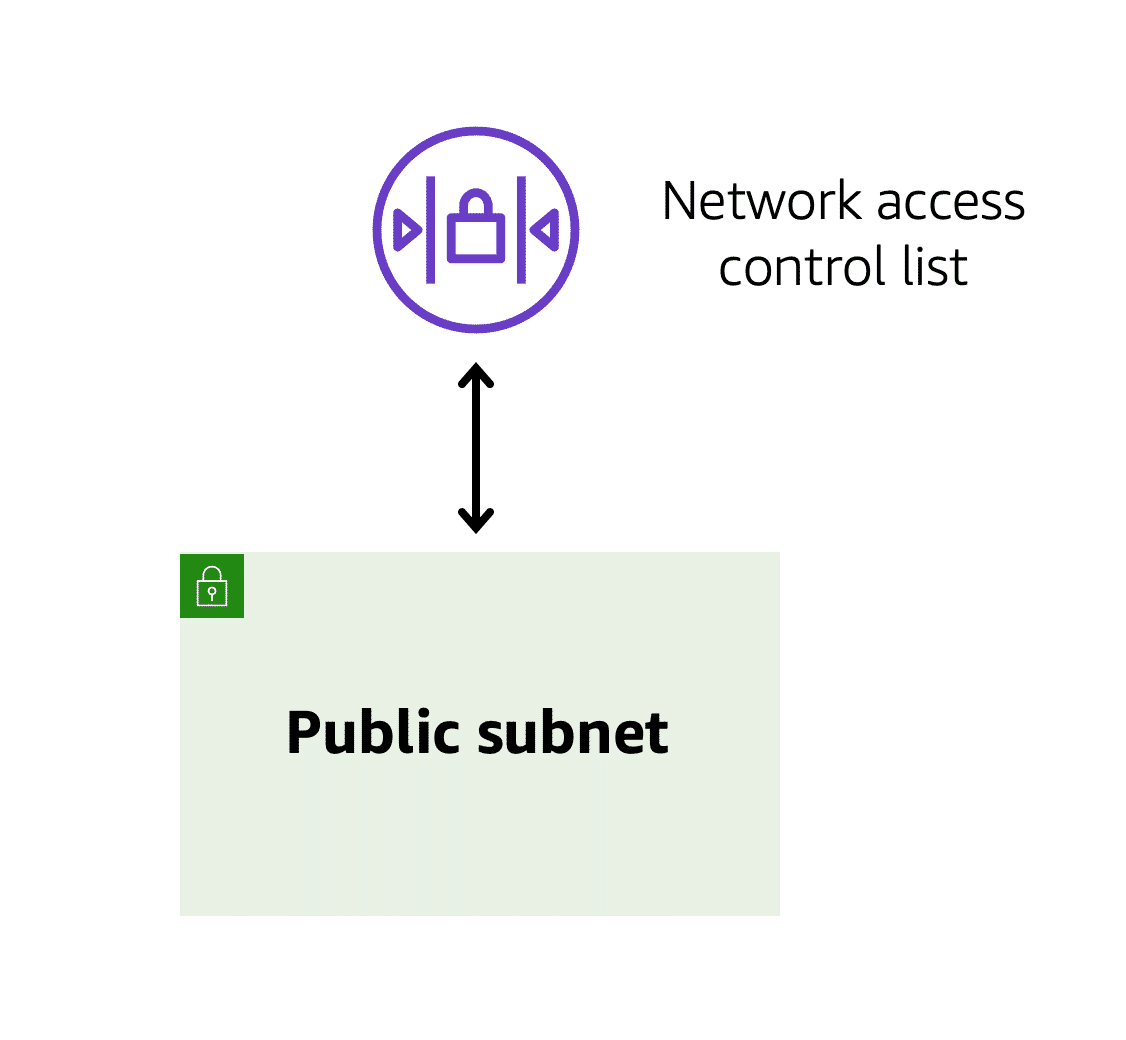
It enters into a VPC through an internet gateway. Before a packet can enter into a subnet or exit from a subnet, it checks for permissions. These permissions indicate who sent the packet and how the packet is trying to communicate with the resources in a subnet.

The VPC component that checks packet permissions for subnets is a [**network access control list (ACL)**](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html).

**Network access control lists (ACLs)**

A network access control list (ACL) is a virtual firewall that controls inbound and outbound traffic at the subnet level.

