AWS Cloud Practisioner

Module 1 Introduction

I'm Blaine Sundrud, AWS Training and Certification. I've been teaching technology for more years than I'm willing to admit. After spending time teaching the newspaper industry, I moved to AWS, where I've taught classes globally on many different disciplines, such as security, cloud architecture, DevOps, big data, AI and ML, and theater history. My momma was a teacher. My daddy was a teacher. My grandpa was a bartender. I was born for this.

Hi, I'm Morgan Willis, a Senior Cloud Technologist at AWS. I started in the IT world about 10 years ago. And along the way, I decided that I was missing something. I missed the help and teaching aspect of IT that I had in my first job in IT support. So, I went into teaching software development in different areas around the U.S. And then I eventually landed here at AWS, where, as a Cloud Technologist, I get to support others in their cloud journey every day.

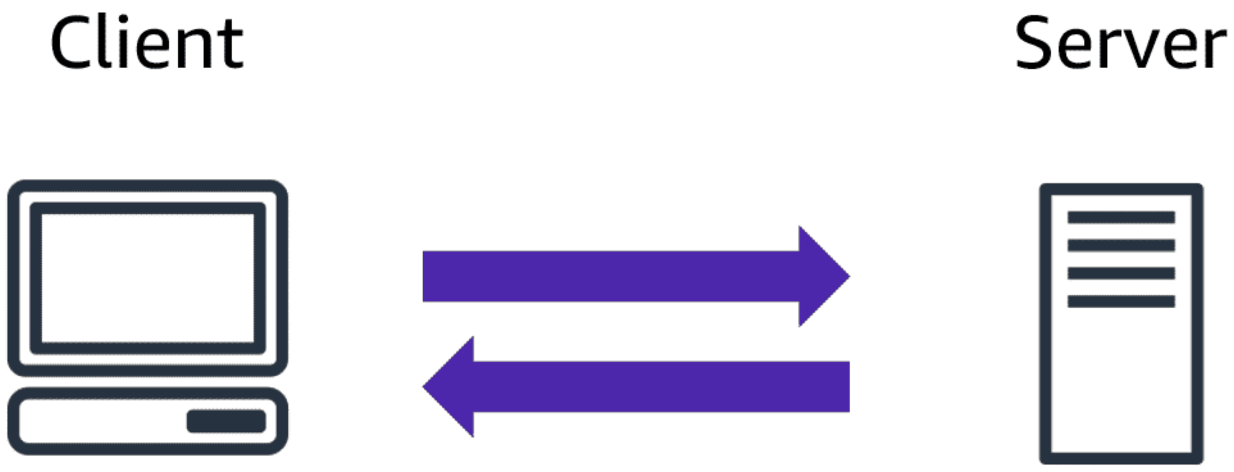
And I'm Rudy Chetty. I come from sunny Cape Town, South Africa, home of biltong, boerewors, and bunny chow. I'm a Solutions Architect, and have been with AWS for over three years. Teaching is my passion. And I can't wait for you to dive into the course, and learn. Thank a lot. and good luck.

This course is gonna cover all the essential information that you need to understand, to be comfortable discussing AWS, to know why it's beneficial to your business.

AWS offers a massive range of services for every business, starting with basic elements, like compute, storage, and network security tools, through complex solutions like blockchain, machine learning, or artificial intelligence, and robot development platforms, all the way through very specialized tool sets, like video production management systems, and orbital satellites you can rent by the minute.

All that, however, is way more than we have time to cover in a foundational class like this one. So let's simplify the conversation by starting with the fundamental cloud compute model.

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.

Almost all modern computing centers around a basic client-server model. Now I know it can be more complicated than that, so let's take a look at our coffee shop.

This coffee shop is going to give us some real world metaphors to help you understand why AWS can change the way your IT operates.

Let's make Morgan the server, the barista. And I am the client, the customer. I make a request. In this case, it is for coffee. Now in the computing world, the request could be anything. It could be rain pattern analysis in South Africa, or the latest x-rays of your knee, or videos of kittens. Whatever is the business, basically a customer makes a request, and with permissions, the server responds to that request. All I want is a caffeinated beverage.

Morgan represents the server part of the client-server model. In AWS, she would be called an Amazon Elastic Compute Cloud, or EC2, an EC2 instance, a virtual server. So from an architectural point of view, the transaction we did is really simple to explain. I, the user, made a request to Morgan, the server. Morgan validated that the request was legitimate, in this case, did I give her money? Then she returned a response, which in this case, is a berry blaster with extra caramel shots.

Now in the real world, applications can get more complicated than just a single transaction with a single server. In a business solution that is more mature, it can get beautifully complex.

To avoid this complexity, we're going to start simple. We will build this discussion out so that it is easy for anyone to understand how these concepts build on each other. So, by the end, those complex concepts, they'll be easy to understand. Let's start with a key concept to AWS, and that is, you only pay for what you use.

This principle makes sense when you run a coffee shop. Employees are only paid when they're in the store working. If Rudy and Morgan are off the clock, well then they don't get paid. The store owner simply decides how many baristas are needed and then just pays for the hours they work. For example, the coffee shop is about to release a new drink, the Pumpkin Monster Spice. In anticipation of this launch, you could always staff your shop with a dozen baristas all day long, just in case you suddenly get an unexpected rush at some point in the day. Only, let's be honest. For most of your day, you don't have near enough customers to justify paying for all those employees.

And yet, the is exactly what happens in an on-premises data center. You can't just snap your fingers and triple your capacity. At AWS, you don't pre-pay for anything. And you don't have to worry about capacity constraints.

When you need instances, or baristas, you just click a button, and you have them. And when you don't need them, another click, and they go away, and you stop paying for them. The same way you don't pay for employees for hours that they're not working.

So, pay for what you need, becomes the first key value of many for running your business on AWS. And that is really why we're here, to help you understand how AWS is built to help you run your business better.

We hope you stick around for the entire course, as we dive deeper into these concepts, and help launch you on your journey to being a Cloud Practitioner.

Before we get deeper into the pieces and parts of AWS, let's zoom out and get a good working definition of cloud. Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing. Let's break this down. On-demand delivery indicates that AWS has the resources you need, when you need them. You don't need to tell us in advance that you're going to need them. Suddenly you find yourself needing 300 virtual servers. Well, just a few clicks and launch them. Or you need 2000 terabytes of storage. You don't have to tell us in advance, just start using the storage you need, when you need it. Don't need them anymore, just as quickly, you can return them and stop paying immediately. That kind of flexibility is just not possible when you're managing your own data centers.

The idea of IT resources is actually a big part of the AWS philosophy. We often get asked why AWS has so many products and the answer is really simple: Because businesses need them. If there are IT elements that are common across a number of businesses, then this is not a differentiator.

Take a MySQL database as an example. If your business runs a MySQL database, does your ability to install the MySQL engine make you a better company than your competitors? Well, probably not that. Do you keep backups in a way that makes you superior to other players in your vertical? Again, doubtful. The data inside your database, now that's critically different. The way you build your tables and manage the structures, absolutely separates you from the competition. But the engine is just the engine.

At AWS, we call that the undifferentiated heavy lifting of IT. Tasks that are common, often repetitive and ultimately time-consuming; these are the tasks AWS wants to help you with. So you can focus on what makes you unique. Over the internet, seems simple enough, but it implies that you can access those resources using a secure webpage console or programmatically.

No additional contracts or sales calls are needed. With pay-as-you-go pricing, we re-emphasize what we pointed out here in the coffee shop. You don't staff a shop with employees 24 hours a day at the same levels you do during peak hours. In fact, some hours, you might not even staff them at all. So why pay for developer environments, for example, on weekends, if your developers aren't working on the weekends?

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.
To learn more about deployment models, choose each of the following three tabs.

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 center. 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.
To learn more about the benefits, expand each for the following six categories.

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.

Additional resources

To learn more about the concepts that were explored in Module 1, review these resources.

- AWS glossary
- Whitepaper: Overview of Amazon Web Services
- AWS Fundamentals: Overview
- What is cloud computing?

Types of Cloud Computing

What are the types of cloud computing?

Cloud computing is providing developers and IT departments with the ability to focus on what matters most and avoid undifferentiated work like procurement, maintenance, and capacity planning. As cloud computing has grown in popularity, several different models and deployment strategies have emerged to help meet the specific needs of different users. Each type of cloud service, and deployment method, provides you with different levels of control, flexibility, and management. Understanding the differences between traditional cloud computing models—Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS)—and what deployment strategies you can use can help you decide what set of services is right for your needs.

While the industry has traditionally used terms like IaaS, PaaS, and SaaS to group cloud services, at AWS we focus on solutions to your needs, which can span many service types. This page uses the traditional service grouping of IaaS, PaaS, and SaaS to help you decide which set is right for your needs and the deployment strategy that works best for you.

Cloud computing models

Although definitions vary from company to company, traditionally there have been three main models for cloud computing. Each model represents a different aspect of cloud computing.

IaaS

IaaS contains the basic building blocks for cloud IT and typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS vendors can help you with the highest level of flexibility and management control over your IT resources and is the type most similar to existing IT resources that many IT departments and developers are familiar with.

