

A Hands-On Guide to Key Features of Amazon EC2 (Elastic Compute Cloud)

- **Introduction and Purpose of having EC2 Instances**
- **Instance Types**
- **Regions and Availability Zones**
- **Purchasing Options**
- **AMI**
- **Key Pairs**
- **Launch Templates**
- **Placement Groups**
- **Elastic Block Store (EBS) and Utilizing Snapshots**
- **Elastic IPs**
- **Load Balancers**
- **Elastic Load Balancer (ELB)**
- **Target Groups**
- **CloudWatch Monitoring**
- **Auto Scaling Groups (ASG)**

Created by Athira KK

A Brief Introduction to EC2 Instance:

An EC2 instance, short for Elastic Compute Cloud instance, is a virtual server offered by Amazon Web Services (AWS) that provides resizable compute capacity in the cloud. Essentially, it allows users to rent virtual servers to run their applications or host their websites without the need to invest in physical hardware.

- EC2 instances offer a variety of configurations for CPU, memory, storage, and networking, catering to diverse workload requirements.
- Users can select from optimized instance types, such as general-purpose, memory-intensive, or high-performance computing, to suit their specific needs.
- Deployment and management are simplified through AWS Management Console, CLI, or SDKs, facilitating easy provisioning, monitoring, and scaling of instances.
- EC2 serves a wide range of use cases, from basic web hosting to complex data analytics, meeting the computing requirements of businesses across industries and sizes.
- EC2 follows a pay-as-you-go pricing model, ensuring cost-effectiveness by billing users based on instance type, usage duration, and other factors, thus accommodating varying business needs and budget constraints.

Purpose of having an EC2 Instance

- **Flexibility:** With a wide range of instance types and configurations, EC2 instances cater to diverse workload requirements, from basic web hosting to high-performance computing and data analytics.
- **Scalability:** EC2 instances offer scalable compute capacity, allowing businesses to easily adjust resources based on demand fluctuations. This scalability ensures efficient resource utilization and cost-effectiveness.
- **Global Reach:** With AWS's global infrastructure, EC2 instances are available in multiple regions worldwide. This enables businesses to deploy applications closer to their target audience for improved performance and latency optimization.
- **Security and Reliability:** EC2 instances provide a secure and reliable environment for running applications, with features like built-in encryption, firewalls, and monitoring tools to safeguard data and ensure high availability.
- **Ease of Management:** EC2 instances can be quickly deployed and managed through AWS Management Console, CLI, or SDKs, streamlining the process of provisioning, monitoring, and scaling instances as needed.

Amazon EC2 offers a variety of instance types tailored for diverse needs. These types feature different configurations of CPU, memory, storage, and networking capacity, providing flexibility in selecting resources for your applications. Additionally, each instance type includes multiple sizes, enabling easy scaling according to your workload's demands.

EC2 Instance Types Overview:

General Purpose:

- These instances offer a balanced mix of compute, memory, and networking resources.
- Ideal for applications like web servers and code repositories that require equal proportions of these resources.

Burstable Performance:

- Also known as the T instance family, these instances provide baseline CPU performance with the ability to burst above it when needed.
- Refer to the Amazon EC2 User Guide for Linux Instances for more details.

Compute Optimized:

- Designed for compute-intensive applications requiring high-performance processors.
- Suitable for tasks such as batch processing, media transcoding, and high-performance web servers.

Memory Optimized:

- Tailored for workloads processing large datasets in memory.
- Deliver fast performance for applications with substantial memory requirements.

Storage Optimized:

- Intended for workloads demanding high, sequential read and write access to large data sets on local storage.
- Optimized to provide high IOPS for applications requiring rapid random I/O operations.

Accelerated Computing:

- Utilizes hardware accelerators or co-processors for efficient processing of functions like floating-point calculations and graphics processing.
- Enhances performance compared to software-based processing on CPUs.

High-Performance Computing (HPC):

- Purpose-built for running HPC workloads at scale on AWS, offering optimal price-performance.
- Ideal for complex simulations and deep learning tasks requiring high-performance processors.

Previous Generation:

- AWS continues to support previous generation instance types for users who have optimized their applications around them.
- While current generation instance types offer better performance, previous generation types are still available for legacy applications.

Understanding AWS Regions, Availability Zones, and Deployment Strategies

Regions:

- AWS divides its global infrastructure into geographical regions, such as North America, Europe, Asia Pacific, etc.
- Each region is completely isolated from others and consists of multiple Availability Zones.
- Users select a region when deploying resources like EC2 instances. The choice of region affects latency, compliance, and data residency.

Availability Zones (AZs):

- Availability Zones are distinct data centers within a region, designed for fault tolerance and high availability.
- They are physically separate from each other and are connected through high-speed, low-latency links.
- Deploying resources across multiple AZs enhances fault tolerance. If one AZ experiences an issue, applications can continue running in others without interruption.

Deployment Strategy:

- To ensure high availability and fault tolerance, it's recommended to distribute resources (like EC2 instances) across multiple AZs within a region.
- This strategy protects against failures or outages in a single AZ and ensures that applications remain accessible even if one AZ becomes unavailable.

Scalability and Performance:

- Deploying resources across multiple AZs allows for better scalability and performance.
- Load balancers can distribute incoming traffic evenly across multiple AZs, optimizing resource utilization and improving response times for users.

Data Residency and Compliance:

- Different regions may have varying data residency regulations and compliance requirements.
- Users need to select the appropriate region and AZs to ensure compliance with local regulations and keep data within designated geographic boundaries.

Cost Considerations:

- Costs for AWS services may vary across regions, so users should consider pricing differences when selecting a region for deployment.
- Additionally, data transfer costs may apply when transferring data between regions or AZs.

EC2 Major Purchasing options:

1. **On Demand Instances** – Pay for the compute capacity per second , no up-front payment.We can decide when to launch,terminate,start.stop,or reboot.

Use cases: short-term work-loads , irregular workloads that cannot be interrupted.

2. **Reserved Instances** – We can reserve instances in terms of 1year and 3year. Provide up to 70% discount compared to On-Demand. Payment method can be all upfront, partial upfront,no upfront. If you purchase Reserved Instances of a specific type in an availability zone and the same type is available in same zone then automatically charge at lower rate.

Use cases – Long-term workloads , Applications with steady state or predictable usage.

3. **Spot Instances** – Most cost effective , We can request for unused instances up to 90% discount. But if someone is paying more than of yours, then instance will get loosed. So it is less reliable.

Use cases – Image processing , non-critical applications.

4. **Dedicated Hosts**– Physical server completely dedicated for your use. More expensive and can be purchase as on demand or reserved. Billing will be per-host.

Use cases – Companies that have strong regulatory or compliance needs.

5.**Dedicated Instances** – Instances running on hardware that's dedicated to a single user. It can share hardware with other instances in same account. Billing will be per-instance.

6.**Savings Plans** – This will assist the user to lower the EC2 costs if they agree to keep it for a specific period – ideally, one or three years. The number of active instances are unrestricted.

AMI (Amazon Machine Image):

- An AMI is a template used to create virtual machines (instances) within the Amazon EC2 service.
- It contains the operating system, application server, and applications necessary to launch an instance.
- You can create custom AMIs or use pre-built AMIs provided by AWS and other users.

AMI Images:

- AMI images are the actual files stored in Amazon S3 that represent the templates used to launch EC2 instances.
- These images include the root volume snapshot and metadata required to launch an instance.
- AMI images are versioned, allowing you to maintain multiple versions of an AMI and roll back to previous versions if needed.

Placement Groups

- Placement groups are the way of logically grouping interdependent instances together in a selected region.
- In other words, instances that are existing within a common availability zone can be grouped under placement group in order to suit workload requirements.
- By using the placement group, we will increase the performance or improve the availability.

Types

- Cluster placement group
- Spread placement group
- Partition placement group

Benefits

- Can be able to launch multiple instances within the same AZ.
- Low network latency.
- High network throughput.

Limitations

- Cannot merge placement groups.
- Cannot launch dedicated hosts in placement groups.
- Instances cannot span into multiple placement groups.

Cluster Placement Group

- All the instances are placed in a same hardware rack inside an availability zone.
- But if the rack fails, all instances fail at the same time.
- This type of group will enable to achieve low latency network.
- Good for High Performance applications.
- **Use Case** – Application with low latency and high throughput.

Spread Placement Group

- All EC2 instances are placed in different hardware racks in a single availability zone.
- A rack failure will not affect more than one instance.
- We can create up to 7 EC2 instances per availability zone in spread placement group.
- Good for high availability applications and not suitable for high performance applications.
- It can span multiple availability zones in the same region.
- **Use Case**- Critical applications that needs to be isolated from failure from each other.

Partition Placement Group

- Group of instances spread across racks, and the instances in one partition do not share the underlying hardware with instances in different partition.
- If a rack fails, it will affect on the instances within that particular partition only.
- Partitions can be in different availability zones in the same region.
- It strikes a balance between high performance and high availability.
- **Use Case** – Big data applications like HDFS, HBase, Cassandra, Kafka.

In AWS EC2, there are three main types of load balancers:

Classic Load Balancer (CLB):

- CLB is the original load balancer service provided by AWS.
- It operates at both the application and transport layers (Layer 4 and Layer 7) of the OSI model.
- CLB supports distributing incoming traffic across multiple EC2 instances in multiple Availability Zones.
- It is suitable for applications that require simple load balancing without advanced features.

Application Load Balancer (ALB):

- ALB operates at the application layer (Layer 7) of the OSI model.
- It supports advanced routing features such as host-based routing, path-based routing, and routing based on HTTP headers.
- ALB can handle HTTP and HTTPS traffic and provides support for WebSocket and HTTP/2 protocols.
- It is ideal for modern web applications with multiple microservices or containers.

Network Load Balancer (NLB):

- NLB operates at the transport layer (Layer 4) of the OSI model.
- It provides ultra-low latency and high throughput, making it suitable for handling millions of requests per second.
- NLB supports routing traffic to targets using IP addresses and TCP ports, making it ideal for TCP and UDP-based applications.
- It is commonly used for applications that require high performance, scalability, and reliability, such as gaming, IoT, and real-time communication applications.

Target Groups are a fundamental component of Elastic Load Balancing (ELB) services in AWS, specifically associated with the Application Load Balancer (ALB) and Network Load Balancer (NLB).

Here's a concise explanation:

Routing:

- When configuring a load balancer (ALB or NLB), you specify one or more target groups to route traffic to.
- Each target group defines the criteria for routing requests to its associated targets, such as based on a specific path, host, or HTTP header.

Health Checks:

- Target Groups perform health checks on their registered targets to ensure they are capable of handling requests.
- Unhealthy targets are automatically removed from the target group until they pass health checks again.

Load Balancing:

- Target Groups distribute incoming traffic evenly across their registered targets according to the configured load balancing algorithm (e.g., round-robin or least outstanding requests).

Port Configuration:

- Target Groups allow you to specify the port on which the targets receive traffic, allowing flexibility in routing requests to different ports on the same target.

Autoscaling:

- Autoscaling is a feature provided by AWS that automatically adjusts the number of EC2 instances in a fleet based on user-defined policies.
- It helps maintain the desired number of instances to handle varying levels of application traffic or workload demand.
- Autoscaling can scale both up (adding more instances) and down (removing instances) dynamically, ensuring optimal resource utilization and cost efficiency.

Autoscaling Groups:

- An Autoscaling Group (ASG) is a logical grouping of EC2 instances managed by the Auto Scaling service.
- ASGs define the scaling policies, launch configurations, and other parameters that govern the behavior of the instances within the group.
- ASGs automatically launch or terminate EC2 instances based on predefined scaling policies and health checks.
- ASGs can be configured to distribute instances across multiple Availability Zones to improve fault tolerance and high availability.

Ways to connect to EC2 instances via the command line:

- SSH connection
- EC2 Instance Connect:
- AWS CLI (Command Line Interface):
- Third-Party Tools (like PuTTY or MobaXterm)

and more.....

The launch of an ec2 instance:

- Choose an AMI (Amazon Machine Image): This is a template for the virtual machine you want to create. It contains the operating system, software, configuration, and sometimes data.
- Select an Instance Type: This determines the computational resources (CPU, RAM, storage, etc.) of your EC2 instance. There are various types optimized for different use cases.
- Configure Instance Details: This includes settings like networking (VPC, subnet), IAM role, and user data (scripts or data to be run at instance launch).
- Add Storage: Configure the amount and type of storage for your instance. This is typically done using Amazon EBS (Elastic Block Store), which provides block-level storage volumes.
- Add Tags: Assign metadata to your instance in the form of key-value pairs. This helps with organization, management, and billing.
- Configure Security Group: Define firewall rules for controlling inbound and outbound traffic to your instance. Security groups act as virtual firewalls.
- Review and Launch: Review your instance configuration, make any necessary changes, select an existing key pair or create a new one for SSH access (for Linux instances), and then launch the instance.

After launching the instance, you can access it using SSH (for Linux instances) or RDP (for Windows instances) using the key pair you selected or created. Remember to properly secure your instance and follow best practices for managing AWS resources.

Let's walk through the process of launching an EC2 instance.

An EC2 instance essentially functions as a virtual machine.

Firstly, log in to your AWS console.

Next, choose your desired region. For instance, I'll be selecting the North Virginia region. It's advisable to select a region based on your current location, as some regions may incur higher costs compared to those in the US.

Each region is identified by a code, such as US East 1 for North Virginia.

To navigate to the EC2 service, click on "Services" and locate EC2 either by browsing through all services or by directly searching for EC2.

The screenshot shows the AWS search interface with the query 'ec2'. The left sidebar has sections for Snapshot, Launch i (with 'Launch' highlighted), and Instance. The main search results page shows 'Services (13)' and a list of services including EC2, EC2 Image Builder, Recycle Bin, and Amazon Inspector.

Service	Description
EC2	Virtual Servers in the Cloud
EC2 Image Builder	A managed service to automate build, customize and deploy OS images
Recycle Bin	Protect resources from accidental deletion
Amazon Inspector	Continual vulnerability management at scale

Once in the EC2 dashboard, you'll find an overview of your instances, volumes, and other relevant information. Keep in mind that the dashboard doesn't auto-refresh, so be sure to manually refresh after making any changes.

Scrolling down, you'll notice zones within the region, each identified by a code like 1A, 1B, etc. Typically, a region will have at least two zones.

