



- **AWS - A Brief Overview**
- **Managed Server**
- **Serverless**
- **OS System Requirement**
- **Meta Data**
- **User Data**



[linkedin.com/in/kumar--himanshu](https://www.linkedin.com/in/kumar--himanshu)



Follow

 **Share**

Introduction

Welcome to our easy-to-understand guide about making websites work on the internet using a special service called Amazon Web Services, or AWS for short.

Imagine the internet as a big digital playground where websites live. AWS is like a super toolkit that helps us create, organize, and manage these websites easily. It's like having a magical box full of tools for building amazing things online.

In this guide, we will explore how to use AWS in the simplest way possible. Whether you are someone who loves computers, works with them, or is just curious, this guide is here to help you understand the basics. We will learn about different parts of AWS, how it helps websites run smoothly, and how to set up your own little space on the internet.

Get ready for an adventure where we'll use easy words and simple ideas to make the internet world a bit more understandable for everyone. Let's start our journey into the online universe with AWS!

AWS is Like a Super Internet Toolkit:

Imagine AWS as a special toolbox on the internet. It helps people and businesses do lots of things online without needing to own big, physical computers. With AWS, you can build websites and other cool stuff without worrying about technical details.

AWS Services

- **Computing Power** (Like a Virtual Computer):

AWS has a tool called EC2, which is like a virtual computer. You can customize it based on what you need, just like picking different parts for a computer. It helps you do tasks on the internet without needing your own computer.

- **Storage** (Like Digital Filing Cabinets):

There's a tool called S3 that's like a big digital filing cabinet. You can store lots of information there, like photos or documents. It's safe and easy to access from anywhere on the internet.

- **Internet Networks (Like Digital Roads):**

AWS has a tool called VPC, which creates private, safe roads on the internet for your data. It also helps connect your online stuff to your real-world office or home computer securely. Another tool, Route 53, helps people find your website easily on the internet by translating easy names into computer addresses.

Additional AWS Services

- **Databases (Like Digital Libraries):** AWS helps you set up and organize digital libraries for storing information.
- **Machine Learning and AI (Like Smart Helpers):** AWS has tools that help you teach computers to do smart things, like recognizing pictures or helping with predictions.
- **Containers (Like Digital Boxes):** These tools help you run applications (like games or websites) in special digital boxes, making them work smoothly online.
- **Security and Identity (Like Digital Locks):** AWS keeps everything safe with digital locks, allowing only the right people to access information.

Why AWS is Great

- **Change Size Easily (Like Magic Growth):** You can make your online tools bigger or smaller whenever you need to, saving money and resources.
- **Always Works (Like a Reliable Friend):** AWS is always available, so your website or app won't suddenly disappear. It's reliable like a good friend.
- **Fits Your Needs (Like a Custom-Made Shirt):** You can choose exactly what you need from AWS, making it perfect for your specific projects.

- **Pays as You Use (Like Paying for a Snack):** You only pay for what you use, just like buying a snack. No need to pay for things you don't use.

Accessing AWS Servers: Understanding Managed Servers and Serverless Architecture

Managed Servers: Building Blocks Made Easy

Think of managed servers on AWS like ready-made LEGO sets. These servers are like complete, pre-built structures that include all the essential parts of a computer: CPU (the brain), memory (short-term memory), storage (long-term memory), and network components (communication lines). AWS provides managed servers like Amazon EC2, which are akin to these pre-assembled LEGO sets. Just like you can start building with LEGO right away, these servers are ready for you to use immediately. They're excellent when you need specific and predictable resources for your online projects.

Example: Amazon EC2 (Elastic Compute Cloud)

Amazon EC2 is like a specialized LEGO set provided by AWS. It comes with various pre-assembled servers, each tailored for specific tasks. For instance, if you need a server to handle a website, there's a pre-built one with all the necessary components. You don't have to worry about putting the pieces together; it's ready to use, just like a complete LEGO structure.

Serverless Architecture: Embracing Simplicity and Efficiency

Now, consider serverless architecture as having magical assistants at your disposal. In this scenario, you don't have to manage servers at all. It's as if you have a team of helpers who appear whenever you need them. Instead of dealing with entire servers, you focus only on specific tasks or functions your website or application needs to perform. AWS provides serverless services like AWS Lambda, which allow you to run code without provisioning or managing servers.

Example: AWS Lambda

Imagine you want to create a function that sends you an email whenever someone fills out a form on your website. With AWS Lambda, you can write the code for this function, and AWS takes care of executing it whenever the form is filled out. You don't have to worry about servers; you only focus on the task you want to accomplish. It's like having magical helpers handling the work behind the scenes, making things happen without the complexity of managing servers.