Learn more about IaaS

PaaS

PaaS vendors remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems), and this integration allows you to focus on the deployment and management of your applications. This helps you be more efficient, as you don’t need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Learn more about PaaS

SaaS

SaaS vendors provide you with software applications that are run and managed by the vendor. In most cases, people referring to SaaS are referring to third-party end-user applications. With a SaaS offering you do not have to worry about how the service is maintained or how the

underlying infrastructure is managed; you only need think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions or maintain the servers and operating systems that the email program is running on.
Learn more about SaaS

IT deployment models

Cloud

A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud. Applications in the cloud have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing. Cloud-based applications can be built on low-level infrastructure pieces or can use higher-level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.
Learn more about cloud computing

Hybrid

A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure, to extend and grow an organization's infrastructure into the cloud while connecting cloud resources to internal system. For more information on how AWS can help you with your hybrid deployment, visit Hybrid Cloud with AWS.
Learn more about hybrid cloud

On premises

Deploying resources on premises using virtualization and resource management tools does not provide many of the

- Cloud computing with AWS

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.

1. 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

2. What is another name for on-premises deployment?
   - Private cloud deployment
   - Cloud-based application
   - Hybrid deployment
   - AWS Cloud

3. **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.

Mordule 2 Introduction

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.

In this video, we are going to talk at a high level about a service called Amazon Elastic Compute Cloud or EC2. If you remember from our coffee shop, the employees are a metaphor for the client/server model where a client sends a request to the server; the server does some work, and then sends a response. That example is for the coffee shop. But the same idea applies to other businesses. Your business, whether it be in healthcare, manufacturing, insurance, or delivering video content to millions of users all around the world, are also using this model to deliver products, resources, or data to your end users. And you're going to need servers to power your business and your applications. You need raw compute capacity to host your applications and provide the compute power that your business needs. When you're working with AWS, those servers are virtual. And the service you use to gain access to virtual servers is called EC2.

Using EC2 for compute is highly flexible, cost effective, and quick when you compare it to running your own servers on premises in a data center that you own. The time and money it takes to get up and running with on-premises resources is fairly high. When you own your own fleet of physical servers, you first have to do a bunch of research to see what type of servers you want to buy and how many you'll need. Then you purchase that hardware up front. You'll wait for multiple weeks or months for a vendor to deliver those servers to you. You then take them to a data center that you own or rent to install them, rack and stack them, and wire them all up. Then you make sure that they are secure and powered up and then they're ready to be used. Only then can you begin to host your applications on top of these servers. The worst part is, once you buy these servers you are stuck with them whether you use them or not.

With EC2, it's much easier to get started. AWS took care of the hard part for you already. AWS already built and secured the data centers. AWS has already bought the servers, racked and stacked them, and they are already online ready to be used. AWS is constantly operating a massive amount of compute capacity. And you can use whatever portion of that capacity when you need it. All you have to do is request the EC2 instances you want and they will launch and boot up, ready to be used within a few minutes. Once you're done, you can easily stop or terminate the EC2 instances. You're not locked in or stuck with servers that you don't need or want. Your usage of EC2 instances can vary greatly over time. And you only pay for what you use. Because with EC2, you only pay for running instances, not stopped or terminated instances.

EC2 runs on top of physical host machines managed by AWS using virtualization technology. When you spin up an EC2 instance, you aren't necessarily taking an entire host to yourself. Instead, you are sharing the host with multiple other instances, otherwise known as virtual machines. And a hypervisor running on the host machine is responsible for sharing the underlying physical resources between the virtual machines. This idea of sharing underlying hardware is called multitenancy. The hypervisor is responsible for coordinating this multitenancy and it is managed by AWS. The hypervisor is responsible for isolating the virtual machines from each other as they share resources from the host. This means EC2 instances are secure. Even though they may be sharing resources, one EC2 instance is not aware of any other EC2 instances also on that host. They are secure and separate from each other.

Luckily, this is not something you, yourself, need to set up. But it's important to know the idea of multitenancy and have a high level understanding of how this works. EC2 gives you a great deal of flexibility and control. Not only can you spin up new servers or take them offline at will, but you also have the flexibility and control over the configuration of those instances.

When you provision an EC2 instance, you can choose the operating system based on either Windows or Linux. You can provision thousands of EC2 instances on demand. With a blend of operating systems and configurations to power your business' different applications.

Beyond the OS, you also configure what software you want running on the instance. Whether it's your own internal business applications, simple web apps, or complex web apps, databases or third party software like enterprise software packages, you have complete control over what happens on that instance. EC2 instances are also resizable. You might start with a small instance, realize the application you are running is starting to max out that server, and then you can give that instance more memory and more CPU. Which is what we call vertically scaling an instance.

In essence, you can make instances bigger or smaller whenever you need to. You also control the networking aspect of EC2. So what type of requests make it to your server and if they are publicly or privately accessible is something you decide.

We will touch more on this later in the course in detail. Virtual machines are not a new thing. But the ease of provisioning EC2 instances allows for programmers and businesses to innovate more quickly. AWS has just made it much, much easier and more cost effective for you to acquire servers through this Compute as a Service model. There's a lot more to learn about EC2. We talked about virtualization and the types of software you can run on an EC2 instance. But there is more you can configure with EC2 as well.

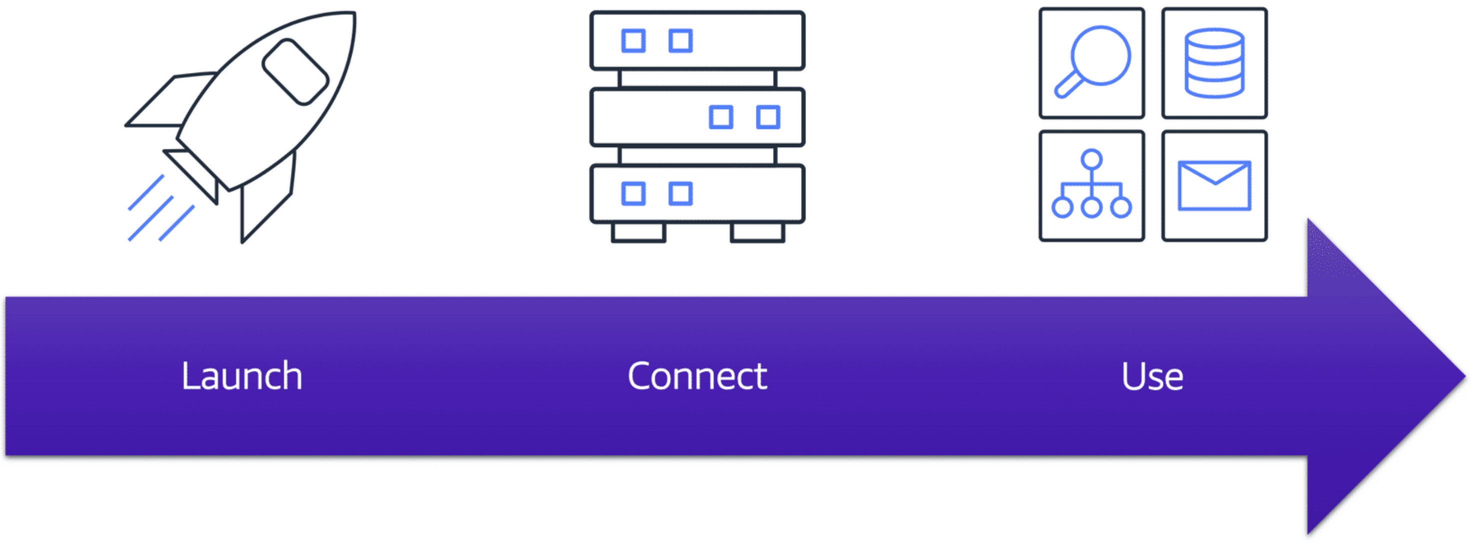
Amazon Elastic Compute Cloud (Amazon EC2)
Amazon Elastic Compute Cloud (Amazon EC2)
(opens in a new tab)
provides secure, resizable compute capacity in the cloud as Amazon EC2 instances.
Imagine you are responsible for the architecture of your company's resources and need to support new websites. With 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, choose each numbered marker.



1. **Launch**

First, you launch an instance. Begin by selecting a template with basic configurations for your instance. These configurations include the operating system, application server, or applications. You also select the instance type, which is the specific hardware configuration of your instance.

As you are preparing to launch an instance, you specify security settings to control the network traffic that can flow into and out of your instance. Later in this course, we will explore Amazon EC2 security features in greater detail.

2. **Connect**

Next, connect to the instance. You can connect to the instance in several ways. Your programs and applications have multiple different methods to connect directly to the instance and exchange data. Users can also connect to the instance by logging in and accessing the computer desktop.

3. **Use**

After you have connected to the instance, you can begin using it. You can run commands to install software, add storage, copy and organize files, and more.

Amazon EC2 Instance Types

Now that we've learned about EC2 instances and the crucial role they play in AWS, let's talk about the different types of EC2 instances that are available. Thinking back to our coffee shop analogy, you'll remember that EC2 instances are like our employees and that they serve client requests. If we want to have a cafe that can serve a lot of customers, then we're probably going to need multiple employees, right? And they all can't just be cashiers. We also need someone to make the drinks, someone to handle the food, and maybe someone to do that cool latte art that our customers love so much. Like any business, there are a variety of tasks that need to be done, and they often require different skillsets.

If we want our business to operate as efficiently as possible, it's important to make sure that an employee's skillset suits their role. In the same way that our coffee shop has different kinds of employees, AWS has different types of EC2 instances that you can spin up and deploy into your AWS environment.

Each instance type is grouped under an instance family and are optimized for certain types of tasks. Instance types offer varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. The different instance families in EC2 are general purpose, compute optimized, memory optimized, accelerated computing, and storage optimized.

General purpose instances provide a good balance of compute, memory, and networking resources, and can be used for a variety of diverse workloads like web service or code repositories.

