**Importance of AWS**

**Amazon Web Services (AWS CLF-C01)**

<https://degreed.com/pathway/1pnlvmk78n/pathway>

<https://partnercentral.awspartner.com>

<https://degreed.com/pathway/dp64vokj87/pathway?newWindow=true>

<https://explore.skillbuilder.aws/learn/course/external/view/elearning/134/aws-cloud-practitioner-essentials>

**Exam guid of CLF-C01**  
<https://capgemini-my.sharepoint.com/personal/payal_krishna-das_capgemini_com/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fpayal%5Fkrishna%2Ddas%5Fcapgemini%5Fcom%2FDocuments%2FMicrosoft%20Teams%20Chat%20Files%2FAWS%2DCertified%2DCloud%2DPractitioner%5FExam%2DGuide%20%282%29%2Epdf&parent=%2Fpersonal%2Fpayal%5Fkrishna%2Ddas%5Fcapgemini%5Fcom%2FDocuments%2FMicrosoft%20Teams%20Chat%20Files>

<https://explore.skillbuilder.aws/learn/course/9449/exam-prep-aws-certified-cloud-practitioner-clf-c01>

**Question and answers:**  
<https://capgemini-my.sharepoint.com/:w:/r/personal/shivakant_choudhary_capgemini_com/Documents/Microsoft%20Teams%20Chat%20Files/Questions_Answers.docx?d=w1a28526b06434b35b3ddaa2560cbfdc0&csf=1&web=1&e=cTc9m9>

**Amazon Web Services (AWS CLF-C02):**

**First mentoring session -** [**https://capgemini-my.sharepoint.com/:v:/p/amal\_thambi/EUn6G4ekjjRJlpy-WkDvo34BUhanQbHj2CoSffswH8Doog?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view**](https://capgemini-my.sharepoint.com/:v:/p/amal_thambi/EUn6G4ekjjRJlpy-WkDvo34BUhanQbHj2CoSffswH8Doog?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view)

**Second mentoring session -** [**https://capgemini-my.sharepoint.com/:v:/p/amal\_thambi/Ea75EpkfpkdIqymmMNOIxRUBO7SL3xpP44IhPxYdHcIm0A?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view**](https://capgemini-my.sharepoint.com/:v:/p/amal_thambi/Ea75EpkfpkdIqymmMNOIxRUBO7SL3xpP44IhPxYdHcIm0A?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view)

**Third mentoring session -** [**https://capgemini-my.sharepoint.com/:v:/p/amal\_thambi/EXt0EPlp-bJLpa9Ifkhldv8B4BX-\_lgL67uFUB7rrQSAcA?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view**](https://capgemini-my.sharepoint.com/:v:/p/amal_thambi/EXt0EPlp-bJLpa9Ifkhldv8B4BX-_lgL67uFUB7rrQSAcA?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view.view)