For example, step outside of the coffee shop and imagine that you are in an airport. In the airport, travelers are trying to enter into a different country. You can think of the travelers as packets and the passport control officer as a network ACL. The passport control officer checks travelers’ credentials when they are both entering and exiting out of the country. If a traveler is on an approved list, they are able to get through. However, if they are not on the approved list or are explicitly on a list of banned travelers, they cannot come in.



Each AWS account includes a default network ACL. When configuring your VPC, you can use your account’s default network ACL or create custom network ACLs.

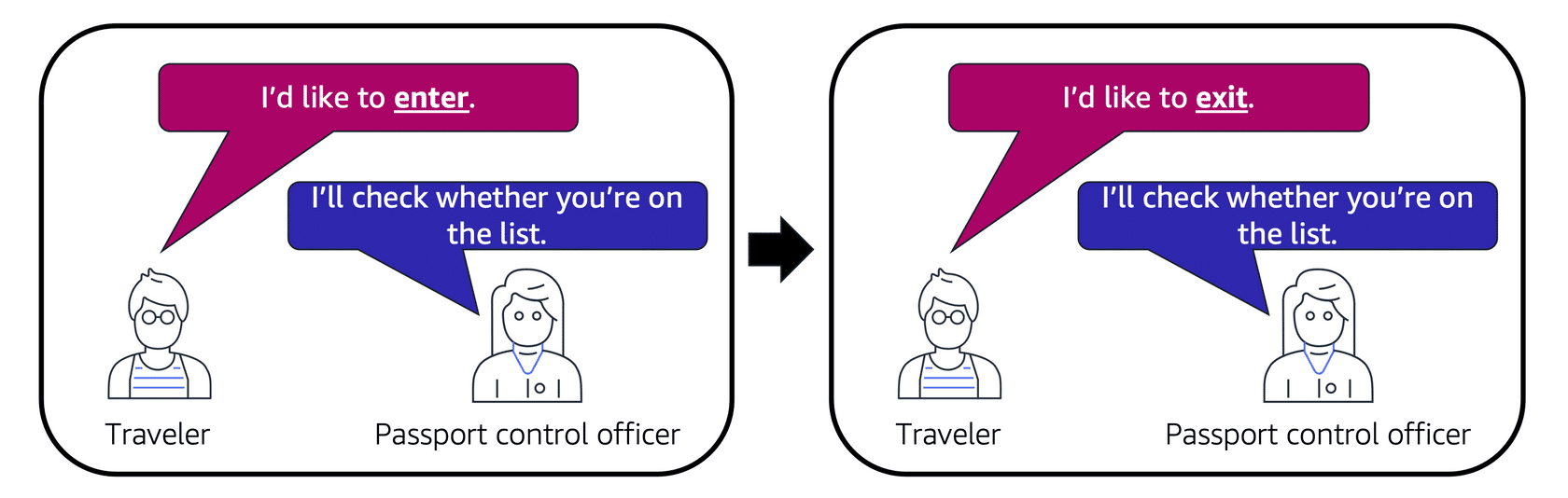
By default, your account’s default network ACL allows all inbound and outbound traffic, but you can modify it by adding your own rules. For custom network ACLs, all inbound and outbound traffic is denied until you add rules to specify which traffic to allow. Additionally, all network ACLs have an explicit deny rule. This rule ensures that if a packet doesn’t match any of the other rules on the list, the packet is denied.

**Stateless packet filtering**

Network ACLs perform **stateless** packet filtering. They remember nothing and check packets that cross the subnet border each way: inbound and outbound.

Recall the previous example of a traveler who wants to enter into a different country. This is similar to sending a request out from an Amazon EC2 instance and to the internet.

When a packet response for that request comes back to the subnet, the network ACL does not remember your previous request. The network ACL checks the packet response against its list of rules to determine whether to allow or deny.

