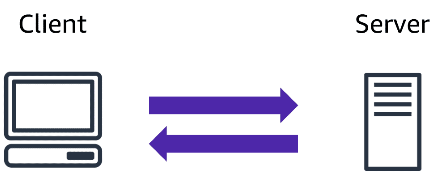
Table of content:

1. [Module 1](#Module1):
2. [Client server model](#client_server_model)
3. [Cloud computing](#cloud_computing)
4. [Deployment models for cloud computing](#deployement_model_for_cc)
5. [Benefits of cloud computing](#benfits_of_cloud_computing)
6. [Module1 quiz](#module1_quiz)
7. [Module2:](#Module_2)
8. [EC2 introduction](#amazon_elastic_cloud)
9. [Virtualization](#virtualization)
10. [AWS instance type](#aws_instance_type)
11. **Module 1:**

**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.

**What is cloud computing?**

**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.

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.

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**

|  |  |  |
| --- | --- | --- |
| Cloud Based deployement | ON-premise deployement | Hybrid deployement |
| * Run all parts of the application in the cloud. * Migrate existing applications to the cloud. * Design and build new applications in the cloud. | * Deploy resources by using virtualization and resource management tools. * Increase resource utilization by using application management and virtualization technologies. | * Connect cloud-based resources to on-premises infrastructure. * Integrate cloud-based resources with legacy IT applications. |
| * 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 | * **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. | * 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****

## 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

Modul1 quiz:

1. **What is cloud computing?**

The correct response option is **On-demand delivery of IT resources and applications through the internet with pay-as-you-go pricing**.

The other response options are incorrect because:

* It is possible to back up files to the cloud, but this response option does not describe cloud computing as a whole.
* Deploying applications connected to on-premises infrastructure is a sample use case for a hybrid cloud deployment. Remember that cloud computing also has cloud and on-premises (or private cloud) deployment models.
* AWS Lambda is an AWS service that lets you run code without needing to manage or provision servers. This description does not describe cloud computing as a whole. AWS Lambda is explained in greater detail later in the course.

1. **What is another name for on-premises deployment?**

The correct response option is **Private cloud deployment**.

The other response options are incorrect because:

* Cloud-based applications are fully deployed in the cloud and do not have any parts that run on premises.
* A hybrid deployment connects infrastructure and applications between cloud-based resources and existing resources that are not in the cloud, such as on-premises resources. However, a hybrid deployment is not equivalent to an on-premises deployment because it involves resources that are located in the cloud.
* The AWS Cloud offers three cloud deployment models: cloud, hybrid, and on-premises. This response option is incorrect because the AWS Cloud is not equivalent to only an on-premises deployment.

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

The correct response option is **The aggregated cloud usage from a large number of customers results in lower pay-as-you-go prices**.

This answer describes how customers can benefit from massive economies of scale in cloud computing.

The other response options are incorrect because:

* Not having to invest in technology resources before using them relates to *Trade upfront expense for variable expense*.
* Accessing services on-demand to prevent excess or limited capacity relates to *Stop guessing capacity*.
* Quickly deploying applications to customers and providing them with low latency relates to *Go global in minutes*.

**Module 2:**

## Virtualization:

## Making virtual machine on a host machine(high level computer/computational unit)

## Virtualization is feasible using Hypervisor:

## Hypervisor is in between host and virual machines.

## Hypervisor allocates computation resouces to virtual machine based on their need from the physical server.

## Hypervisor are of two types: type 1 and type 2

## Type 1 are directly installed on the host machine, they are fast ,secure and has low latency(ability to delay the work) ex: Microsoft hyper V or open source KVM

## Type 2 hypervisor has an OS in between the hypervisor and host machine, they are less secure , slow and has high latency ex: VMware workstation

## Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon 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.

EC2 and Hypervisor:

* 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.
* 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.

**AWS instance types:**

|  |  |
| --- | --- |
| Ideal for high-performance databases | Memory purpose instance |
| Suitable for data warehousing applications | Storage purpose |
| Balances compute, memory, and networking resources | General purpose |
| Offers high-performance processors | Compute purpose |

Amazon EC2 pricing:

With Amazon EC2, you pay only for the compute time that you use. Amazon EC2 offers a variety of pricing options for different use cases.

* **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.
* **Amazon EC2 Savings Plans** enable you to reduce your compute costs by committing to a consistent amount of compute usage for a 1-year or 3-year term.
* **Spot Instances** are ideal for workloads with flexible start and end times, or that can withstand interruptions.
* **Dedicated Hosts**are physical servers with Amazon EC2 instance capacity that is fully dedicated to your use.

Scaling Amazon EC2:

**Scalability** involves beginning with only the resources you need and designing your architecture to automatically respond to changing demand by scaling out or in.