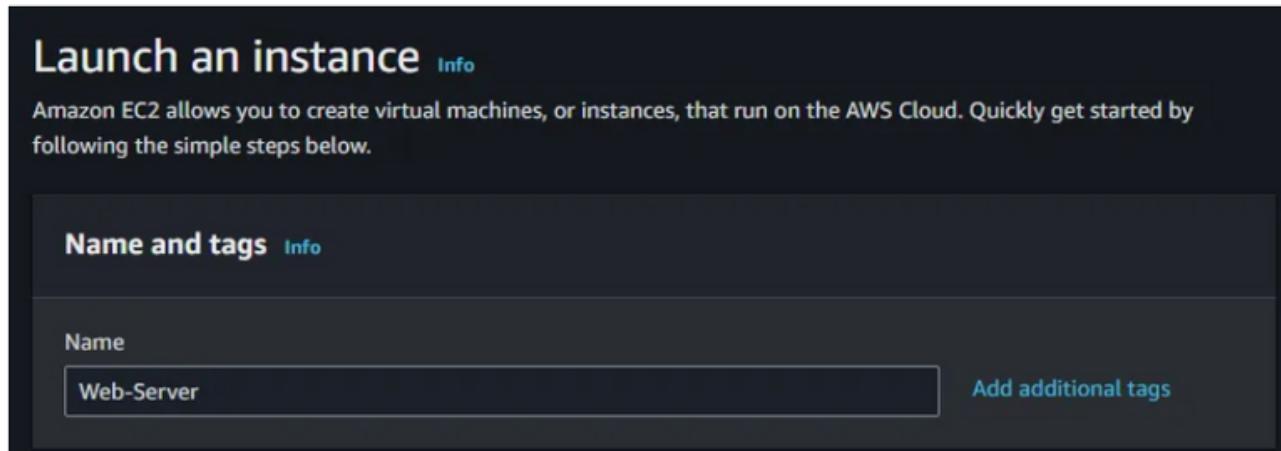
The screenshot shows the AWS Health Dashboard for the EC2 service. It displays the Region (US East (N. Virginia)) and Status (This service is operating normally). Below this, the 'Zones' section lists six zones with their corresponding Zone IDs:

Zone name	Zone ID
us-east-1a	use1-az1
us-east-1b	use1-az2
us-east-1c	use1-az4
us-east-1d	use1-az6
us-east-1e	use1-az3
us-east-1f	use1-az5

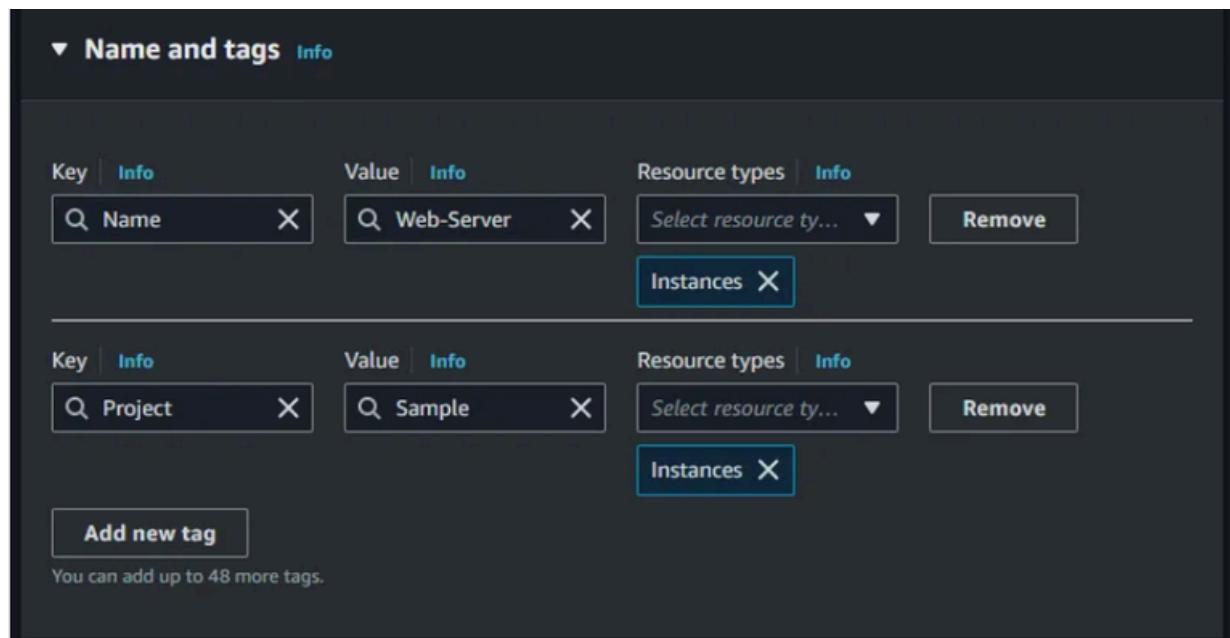
[Enable additional Zones](#)

Click on “Instances” and then “Launch Instance” to initiate the instance creation process.

Before proceeding, it’s worth noting that the AWS console interface may evolve over time, but the core options and functionalities remain consistent.



Now, let's begin by assigning a name and optional tags to the instance. Tags can be useful for filtering and organization purposes.



Moving on to selecting the Amazon Machine Image (AMI), you'll find a variety of options including quick start AMIs like Amazon Linux, Ubuntu, and Windows, as well as community AMIs and those from the AWS Marketplace. Carefully review the details and charges associated with each AMI.

For this example, let's choose the Ubuntu AMI provided by Amazon Web Services, which is free.

▼ 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.

 [Search our full catalog including 1000s of application and OS images](#)

Quick Start



[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-080e1f13689e07408 (64-bit (x86)) / ami-0a55ba1c20b74fc30 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-03-01

Architecture

64-bit (x86) ▾

AMI ID

ami-080e1f13689e07408

Verified provider

Ensure that the instance type selected aligns with your requirements. For free-tier usage, T2 micro instances are suitable for most learning purposes.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

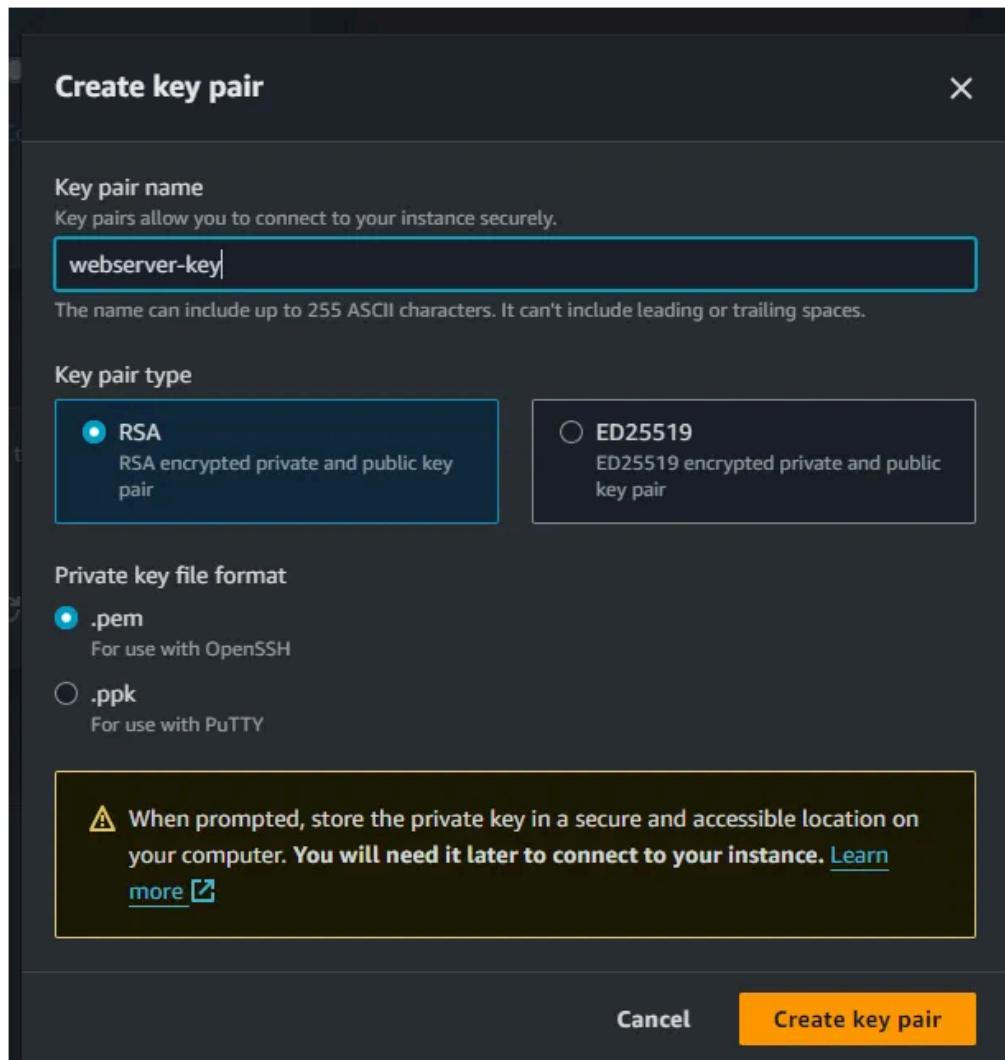
Free tier eligible

All generations

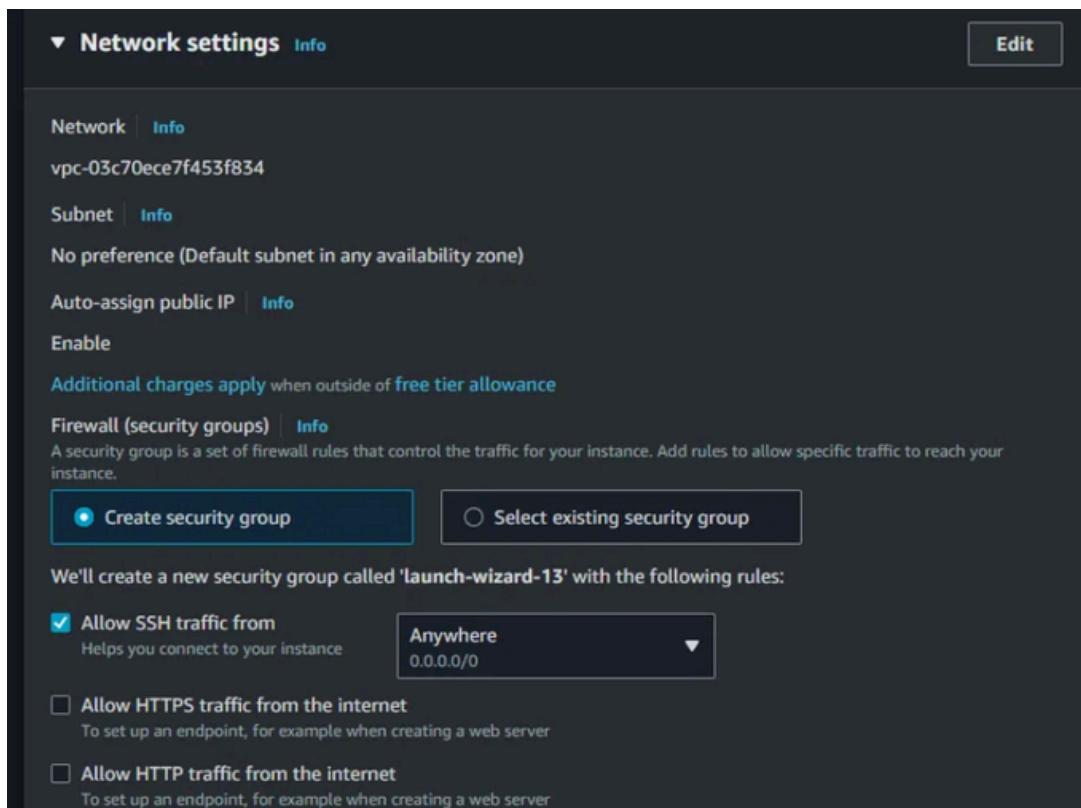
[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

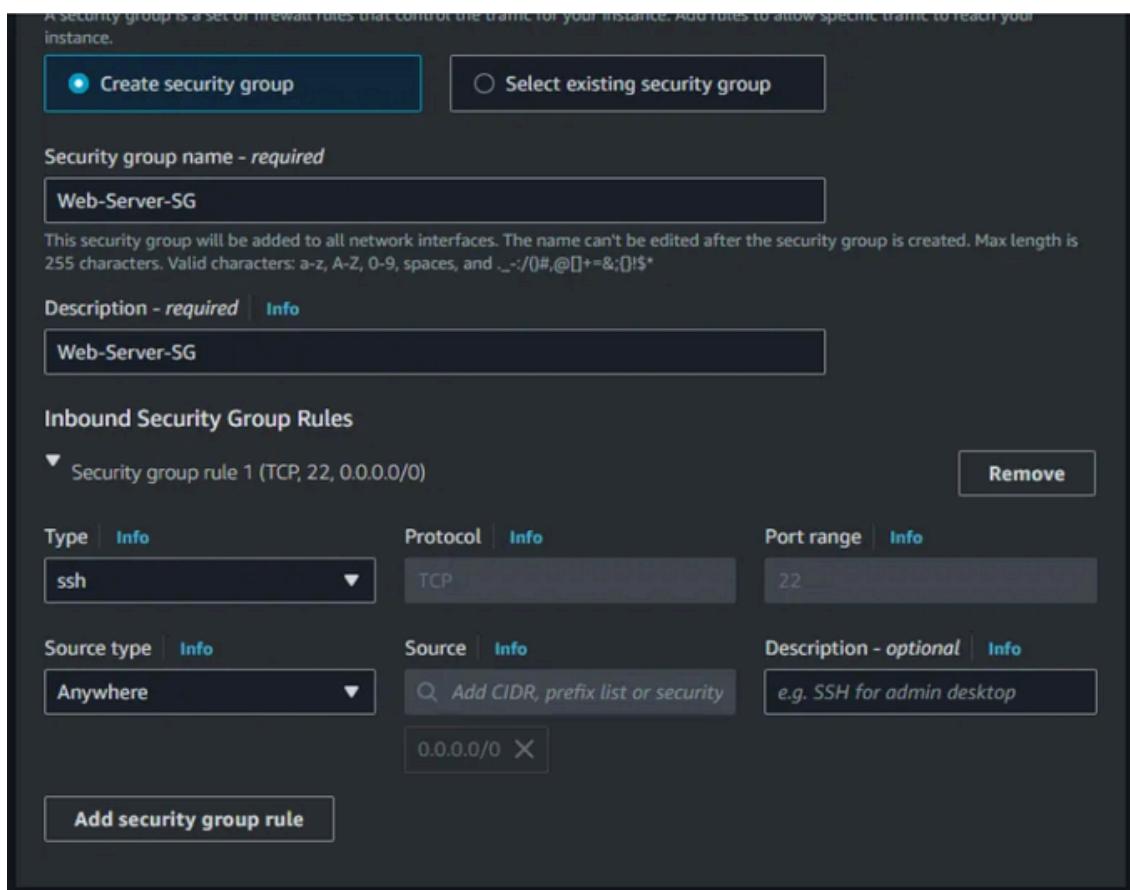
- Key pairs are used primarily for securely accessing EC2 instances. When you create an EC2 instance, you can specify a key pair to use for SSH access to the instance. The key pair consists of a public key that AWS stores, and a private key file that you download to your local machine.
- It's important to note that AWS doesn't store your private key, so if you lose it, there's no way to retrieve it from AWS. Make sure to securely store your private keys and avoid sharing them with unauthorized users.



Proceed to configure the network settings, including security groups. Security groups act as mandatory firewalls for EC2 instances.



Customize the security group to allow necessary inbound and outbound traffic.



You can specify the storage size, typically defaulted to 10 GB, and further advanced details like user data, which allows for automated provisioning upon instance launch. Finally, review your configuration and launch the instance.

Upon launching, the instance may take a few moments to become available. You can monitor its status in the EC2 dashboard.

Next, establish a connection to the instance; in this example, I'm using EC2 Instance Connect.

There are two methods to install Apache2:

- By incorporating commands in the user data section before initiating the instance launch, as demonstrated below:

The screenshot shows the 'User data - optional' section. It includes a note: 'Upload a file with your user data or enter it in the field.' Below is a 'Choose file' button. A code editor displays the following commands:
sudo apt update
sudo apt install apache2
sudo systemctl start apache2
sudo systemctl status apache2

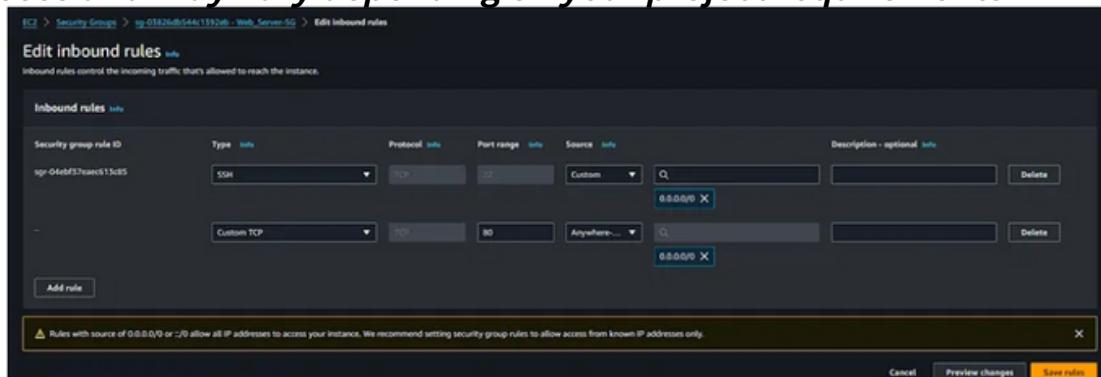
- Or, by executing the commands after connecting the ec2 instance via terminal.

- *sudo apt update*
- *sudo apt install apache2 -y*
- *sudo systemctl start apache2*
- *sudo systemctl enable apache2*
- *sudo systemctl status apache2*

Troubleshooting tip: If encountering connection issues, ensure the correct path to the SSH key or navigate to the directory containing the key before attempting to connect.

Once connected, you can verify the instance status, access web services like Apache, and configure firewall rules as needed to enable external access to your services.

Additionally, click on the Security Group, edit inbound rules, add the following rule, and save the rules. It's important to note that allowing all IPs is solely for learning purposes and may vary depending on your project requirements.



Go to the browser and copy the public IP followed by ':80' to view the Apache page.