**Difference of local deployment and Aws deployment**

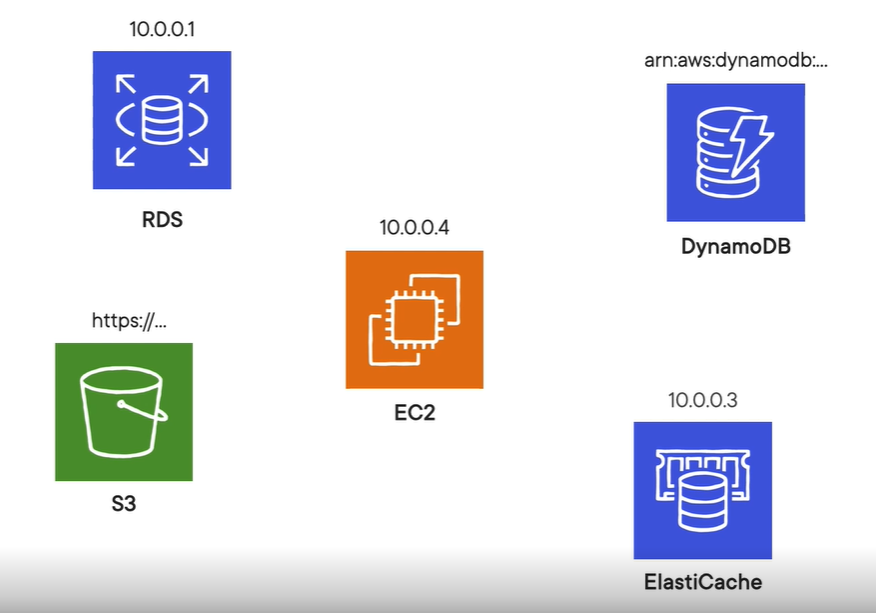
Local Deployment Aws Deployment

\* Everything runs locally and same machine \* one service=one responsibility

\* Files stored on server \* Files served from s3

\* Database on same server as application \* Database as a service

\* Connections done manually \* Connections done AWS software development kit

For AWS have 170+ services , How can interact with these services

**The core of AWS is Elastic cloud computing (EC2)**

**Amazon Elastic Cloud Compute (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 must 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 centre.

Make all the necessary configurations.

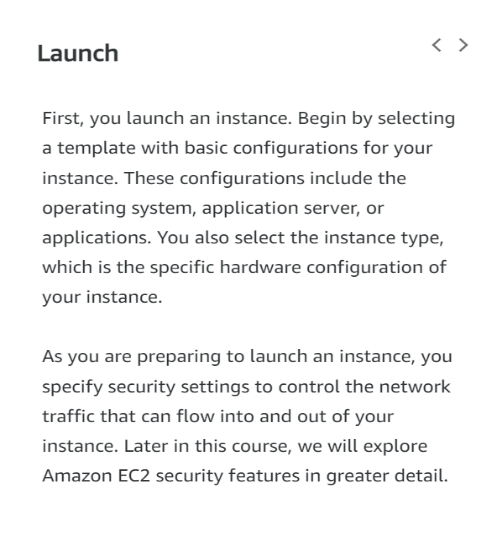
**By comparison, with an Amazon EC2 instance you can use a virtual server to run applications in the AWS Cloud.**

You can provision and launch an Amazon EC2 instance within minutes.

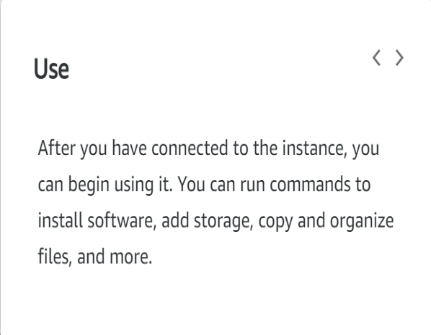
You can stop using it when you have finished running a workload.

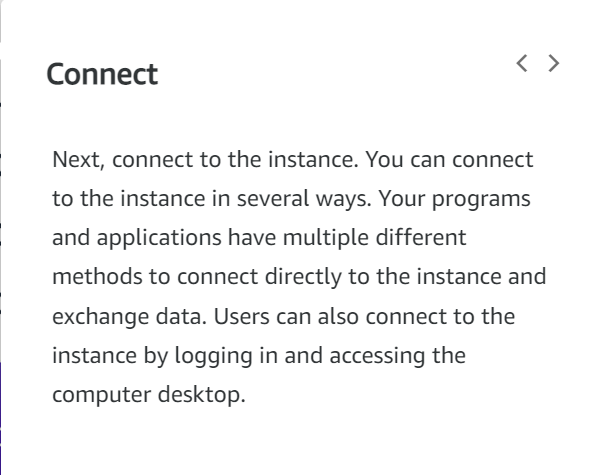
You pay only for the compute time you use when an instance is running, not when it is stopped or terminated.

You can save costs by paying only for server capacity that you need or want.

**How Amazon EC2 works:**

1) Launch

2) connect

3) use

**Amazon EC2 instance types:**

Amazon EC2 instance types are optimized for different tasks.