Operating System Requirements: Picking the Right Digital Foundation

Imagine your computer or phone as a house, and the operating system (OS) as the foundation. The OS is like the solid ground upon which everything else stands. It's important to choose the right foundation because different tasks need different kinds of support. Let's break this down:

1. Importance of OS Selection:

Think of it like choosing a house for your needs:

Just like you wouldn't build a skyscraper on soft soil, you don't use the same OS for all tasks. For example, some OS are best for running games, while others are great for handling data in databases. Picking the right OS ensures your computer runs smoothly and efficiently for what you want to do.

2. Examples of OS Requirements for Specific Services:

Different rooms for different purposes:

- **Web Browsing and Emails:** For basic internet usage, you can use widely used OS like Windows, macOS, or various versions of Linux. They are like comfortable rooms where you can access the internet and send emails easily.

- **Web Development:** If you're building websites, you might prefer macOS or Linux. These OS have tools that developers love, making it like having a room full of advanced building tools and blueprints.
- **Heavy Graphics and Gaming:** For high-end gaming or graphic design work, Windows is often the preferred OS. It's like a specially designed gaming room with all the latest gadgets.
- **Server Operations:** When you're managing servers (like those on AWS), Linux is often the choice. It's like the control room of a big building where you can monitor and manage everything effectively.
- **Mobile Apps:** For creating mobile apps, macOS is popular among developers. It's like a creative workshop where you can design apps for phones and tablets.

In simple terms, just like you choose a house with the right number of rooms and facilities for your family, selecting an OS tailored to your tasks ensures your digital activities run smoothly and efficiently. It's like customizing your home to fit your needs perfectly.

Metadata in AWS

In the world of AWS, metadata serves as the digital blueprint of your virtual space. Imagine it as a special map that tells you exactly where your online tools are located, what kind of tools they are, and how they communicate with the outside world. This includes essential information like the location of your server, what type of server it is, and its public IP.

For instance, think of metadata as the digital nameplate outside your virtual office, providing crucial details about the services running inside.

How to retrieve Metadata of Instance?

In Amazon Web Services (AWS), you can retrieve metadata of an instance by accessing a special URL provided within the instance. The metadata is available at a well-known endpoint, and you can retrieve it using HTTP requests. Here's how you can do it:

To retrieve the metadata of an AWS instance, you need to make an HTTP request to the following URL using curl command.

<http://169.254.169.254/latest/meta-data/>

```
ubuntu@ip-172-31-      :~$ curl http://169.254.169.254/latest/meta-data/  
ami-id  
ami-launch-index  
ami-manifest-path  
block-device-mapping/  
events/  
hostname  
identity-credentials/  
instance-action  
instance-id  
instance-life-cycle  
instance-type  
local-hostname  
local-ipv4  
mac  
metrics/  
network/  
placement/  
profile  
public-hostname  
public-ipv4  
public-keys/  
reservation-id  
security-groups  
services/
```

This IP address, 169.254.169.254, is a link-local address that AWS instances recognize. By appending specific paths to this base URL, you can retrieve various metadata about the instance.

For example, if you want to retrieve the metrics of the instance, you will make an HTTP GET request to:

```
ubuntu@ip-172-31-22-151:~$ curl http://169.254.169.254/latest/meta-data/metrics; echo "";  
vhostmd
```

To retrieve the **Instance-type** or **AMI-Id** of the instance, we can retrieve it by using the following command.

```
ubuntu@ip-172-31-22-151:~$ curl http://169.254.169.254/latest/meta-data/instance-type; echo "";  
t2.micro  
ubuntu@ip-172-31-22-151:~$ curl http://169.254.169.254/latest/meta-data/ami-id; echo "";  
ami-0694d931cee176e7d
```

Use of Metadata in Automation Scripts

You can incorporate this metadata retrieval into your automation scripts. For instance, in a shell script, you can store the instance ID in a variable like this:

```
root@ip-172-31-27-202:~# INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)  
root@ip-172-31-27-202:~# echo "Instance ID: $INSTANCE_ID"  
Instance ID: i-0e1de735f975b7d82  
root@ip-172-31-27-202:~#
```

Once you have the metadata values, you can use them in your automation tasks. For example, you can dynamically configure software based on the instance type, perform conditional actions based on instance tags, or update load balancer settings dynamically.