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

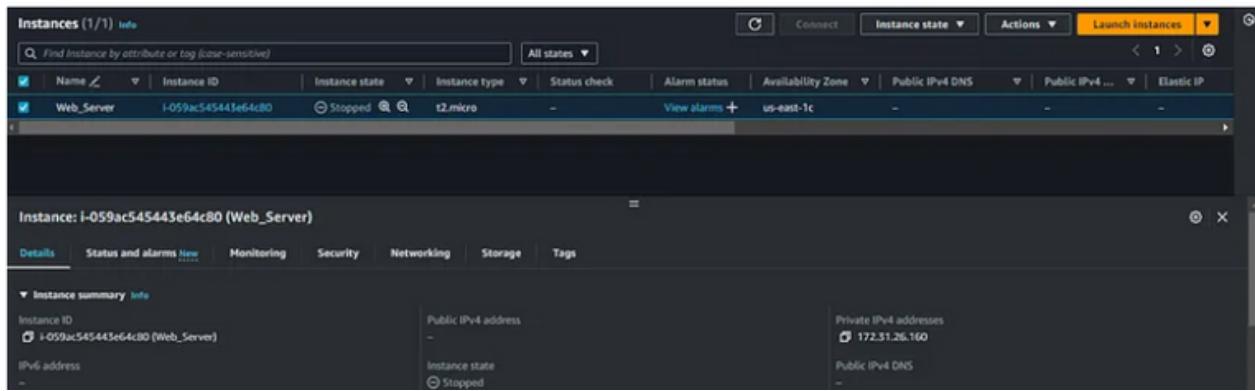
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   '-- ports.conf
|-- mods-enabled
|   '-- *.Load
|   '-- *.conf
|-- conf-enabled
|   '-- *.conf
|-- sites-enabled
|   '-- *.conf
```

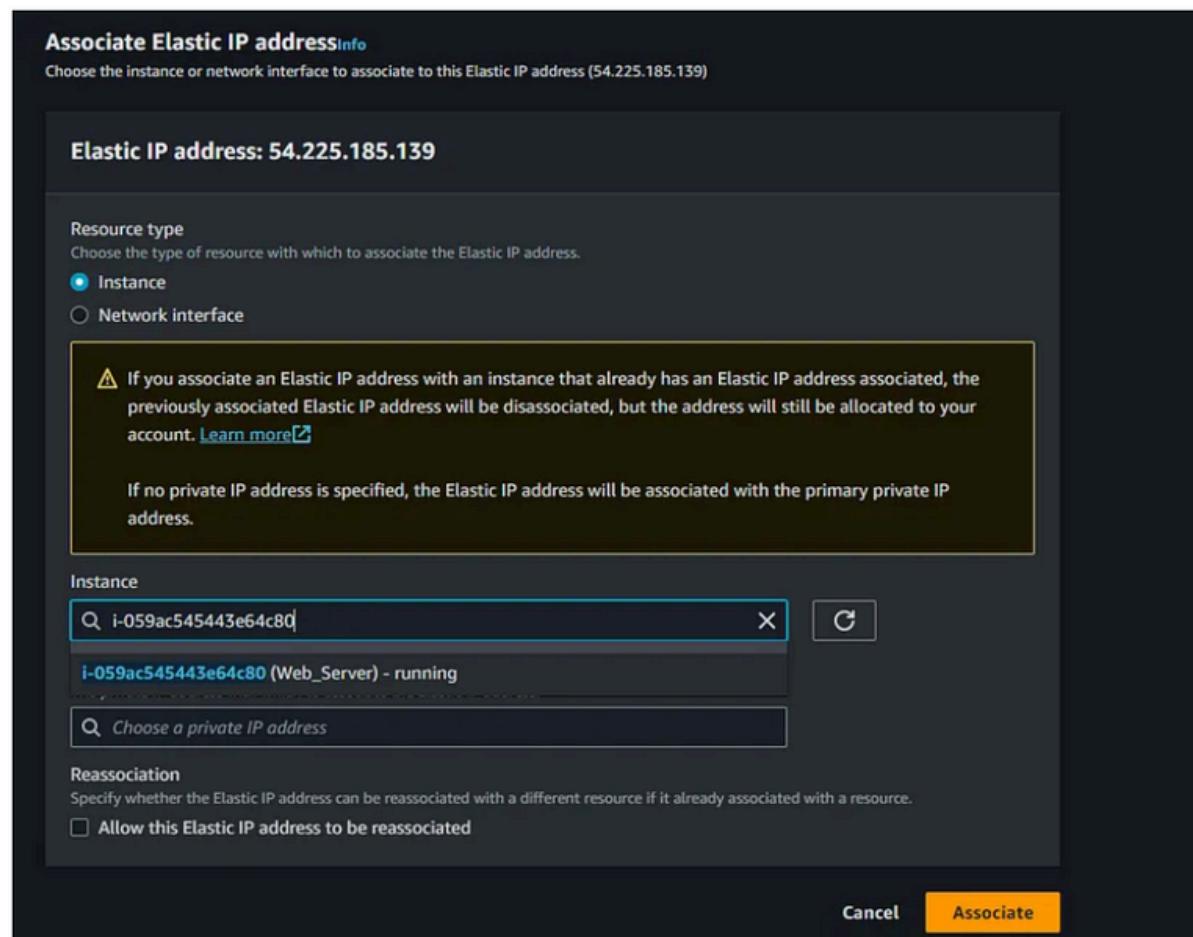
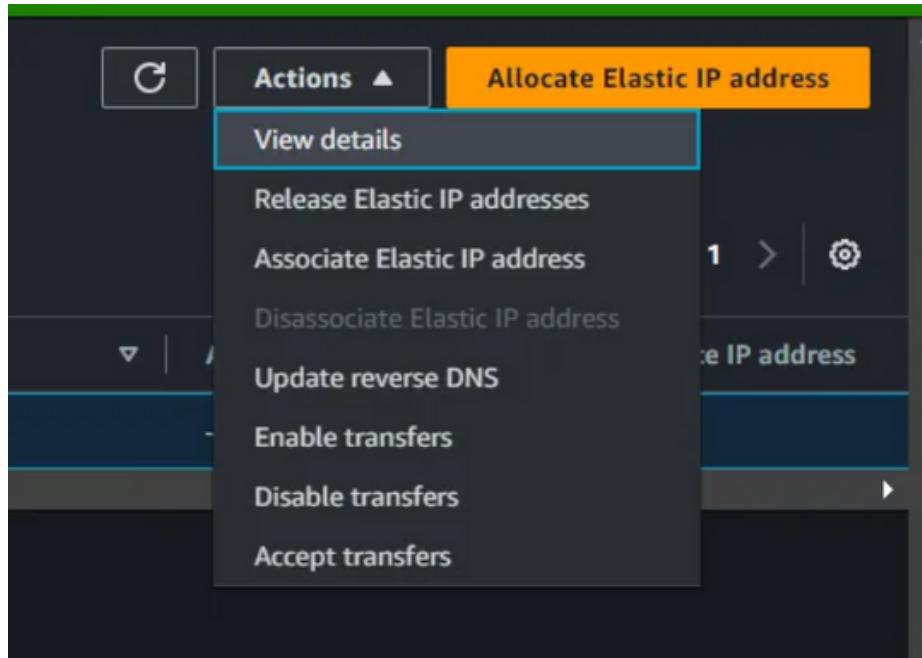
Elastic IP

- Powering Off the EC2 Instance:
- Take note of the current public and private IP addresses of the instance.
- Stop the instance using the AWS Management Console.
- Observing IP Address Changes:
- After stopping the instance, observe that the public IP address is released while the private IP address remains unchanged.
- Refresh the console to verify the changes.



Elastic IP Benefits:

- Elastic IP addresses offer the advantage of maintaining a static public IP address even in scenarios where EC2 instances are restarted or recreated.
- They provide a stable endpoint for applications or services that require a consistent public IP address.
- With Elastic IP, there's no need to update DNS records or reconfigure applications each time an instance is stopped or replaced.
- It simplifies infrastructure management and enhances reliability by ensuring uninterrupted connectivity to resources hosted on EC2 instances.
- Elastic IP addresses are particularly useful for scenarios requiring a fixed public IP address, such as hosting websites, APIs, or VPN gateways.
- Allocating an Elastic IP:
- Allocate an Elastic IP address from the AWS pool.
- Associate the Elastic IP with the desired EC2 instance.



EC2 Instance Components:

- In addition to the EC2 instance itself, other components include the network interface and volumes (virtual hard disks).
- The network interface manages connectivity and is associated with firewall rules, public IP addresses, and security groups.
- Volumes represent the virtual hard disks attached to the instance, storing data and operating system files.
- Understanding these components is essential for managing and configuring EC2 instances effectively.

Network interfaces (1) Info							Last updated less than a minute ago		Actions ▾	Create network interface	
	Name	Network interface ID	Subnet ID	VPC ID	Availability Zone	Security group n....	Security group IDs	Interface Type			
	eni-077ba05a6403fc49	subnet-08d3ae969e060e0c459	vpc-03c70ece7f453fb34	us-east-1c	Web_Server-SG	sg-03826db544c139...	Elastic network interface				

Navigating the EC2 Dashboard:

- Encourage exploration of the EC2 dashboard to view information about resources like key pairs, security groups, and volumes.

Resources		EC2 Global view	Region	CloudWatch Metrics	
You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:					
Instances (running)	1	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	1	Instances	1	Key pairs	1
Load balancers	0	Placement groups	0	Security groups	3
Snapshots	0	Volumes	1		

Adjusting Instance Settings:

- Demonstrate how to change instance type and other settings like Elastic IP disassociation, network interface attachment/detachment, and security group changes.

Reverting Instance Changes:

- Revert any instance changes made during the demonstration, such as changing the instance type, by powering on the instance and reverting the settings back to their original state.

The screenshot shows the AWS Lambda Instances page. At the top, there's a search bar with placeholder text "Find Instance by attribute or tag (case-sensitive)" and a dropdown menu set to "All states". Below the header, a table lists one instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Web_Server	i-059ac545443e64c80	Running	t2.micro	2/2 checks passed	View alarms	us-east-1	amazonaws.com

To the right of the table, a context menu is open under the "Actions" heading. The menu items are:

- Connect
- View details
- Manage instance state
- Instance settings (highlighted with a blue box)
- Networking
- Security
- Image and templates
- Monitor and troubleshoot

Below the table, a section titled "Instance: i-059ac545443e64c80 (Web_Server)" contains tabs for "Details", "Status and alarms", "Monitoring", "Security", "Networking", "Storage", and "Tags". The "Details" tab is selected. Under "Instance summary", it shows the instance ID (i-059ac545443e64c80) and the public IPv4 address (54.225.185.159). It also lists the instance state as "Running", the private IP DNS name (ip-172-31-26-160.ec2.internal), and the instance type (t2.micro). There are also links for modifying instance metadata and tags.

Elastic Block Store (EBS)

EBS serves as the virtual hard disk for EC2 instances, offering both EBS volumes and snapshots for data storage and backup.

EBS Volumes:

- EBS volumes, akin to physical hard disks, house the operating system data of EC2 instances and can store various types of data such as databases, web servers, and files.
- During volume creation, users specify an availability zone, ensuring alignment with the instance's location.
- Data stored in EBS volumes is automatically replicated within the same availability zone, mitigating the risk of data loss due to local failures.

Snapshots:

- Snapshots, used for backing up entire volumes, provide an additional layer of data protection.
- These snapshots capture the state of the EBS volume at a specific point in time, facilitating data recovery and restoration.

Types of EBS Volumes:

- EBS volumes come in various types catering to different workload requirements and performance needs.
- General purpose volumes, SSD-based, are recommended for most workloads due to their balanced pricing and speed.
- Provisioned IOPS volumes offer enhanced performance and are ideal for large databases requiring high input-output operations per second (IOPS).
- Throughput optimized HDD volumes suit big data and data warehousing applications, while cold HDD volumes are cost-effective solutions for infrequently accessed data.
- Magnetic volumes, the cheapest option, are suitable for backups and archives but offer lower performance.

Choosing the Right Volume Type:

- The selection of an appropriate volume type depends on factors such as workload characteristics, performance requirements, and budget considerations.
- AWS documentation provides detailed information and use cases for each volume type, aiding users in making informed decisions.

After establishing a connection to the instance, execute the following commands. Please note that these commands can also be added to the instance creation process in the user data area:

- `systemctl status apache2`
- `cd /tmp`
- `wget https://www.tooplate.com/zip-templates/2119_gymso_fitness.zip`
- `unzip -o 2119_gymso_fitness.zip`
- `cp -r 2119_gymso_fitness/* /var/www/html/`
- `systemctl restart apache2`

These commands are designed to check the status of the Apache web server, navigate to the temporary directory, download a ZIP file containing website templates from a specified URL, unzip the downloaded file, copy the contents to the Apache document root directory, and finally restart the Apache service to apply the changes.

Here, we can observe the available volume types for creating an EC2 instance.

The screenshot shows the 'Configure storage' section of the AWS Lambda function configuration interface. It displays the following details:

- Root volume (Not encrypted):** Set to 1x 8 GiB gp3.
- Volume Type Selection:** A dropdown menu is open, showing "General purpose SSD (gp3)" selected. Other options include "General purpose SSD (gp2)", "Provisioned IOPS SSD (io1)", and "Provisioned IOPS SSD (io2)".
- Volume Type Legend:** A legend indicates that "Free tier eligible" volumes are shown in blue, while "Choose (SSD) or Magnetic storage" volumes are shown in grey.
- Add new volume:** A button to add additional volumes.
- Tags:** A note says "Click refresh to view tags assigned to this function".
- Data Lifecycle Manager:** A note says "The tags that you assign to this function are used by the Data Lifecycle Manager to determine which files are deleted when the function's storage quota is exceeded." A "by any" filter is applied.
- File Systems:** A section showing 0x File systems with an "Edit" button.

Navigate to the Volumes section located on the left-hand side of the EC2 instances page and update the name of the volume associated with the instance mentioned earlier.

The screenshot shows the AWS CloudWatch Metrics Insights search results. At the top, there's a search bar and a 'Create new' button. Below it is a table with columns: Metric Name, Metric Type, Value, and Last Value. One row is highlighted in yellow, showing the metric 'AWS CloudWatch Metrics Insights Metrics/MetricValue' with a value of '1' and a last value of '1'. There are also sections for 'Metrics' and 'Logs' with their respective tables.

Adding extra storage to an EC2 instance, it's crucial to consider the availability zone. Both the instance and the volume need to be in the same zone to ensure connectivity.

When gathering requirements for additional storage, such as for web server images, it's important to adhere to the free tier limit of 30 GB for EBS volumes. For instance, if there's a need for five gigabytes of storage for web server images, it's imperative to ensure that the total storage remains within the free tier limit.

Click on create volume:

Volume settings

Volume type | [Info](#)

General Purpose SSD (gp3)



ⓘ General Purpose SSD gp3 is now the default selection. gp3 provides up to 20% lower cost per GB than gp2.
[Learn More](#) ⓘ

Size (GiB) | [Info](#)

5

Min: 1 GiB, Max: 16384 GiB. The value must be an integer.

IOPS | [Info](#)

3000

Min: 3000 IOPS, Max: 16000 IOPS. The value must be an integer.

Throughput (MiB/s) | [Info](#)

125

Min: 125 MiB, Max: 1000 MiB. Baseline: 125 MiB/s.

Availability Zone | [Info](#)

us-east-1c



Snapshot ID - *optional* | [Info](#)

Don't create volume from a snapshot



Encryption | [Info](#)

Use Amazon EBS encryption as an encryption solution for your EBS resources associated with your EC2 instances.

Encrypt this volume

Tags, you can give upon your wish, then click on Create Volume

Tags - *optional* [Info](#)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Project

Value - *optional*

Sample



[Remove](#)

[Add tag](#)

You can add 49 more tags.

Choose the created volume, navigate to actions, and then opt for attaching the volume once it appears as available on the dashboard.

Select the instance , device name and click on attach volume.

Attach volume Info

Attach a volume to an instance to use it as you would a regular physical hard disk drive.

Basic details

Volume ID

vol-0c72a77fc31471612

Availability Zone

us-east-1c

Instance | [Info](#)

i-059ac545443e64c80 ▾



Only instances in the same Availability Zone as the selected volume are displayed.

Device name | [Info](#)

/dev/sdb ▾

Recommended device names for Linux: /dev/sda1 for root volume. /dev/sd[f-p] for data volumes.

ⓘ Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.

Cancel

Attach volume

(Please note I've selected /dev/xvdf in above screenshot as Device name).

If we run the “fdisk -l” command, we could see the new volume listed:

- *fdisk -l*

List Disks: Start by running the command *fdisk -l* to list all disks. For instance, /dev/xvdf represents the root volume with a size of 8 GB, and it has a partition /dev/xvdaf1.