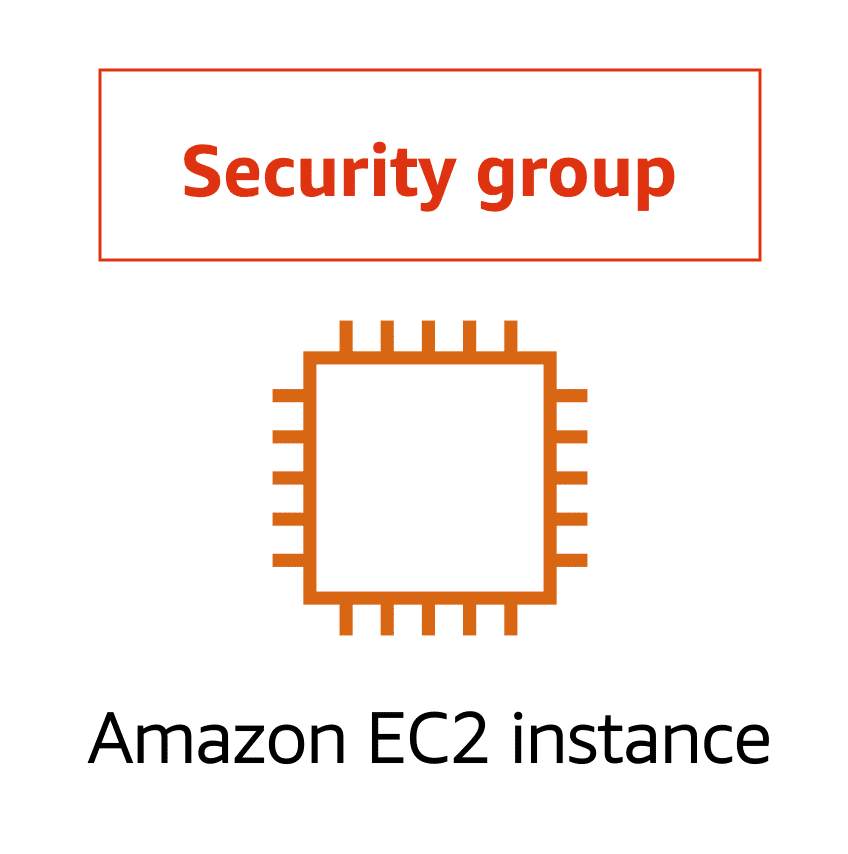


After a packet has entered a subnet, it must have its permissions evaluated for resources within the subnet, such as Amazon EC2 instances.

The VPC component that checks packet permissions for an Amazon EC2 instance is a [**security group**](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html).

**Security groups**

A security group is a virtual firewall that controls inbound and outbound traffic for an Amazon EC2 instance.



By default, a security group denies all inbound traffic and allows all outbound traffic. You can add custom rules to configure which traffic to allow or deny.

For this example, suppose that you are in an apartment building with a door attendant who greets guests in the lobby. You can think of the guests as packets and the door attendant as a security group. As guests arrive, the door attendant checks a list to ensure they can enter the building. However, the door attendant does not check the list again when guests are exiting the building

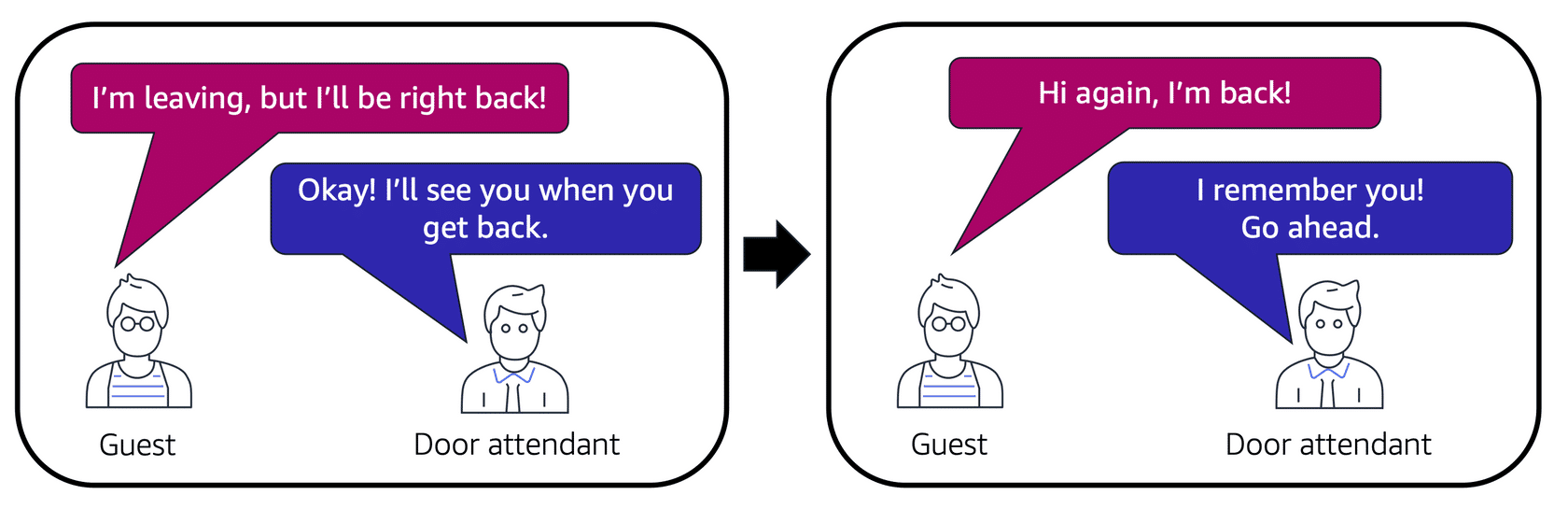
If you have multiple Amazon EC2 instances within a subnet, you can associate them with the same security group or use different security groups for each instance.