Compute optimized instances are ideal for compute-intensive tasks like gaming servers, high performance computing or HPC, and even scientific modeling.

Similarly, memory optimized instances are good for memory-intensive tasks. Accelerated computing are good for floating point number calculations, graphics processing, or data pattern matching, as they use hardware accelerators.

And finally, storage optimized are good for, can you guess it? Workloads that require high performance for locally stored data.

Now, if we map this back to our coffee shop, our cashier becomes a memory optimized EC2 instance, baristas become compute optimized instances, and our latte art employee is an accelerated computing instance type. And there you have it, EC2 instance types.

Amazon EC2 instance types
Amazon EC2 instance types
(opens in a new tab)
 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 about Amazon EC2 instance types, expand each of the following five categories.
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.

Knowledge check

For guidance on navigating these questions using the keyboard, expand the following keyboard instructions.

1. Which Amazon EC2 instance type is suitable for data warehousing applications?
   - ○ Memory optimized
   - ○ Storage optimized
   - ○ General purpose
   - ○ Compute optimised

2. Which Amazon EC2 instance type balances compute, memory, and networking resources?
   - ○ Memory optimized
   - ○ Storage optimized
   - ○ General purpose
   - ○ Compute optimized

3. Which Amazon EC2 instance type is ideal for high-performance databases?
   - ○ Memory optimized
   - ○ Storage optimized
   - ○ General purpose

- ○ Compute optimized

4. Which Amazon EC2 instance type offers high-performance processors?

- ○ Memory optimized
- ○ Storage optimized
- ○ General purpose
- ○ Compute optimized

Amazon EC2 Pricing

We talked about EC2 instance types, but you're all probably wondering how much is this gonna cost me? Well, don't fret. For EC2, we have multiple billing options available.

The first one and the one that most people are familiar with is called On-Demand. What that means is that you only pay for the duration that your instance runs for. This can be per hour or per second, depending on the instance type and operating system you choose to run. Plus, no long-term commitments or upfront payments are needed. This type of pricing is usually for when you get started and want to spin up servers to test out workloads and play around. You don't need any prior contracts or communication with AWS to use On-Demand pricing. You can also use them to get a baseline for your average usage, which leads us to our next pricing option, Savings Plan.

Savings Plan offers low prices on EC2 usage in exchange for a commitment to a consistent amount of usage measured in dollars per hour for a one or three-year term. This flexible pricing model can therefore provide savings of up to 72% on your AWS compute usage. This can lower prices on your EC2 usage, regardless of instance family, size, OS, tenancy, or AWS region. This also applies to AWS Fargate and AWS Lambda usage, which are serverless compute options that we will cover later in this course.

Another option is Reserved Instances. These are suited for steady-state workloads or ones with predictable usage and offer you up to a 75% discount versus On-Demand pricing. You qualify for a discount once you commit to a one or three-year term and can pay for them with three payment options: all upfront, where you pay for them in full when you commit; partial upfront, where you pay for a portion when you commit; and no upfront, where you don't pay anything at the beginning.

The next option is Spot Instances, and they allow you to request spare Amazon EC2 computing capacity for up to 90% off of the On-Demand price. The catch here is that AWS can reclaim the instance at any time they need it, giving you a two-minute warning to finish up work and save state. You can always resume later if needed. So when choosing Spot Instances, make sure your workloads can tolerate being interrupted. A good example of those are batch workloads.

And finally, we have Dedicated Hosts, which are physical hosts dedicated for your use for EC2. These are usually for meeting certain compliance requirements and nobody else will share tenancy of that host.

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.
To learn more Amazon EC2 pricing, choose each of the following five categories.
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.

Reserved Instances

Reserved Instances are a billing discount applied to the use of On-Demand Instances in your account. There are two available types of Reserved Instances:

- Standard Reserved Instances
- Convertible Reserved Instances

You can purchase Standard Reserved and Convertible Reserved Instances for a 1-year or 3-year term. You realize greater cost savings with the 3-year option.

Standard Reserved Instances: This option is a good fit if you know the EC2 instance type and size you need for your steady-state applications and in which AWS Region you plan to run them. Reserved Instances require you to state the following qualifications:

- **Instance type and size:** For example, m5.xlarge
- **Platform description (operating system):** For example, Microsoft Windows Server or Red Hat Enterprise Linux
- **Tenancy:** Default tenancy or dedicated tenancy

You have the option to specify an Availability Zone for your EC2 Reserved Instances. If you make this specification, you get EC2 capacity reservation. This ensures that your desired amount of EC2 instances will be available when you need them.

Convertible Reserved Instances: If you need to run your EC2 instances in different Availability Zones or different instance types, then Convertible Reserved Instances might be right for you. Note: You trade in a deeper discount when you require flexibility to run your EC2 instances.

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 family and size, Region, platform, and tenancy).

EC2 Instance Savings Plans

AWS offers Savings Plans for a few compute services, including Amazon EC2. **EC2 Instance Savings Plans** reduce your EC2 instance costs when you make an hourly spend commitment to an instance family and Region for a 1-year or 3-year term. This term commitment results in savings of up to 72 percent compared to On-Demand rates. Any usage up to the commitment is charged at the discounted Savings Plans rate (for example, $10 per hour). Any usage beyond the commitment is charged at regular On-Demand rates.

The EC2 Instance Savings Plans are a good option if you need flexibility in your Amazon EC2 usage over the duration of the commitment term. You have the benefit of saving costs on running any EC2 instance within an EC2 instance family in a chosen Region (for example, M5 usage in N. Virginia) regardless of Availability Zone, instance size, OS, or tenancy. The savings with EC2 Instance Savings Plans are similar to the savings provided by Standard Reserved Instances.

Unlike Reserved Instances, however, you don't need to specify up front what EC2 instance type and size (for example, m5.xlarge), OS, and tenancy to get a discount. Further, you don't need to commit to a certain number of EC2 instances over a 1-year or 3-year term. Additionally, the EC2 Instance Savings Plans don't include an EC2 capacity reservation option.

Later in this course, you'll review AWS Cost Explorer, which you can use to visualize, understand, and manage your AWS costs and usage over time. If you're considering your options for Savings Plans, you can use AWS Cost Explorer to 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 Savings Plan.

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.

Knowledge check
For guidance on navigating these questions using the keyboard, expand the following keyboard instructions.

1. Which Amazon EC2 pricing option provides a discount when you specify a number of EC2 instances to run a specific OS, instance family and size, and tenancy in one Region?
   - ○ Convertible Reserved Instances
   - ○ EC2 Instance Savings Plans
   - ○ Spot Instances
   - ○ Standard Reserved Instances

2. Which Amazon EC2 pricing option provides a discount when you make an hourly spend commitment to an instance family and Region for a 1-year or 3-year term?
   - ○ On-demand
   - ○ EC2 Instance Savings Plans
   - ○ Spot Instances
   - ○ Reserved Instances

Scaling Amazon EC2

So we have a good idea now on the basics of EC2 and how it can help with any compute needs, like making coffee. Well, like metaphorically making coffee. The coffee represents the, whatever your instance is producing. The next thing we want to talk about is another major benefit of AWS, scalability and elasticity, or how capacity can grow and shrink, based on business needs.

Here is the on-prem data center dilemma. If your business is like 99% of all businesses out in the world, your customer workloads vary over time: perhaps over a simple 24 hour period, or you might have seasons where you're busy, and weeks that are not in demand. If you're building out a data center, the question is, what is the right amount of hardware to purchase? If you buy for the average amount, the average usage, you won't be wasting money on average. But when the peak loads come in, you won't have the hardware to service the customers, especially during the critical moments to expect to be making all your results.

Now, if you buy for the top max load, you might have happy customers, but for most of the year, you'll have idle resources. Which means your average utilization is very low. And I've seen data centers with average utilization under 10%, just out of fear of missing peak demand. So how do you solve the problem on-premises? Well the truth is, you can't. And here's where AWS changes the conversation entirely.

What if you could provision your workload to exactly the demand, every hour, every day? Well, now you have happy customers, because they can always get the services they want. And you have a happy financial officer because they get the ROI your company needs.

And here's how it works. Morgan is behind the counter. She's taking orders and we have a decoupled system. Morgan isn't doing all of the work here, so we need somebody making the drinks. It looks like Rudy's up. Now, the first problem I wanna solve, is the idea that we plan for a disaster. There's a great quote by Werner Vogels that says, "Everything fails all the time, so plan for failure and nothing fails." Or in other words, we ask ourselves what would happen if we lost our order taking instance? Well, we'd be out of business until we'd get another person to work the line, sorry, another instance up and running.