```
Disk /dev/xvdf: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@ip-172-31-26-160:/tmp#
```

- Understand Partitions: Linux assigns partitioning numbers (e.g., xvda1) rather than names. Verify the root volume by checking its mount point using *df -h*. If /dev/xvda1 is mounted to the root directory, it contains the operating system data.
- Create Partition: Identify the additional disk, such as /dev/xvdf, and use *fdisk* to create a partition. Choose the disk path (e.g., /dev/xvdf), follow the prompts to create a new partition, and write the changes using *w*.

```

Command (m for help): m

Help:

DOS (MBR)
a toggle a bootable flag
b edit nested BSD disklabel
c toggle the dos compatibility flag

Generic
d delete a partition
F list free unpartitioned space
l list known partition types
n add a new partition
p print the partition table
t change a partition type
v verify the partition table
i print information about a partition

Misc
m print this menu
u change display/entry units
x extra functionality (experts only)

Script
I load disk layout from sfdisk script file
O dump disk layout to sfdisk script file

Save & Exit
w write table to disk and exit
q quit without saving changes

Create a new label
g create a new empty GPT partition table
G create a new empty SGI (IRIX) partition table
o create a new empty DOS partition table
s create a new empty Sun partition table

```

```
Command (m for help): n
```

```

Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-10485759, default 10485759):
Created a new partition 1 of type 'Linux' and of size 5 GiB.

Command (m for help): p
Disk /dev/xvdf: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe3548d72

Device      Boot Start      End Sectors Size Id Type
/dev/xvdf1        2048 10485759 10483712    5G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@ip-172-31-26-160:~# 
```

- fdisk -l

```
Device      Boot Start     End Sectors Size Id Type
/dev/xvdf1            2048 10485759 10483712   5G 83 Linux
root@ip-172-31-26-160:~#
```

Format Partition: Use the appropriate formatting utility, such as mkfs.ext4, to format the partition. For example, run mkfs.ext4 /dev/xvdf1 to format the partition with ext4 format.

```
mkfs.cramfs  mkfs.ext2  mkfs.ext3  mkfs.ext4  mkfs.fat  mkfs.minix  mkfs.msdos  mkfs.
```

- mkfs.ext4 /dev/xvdf1

```
root@ip-172-31-26-160:~# mkfs.ext4 /dev/xvdf1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 1310464 4k blocks and 327680 inodes
Filesystem UUID: 18c5893b-0f5d-49f8-bb29-167b8dbff181
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@ip-172-31-26-160:~#
```

Temporary Mount: Mount the partition temporarily using the mount command. For instance, mount /dev/xvdf1 /var/www/html/images.

Verify the mount using **df -h**.

Create a directory: “ mkdir /tmp/image_backups”, and move all data from Images to newly created one

```
root@ip-172-31-26-160:/var/www/html#
root@ip-172-31-26-160:/var/www/html# mv images/* /tmp/image_backups/
root@ip-172-31-26-160:/var/www/html#
```

Then mount the volume and verify as below:

```
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# mount /dev/xvdf1 /var/www/html/images/
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.4G  5.3G  31% /
tmpfs          475M    0  475M   0% /dev/shm
tmpfs          190M  868K  190M   1% /run
tmpfs          5.0M    0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs          95M  4.0K   95M   1% /run/user/1000
/dev/xvdf1       4.9G  24K   4.6G   1% /var/www/html/images
root@ip-172-31-26-160:~#
```

Permanent Mount: Edit the /etc/fstab file to configure permanent mounting. Add an entry with the partition path, mount point, format type, and other options. Save the file and test the configuration using mount -a.

```
LABEL=cloudimg-rootfs   /      ext4  discard,errors=remount-ro      0 1
LABEL=UEFI      /boot/efi    vfat   umask=0077      0 1
/dev/xvdf1     /var/www/html/images  ext4   defaults      0 0
```

Save the file and exit.

```
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# mount -a
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

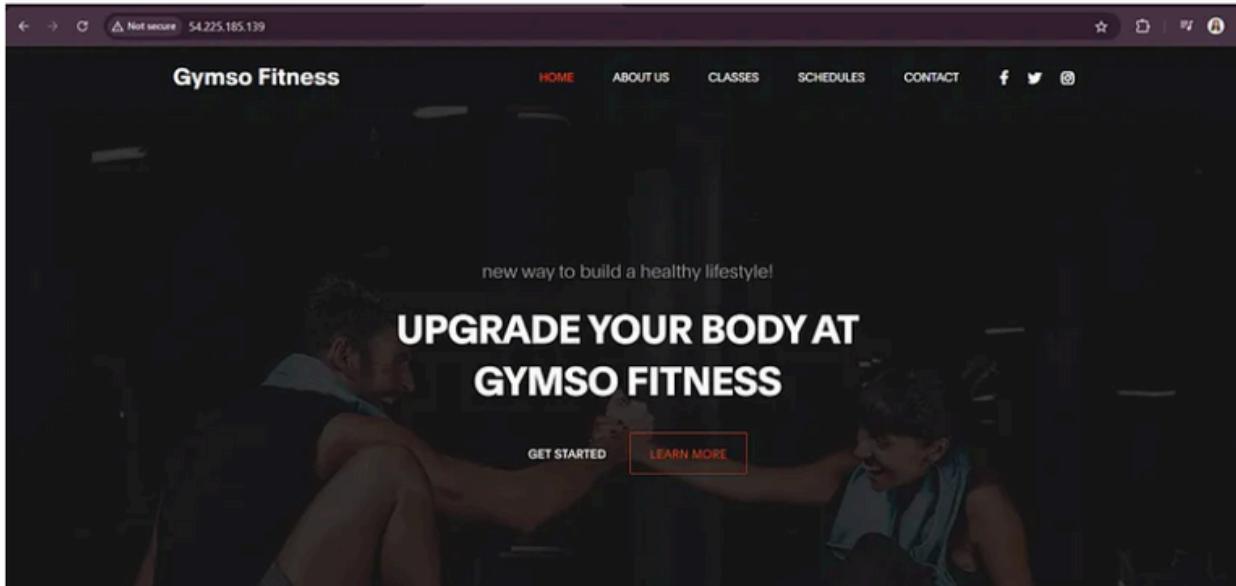
```
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.4G  5.3G  31% /
tmpfs           475M    0  475M   0% /dev/shm
tmpfs           190M  868K  190M   1% /run
tmpfs           5.0M    0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
/dev/xvdf1       4.9G  24K  4.6G   1% /var/www/html/images
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

Move Data: Move data from the backup directory to the mounted partition. For example, use mv command to move data from /tmp/img-backups to /var/www/html/images., and restart the apache2 service

```
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# mv /tmp/image_backups/* /var/www/html/images/
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# systemctl restart apache2
root@ip-172-31-26-160:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-04-15 00:48:20 UTC; 13s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 3770 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 3776 (apache2)
   Tasks: 55 (limit: 1121)
  Memory: 4.8M
    CPU: 28ms
   CGroup: /system.slice/apache2.service
           ├─3776 /usr/sbin/apache2 -k start
           ├─3777 /usr/sbin/apache2 -k start
           └─3778 /usr/sbin/apache2 -k start

Apr 15 00:48:20 ip-172-31-26-160 systemd[1]: Starting The Apache HTTP Server...
Apr 15 00:48:20 ip-172-31-26-160 systemd[1]: Started The Apache HTTP Server.
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

Verify: Check if the data is accessible through the browser. If not, consider disabling SELinux temporarily by modifying the /etc/selinux/config file and rebooting the system. Take the public IP and paste in the browser: you could see the page.



EBS Snapshots

Unmount the volume we created above as below:

```
root@ip-172-31-26-160:~# umount /var/www/html/images
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.4G  5.3G  31% /
tmpfs           475M    0  475M   0% /dev/shm
tmpfs           190M  868K  190M   1% /run
tmpfs            5.0M    0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
root@ip-172-31-26-160:~#
```

Next, navigate to the AWS console, choose the volume that was created, and proceed to detach it by selecting the respective option.

Create a new volume same as shown above and attach it to the same ec2 instance.

Follow the same steps as mentioned above to create a new partition, format , and then create a directory

```
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# mkdir -p /var/lib/mysql
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

Add the new volume to the /etc/fstab.

```
root@ip-172-31-26-160:~# cat /etc/fstab
LABEL=cloudimg-rootfs   /           ext4    discard,errors=remount-ro      0 1
LABEL=UEFI     /boot/efi       vfat    umask=0077      0 1
/dev/xvdbf1   /var/lib/mysql   ext4    defaults      0 0

~
```

Then mount it.

```
root@ip-172-31-26-160:~# mount -a
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        7.6G  2.4G  5.3G  31% /
tmpfs            475M    0  475M   0% /dev/shm
tmpfs            190M  868K  190M   1% /run
tmpfs             5.0M    0  5.0M   0% /run/lock
/dev/xvda15      105M  6.1M   99M   6% /boot/efi
tmpfs            95M  4.0K   95M   1% /run/user/1000
/dev/xvdbf1      2.9G   24K  2.8G   1% /var/lib/mysql
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

Install mariadb-server package with the below steps:

- *apt install mariadb-server*
- *systemctl start mariadb*
- *systemctl enable mariadb*
- *systemctl status mariadb*

List the /var/lib/mysql contents:

- *ls /var/lib/mysql/*

```
root@ip-172-31-26-160:~# ls /var/lib/mysql/
aria_log.00000001  aria_log_control  dd1_recovery.log  debian-10.6.flag  ib_buffer_pool  ib_logfile0  ibdata1  ibtmp1  lost+found  multi-master.info  mysql  mysql_upgrade_info  performance_schema  sys
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
root@ip-172-31-26-160:~#
```

Snapshot functionality primarily serves as a backup and restoration tool, crucial for mitigating data loss in the event of system failures. However, snapshots offer more than just backup capabilities.

In the event of data loss, the initial step involves unmounting the partition to prevent data overwriting. If the goal is to restore the existing partition, snapshots may not be applicable, as they provide a new volume containing the data. In such cases, unmounting the partition and employing recovery tools become essential.

When utilizing snapshots, unmounting the partition is the preliminary action, followed by detaching the corrupted volume. Subsequently, a new volume is created from the snapshot, retaining all data and partition configurations. Upon creation, the new volume is ready for attachment and mounting, effectively replacing the corrupted volume.

In essence, snapshots facilitate data restoration through volume replacement rather than repair, streamlining the recovery process and ensuring data integrity.

Take a Snapshot:

- Navigate to the Volume section in the AWS console.
- Select the relevant volume, then click on “Actions” and choose “Create Snapshot.”
- Provide a meaningful description and add tags for easy identification.
- Click on “Create Snapshot” and wait for it to enter the completed state.

The screenshot shows the AWS Volumes (1/2) page. There are two volumes listed: 'Web-Sererv_Root-Volume' and 'vol-045649e72d6bf20e0'. The second volume has its Actions menu open, with 'Create snapshot' highlighted. Other options in the menu include Modify volume, Delete volume, Attach volume, Detach volume, Force detach volume, Manage auto-enabled I/O, Manage tags, and Fault injection.

The screenshot shows the 'Create snapshot' wizard in the AWS EC2 console. The 'Details' step is active, showing a selected Volume ID (vol-045649e72d6bf20e0) and a description 'db-volume-snapshot'. The 'Encryption' section indicates 'Not encrypted'. The 'Tags' step is visible below, stating 'No tags associated with the resource' and providing an 'Add tag' button. At the bottom are 'Cancel' and 'Create snapshot' buttons.

Snapshots (1) Info										
Owned by me		Search		Actions						
Name	Snapshot ID	Volume size	Description	Storage tier	Snapshot status	Started	Progress	Encryption		
-	snap-0a19584a2ebc0558	100 GB	db-volume-snapshot	Standard	Completed	2024/04/15 07:18 GMT+5:30	Available (100%)	Not encrypted		

Delete Data:

- If you accidentally delete data or encounter corruption, proceed with data deletion.
- Access the directory containing the data you wish to remove, e.g., /var/lib/mysql.
- Use the command `rm -rf *` to remove all files and directories within the specified location.

```
[mysql]#
[mysql]# ls
[mysql]# dd_recovery.log  debian-10.6.flag  ib_buffer_pool  ib_logfile0  ibdata1  ibtmp1  lost+found  multi-master.info  mysql  mysql_upgrade
[mysql]# rm -rf *
[mysql]#
```

Unmount:

- Stop any relevant services to prevent further data overwrites.
- Unmount the affected directory, e.g., /var/lib/mysql.

```
root@ip-172-31-26-160:/var/lib/mysql# rm -rf *
root@ip-172-31-26-160:/var/lib/mysql# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/zroot    7.6G  2.5G  5.1G  33% /
tmpfs          475M   0  475M  0% /dev/shm
tmpfs          190M  880K  189M  1% /run
tmpfs          5.0M   0  5.0M  0% /run/lock
/dev/xvda1S   105M   1M  99M  6% /boot/efi
/dev/xvdb1     2.9G  124M  2.6G  5% /var/lib/mysql
tmpfs          95M   4.0K  95M  1% /run/utmp/1000
root@ip-172-31-26-160:/var/lib/mysql#
root@ip-172-31-26-160:/var/lib/mysql#
root@ip-172-31-26-160:/var/lib/mysql#
root@ip-172-31-26-160:/var/lib/mysql# systemctl stop mariadb
root@ip-172-31-26-160:/var/lib/mysql#
root@ip-172-31-26-160:/var/lib/mysql# systemctl status mariadb
● mariadb.service - MariaDB 10.6.16 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Mon 2024-04-15 01:51:02 UTC; 5s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
Process: 4733 ExecStart=/usr/bin/mariadb ${MYSQLD_OPTS} ${_WSREP_NEW_CLUSTER} ${_WSREP_START_POSITION} (code=exited, status=0/SUCCESS)
Main PID: 4733 (code=exited, status=0/SUCCESS)
   Status: "MariaDB server is down"
      CPU: 567ms
```

```
root@ip-172-31-26-160:~# 
root@ip-172-31-26-160:~# umount /var/lib/mysql/
root@ip-172-31-26-160:~# 
root@ip-172-31-26-160:~# 
root@ip-172-31-26-160:~# 
root@ip-172-31-26-160:~#
```

Detach Corrupted Volume:

- Navigate to the Volumes section and locate the corrupted volume.
- Rename the volume for easy identification, e.g., “corrupted.”
- Select “Actions” and choose “Detach Volume” to disconnect it.

Volumes (1/2) Info											
Actions Create volume 1											
Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created	Availability Zone	Volume state		
Web-Servr_Root-Volume	vol-05060db3eeabd45bf	gp2	8 GB	100	-	snap-057fda6...	2024/04/12 10:35 GMT+5...	us-east-1c	In-use		
Corrupted	vol-045649e72d6bf20e0	gp3	100 GB	3000	125	-	2024/04/15 06:35 GMT+5...	us-east-1c	In-use		

Recover from Snapshot:

- Head to the Snapshots section and select the desired snapshot.
- Click on “Actions” and choose “Create snapshots from volume”.
- Customize volume settings such as type, size, and zone if needed.
- Add appropriate tags for clarity
- Click on “Create Volume” to finalize the recovery process.

Snapshot ID
 snap-0a19584a2ebcb0338

Volume type | [Info](#)
 General Purpose SSD (gp3)

General Purpose SSD gp3 is now the default selection. gp3 provides up to 20% lower cost per GB than gp2.
[Learn More](#)

Size (GiB) | [Info](#)
 100
Min: 1 GiB, Max: 16384 GiB. The value must be an integer.

IOPS | [Info](#)
 3000
Min: 3000 IOPS, Max: 16000 IOPS. The value must be an integer.

Throughput (MiB/s) | [Info](#)
 125
Min: 125 MiB, Max: 1000 MiB. Baseline: 125 MiB/s.

Availability Zone | [Info](#)
 us-east-1a

Fast snapshot restore | [Info](#)
 Not enabled for selected snapshot

Encryption |
Use Amazon EBS encryption as an encryption solution for your EBS resources associated with your EC2 instances.
 Encrypt this volume

In the Volumes console. goto Volumes and rename the new volume as “recovered”.