**Stateful packet filtering**

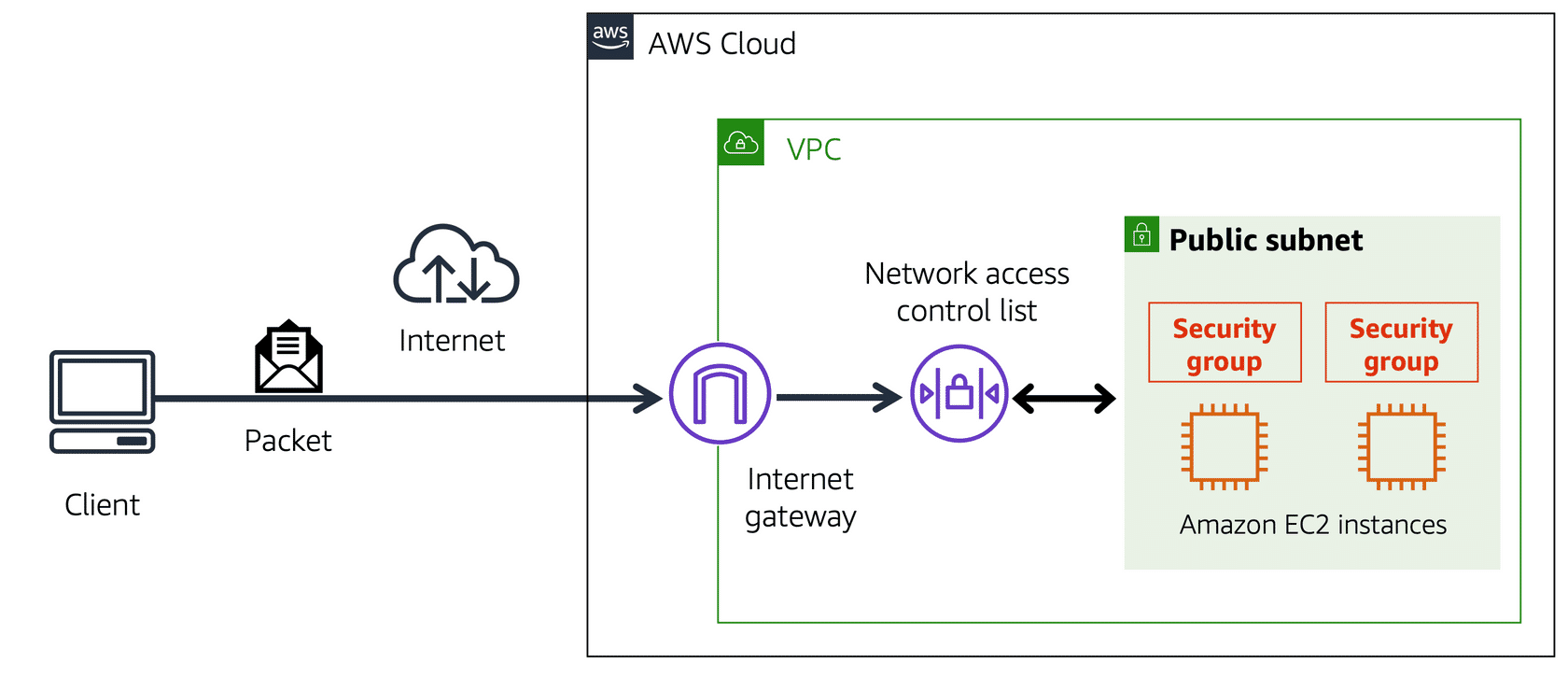
Security groups perform **stateful** packet filtering. They remember previous decisions made for incoming packets.