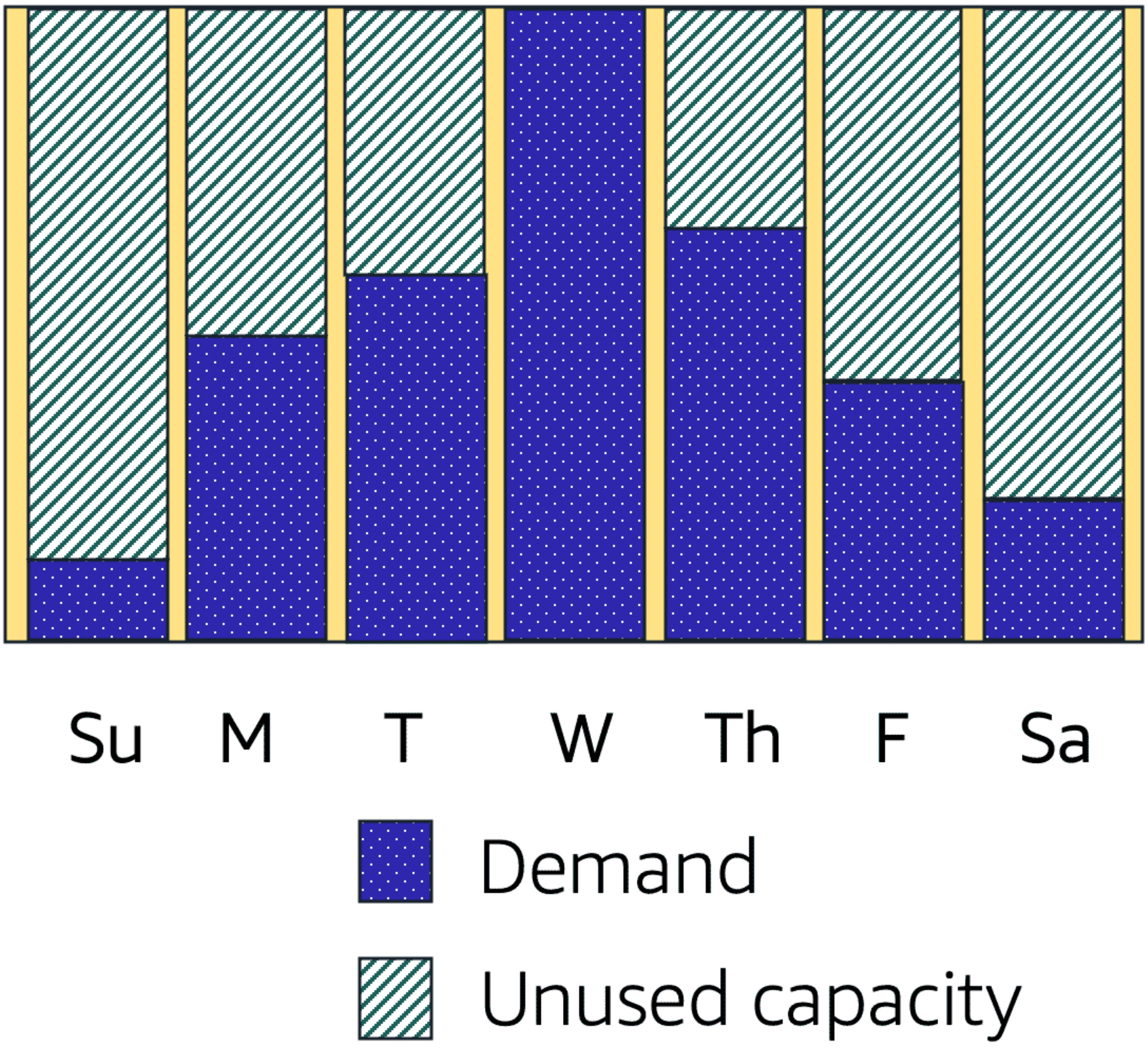
So here is where AWS makes it really simple. Using the same programmatic method we used to create the original Morgan, we can create a second Morgan. So if one fails, we have another one already on the front line and taking orders. The customers never lose service. Well, don't forget about the backend. Let's make our processing instances redundant as well. So that gets our regular operating capacity. We now have a highly available system, with no single point of failure. And as long as the number of customers in line stays the same, we're good. But you know that's gonna change, right? So let's take a look at what's going to happen when we have a rush of customers, an increase in demand.

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 can 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.

Now there are two ways to handle growing demands. You can scale up, or scale out. Scaling up means adding more power to the machines that are running, which might make sense in a few cases but think about it. When you have an increase in customers, a bigger instance of Morgan really can't take a customer's order any faster. Because that depends on the customer more than Morgan.

"I'll take an espresso. Oh, wait, is that organic? Make it a soy latte? Actually, I don't know. Do you just have tea?"

What we need are, well, more Morgans. Customers? Oh, no! Looks like the processing instances are about to get overloaded. So let's scale them as well.

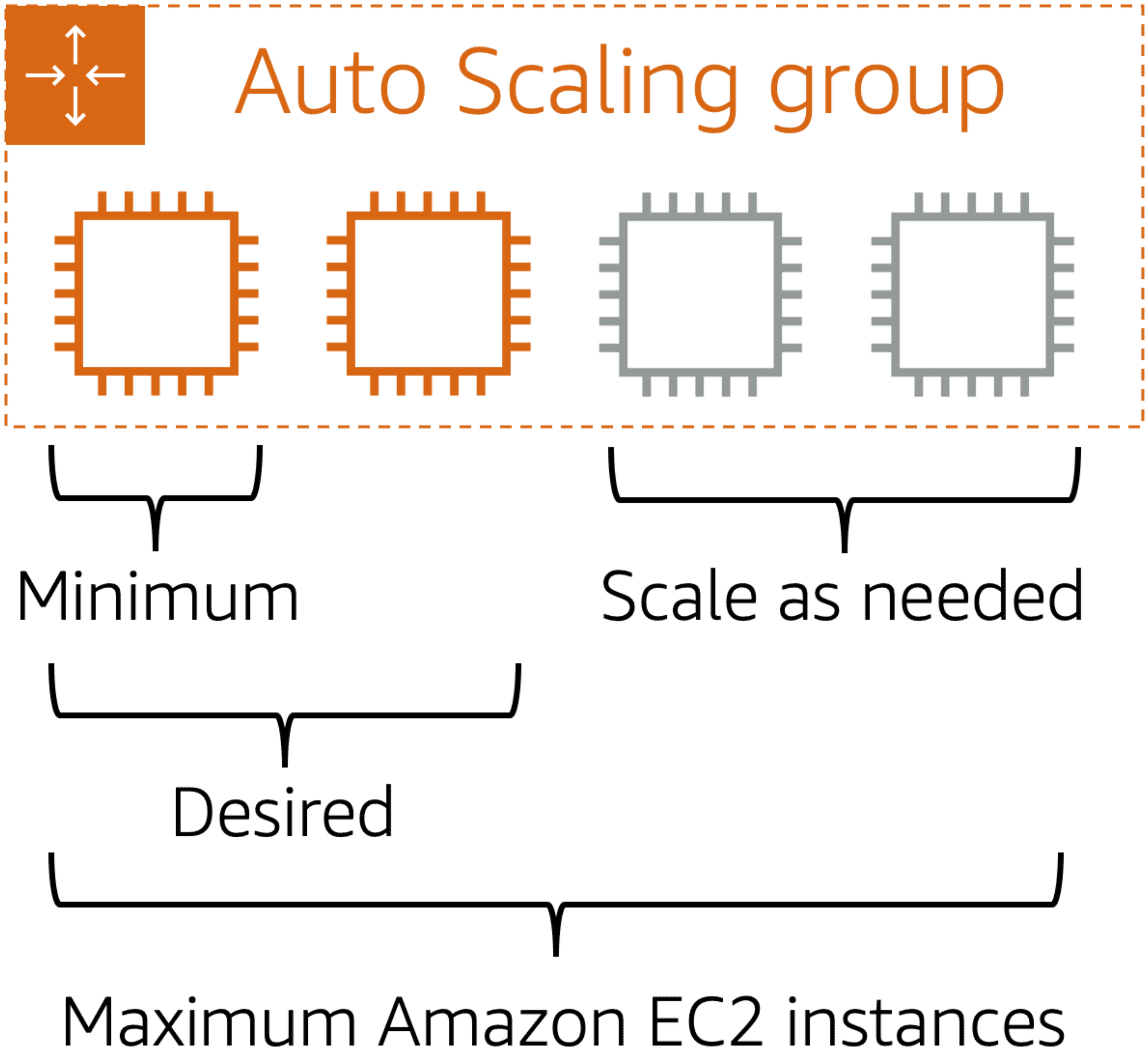
Obvious question is obvious. How come they are more order taking instances than order makers?

Well, in this case, the amount of work you can get done is still more than the order machines can send your way. You don't have a backlog. So no reason to add more worker instances. This is one of the great things about decoupling the system. You can end up with exactly the right amount of power for each part of your process, rather than over provisioning to solve a separate problem. Okay, looks like we just cleared that rush.

Now here is where AWS really makes a difference to your business. All these extra workers that are sitting around idle, if you don't need them, send them home or stop the instances. Amazon EC2 Auto Scaling. Adds instances based on demand and then decommissions instances when they are no longer needed. This means that every minute of the day, you always have the correct number of instances. Happy customers. Happy CFO. Happy architecture.

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.

So we solved the scaling problem with Amazon EC2 Auto Scaling. But now we've got a bit of a traffic problem, don't we? Let's take a look at the situation. When customers come into the coffee shop, right now they have three options for which cashier to talk to to place an order. And oddly enough, most of them are lining up in one line, causing an uneven distribution of customers per line. Even though we have other cashiers waiting to take orders, standing around doing nothing. Customers come in and aren't sure exactly where to route their order. It would help a lot if we added a host to the situation.

A host stands at the door and when customers come into the coffee shop, they tell them which line to proceed to for placing their order. The host keeps an eye on the cashier's taking orders, and counts the number of people in line each cashier is serving. Then it will direct new customers to the cashier that has the shortest line, that's the least bogged down, thus allowing the lines to be even across cashiers and helping customers be served in the most efficient manner possible.