Volumes (1/3) Info											
Actions Create volume 1											
Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created	Availability Zone	Volume state		
Web-Servr_Root-Volume	vol-05060db3eeabd45bf	gp2	8 GB	100	-	snap-057fda6...	2024/04/12 10:35 GMT+5...	us-east-1c	In-use		
Corrupted	vol-045649e72d6bf20e0	gp3	100 GB	3000	125	-	2024/04/15 06:35 GMT+5...	us-east-1c	Available		
Recovered	vol-012c32476a188b164	gp3	100 GB	3000	125	snap-0a19584...	2024/04/15 07:30 GMT+5...	us-east-1c	Available		

Goto actions and attach the volume

The screenshot shows the 'Attach volume' step in the AWS EC2 console. It displays the 'Basic details' section where a volume ID is selected (vol-012c32476a188b164 (Recovered)) and an instance (i-059ac545443e64c80) is chosen from a dropdown. A note indicates that only instances in the same availability zone will be shown. The 'Device name' field is set to /dev/xvdbf. A warning message in a callout box states: 'Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.' At the bottom right are 'Cancel' and 'Attach volume' buttons.

Below, we examined the disks and found that they were not mounted. Consequently, we executed the mount -a command to rectify the issue. Upon verification, the disks were automatically mounted, allowing us to recover the deleted data by executing ls /var/lib/mysql/.

```
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.5G  5.1G  33% /
tmpfs          475M   0K  475M  0% /dev/shm
tmpfs          190M  872K  189M  1% /run
tmpfs          5.0M   0K  5.0M  0% /run/lock
/dev/xvda15    105M  6.2M  93M  6% /boot/efi
tmpfs          95M  4.0K  95M  1% /run/user/1000
root@ip-172-31-26-160:~# mount -a
root@ip-172-31-26-160:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  2.5G  5.1G  33% /
tmpfs          475M   0K  475M  0% /dev/shm
tmpfs          190M  872K  189M  1% /run
tmpfs          5.0M   0K  5.0M  0% /run/lock
/dev/xvda15    105M  6.2M  93M  6% /boot/efi
tmpfs          95M  1.0K  95M  1% /run/user/1000
/dev/xvdbf     2.0G  1.0M  2.0G  5% /var/lib/mysql
root@ip-172-31-26-160:~# ls /var/lib/mysql/
aria_log_00000001  aria_log_control  dd1_recovery.log  debian-10.6.flag  ib_buffer_pool  ib_logfile0  ibdata1  ibtmp1  lost+found  multi-master.info  mysql  mysql_upgrade_info  performance_schema  sys
```

In the snapshot console, you'll find various options beyond basic backup functionality. These include deleting snapshots, creating volumes from existing snapshots (which we've already covered), and managing fast snapshot restores for large volumes, albeit with additional charges.

Additionally, you can leverage snapshots to create images, known as Amazon Machine Images (AMIs). If the snapshot originates from a root volume, you can directly create an image from it.

Another useful feature is the ability to copy snapshots. By clicking the “copy” button, you can duplicate snapshots to different regions, which is particularly handy for multi-site setups. Moreover, during the copying process, you have the option to enable encryption for added security.

Furthermore, the snapshot interface offers a “modify permissions” button, allowing you to adjust access settings. This enables you to make snapshots public or share them with specific AWS accounts by providing their account IDs.

In summary, snapshots offer a simple yet powerful toolset with various benefits, making them an indispensable asset for managing AWS resources effectively.

Elastic Load Balancer

- Load balancing is a crucial aspect of managing clusters of servers efficiently. It involves distributing incoming network traffic across multiple servers to ensure optimal resource utilization, reliability, and scalability.
- In AWS, Elastic Load Balancer (ELB) provides a scalable and reliable solution for load balancing, allowing users to distribute traffic across multiple targets seamlessly.

- Key Components of ELB:

■Frontend Port and Backend Port:

- The frontend port is the port on which the load balancer listens for incoming traffic. For example, for HTTPS connections, the frontend port would typically be 443.
- The backend port is the port on which the load balancer forwards traffic to the backend servers. It corresponds to the port where the application/service is running on the backend servers. For instance, if a Tomcat server is running on port 8080 on backend instances, the backend port would be set to 8080.

■Types of ELB:

- Classic Load Balancer: Operates at the network layer (Layer 4) of the OSI model. It provides basic load balancing capabilities and distributes incoming traffic to backend instances based on configured rules.
- Application Load Balancer (ALB): Operates at the application layer (Layer 7) of the OSI model. ALB supports advanced routing based on URL paths, hostnames, and other application-level attributes, making it ideal for HTTP and HTTPS traffic.

- Network Load Balancer (NLB): Operates at the transport layer (Layer 4) of the OSI model. NLB is designed to handle extremely high volumes of traffic and offers ultra-low latency, making it suitable for TCP, UDP, and TLS traffic.

■ Load Balancer Targets:

- ELB distributes traffic to backend targets, which can include EC2 instances, containers, IP addresses, or Lambda functions.
- Targets are distributed across multiple availability zones to improve fault tolerance and high availability.

Benefits of ELB:

- Scalability: ELB automatically scales in response to incoming traffic, ensuring that applications remain responsive even during traffic spikes.
- Fault Tolerance: By distributing traffic across multiple targets and availability zones, ELB enhances fault tolerance and resiliency.
- Simplified Management: ELB abstracts away the complexities of managing load balancers, allowing users to focus on building and scaling their applications.

Let's proceed with launching an instance and naming it as "web-health".

You have the flexibility to choose between Amazon Linux, Ubuntu, or CentOS. The script provided in the resource section is compatible with both RPM-based and Debian-based operating systems. Since Amazon Linux is RPM-based, we'll stick with it for this instance.

The screenshot shows the AWS EC2 'Launch an instance' wizard. The top navigation bar shows 'EC2 > Instances > Launch an instance'. The main title is 'Launch an instance' with an 'Info' link. A sub-instruction reads: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.' The first step is 'Name and tags' with an 'Info' link. The 'Name' field contains 'web-health'. To the right is a 'Add additional tags' button. The next step is 'Application and OS Images (Amazon Machine Image)' with an 'Info' link. It includes a search bar and a list of 'Quick Start' AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. On the right, there's a 'Browse more AMIs' button with a note: 'Including AMIs from AWS, Marketplace and the Community'.

Let's opt for a t2.micro instance type and create a new key pair named "web-health-key" for secure access.

The screenshot shows the "Instance type" section of the AWS Lambda configuration interface. It lists the "t2.micro" instance type, which is described as "Free tier eligible". Below the instance type, there is detailed pricing information for various operating systems. A note at the bottom states "Additional costs apply for AMIs with pre-installed software". To the right, there is a button labeled "All generations" and a link to "Compare instance types".

The screenshot shows the "Create key pair" dialog box. In the "Key pair name" field, the value "web-health-key" is entered. A note below says, "The name can include up to 255 ASCII characters. It can't include leading or trailing spaces." Under "Key pair type", the "RSA" option is selected. In the "Private key file format", the ".pem" option is selected. A warning message in a callout box states: "⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) ↗". At the bottom, there are "Cancel" and "Create key pair" buttons.

Now, for the security group, let's create a new one named "web-health-sg", allowing inbound traffic on port 22 for SSH from your IP, and an additional rule for port 80, allowing traffic only from your IP.

Security group name - required

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _~-!/#,@[]+=&{}!\$*

Description - required | [Info](#)

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) [Remove](#)

Type Info	Protocol Info	Port range Info
ssh	TCP	22
Source type Info	Source Info	Description - optional Info
Anywhere	<input type="text" value="Add CIDR, prefix list or security"/>	e.g. SSH for admin desktop
0.0.0.0/0 X		

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) [Remove](#)

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	80
Source type Info	Source Info	Description - optional Info
Anywhere	<input type="text" value="Add CIDR, prefix list or security"/>	e.g. SSH for admin desktop
0.0.0.0/0 X		

[Add security group rule](#)

[EC2](#) > [Security Groups](#) > sg-0f64c5d63b04ab3 - web-health-sg > [Edit inbound rules](#)

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

[Inbound rules](#) [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-0e1bca5d0cb8ed719c	HTTP	TCP	80	Custom	<input type="text" value="103.205.72.75/32"/> X
-	SSH	TOP	22	Anywhere... V	<input type="text" value="0.0.0.0"/> X

[Add rule](#)

Moving forward, let's proceed with the default storage settings and navigate to advanced details. Here, we'll utilize the script in the "User data" section under "Advanced details". Simply copy and paste it as instructed.

Script:

```
#!/bin/bash

# Variable Declaration
#PACKAGE="httpd wget unzip"
#SVC="httpd"
URL='https://www.tooplate.com/zip-templates/2098_health.zip'
ART_NAME='2098_health'
TEMPDIR="/tmp/webfiles"

yum --help &> /dev/null

if [ $? -eq 0 ]
then
# Set Variables for CentOS
PACKAGE="httpd wget unzip"
SVC="httpd"

echo "Running Setup on CentOS"
# Installing Dependencies
echo "#####
echo "Installing packages."
echo "#####
sudo yum install $PACKAGE -y > /dev/null
echo

# Start & Enable Service
echo "#####
echo "Start & Enable HTTPD Service"
echo "#####
sudo systemctl start $SVC
sudo systemctl enable $SVC
echo

# Creating Temp Directory
echo "#####
echo "Starting Artifact Deployment"
echo "#####
mkdir -p $TEMPDIR
cd $TEMPDIR
echo

fi
```

```
wget $URL > /dev/null
unzip $ART_NAME.zip > /dev/null
sudo cp -r $ART_NAME/* /var/www/html/
echo

# Bounce Service
echo "#####
echo "Restarting HTTPD service"
echo "#####
systemctl restart $SVC
echo

# Clean Up
echo "#####
echo "Removing Temporary Files"
echo "#####
rm -rf $TEMPDIR
echo

sudo systemctl status $SVC
ls /var/www/html/

else
# Set Variables for Ubuntu
PACKAGE="apache2 wget unzip"
SVC="apache2"

echo "Running Setup on CentOS"
# Installing Dependencies
echo "#####
echo "Installing packages."
echo "#####
sudo apt update
sudo apt install $PACKAGE -y > /dev/null
echo

# Start & Enable Service
echo "#####
echo "Start & Enable HTTPD Service"
echo "#####
sudo systemctl start $SVC
sudo systemctl enable $SVC
echo
```

```

# Creating Temp Directory
echo "#####
echo "Starting Artifact Deployment"
echo "#####
mkdir -p $TEMPDIR
cd $TEMPDIR
echo

wget $URL > /dev/null
unzip $ART_NAME.zip > /dev/null
sudo cp -r $ART_NAME/* /var/www/html/
echo

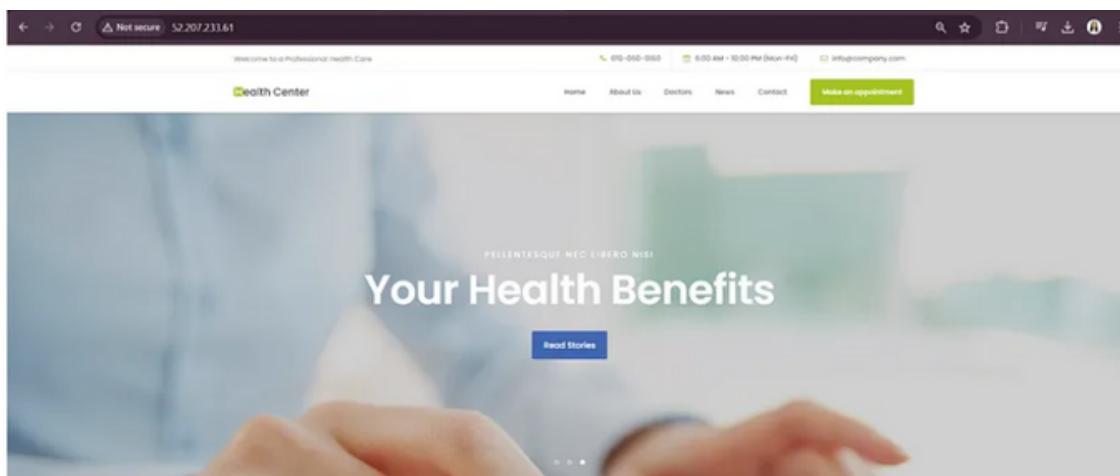
# Bounce Service
echo "#####
echo "Restarting HTTPD service"
echo "#####
systemctl restart $SVC
echo

# Clean Up
echo "#####
echo "Removing Temporary Files"
echo "#####
rm -rf $TEMPDIR
echo

sudo systemctl status $SVC
ls /var/www/html/

```

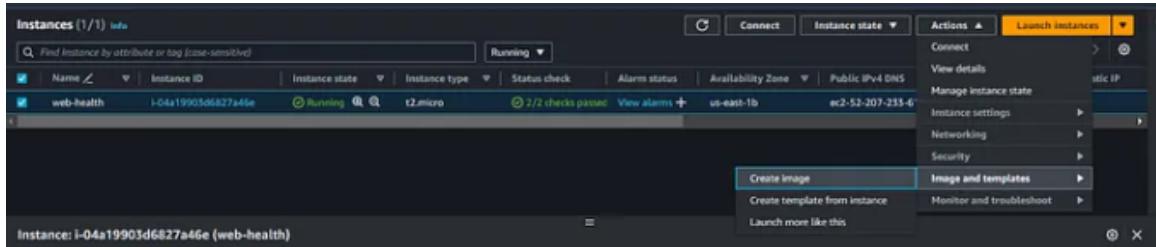
After launching the instance, it typically takes about 5 minutes to initialize. Once it's up, we can check the public IP address and access the website via a browser.



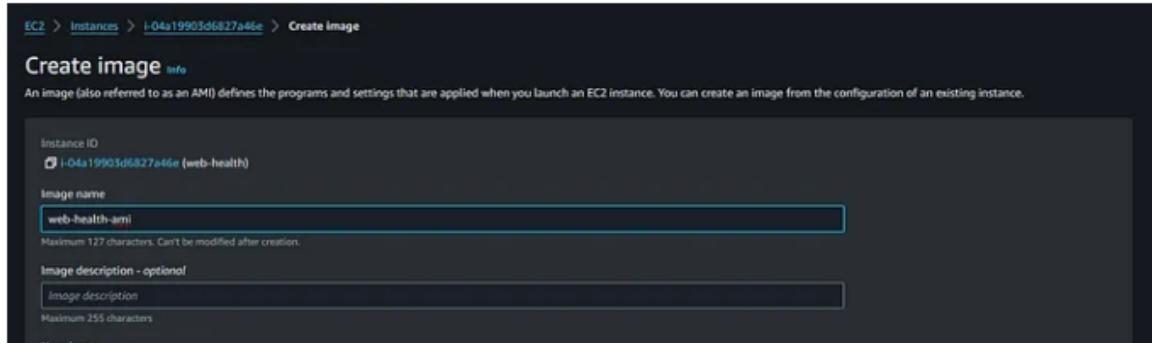
In case you encounter any issues accessing the website, ensure that the security group allows traffic from your IP and that the HTTP service is running on the instance.

Following the successful setup of our website on the EC2 instance, we'll proceed with creating an AMI and a launch template for future deployments.

Creating an AMI involves selecting the instance, navigating to Actions > Image and templates > Create image, and specifying a name for the AMI.

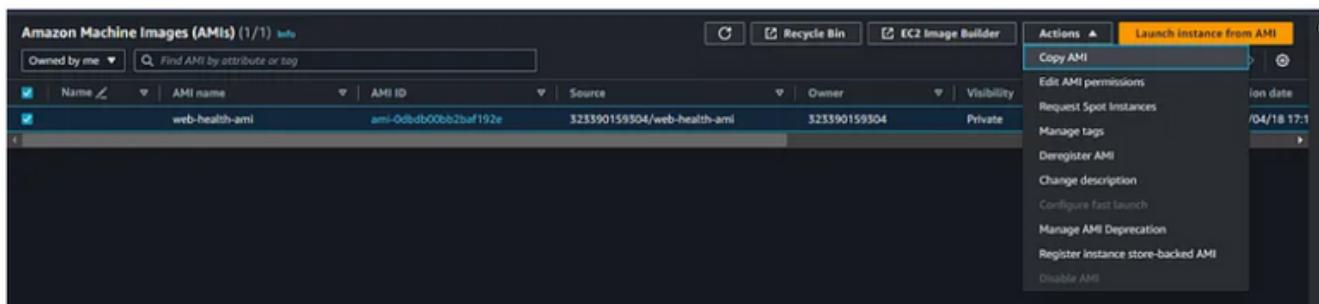


Provide a name for the AMI – Here I gave – “web-health-ami”.



Once the AMI is created, we can leverage it to launch instances with identical configurations, ensuring consistency across our cluster.

Additionally, we can explore options like copying the AMI to other regions or sharing it with other AWS accounts for collaboration.