Consider the same example of sending a request out from an Amazon EC2 instance to the internet.

When a packet response for that request returns to the instance, the security group remembers your previous request. The security group allows the response to proceed, regardless of inbound security group rules.



Both network ACLs and security groups enable you to configure custom rules for the traffic in your VPC. As you continue to learn more about AWS security and networking, make sure to understand the differences between network ACLs and security groups.

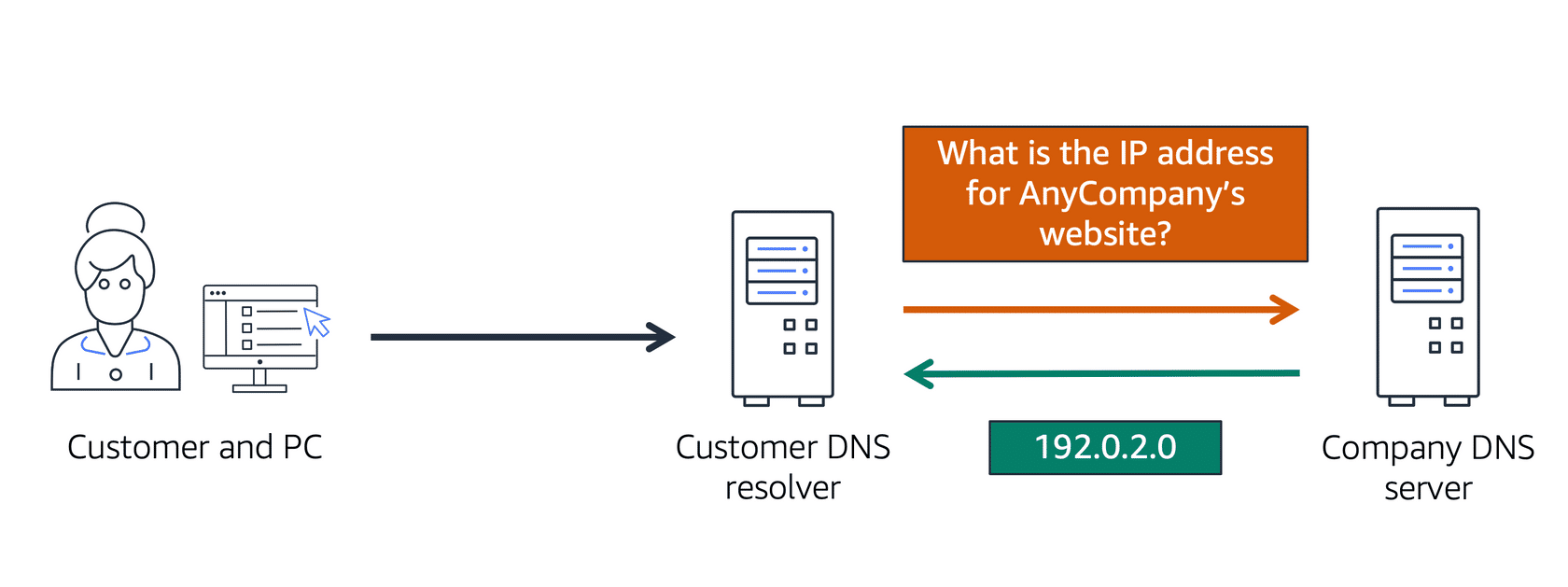


.

**Domain Name System (DNS)**

Suppose that AnyCompany has a website hosted in the AWS Cloud. Customers enter the web address into their browser, and they are able to access the website. This happens because of **Domain Name System (DNS)** resolution. DNS resolution involves a customer DNS resolver communicating with a company DNS server.

You can think of DNS as being the phone book of the internet. DNS resolution is the process of translating a domain name to an IP address.



For example, suppose that you want to visit AnyCompany’s website.

* 1

1

When you enter the domain name into your browser, this request is sent to a customer DNS resolver.

* 2

2

The customer DNS resolver asks the company DNS server for the IP address that corresponds to AnyCompany’s website.

* 3

3

The company DNS server responds by providing the IP address for AnyCompany’s website, 192.0.2.0.