Remember that this metadata is accessible only from within the EC2 instance itself. It provides a convenient and dynamic way to fetch information about the instance, enabling you to automate tasks without hardcoding specific instance details.

User data in AWS

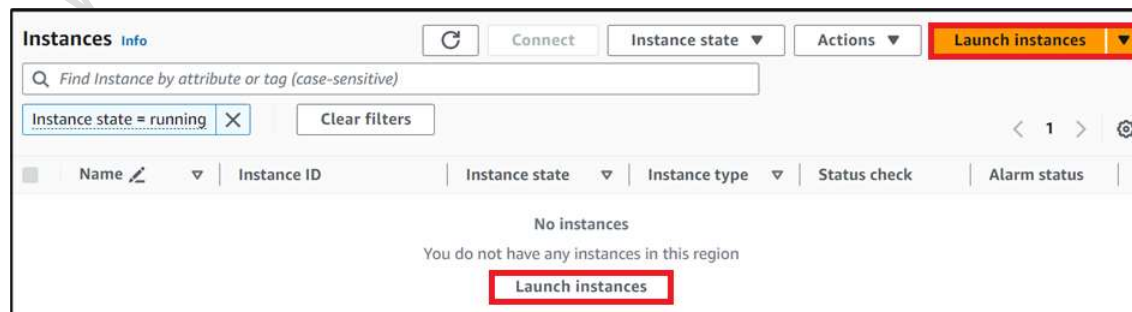
In Amazon Web Services (AWS), user data refers to the information that you can pass to an Amazon EC2 (Elastic Compute Cloud) instance during its launch. This data can be used to perform various automated tasks and configurations on the instance at the time of its initialization.

User data is commonly utilized for tasks such as:

- **Bootstrap Scripting:** You can provide a script or a set of commands as user data. When the EC2 instance starts, these commands are executed automatically. This is particularly useful for tasks like software installation, configuration, or system updates.
- **Cloud-Init Configuration:** AWS uses a tool called "cloud-init" to handle user data. Cloud-init interprets the user data and applies configurations such as creating users, setting up SSH keys, installing packages, and more, all based on the provided instructions.
- **Customizing Instances:** User data allows you to customize instances based on specific requirements without manual intervention. For example, you can pass user data to configure a web server, set up a database, or join an instance to a specific domain.
- **Automation and Scaling:** User data is essential for automation, especially in scenarios involving Auto Scaling groups or launching instances via AWS Management Console, AWS CLI, or SDKs. It ensures that instances are automatically configured based on your specifications.

Implementation of User Data

Let's explore how to set up user data while launching an instance in AWS, allowing us to automate configurations and tasks during the initialization process.



1) Click on Launch Instance Button

Himanshu-WebServer [Add additional tags](#)

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat

Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.2.2...[read more](#)
ami-05c0f5389589545b7

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

[Cancel](#) [Launch instance](#)

When launching an instance, it's a best practice to assign a descriptive tag name. In my example, I've named it 'Himanshu-WebServer.' I've chosen the Amazon Linux operating system and selected the instance type as 't2.micro'.

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-12' with the following rules:

☒ Allow SSH traffic from
Helps you connect to your instance 0.0.0.0/0

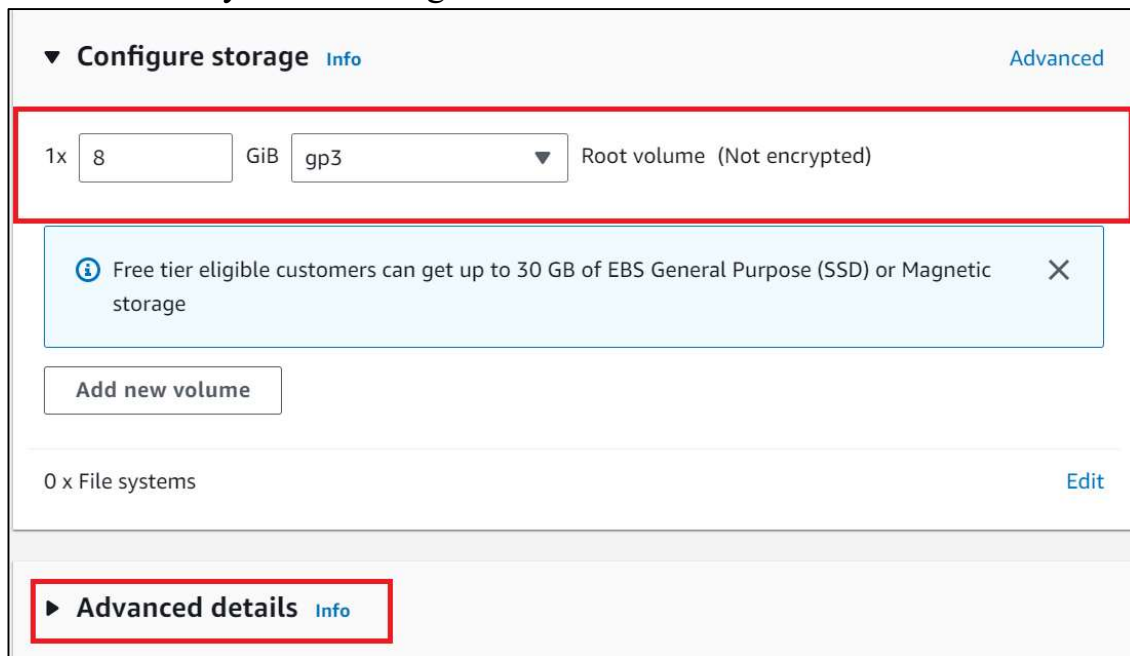
☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