Also, AWS offers a service known as EC2 Image Builder, allowing you to establish automated pipelines for image creation.

The image pipeline in Image Builder defines all aspects of the process to customize images. It consists of the image recipe, infrastructure configuration, distribution, and test settings.

With the AMI in place, we can streamline instance launches by creating a launch template. Given name as “web-health-template” and version as “V1”

The screenshot shows the 'Create launch template' wizard. At the top, there's a breadcrumb navigation: EC2 > Launch templates > Create launch template. The main title is 'Create launch template'. A descriptive text below it says: 'Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.' The first step is 'Launch template name and description'. It has a 'Launch template name - required' field containing 'web-health-template'. Below it is a note: 'Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.' The next field is 'Template version description' with 'V1' entered. A note below says: 'Max 255 chars'. The third section is 'Auto Scaling guidance' with an 'Info' link. It says: 'Select this if you intend to use this template with EC2 Auto Scaling'. There's a checkbox labeled 'Provide guidance to help me set up a template that I can use with EC2 Auto Scaling'. At the bottom, there are two expandable sections: 'Template tags' and 'Source template'.

Then choose the “My AMIs” and select the AMI we created just before “web-health-ami”

▼ 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.

Search our full catalog including 1000s of application and OS images

Recents My AMIs Quick Start

Don't include in launch template Owned by me Shared with me

Search icon [Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Select the instance type as t2.micro

▼ Instance type		Info Get advice	Advanced
Instance type			
t2.micro		Free tier eligible	<input checked="" type="radio"/> All generations
Family: t2	1 vCPU	1 GiB Memory	Current generation: true
On-Demand Windows base pricing:	0.0162 USD per Hour		
On-Demand SUSE base pricing:	0.0116 USD per Hour		
On-Demand RHEL base pricing:	0.0716 USD per Hour		
On-Demand Linux base pricing:	0.0116 USD per Hour		
▼ Additional costs apply for AMIs with pre-installed software			

Select our key pair we created for the instance “web-health”

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

web-health-key

▼ C Create new key pair

Select the security group as “web-health-sg” we created for our instance “web-health”

Network settings [Info](#)

Subnet [Info](#)

Don't include in launch template [▼](#) [Create new subnet](#)

When you specify a subnet, a network interface is automatically added to your template.

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.

Select existing security group [▼](#) Create security group

Security groups [Info](#)

Select security groups [▼](#)

web-health-sg sg-05e4c36b63b6cab8 [X](#)
VPC: vpc-03c70ece7f453f834

[Compare security group rules](#)

► Advanced network configuration

Just give as below: When launching an instance from this template, I'll modify the name to follow a format such as 2,3,4 etc.

Resource tags [Info](#)

Key [Info](#) Value [Info](#) Resource types [Info](#)

Name [X](#) web-health01 [X](#) Select resource ty... [▼](#) [Remove](#)

[Instances](#) [X](#)

[Add new tag](#)

You can add up to 49 more tags.

Launch Templates (1/1) [Info](#)

Q Search

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0ff650c33280754c2	web-health-template	1	1	2024-04-18T12:02:07.000Z	arn:aws:iam::3233901593

Actions [▼](#) [Create launch template](#)

[Launch instance from template](#) [▼](#)

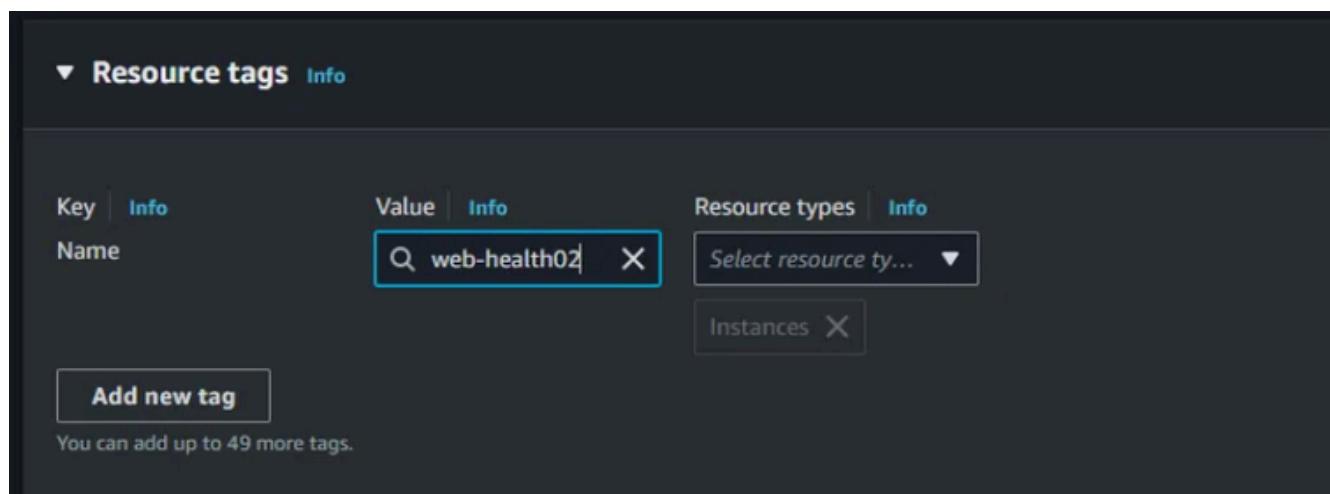
- [Modify template \(Create new version\)](#)
- [Delete template](#)
- [Delete template version](#)
- [Set default version](#)
- [Manage tags](#)
- [Create Spot Fleet](#)
- [Create Auto Scaling group](#)
- [View details](#)

With the AMI in place, we can streamline instance launches by creating a launch template. This template encapsulates all the configuration details, allowing for quick and consistent instance deployments.

By utilizing launch templates, we can expedite the process of launching instances and maintain standardized configurations across our infrastructure.

After configuring our instances, AMI, and launch template, we'll proceed to set up a load balancer to distribute traffic efficiently across our web servers.

Simply navigate to the template section, locate your desired template, and select it. Then, choose the action "launch instance from template." During the launch process, you'll have the opportunity to make adjustments as needed. For example, you can modify the tag to "web-health01 to web-health02 two" before proceeding to launch the instance. That's all there is to it.



Now we have 2 instances which we created from templates and we can create 'n' number of instances upon our business needs.

Instances (2) Info										
Find Instance by attribute or tag (case-sensitive)										
Running										
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP	
web-health02	i-024c082254f908505	Running	t2.micro	Initializing	View alarms +	us-east-1b	ec2-54-254-144-31.co...	54.254.144.31	-	Actions
web-health	i-04a19903d6827a46e	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-52-207-233-61.co...	52.207.233.61	-	Actions

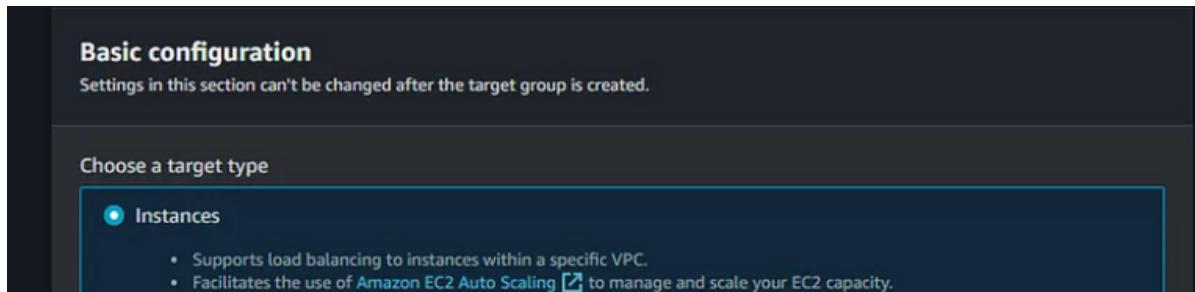
We have successfully created an AMI and established a launch template utilizing this custom AMI.

Presently, we have two web servers operating here. These servers are not individually accessible by users. Instead, users must access them through a unified endpoint, which directs requests to either of these instances. This single endpoint is facilitated by a load balancer.

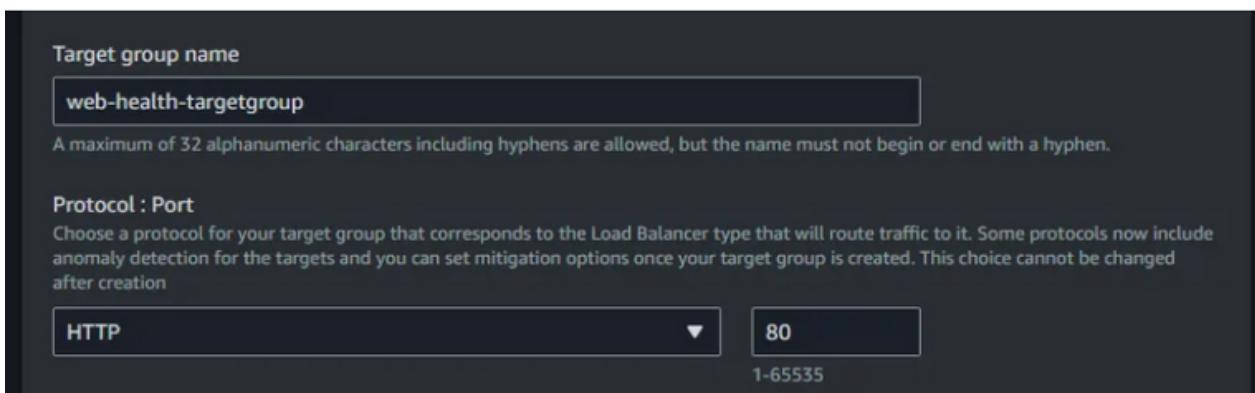
In the Load Balancer section, we will first navigate to target groups. Target groups function as collections of instances, each subject to health checks. Essentially, they are groups comprising instances. Let's proceed to create a target group.

Starting with creating a target group, we'll define health checks and specify the instances to include.

Choose target type as “Instances”



Provide a name as below and Port number 80, protocol HTTP.



In our setup, we've implemented health checks within the target group. These health checks allow the target group to assess the status of each instance. If an instance is deemed unhealthy during these checks, the target group will refrain from routing requests to that specific instance.

Now, you might wonder, how does the target group determine if an instance is healthy or not? It's a similar process to how we, as humans, assess the health of a website. We simply open the website in a browser and check if it's functioning properly. If everything looks good, we consider the website to be healthy.

Similarly, the target group conducts health checks by sending HTTP requests to the instances. When configuring these health checks, we specify the protocol (HTTP) and the health check path (/).

Let's break down what the health check path (/) signifies. Imagine your website's URL is http://IP:80. The slash at the end of this URL represents the root directory of the website. It's where the homepage resides.

Now, if your website has subfolders, such as “videos” or “images,” the health check path would be adjusted accordingly (e.g., /videos or /images).

In essence, by specifying the health check path as “/”, we're instructing the target group to check the root directory of our website. When the target group receives a successful response (HTTP 200 OK) from this path, it considers the instance to be healthy.

The screenshot shows the 'Health checks' configuration page. At the top, it says 'The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.' Below this, the 'Health check protocol' is set to 'HTTP'. The 'Health check path' is set to '/'. A note says 'Up to 1024 characters allowed.' At the bottom, there is a link '► Advanced health check settings'.

In our setup, traffic is expected to flow through the default port, which is port 80. However, if your website operates on a different port, you would need to specify it here.

To do so, you would select the “override” option and provide the appropriate port number, such as 8090. However, since we know that our website operates on port 80, we will leave it as is.

The screenshot shows the 'Advanced health check settings' section. It includes a 'Restore defaults' button. Under 'Health check port', it says 'The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.' There are two options: 'Traffic port' (selected) and 'Override'.

The “Healthy threshold” parameter determines the number of consecutive successful health checks required before declaring the instance as healthy. By default, it is set to five, but you can adjust it to two or ten as per your preference. With a threshold of five, the system checks the instance’s health five times, and if all checks pass successfully, the instance is marked as healthy.

In our case, I will reduce it to two. This means that the instance will be considered healthy if it passes two consecutive health checks. Similarly, the “Unhealthy threshold” specifies the number of consecutive failed health checks required to declare the instance as unhealthy. If an instance returns an unhealthy status twice in a row, it will be marked as unhealthy.

The “Timeout” parameter determines how long the system waits for a response from the website during each health check. If the website does not respond within the specified timeout period (default is 5 seconds), the system moves on to the next check.

Health checks, whether for determining the instance's health or unhealthiness, occur at regular intervals, typically every 30 seconds. This interval can be adjusted within a range of 5 to 300 seconds.

When performing health checks, the system looks for a success exit code, typically HTTP 200, indicating that the website is functioning properly. While we can visually confirm this by checking the website in a browser, the target group relies on these exit codes to determine the instance's health status.

Healthy threshold
The number of consecutive health checks successes required before considering an unhealthy target healthy.

2-10

Unhealthy threshold
The number of consecutive health check failures required before considering a target unhealthy.

2-10

Timeout
The amount of time, in seconds, during which no response means a failed health check.

seconds 2-120

Interval
The approximate amount of time between health checks of an individual target

seconds 5-300

Success codes
The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").

So the below are the instances running and click on the “Include as pending below option”.

Register targets
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (2/2)

Instance ID	Name	State	Security groups	Zone	Private IPv4 address
i-024c082254f908505	web-health02	Running	web-health-sg	us-east-1b	172.31.93.171
i-04a19903d6827a46e	web-health	Running	web-health-sg	us-east-1b	172.31.86.181

2 selected
Ports for the selected instances
Ports for routing traffic to the selected instances.
 1-65535 (separate multiple ports with commas)

Include as pending below

And the targets are in running state.

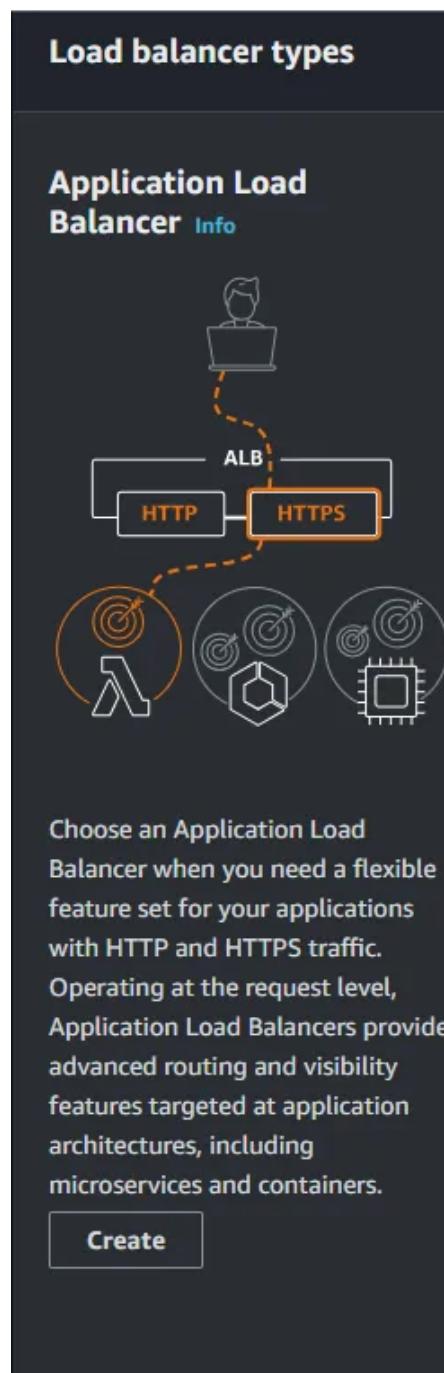
Review targets							
Targets (2)							
<input type="text"/> Filter targets				<input checked="" type="radio"/> Show only pending			
Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet ID
i-024c082254f908505	web-health02	80	Running	web-health-sg	us-east-1b	172.31.93.171	subnet-0c29abf08d0b9642a
i-04a19903d6827a46e	web-health	80	Running	web-health-sg	us-east-1b	172.31.86.181	subnet-0c29abf08d0b9642a

2 pending

Cancel Previous Create target group

Once the target group is set up, we'll proceed to create an application load balancer (ALB), specifying its name, internet-facing nature, and availability zones.

We will go with application load balancer for http, https traffic.



Give a load balancer name “web-health-elb” and We want it to be available on the Internet, so we’ll keep it Internet facing.

The screenshot shows the 'Create Application Load Balancer' wizard in the AWS Management Console. The 'Basic configuration' step is selected. Key fields shown include:

- Load balancer name:** web-health-elb
- Scheme:** Internet-facing (selected)
- IP address type:** IPv4 (selected)

Below these fields, there is descriptive text about the requirements for Internet-facing load balancers.