**Amazon Route 53**

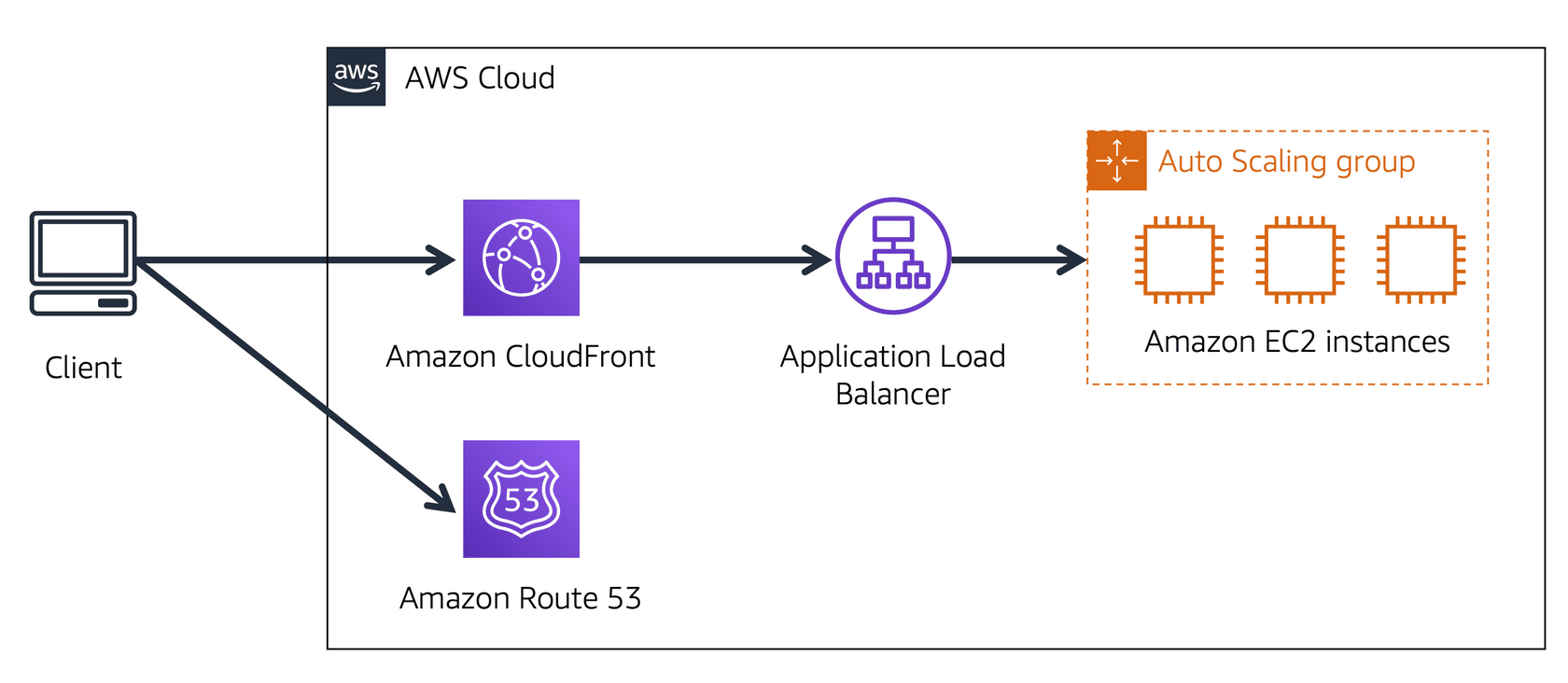
[**Amazon Route 53**](https://aws.amazon.com/route53) is a DNS web service. It gives developers and businesses a reliable way to route end users to internet applications hosted in AWS.

Amazon Route 53 connects user requests to infrastructure running in AWS (such as Amazon EC2 instances and load balancers). It can route users to infrastructure outside of AWS.

Another feature of Route 53 is the ability to manage the DNS records for domain names. You can register new domain names directly in Route 53. You can also transfer DNS records for existing domain names managed by other domain registrars. This enables you to manage all of your domain names within a single location.

In the previous module, you learned about Amazon CloudFront, a content delivery service. The following example describes how Route 53 and Amazon CloudFront work together to deliver content to customers.

**Example: How Amazon Route 53 and Amazon CloudFront deliver content**



Suppose that AnyCompany’s application is running on several Amazon EC2 instances. These instances are in an Auto Scaling group that attaches to an Application Load Balancer.

1.A customer requests data from the application by going to AnyCompany’s website.

2.Amazon Route 53 uses DNS resolution to identify AnyCompany.com’s corresponding IP address, 192.0.2.0. This information is sent back to the customer.