**Amazon EC2 Auto Scaling**: Automate scaling process

## ****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

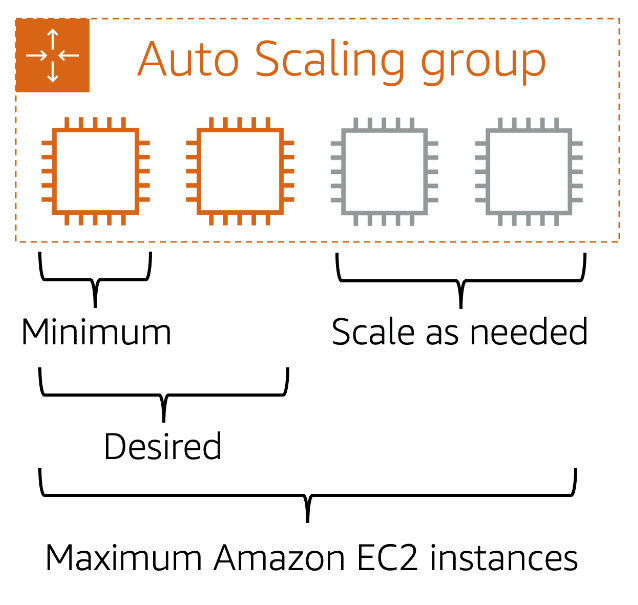
mazon EC2 Auto Scaling enables you to automatically add or remove Amazon EC2 instances in response to changing application demand. By automatically scaling your instances in and out as needed, you are able to maintain a greater sense of application availability.

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.

## ****Example: Amazon EC2 Auto Scaling****



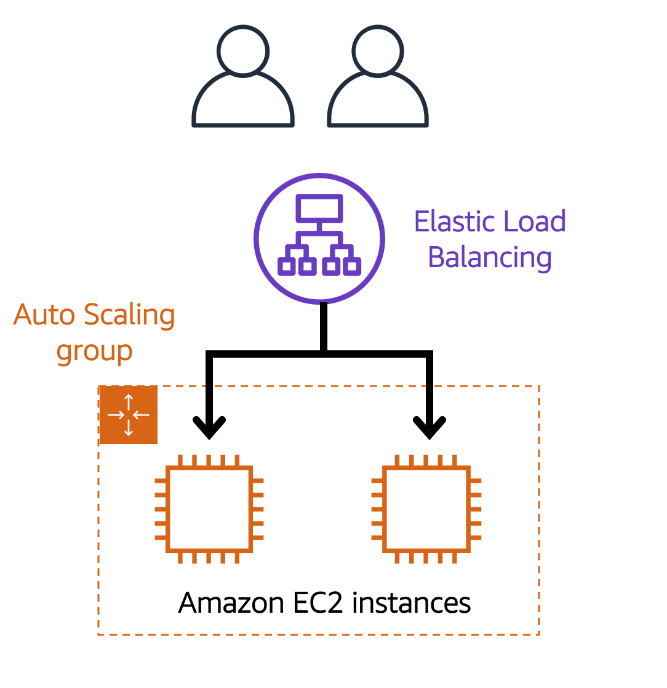
If you do not specify the desired number of Amazon EC2 instances in an Auto Scaling group, the desired capacity defaults to your minimum capacity.

**Directing traffic with 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.

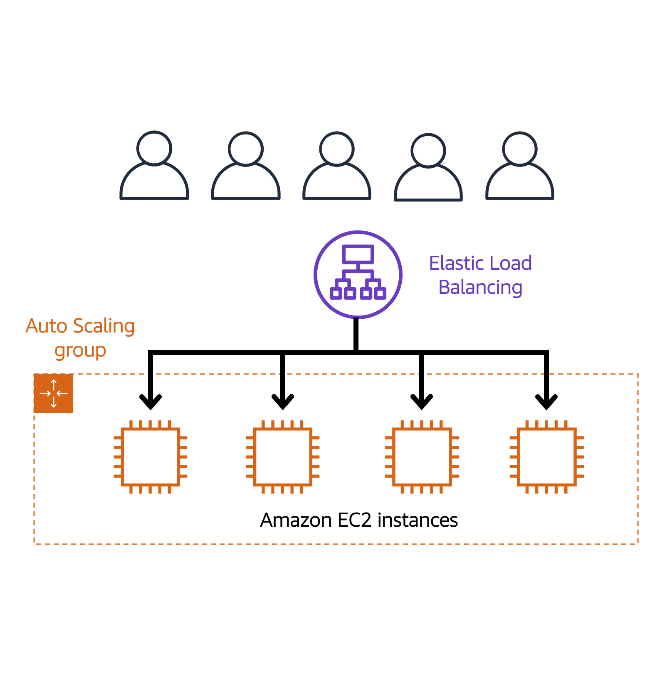
Although Elastic Load Balancing and Amazon EC2 Auto Scaling are separate services, they work together to help ensure that applications running in Amazon EC2 can provide high performance and availability.



**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.

## ****Monolithic applications and microservices****

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

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

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

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

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

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

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

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

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

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

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

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