When setting up the load balancer, it's necessary to specify the availability zones where the load balancer will operate. The system requires a minimum of two zones to ensure redundancy and high availability.

For our configuration, I will choose to select all available zones. This ensures that the load balancer operates across multiple zones, maximizing its availability and fault tolerance. This approach provides a high level of availability at the load balancer level, which is crucial for maintaining continuous service availability.

By selecting all zones, we distribute the load balancing functionality across multiple data centers or regions, reducing the risk of downtime due to failures in any single zone. This enhances the resilience and reliability of our application infrastructure, allowing it to withstand potential disruptions in individual zones.

Network mapping [Info](#)
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

vpc-03c70ece7f453f834
IPv4 VPC CIDR: 172.31.0.0/16

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

- us-east-1a (use1-az1)
- us-east-1b (use1-az2)
- us-east-1c (use1-az4)
- us-east-1d (use1-az6)
- us-east-1e (use1-az3)
- us-east-1f (use1-az5)

Click on the “Create a new security group”

Security groups [Info](#)
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

Security groups
Select up to 5 security groups

default sg-0d57201934000c41e VPC: vpc-03c70ece7f453f834

For the security configuration of our load balancer, we'll create a new security group to define its access permissions. It's important to name this security group appropriately for easy identification and management.

We'll name it “web-health-elb-sg” to indicate that it's the security group for our health project's load balancer. Using descriptive names helps maintain clarity and organization within our AWS environment.

[EC2](#) > [Security Groups](#) > [Create security group](#)

Create security group [Info](#)
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)
web-health-elb-sg
Name cannot be edited after creation.

Description [Info](#)
web-health-elb-sg

VPC [Info](#)
vpc-03c70ece7f453f834

As for the inbound rule, we want to allow access on port 80 from anywhere, making the load balancer accessible to the public. We'll specify the protocol as IPv6 to accommodate modern internet standards, especially considering the increasing adoption of IPv6 by internet service providers, including mobile networks.

Ensuring broad accessibility while maintaining security is crucial for our load balancer's functionality and usability. Making these considerations and adopting best practices, such as proper naming conventions and access control, contributes to the overall robustness of our infrastructure.

The screenshot shows the 'Inbound rules' section of a cloud provider's management console. It lists two rules:

- Rule 1: Type: Custom TCP, Protocol: TCP, Port range: 80, Source: Anywhere..., Description: optional (empty), Delete button.
- Rule 2: Type: Custom TCP, Protocol: TCP, Port range: 80, Source: Anywhere..., Description: optional (empty), Delete button.

An 'Add rule' button is located at the bottom left.

Now select the security group we created just now:

The screenshot shows the 'Security groups' section. A message at the top says: 'A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.' Below is a list of security groups:

- Select up to 5 security groups
- web-health-elb-sg sg-047954c0c46ea9d2f VPC: vpc-03c70ece7f453f834

Ensuring that the ALB's security group allows inbound traffic on port 80 from any source, we'll configure listeners to route traffic to the target group on port 80.

The screenshot shows the 'Listeners and routing' section for an ALB. It displays a single listener configuration:

- Listener: HTTP:80
- Protocol: HTTP, Port: 80, Default action: Forward to web-health-targetgroup (Target type: Instance, IPv4)
- Actions: Remove, Create target group

Below the listener, there are sections for 'Listener tags - optional' (with a note about adding tags for easier management) and 'Add listener' (with a note about adding up to 50 more tags).

Now let's create a load balancer.

The screenshot shows the AWS EC2 Load Balancers console. At the top, there is a header with 'Actions' and 'Create load balancer'. Below the header, a table lists one load balancer: 'web-health-elb'. The table columns include Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created. The 'Availability Zones' column lists 'us-east-1a', 'us-east-1b', and 'us-east-1c'. The 'Type' column is 'application'. The 'Date created' column shows 'April 18, 2024, 17:58 (UTC+05:30)'. Below the table, a detailed view for 'Load balancer: web-health-elb' is shown. It includes sections for 'Details', 'Scheme' (Internet-facing), 'Hosted zone' (Z555XBDOTRQ7X7K), 'VPC' (vpc-03c70dec7f455ff84), 'Availability Zones' (with links to subnet details), 'IP address type' (IPv4), and 'DNS name info' (web-health-elb-587798721.us-east-1.elb.amazonaws.com (A Record)).

Upon viewing the load balancer, you'll notice that it initially enters the provisioning state, transitioning to the active state once fully provisioned. However, despite reaching the active state, attempting to access the website directly from the load balancer may not yield the expected results.

When encountering this issue, accessing the website via the load balancer's DNS name might result in a 504 Gateway Timeout error, indicating a failure to establish a connection between the load balancer and the instances. Despite verifying that the instances are healthy, the presence of this error suggests a potential obstacle preventing traffic from passing between the load balancer and the instances.

To address this challenge, we need to identify and resolve the underlying cause obstructing traffic flow between the load balancer and the instances. This discrepancy could be attributed to misconfigured security groups, routing issues, or other network-related factors.

By diagnosing and rectifying this barrier, we can ensure seamless connectivity between the load balancer and the instances, thereby resolving the 504 Gateway Timeout error and enabling users to access the website reliably through the load balancer.

Each instance is shielded by a security group, a vital layer of defense governing inbound and outbound traffic. Upon inspection, we observed that both instances share the same security group. However, upon reviewing its inbound rules, we discovered that port 80 was only permitted from our IP address, rather than from the load balancer.

To rectify this, we need to adjust the security group settings to allow traffic from the load balancer. But before proceeding, it's prudent to verify the status of the target group. The target group conducts health checks to determine an instance's well-being. If an instance is deemed unhealthy, the target group will cease routing traffic to it. In this scenario, instances were marked as unhealthy due to the inability to access port 80.

Returning to the security group settings, we locate the security group associated with the instances and modify its inbound rules. Specifically, we add a new rule permitting port 80 access from the security group of the load balancer. By specifying the source as the load balancer's security group, we ensure that any member of the load balancer's group can communicate with the instances over port 80.

Additionally, it's beneficial to include a descriptive label for this rule, such as "allow port 80 from ELB," to enhance clarity and documentation. Descriptive labels streamline management and aid in understanding the purpose of each rule, promoting efficient administration and troubleshooting.

Make the security groups inbound rules as below of "web-health".

The screenshot shows the 'Edit inbound rules' page for a security group named 'sg-01e5a2140927a7828'. It lists four rules:

- sg-029f1fd2d9b98a49: Type: HTTP, Protocol: TCP, Port range: 80, Source: Custom, Description: 'allow port 80 from ELB'.
- sg-0e1bda5d0c8a719c: Type: HTTP, Protocol: TCP, Port range: 80, Source: Custom, Description: '103.208.72.75/32'.
- sg-01e5a2140927a7828: Type: SSH, Protocol: TCP, Port range: 22, Source: Custom, Description: '0.0.0.0/0'.

An 'Add rule' button is visible at the bottom left.

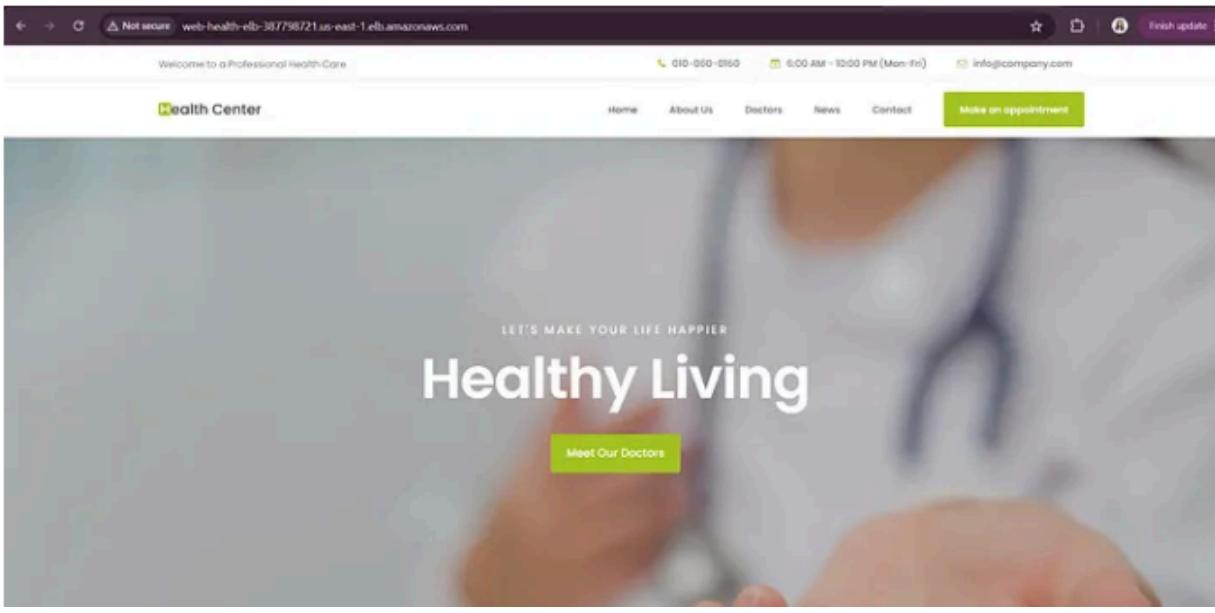
Save the rule.

Make sure that the targets are healthy.

The screenshot shows the 'Targets' tab for a target group named 'web-health-targetgroup'. It displays two registered targets:

Instance ID	Name	Port	Zone	Health status	Last check time	Anomaly detection
i-0240822549080505	web-health02	80	us-east-1b	Healthy	April 18, 2024, 17:33 (UTC+05:30)	Normal
i-04a19905d6827a4fe	web-health	80	us-east-1b	Healthy	April 18, 2024, 17:05 (UTC+05:30)	Normal

Now check it on the browser:



Managing instances within target groups is a critical aspect of optimizing the performance and availability of applications deployed on AWS. Target groups serve as collections of instances that are subject to health checks, ensuring that traffic is routed only to healthy instances. Here are some key points to consider when managing instances within target groups:

1. Health Monitoring: Target groups continuously monitor the health of instances by conducting health checks based on configured parameters. These health checks assess the instance's ability to respond to requests and determine its overall health status.
2. Registration and Deregistration: Instances can be registered or deregistered from target groups dynamically based on operational requirements. This flexibility allows for seamless scaling and maintenance operations without disrupting application availability.
3. Maintenance and Scaling: Registering or deregistering instances from target groups is a common practice during maintenance activities or when scaling the application. By removing unhealthy or underperforming instances from the target group, traffic can be effectively redirected to healthy instances, ensuring consistent application performance.
4. Load Balancing: Target groups play a crucial role in load balancing by distributing incoming traffic across multiple instances based on predefined rules and health check results. This ensures optimal resource utilization and enhances application reliability and scalability.
5. Automation: Managing instances within target groups can be automated using AWS services such as Auto Scaling, which automatically adjusts the number of instances in response to changing demand. This automation helps optimize resource usage and ensures that the application can handle fluctuations in traffic effectively.

Finally, we'll familiarize ourselves with managing instances within the target group, including registering and deregistering instances as needed for maintenance or scaling purposes.

By grasping these concepts and experimenting with the setup, we can gain a comprehensive understanding of deploying and managing web applications on AWS.

AWS CloudWatch

Introduction to CloudWatch:

- AWS CloudWatch is a comprehensive monitoring service offered by Amazon Web Services.
- Originally focused on monitoring, CloudWatch has expanded its capabilities to include logging, events, and more.
- It serves as a central hub for monitoring the performance and health of AWS environments.

Key Features of CloudWatch:

■ Monitoring Service:

- CloudWatch serves as a monitoring solution for AWS resources, tracking performance metrics such as CPU utilization, disk I/O, and network traffic.
- It automatically generates standard metrics for various AWS services used in a region.

■ Logging Solution:

- In addition to monitoring, CloudWatch functions as a logging solution, allowing users to collect, store, and analyze log data generated by AWS resources and applications.
- Logs from services like EC2 instances can be streamed to CloudWatch for centralized log management.

■ Event Monitoring:

- CloudWatch captures real-time events within the AWS environment, such as instance launches, terminations, or volume creations.
- Users can set triggers and notifications based on these events, often integrated with AWS Lambda functions.

■ Standard and Custom Metrics:

- CloudWatch provides both standard and custom metrics for monitoring AWS resources.
- Standard metrics cover common performance indicators like CPU utilization, network traffic, and disk operations.
- Users can define custom metrics tailored to their specific monitoring needs.

■ Alarms and Notifications:

- Users can set alarms on CloudWatch metrics to trigger notifications when predefined thresholds are breached.
- Notifications can be sent via email or integrated with Amazon SNS for broader alerting capabilities.

■ Integration with AWS Services:

- CloudWatch seamlessly integrates with various AWS services, including EC2 instances and EBS volumes.
- Metrics and logs from these services are collected and monitored by CloudWatch, providing insights into resource performance.

Practical Use Cases:

- CloudWatch simplifies monitoring by automatically collecting metrics for AWS resources.
- Users can customize monitoring settings and set up alarms to receive timely notifications of any performance anomalies.
- Practical examples include setting alarms for CPU utilization exceeding a certain threshold, which triggers email notifications via SNS.

If you've already reviewed the previous blog post, we've provided a template for launching EC2 instances. Kindly review it, or proceed to create an EC2 instance.

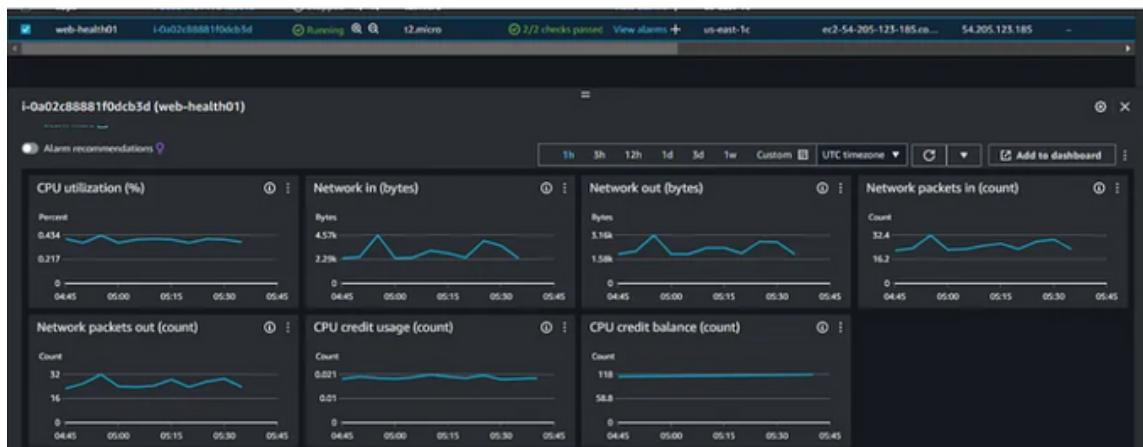
Select the EC2 instance and click on monitoring:

Before proceeding further, take note of the various metric names available, such as CPU utilization, which indicates the percentage of CPU being used over time. Additionally, observe other metrics like status checks, network in and out (in bytes), network packets in and out (count), and disk read operations.

These metrics are automatically generated by CloudWatch when you launch an EC2 instance.

If you require additional metrics, such as RAM or disk utilization, you'll need to create custom metrics. However, for the purpose of this session, we'll focus solely on CPU utilization, as it's one of the most critical metrics to monitor.

By default, CloudWatch checks and updates these metrics every 5 minutes, populating the corresponding graphs. If you prefer more frequent updates, you can enable detailed monitoring, though it's important to note that this option isn't free.



For this hands-on, enabling detailed monitoring is optional.

i-0a02c88881f0dcb3d (web-health01)

Details Status and alarms New Monitoring Security Networking Storage Tags

CloudWatch agent metrics
The monitoring tab will now include metrics related to a single instance in the CWAgent namespace. If you want metrics that are emitted from the CloudWatch agent to be displayed, include them in the CWAgent namespace.

Include metrics in the CWAgent namespace [Learn more](#)

Manage detailed monitoring

1. Select the option to “Enable” detailed monitoring, indicating that CloudWatch should monitor metrics every minute.
2. Keep in mind that detailed monitoring incurs additional charges compared to the default 5-minute monitoring interval.
3. Despite the additional cost, enabling detailed monitoring offers more granular insights into your AWS environment’s performance.