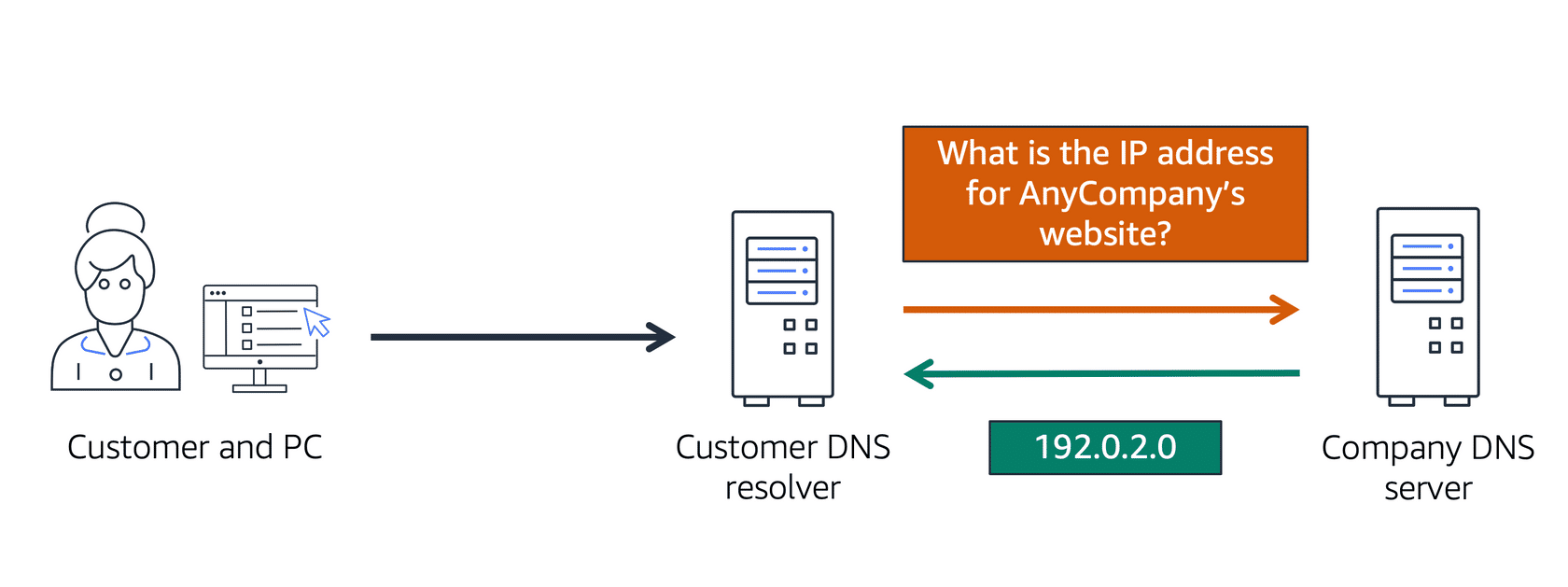
3.The customer’s request is sent to the nearest edge location through Amazon CloudFront.

* 4

**Domain Name System (DNS)**

Suppose that AnyCompany has a website hosted in the AWS Cloud. Customers enter the web address into their browser, and they are able to access the website. This happens because of **Domain Name System (DNS)** resolution. DNS resolution involves a customer DNS resolver communicating with a company DNS server.

You can think of DNS as being the phone book of the internet. DNS resolution is the process of translating a domain name to an IP address.



For example, suppose that you want to visit AnyCompany’s website.

1111

111.When you enter the domain name into your browser, this request is sent to a customer DNS resolver.

2.The customer DNS resolver asks the company DNS server for the IP address that corresponds to AnyCompany’s website.

* 3

3.The company DNS server responds by providing the IP address for AnyCompany’s website, 192.0.2.0.