While Configuring the OS, it is recommended to set what are the Ports you want to open for public world by which they can access your application. It doesn't mean you allowed all the public guys to connect with all the Ports in your System, also this is not a good practice. In my case, I have allowed SSH, HTTP, HTTPS for my webserver.

Now, scroll down to the same page to find the storage configuration. In my case, I've selected **8 GB of storage**. Next, click on the '**Advanced details**'

section where you can configure the User Data.



▼ **Configure storage** [Info](#) Advanced

1x 8 GiB gp3 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

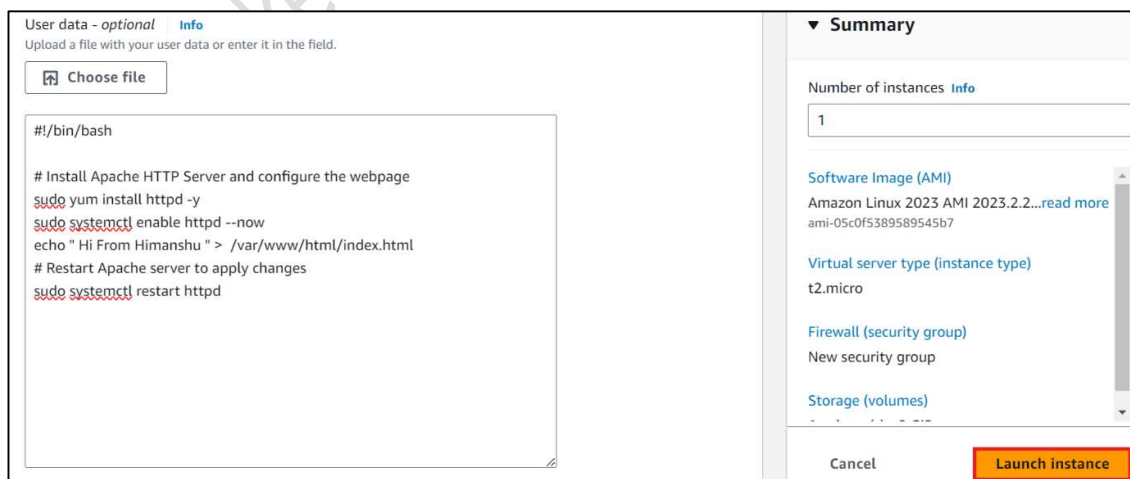
0 x File systems [Edit](#)

► **Advanced details** [Info](#)

Purpose: This User Data script is used to automate the setup of a web server on an Amazon EC2 instance.

```
# aws_user_data_ec2
#!/bin/bash

# Install Apache HTTP Server and configure the webpage
sudo yum install httpd -y
sudo systemctl enable httpd --now
echo " Hi From Himanshu " > /var/www/html/index.html
# Restart Apache server to apply changes
sudo systemctl restart httpd
```



User data - optional [Info](#)
Upload a file with your user data or enter it in the field.