The same idea applies to your AWS environment. When you have multiple EC2 instances all running the same program, to serve the same purpose, and a request comes in, how does that request know which EC2 instance to go to? How can you ensure there's an even distribution of workload across EC2 instances? So not just one is backed up while the others are idle sitting by. You need a way to route requests to instances to process that request. What you need to solve this is called load balancing.

A load balancer is an application that takes in requests and routes them to the instances to be processed. Now, there are many off the shelf load balancers that work great on AWS. And if you have a favorite flavor that already does exactly what you want, then feel free to keep using it. In which case it will be up to your operations team to install, manage, update, scale, handle fail over, and availability. It's doable. Odds are what you really need is just to properly distribute traffic in a high performance, cost-efficient, highly available, automatically scalable system that you can just set and forget.

Introducing Elastic Load Balancing. Elastic Load Balancing, or ELB, is one of the first major managed services we're going to talk about in this course. And it's engineered to address the undifferentiated heavy lifting of load balancing. To illustrate this point, I need to zoom out a bit here. To begin with, Elastic Load Balancing is a regional construct, and we'll explain more of what that means in later videos. But the key value for you is that because it runs at the Region level rather than on individual EC2 instances, the service is automatically highly available with no additional effort on your part.

ELB is automatically scalable. As your traffic grows, ELB is designed to handle the additional throughput with no change to the hourly cost. When your EC2 fleet auto-scales out, as each instance comes online, the auto-scaling service just lets the Elastic Load Balancing service know that it's ready to handle the traffic, and off it goes. Once the fleet scales in, ELB first stops all new traffic, and waits for the existing requests to complete, to drain out. Once they do that, then the auto-scaling engine can terminate the instances without disruption to existing customers.

ELB is not only used for external traffic. Let's look at the ordering tier, and how it communicates with the production tier. Right now, each front end instance is aware of each backend instance. And if a new back end instance comes online, in this current architecture, it would have to tell every front end instance that it can now accept traffic. This is complicated enough with just half a dozen instances. Now imagine you have potentially hundreds of instances on both tiers. Each tier shifting constantly based on demand. Just keeping them networked efficiently is impossible.

Well, we solve the back end traffic chaos with an ELB as well. Because ELB is regional, it's a single URL that each front end instance uses. Then the ELB directs traffic to the back end that has the least outstanding requests. Now, if the back end scales, once the new instance is ready, it just tells the ELB that it can take traffic, and it gets to work. The front end doesn't know and doesn't care how many back end instances are running. This is true decoupled architecture.

There's even more that the ELB can do that we'll learn about later, but this is good enough for now. The key is choosing the right tool for the right job, which is one of the reasons AWS offers so many different services. For example, back end communication. There are many ways to handle it and ELB is just one method. Next, we'll talk about some other services that might work even better for some architectures.

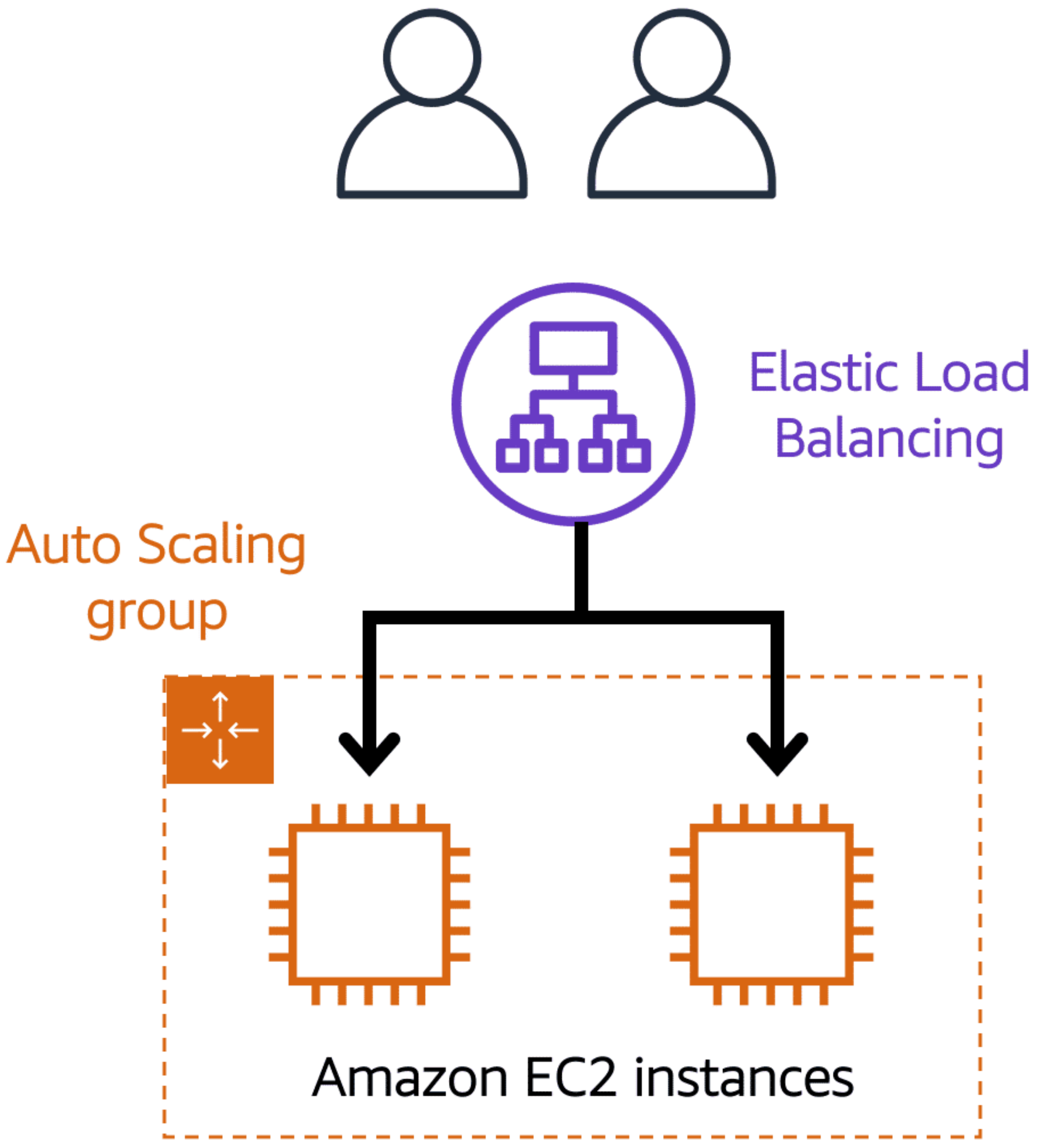
Elastic Load Balancing

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.

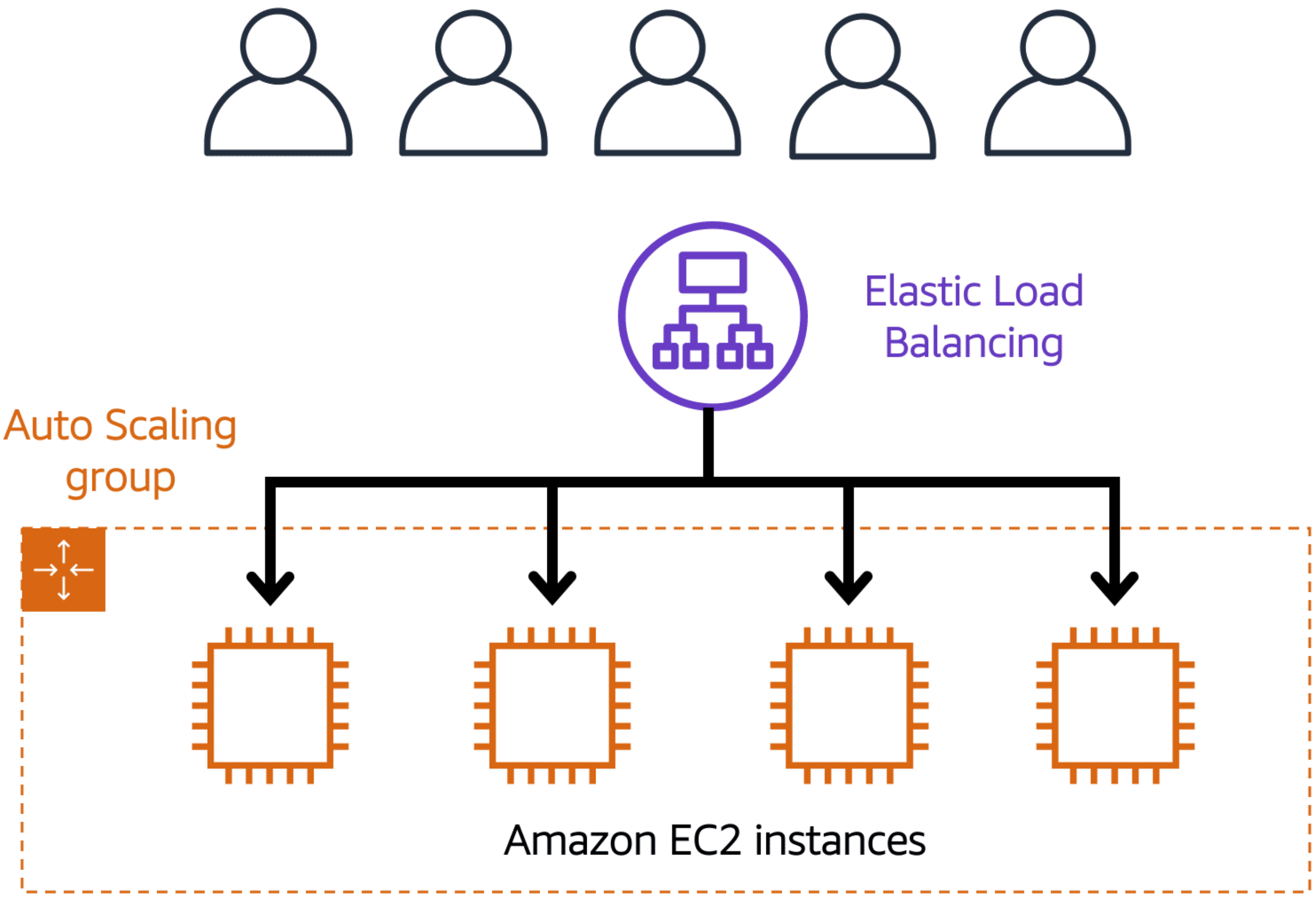
Example: Elastic Load Balancing

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.