**Amazon Route 53**

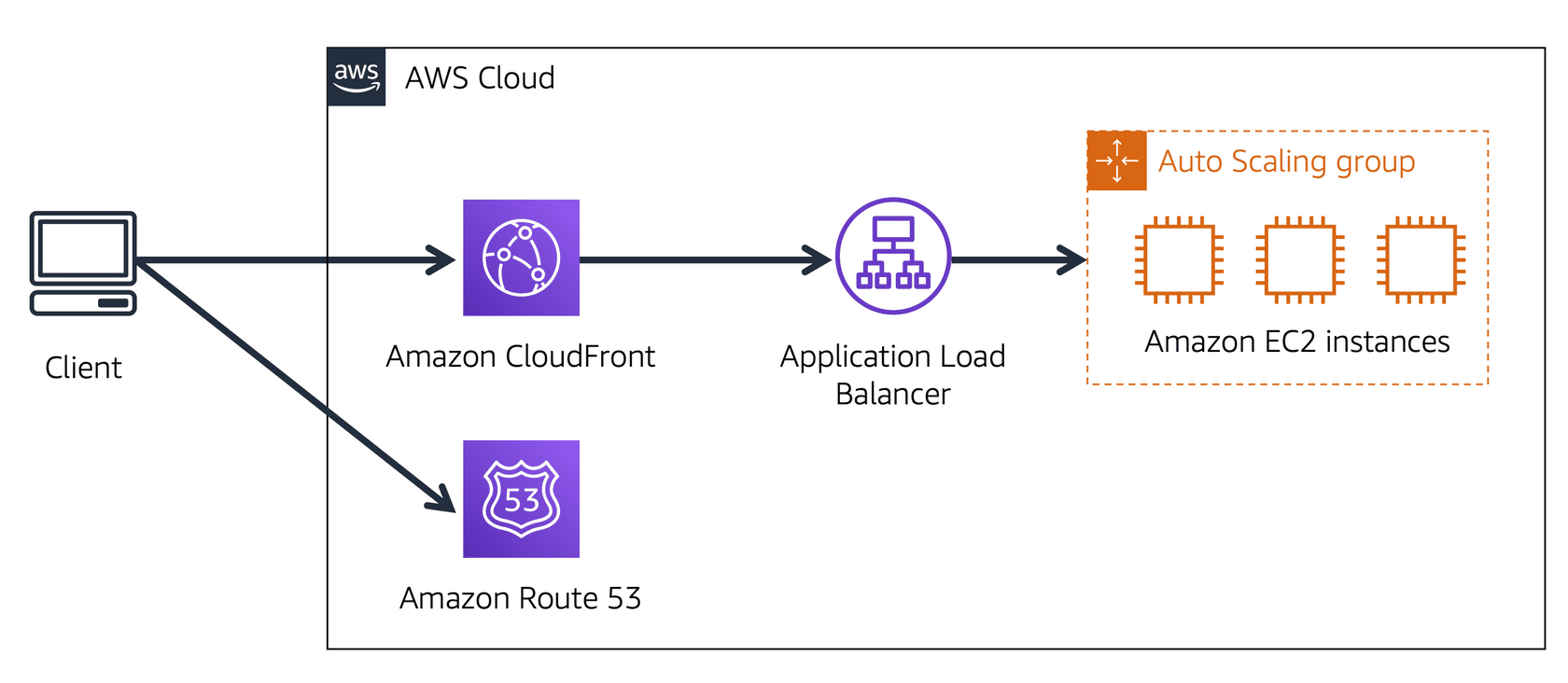
[**Amazon Route 53**](https://aws.amazon.com/route53) is a DNS web service. It gives developers and businesses a reliable way to route end users to internet applications hosted in AWS.

Amazon Route 53 connects user requests to infrastructure running in AWS (such as Amazon EC2 instances and load balancers). It can route users to infrastructure outside of AWS.

Another feature of Route 53 is the ability to manage the DNS records for domain names. You can register new domain names directly in Route 53. You can also transfer DNS records for existing domain names managed by other domain registrars. This enables you to manage all of your domain names within a single location.

In the previous module, you learned about Amazon CloudFront, a content delivery service. The following example describes how Route 53 and Amazon CloudFront work together to deliver content to customers.

**Example: How Amazon Route 53 and Amazon CloudFront deliver content**



Suppose that AnyCompany’s application is running on several Amazon EC2 instances. These instances are in an Auto Scaling group that attaches to an Application Load Balancer.

1.A customer requests data from the application by going to AnyCompany’s website.

2.Amazon Route 53 uses DNS resolution to identify AnyCompany.com’s corresponding IP address, 192.0.2.0. This information is sent back to the customer.

3----3The customer’s request is sent to the nearest edge location through Amazon CloudFront.

* 4

4.Amazon CloudFront connects to the Application Load Balancer, which sends the incoming packet to an Amazon EC2 instance.

Amazon CloudFront connects to the Application Load Balancer, which sends the incoming packet to an Amazon EC2 instance.