Choose file

```
#!/bin/bash

# Install Apache HTTP Server and configure the webpage
sudo yum install httpd -y
sudo systemctl enable httpd --now
echo " Hi From Himanshu " > /var/www/html/index.html
# Restart Apache server to apply changes
sudo systemctl restart httpd
```

▼ **Summary**

Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.2.2...[read more](#)
ami-05c0f5389589545b7

Virtual server type (instance type)
t2.micro

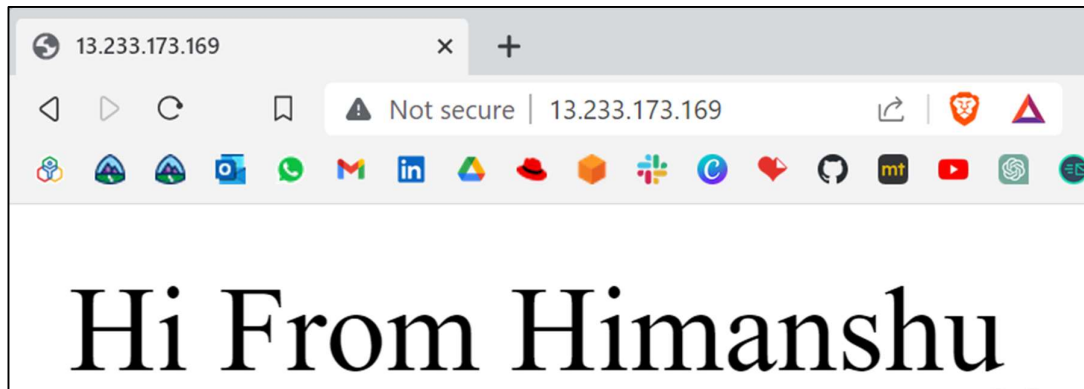
Firewall (security group)
New security group

Storage (volumes)
-

Cancel **Launch instance**

Click on Launch Instance.

When someone visits the EC2 instance's public IP address in a web browser, they will see the message: Hi, From Himanshu



While launching the Instance, The Cloud init program helps to execute the user script, and their Log is maintained inside the location /var/log/ with the name of cloud-init-output.log

```
[ec2-user@ip-172-31-39-89 ~]$ sudo su -
[root@ip-172-31-39-89 ~]# cd /var/log/
[root@ip-172-31-39-89 log]# ls
README  audit  chrony  cloud-init.log  dnf.log  hawkey.log  journal  private  sssd  wtmp
amazon  btmap  cloud-init-output.log  dnf.librepo.log  dnf.rpm.log  httpd  lastlog  sa  tallylog
```

View this file using cat command: cat cloud-init-output.log

```
Cloud-init v. 22.2.2 running 'modules:config' at Thu, 02 Nov 2023 07:00:24 +0000. Up 14.69 seconds.
Cloud-init v. 22.2.2 running 'modules:final' at Thu, 02 Nov 2023 07:00:25 +0000. Up 15.94 seconds.
Amazon Linux 2023 repository 20 MB/s | 21 MB 00:01
Amazon Linux 2023 Kernel Livepatch repository 548 kB/s | 162 kB 00:00
Dependencies resolved.
=====
Package Arch Version Repository Size
=====
Installing:
httpd x86_64 2.4.56-1.amzn2023 amazonlinux 48 k
Installing dependencies:
apr x86_64 1.7.2-2.amzn2023.0.2 amazonlinux 129 k
apr-util x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 98 k
generic-logos-httpd noarch 18.0.0-12.amzn2023.0.3 amazonlinux 19 k
httpd-core x86_64 2.4.56-1.amzn2023 amazonlinux 1.4 M
httpd-filesystem noarch 2.4.56-1.amzn2023 amazonlinux 15 k
httpd-tools x86_64 2.4.56-1.amzn2023 amazonlinux 82 k
libbrotli x86_64 1.0.9-4.amzn2023.0.2 amazonlinux 315 k
mailcap noarch 2.1.49-3.amzn2023.0.3 amazonlinux 33 k
Installing weak dependencies:
apr-util-openssl x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 17 k
mod_http2 x86_64 2.0.11-2.amzn2023 amazonlinux 150 k
mod_lua x86_64 2.4.56-1.amzn2023 amazonlinux 62 k
Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64
apr-util-1.6.3-1.amzn2023.0.1.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.56-1.amzn2023.x86_64
httpd-core-2.4.56-1.amzn2023.x86_64
httpd-filesystem-2.4.56-1.amzn2023.noarch
httpd-tools-2.4.56-1.amzn2023.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_http2-2.0.11-2.amzn2023.x86_64
mod_lua-2.4.56-1.amzn2023.x86_64
Complete!
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service -> /usr/lib/systemd/system/httpd.service.
Cloud-init v. 22.2.2 finished at Thu, 02 Nov 2023 07:00:48 +0000. Datasource DataSourceEc2. Up 38.14 seconds
[root@ip-172-31-39-89 log]#
```

The Httpd software is installed, and the service is also enabled.

Thank You For Reading.