High-demand period
Throughout the day, as the number of customers increases, the coffee shop opens more registers to accommodate them.

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

Let's talk about messaging and queuing. In the coffee shop, there are cashiers taking orders from the customers and baristas making the orders. Currently, the cashier takes the order, writes it down with a pen and paper, and delivers this order to the barista. The barista then takes the paper and makes the order. When the next order comes in, the process repeats. This works great as long as both the cashier and the barista are in sync. But what would happen if the cashier took the order and turned to pass it to the barista and the barista was out on break or busy with another order? Well, that cashier is stuck until the barista is ready to take the order. And at a certain point, the order will probably be dropped so the cashier can go serve the next customer.

You can see how this is a flawed process, because as soon as either the cashier or barista is out of sync, the process will degrade, causing slow downs in receiving orders and failures to complete orders at all. A much better process would be to introduce some sort of buffer or queue into the system. Instead of handing the order directly to the barista, the cashier would post the order to some sort of buffer, like an order board.

This idea of placing messages into a buffer is called messaging and queuing. Just as our cashier sends orders to the barista, applications send messages to each other to communicate. If applications communicate directly like our cashier and barista previously, this is called being tightly coupled.

A hallmark trait of a tightly coupled architecture is where if a single component fails or changes, it causes issues for other components or even the whole system. For example, if we have Application A and it is sending messages directly to Application B, if Application B has a failure and cannot accept those messages, Application A will begin to see errors as well. This is a tightly coupled architecture.

A more reliable architecture is loosely coupled. This is an architecture where if one component fails, it is isolated and therefore won't cause cascading failures throughout the whole system. If we coded the application to use a more loosely coupled architecture, it could look as follows.

Just like our cashier and barista, we introduced a buffer between the two. In this case, we introduced a message queue. Messages are sent into the queue by Application A and they are processed by Application B. If Application B fails, Application A doesn't experience any disruption. Messages being sent can still be sent to the queue and will remain there until they are eventually processed.

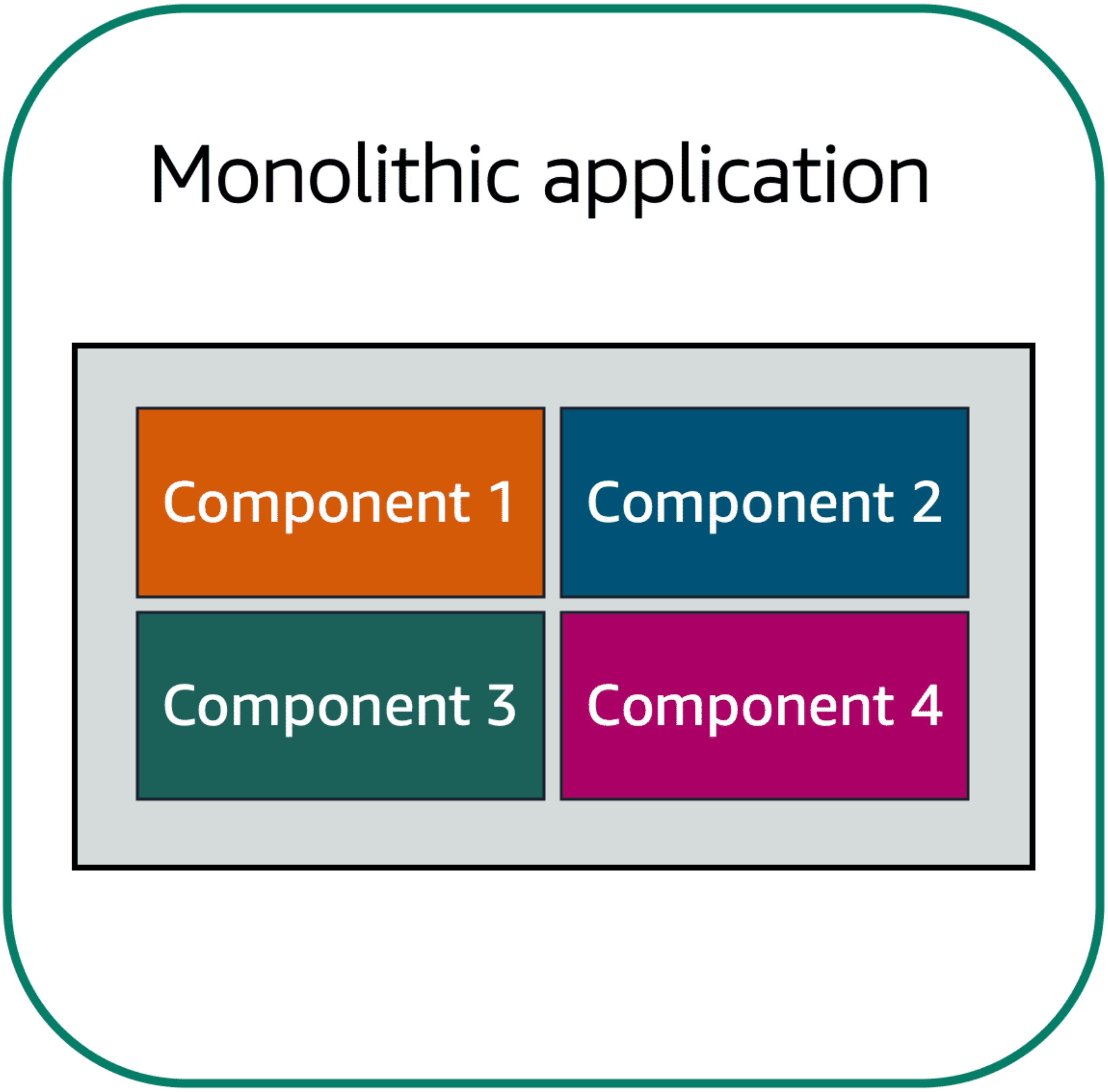
This is loosely coupled. This is what we strive to achieve with architectures on AWS. And this brings me to two AWS services that can assist in this regard. Amazon Simple Queue Service or SQS and Amazon Simple Notification Service or SNS. But before I dive into those two services, let me just order a to-go coffee on our cafe website. Done. All right, well, that's great. I should get a message when that order is ready.

First up, let's discuss Amazon SQS. SQS allows you to send, store, and receive messages between software components at any volume. This is without losing messages or requiring other services to be available. Think of messages as our coffee orders and the order board as an SQS queue. Messages have the person's name, coffee order, and time they ordered. The data contained within a message is called a payload, and it's protected until delivery. SQS queues are where messages are placed until they are processed. And AWS manages the underlying infrastructure for you to host those queues. These scale automatically, are reliable, and are easy to configure and use.

Now, Amazon SNS is similar in that it is used to send out messages to services, but it can also send out notifications to end users. It does this in a different way called a publish/subscribe or pub/sub model. This means that you can create something called an SNS topic which is just a channel for messages to be delivered. You then configure subscribers to that topic and finally publish messages for those subscribers. In practice, that means you can send one message to a topic which will then fan out to all the subscribers in a single go. These subscribers can also be endpoints such as SQS queues, AWS Lambda functions, and HTTPS or HTTP web hooks.

Additionally, SNS can be used to fan out notifications to end users using mobile push, SMS, and email. Taking this back to our coffee shop, we could send out a notification when a customer's order is ready. This could be a simple SMS to let them know to pick it up or even a mobile push. In fact, it looks like my phone just received a message. Looks like my order is ready. See you soon.