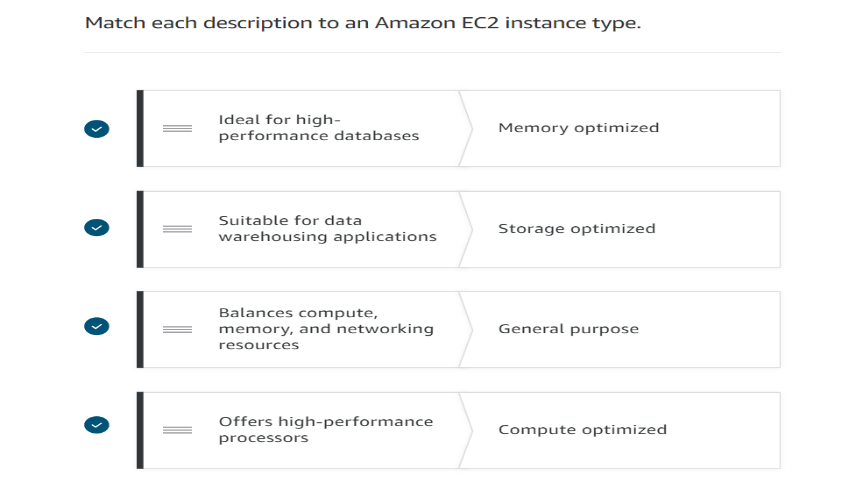
When selecting an instance type, consider the specific needs of your workloads and applications. This might include requirements for compute, memory, or storage capabilities.

1) General purpose Instance

It is used balance compute, memory, and networking resources

2) Compute optimized Instance

Offers high-performance processors



3) Memory optimized Instance

Ideal for high performance databases

4) Storage optimized Instance

Suitable for data warehousing applications

5) Accelerated computing Instance

Use hardware accelerators, or co-processors, or data pattern machine

**Amazon EC2 prising:**

Amazon EC2 purchase options

1) On-Demand

\* It's the pay-as-you-go model

\* Standard rate-no discount; no commitments; dev/test, sort term or unpredictable workloads.

2) Savings Plans

\* Savings Plans are a flexible pricing model that offers low prices on EC2 and Faregate usage,

in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a one - or three-year term.

3) Reserved Instances

\* It purchases are one of the easiest ways to start reducing cloud spends.

\* One- or three-year commitment.

\* In return for that long-term commitment,

\* AWS gives a hefty discount of up to 72% off compared to th On-Demand prices.

4) Spot Instances

\* It allows you to take advantage of Amazon's 'extra' capacity.

\* They are offered at a deep discount of up-to 90% off the On-Demand EC2 prices.

\* Spot Instance are flexible start and end time.

5) Dedicated hosts

\* A Dedicated Host is a physical EC2 server dedicated for your use.

\* Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses,

including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms),

and can also help you meet compliance requirements.

**Scaling Amazon EC2:**

Scalability

Scalability involves beginning with only the resources you need and designing your architecture to automatically

respond to changing demand by scaling out or in.

As a result, you pay for only the resources you use. You don’t have to worry about a lack of computing capacity to meet your customers’ needs.

If you wanted the scaling process to happen automatically, which AWS service would you use?

The AWS service that provides this functionality for Amazon EC2 instances is Amazon EC2 Auto Scaling

**Amazon EC2 Auto Scaling:**

Amazon EC2 Auto Scaling enables you to automatically add or remove Amazon EC2 instances in response to changing application demand.

By automatically scaling your instances in and out as needed, you are able to maintain a greater sense of application availability.

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

Dynamic scaling responds to changing demand.

Predictive scaling automatically schedules the right number of Amazon EC2 instances based on predicted demand.

EX: 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.

**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 must 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.

**Ex:**

Low-demand period:

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

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

High-demand period:

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

the Auto Scaling group represents this.

Additionally, a coffee shop employee directs customers to the most appropriate register so that the number of requests can evenly

distribute across the open registers. You can think of this coffee shop employee as a load balancer.

**Messaging and queuing:**

Monolithic applications and microservices:

Monolithic applications:

Applications are made of multiple components. The components communicate with each other to transmit data, fulfil 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.

Microservices:

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 fulfil 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 like 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.

**AWS CLI (Command line interface) features**

Great for shell shifting, feature complete with console and SDKs, Interact with any service

AWS CLI made for operations engineers more than developers

Usages: Configure local development environment and alternative to console actions.

**AWS SDK (Software Development Kit) features**