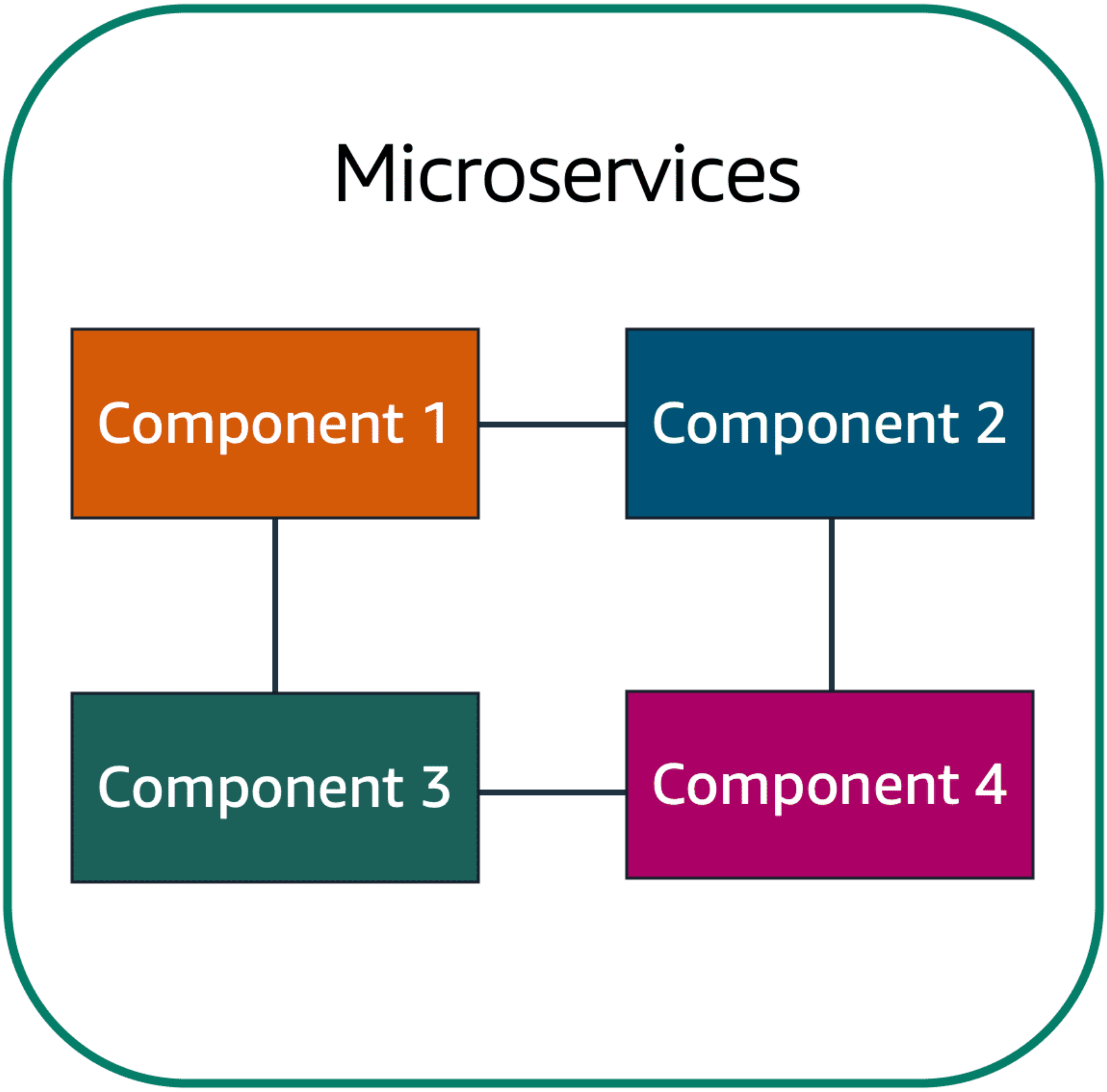
Monolithic applications and microservices



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).

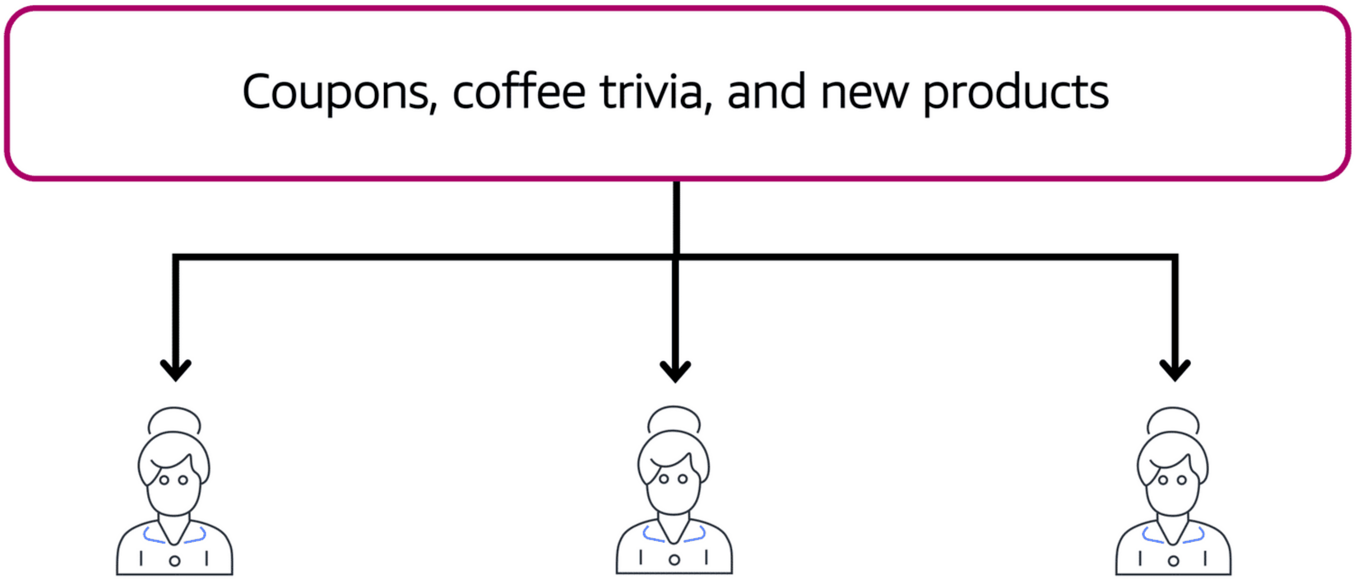
Amazon Simple Notification Service (Amazon SNS)

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.

To review two examples of how to use Amazon SNS, choose the arrow buttons to display each one.

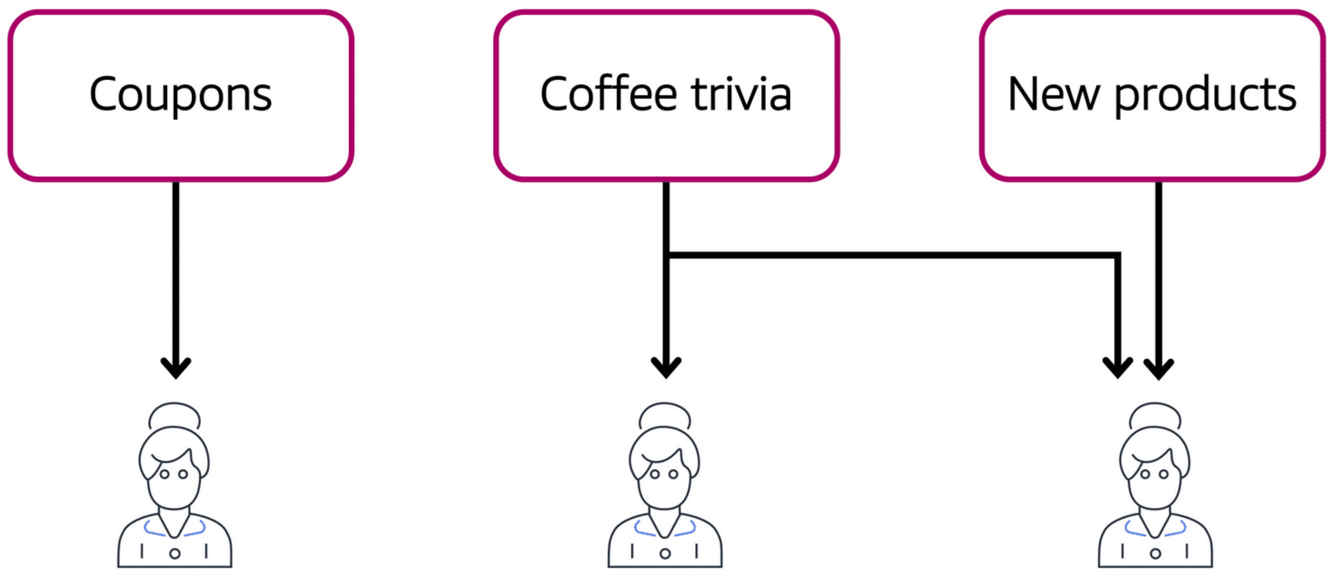
Step 1:
Publishing updates from a single topic



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.

Summary

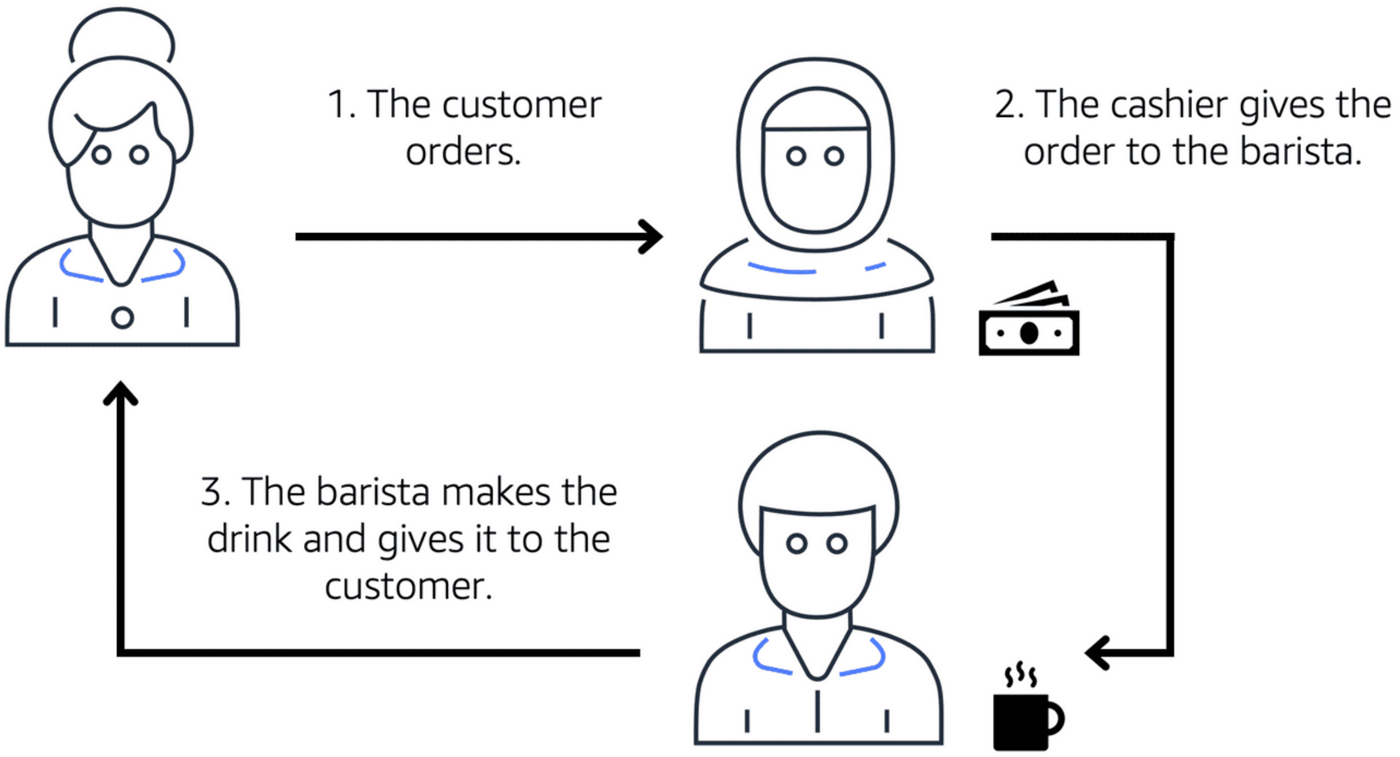
Although these examples from the coffee shop involve subscribers who are people, in Amazon SNS, subscribers can be web servers, email addresses, AWS Lambda functions, or several other options.

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.
To review two examples of how to use Amazon SQS, choose the arrow buttons to display each one.

Step 1

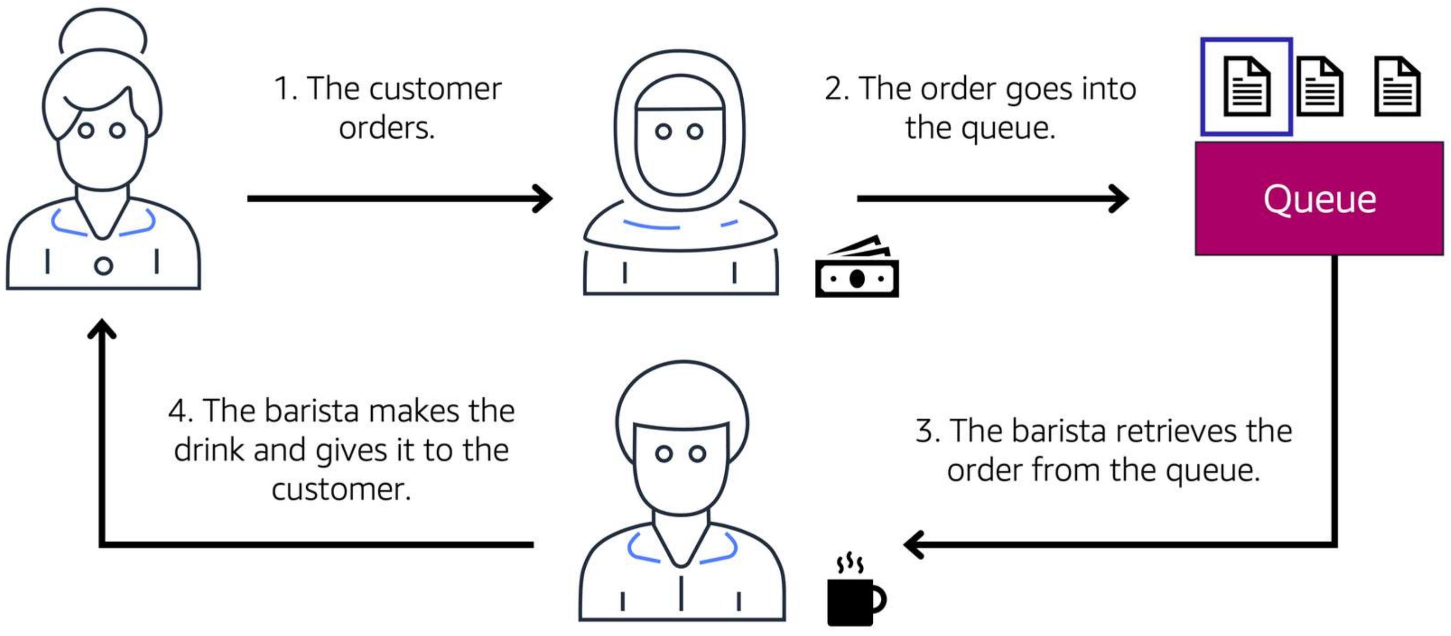
Example 1: Fulfilling an order


1. The customer orders.
2. The cashier gives the order to the barista.
3. The barista makes the drink and gives it to the customer.

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.

2

Example 2: Orders in a queue


1. The customer orders.
2. The order goes into the queue.
Queue
3. The barista retrieves the order from the queue.
4. The barista makes the drink and gives it to the customer.

Recall that the cashier and the barista are two separate components of an application. A message queuing service, such as Amazon SQS, lets 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.

Summary

For decoupled applications and microservices, Amazon SQS enables you to send, store, and retrieve messages between components.
This decoupled approach enables the separate components to work more efficiently and independently.

Knowledge check

For guidance on navigating this question using the keyboard, expand the following keyboard instructions.

1. **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)
Elastic Load Balancing

EC2 instances are virtual machines that you can provision with minimal friction to get up and running on AWS. EC2 is great for all sorts of different use cases like running basic web servers to running high performance computing clusters and everything in between.

That being said, though EC2 is incredibly flexible, reliable, and scalable, depending on your use case, you might be looking at alternatives for your compute capacity. EC2 requires that you set up and manage your fleet of instances over time. When you're using EC2, you are responsible for patching your instances when new software packages come out, setting up the scaling of those instances as well as ensuring that you've architected your solutions to be hosted in a highly available manner. This is still not as much management as you would have if you hosted these on-premises. But management processes will still need to be in place.

You might be wondering what other services does AWS offer for compute that are more convenient from a management perspective. This is where the term serverless comes in. AWS offers multiple serverless compute options. Serverless means that you cannot actually see or access the underlying infrastructure or instances that are hosting your application. Instead, all the management of the underlying environment from a provisioning, scaling, high availability, and maintenance perspective are taken care of for you. All you need to do is focus on your application and the rest is taken care of.

AWS Lambda is one serverless compute option. Lambda's a service that allows you to upload your code into what's called a Lambda function. Configure a trigger and from there, the service waits for the trigger. When the trigger is detected, the code is automatically run in a managed environment, an environment you do not need to worry too much about because it is automatically scalable, highly available and all of the maintenance in the environment itself is done by AWS. If you have one or 1,000 incoming triggers, Lambda will scale your function to meet demand. Lambda is designed to run code under 15 minutes so this isn't for long running processes like deep learning. It's more suited for quick processing like a web backend, handling requests or a backend expense report processing service where each invocation takes less than 15 minutes to complete.

If you weren't quite ready for serverless yet or you need access to the underlying environment, but still want efficiency and portability, you should look at AWS container services like Amazon Elastic Container Service, otherwise known as ECS. Or Amazon Elastic Kubernetes Service, otherwise known as EKS.

Both of these services are container orchestration tools, but before I get too far here, a container in this case is a Docker container. Docker is a widely used platform that uses operating system level virtualization to deliver software in containers. Now a container is a package for your code where you package up your application, its dependencies as well as any configurations that it needs to run. These containers run on top of EC2 instances and run in isolation from each other similar to how virtual machines work. But in this case, the host is an

EC2 instance. When you use Docker containers on AWS, you need processes to start, stop, restart, and monitor containers running across not just one EC2 instance, but a number of them together which is called a cluster.

The process of doing these tasks is called container orchestration and it turns out it's really hard to do on your own. Orchestration tools were created to help you manage your containers. ECS is designed to help you run your containerized applications at scale without the hassle of managing your own container orchestration software. EKS does a similar thing, but uses different tooling and with different features.

Both Amazon ECS and Amazon EKS can run on top of EC2. But if you don't want to even think about using EC2s to host your containers because you either don't need access to the underlying OS or you don't want to manage those EC2 instances, you can use a compute platform called AWS Fargate. Fargate is a serverless compute platform for ECS or EKS. That's a bit high level and it might be confusing, so let's clear that up.

If you are trying to host traditional applications and want full access to the underlying operating system like Linux or Windows, you are going to want to use EC2. If you are looking to host short running functions, service-oriented or event driven applications and you don't want to manage the underlying environment at all, look into the serverless AWS Lambda. If you are looking to run Docker container-based workloads on AWS, you first need to choose your orchestration tool. Do you want to use Amazon ECS or Amazon EKS? After you choose your tool, you then need to chose your platform. Do you want to run your containers on EC2 instances that you manage or in a serverless environment like AWS Fargate that is managed for you?

Those are just some of your compute options with AWS and it's not even a complete list. Check out the notes for more information on AWS compute services as well as others that we didn't get to talk about in this video.