Detailed monitoring

After you enable detailed monitoring for an instance, monitoring data is available in 1-minute periods. [Learn more](#)

Instance ID
 i-0a02c88881f0dcb3d (web-health01)

Detailed monitoring
 Enable

After you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period for the instance.
[Additional charges apply](#)

Cancel Confirm

Log in to the terminal of the EC2 instance and switch as root user.

Then install the stress tool along with its dependencies on your instance.

Stress is a versatile tool designed to test the limits of your Linux operating system, particularly focusing on CPU performance and other system metrics. Simply running the stress command allows you to stress the CPU. You can specify parameters such as the number of CPUs to stress and even utilize functions like the square root function. Additionally, stress can be used to stress other components like IO and RAM, but for our purpose, we’re concentrating on CPU stress testing.

yum install stress -y

```
[root@ip-172-31-29-118 ~]# nohup stress -c 3 -t 100 &
[2] 66003
[root@ip-172-31-29-118 ~]# nohup: ignoring input and appending output to 'nohup.out'
```

```
nohup stress -c 3 -t 100 &
```

This command executes the stress utility in the background using the nohup command, which allows the process to continue running even after the terminal session is terminated. Let's break down the command:

- nohup: Prevents the following command from being terminated when the terminal session ends.
- stress: The stress testing utility.
- -c 3: Specifies that 3 worker threads will be used to stress the CPU cores.
- -t 100: Specifies that the stress test will run for 100 seconds.
- &: Runs the command in the background.

So, this command will run the stress test with 3 worker threads for 100 seconds, and the process will continue running even if the terminal session is closed.

Execute the top command.

```
top
```

You'll notice four stress processes running, indicating the CPU utilization at 100 or close to it, reflected by the load average increment.

Continuously stressing the instance like this will be monitored by CloudWatch, updating the graph every minute.

Repeat this process a few times, running for intervals like 100 seconds and 200 seconds. After a few minutes, you'll observe a pattern forming on the graph.

```
[root@ip-172-31-29-118 ~]# top - 06:50:03 up 16:21,  1 user,  load average: 1.18, 0.30, 0.10
Tasks: 107 total,   4 running, 103 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99.7 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.3 st
MiB Mem :  949.6 total,   538.1 free,   158.2 used,   253.3 buff/cache
MiB Swap:     0.0 total,     0.0 free,     0.0 used.   646.4 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
 65905 root      20   0   3512   116    0 R  33.2  0.0  0:11.25 stress
 65903 root      20   0   3512   116    0 R  32.9  0.0  0:11.24 stress
 65904 root      20   0   3512   116    0 R  32.9  0.0  0:11.24 stress
    1 root      20   0 170708  16540  10084 S  0.0  1.7  0:07.29 systemd
    2 root      20   0     0     0    0 S  0.0  0.0  0:00.01 kthreadd
    3 root      0 -20     0     0    0 I  0.0  0.0  0:00.00 rcu_gp
    4 root      0 -20     0     0    0 I  0.0  0.0  0:00.00 rcu_par_gp
    5 root      0 -20     0     0    0 I  0.0  0.0  0:00.00 slab flushwq
```

Let's create a script for this process.

```
vim stress.sh
```

```
sleep 60 && stress -c 4 -t 60 && sleep 60 && stress -c 4 -t 60 && sleep 60 && stress -c 4 -t 30 && sleep 60 && stress -c 4 -t 100 && sleep 30 && stress -c 4 -t 200
:wq!
```

This sequence of commands progressively varies the duration of the stress tests, interspersed with periods of rest. It's a way to simulate different levels of CPU load on the system over time. designed to stress test the system's CPU using the stress utility. Let's break down each part:

- sleep 60: This command pauses execution for 60 seconds before proceeding to the next command. It introduces a delay in the sequence.
- stress -c 4 -t 60: This command runs the stress utility with the following options:
 - -c 4: Specifies that 4 worker threads will be used to stress the CPU cores.
 - -t 60: Specifies that the stress test will run for 60 seconds.
- sleep 60: Another pause of 60 seconds.
- stress -c 4 -t 60: Similar to the second command, this runs a stress test for another 60 seconds.
- sleep 60: Another pause of 60 seconds.
- stress -c 4 -t 30: This command runs a shorter stress test for 30 seconds.
- sleep 60: Another pause of 60 seconds.
- stress -c 4 -t 100: This command runs a longer stress test for 100 seconds.
- sleep 30: A shorter pause of 30 seconds.
- stress -c 4 -t 200: This command runs an even longer stress test for 200 seconds.

Execute this script:

```
./stress.sh
```

```
[root@ip-172-31-29-118 ~]# nohup ./stress.sh &
[1] 66549
[root@ip-172-31-29-118 ~]# nohup: ignoring input and appending output to 'nohup.out'
```

We generate a graph by running the command. I simply executed it using nohup stress.sh &.

Ensure your script is executable.

```
chmod +x stress.sh
```

Now, when observing the top output, you may intermittently notice the stress command based on the ongoing operations within your script.



Now, let's navigate to the CloudWatch service to configure an alarm for CPU utilization on this instance.

First, head over to the “All alarms” section.

Next, click on the “Create Alarm” button.

To select the CPU utilization metric, start by navigating to the EC2 service.

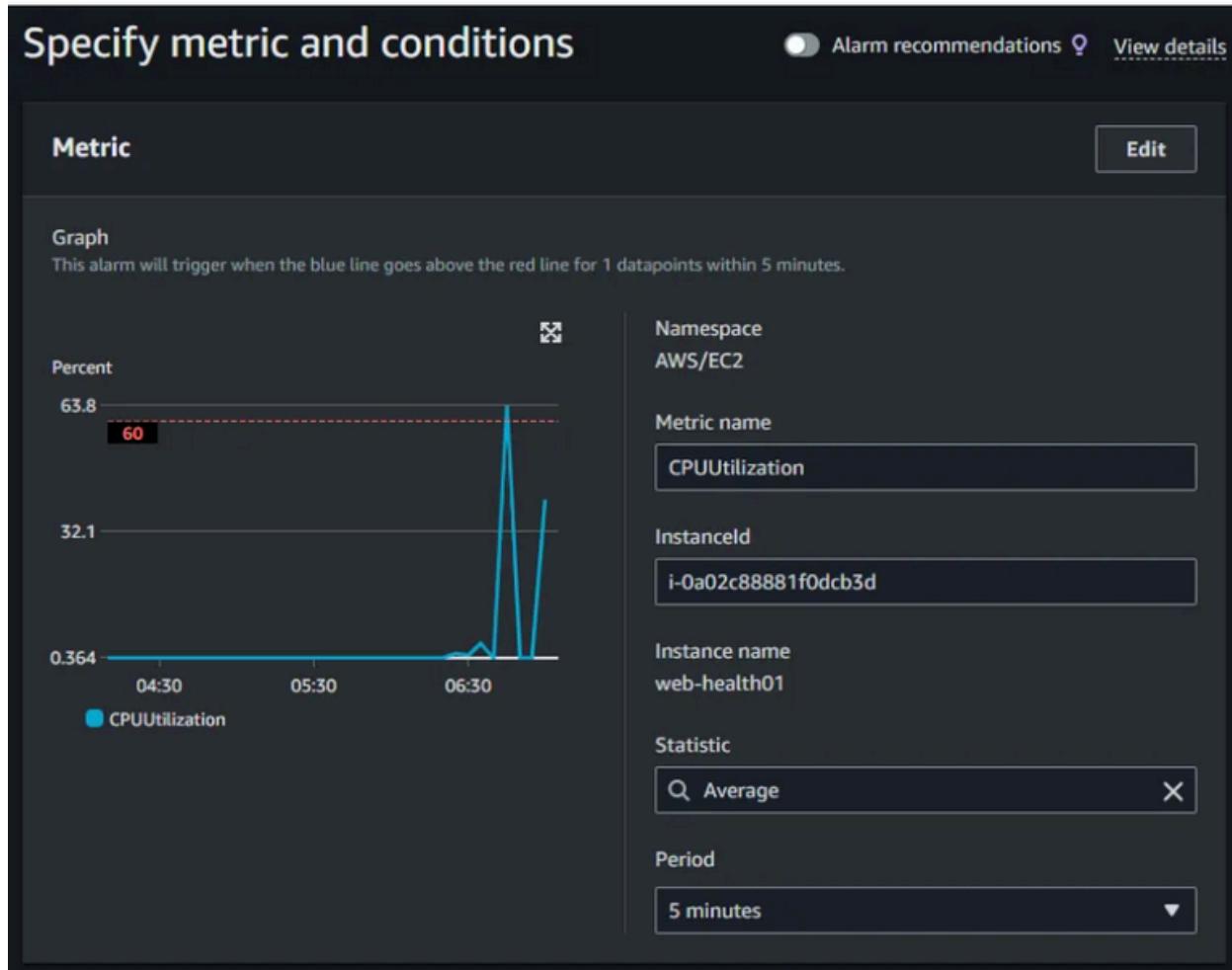
Then, locate the “Per-instance metrics” section and find your instance. If you don’t immediately see your instance, wait for a few moments for the information to load.

Once you’ve found your instance, look for the CPU utilization metric.

Select the CPU utilization metric by clicking on it.

In this step, we can choose the period for which the alarm will be evaluated. For this demonstration, I'll keep the period set to 5 minutes.

Next, we specify the condition for triggering the alarm based on CPU utilization.



For example, we can set the condition to trigger the alarm if the CPU utilization is greater than or equal to 60 for a period of 5 minutes.

Once we've defined the condition, we proceed to the next step.

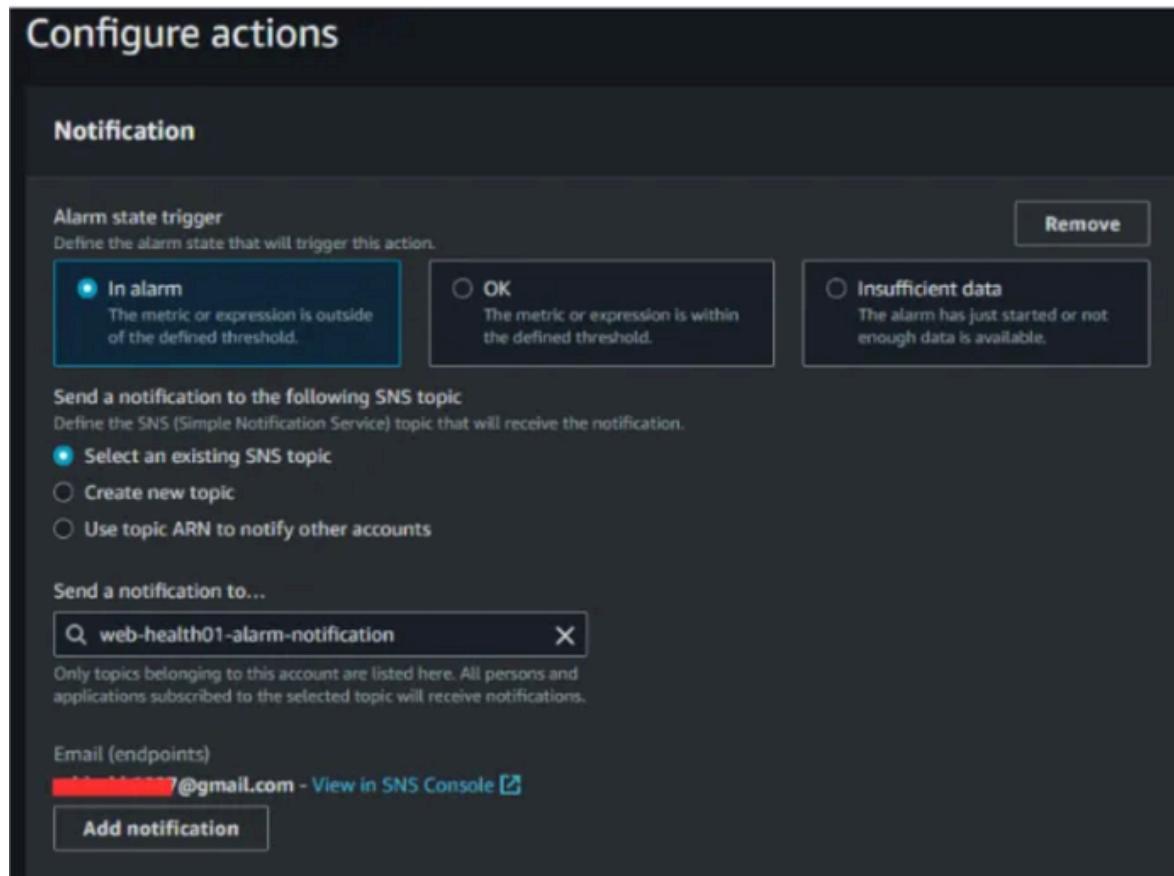
The screenshot shows the 'Conditions' step of the alarm creation wizard. It includes the following fields:

- Threshold type:** Static (selected)
- Whenever CPUUtilization is...:** Greater/Equal (\geq) threshold (selected)
- than...:** 60 (threshold value)
- Additional configuration:** A link at the bottom left.

If you don't find your desired topic listed, you can click on "Create Topic." Provide a name for the topic, enter your email address, and click on "Create Topic."

However, since we already created a topic for billing alarms earlier, I'll select the same topic, which contains my email address.

This topic is designated for notifications. So, when the alarm state is triggered, it will send a notification to this topic, ultimately resulting in an email notification being sent to me.



There are several other actions you can take, such as EC2 actions. For instance, you might want to stop, terminate, or reboot the instance if it triggers an alarm.

In some cases, a high CPU utilization might prevent you from logging into the instance via SSH. Rebooting the instance could be a temporary solution to address this issue.

However, for the purpose of this demonstration, we will skip these additional actions. We will stick to email notifications only and proceed to the next step.

EC2 action

Alarm state trigger

Define the alarm state that will trigger this action.

[Remove](#)

In alarm

The metric or expression is outside of the defined threshold.

OK

The metric or expression is within the defined threshold.

Insufficient data

The alarm has just started or not enough data is available.

Take the following action...

Define what will happen to the EC2 instance with the Instance ID i-0a02c88881f0dc83d when this alarm is triggered.

Recover this instance

You can only recover certain EC2 instance types. [See documentation](#)

Stop this instance

You can only stop an instance if it is backed by an EBS volume. AWS will use the existing Service Linked Role (AWSLambdaRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

Terminate this instance

You will not be able to terminate this instance if termination protection is enabled. AWS will use the existing Service Linked Role (AWSLambdaRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

Reboot this instance

An instance reboot is equivalent to an operating system reboot. AWS will use the existing Service Linked Role (AWSLambdaRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

[Add EC2 action](#)

Let's give the alarm a descriptive name.

The naming convention will be: "warning-web-health-alarm-notification-cpu" for the specific instance.

In our organization, a warning is triggered when the utilization exceeds 60%. We could set up additional alarms for critical situations, such as when it surpasses 80%. However, for now, we'll proceed with just one alarm.

Now, let's move on to the next step.

Add name and description

Name and description

Alarm name

warning-web-health-alarm-notification-cpu

Alarm description - optional [View formatting guidelines](#)

[Edit](#)

[Preview](#)

warning-web-health-alarm-notification-cpu

Up to 1024 characters (41/1024)

 Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.

[Cancel](#)

[Previous](#)

[Next](#)

After a while, CloudWatch will collect the data and display whether the instance is in an alarm state or if it's okay.

Now, this process aligns with the basic principles of any monitoring tool. You have metrics or checks, alarms triggered by those metrics or checks, and actions associated with those alarms, such as sending email notifications.

Whether it's Prometheus, Nagios, Icinga, Zenos, or others, the concept remains similar. However, with CloudWatch, monitoring is already configured, and you only need to set up alarms. In contrast, with other tools, such as those you set up yourself, the entire monitoring system needs configuration, typically by the monitoring or administration team.

The screenshot shows the CloudWatch Alarms interface. At the top, there are several filters: 'Hide Auto Scaling alarms', 'Clear selection', 'Create composite alarm', 'Actions', and a prominent orange 'Create alarm' button. Below the filters is a search bar and dropdown menus for 'Alarm state: Any', 'Alarm type: Any', and 'Actions status: Any'. A navigation bar with icons for back, forward, and refresh is also present. The main table lists one alarm: 'warning-web-health-alarm-notification-cpu'. It shows the alarm is 'In alarm' (indicated by a red triangle icon), last updated on '2024-04-29 07:16:13', and has the condition 'CPUUtilization >= 60 for 1 datapoints within 5 minutes'. The 'Actions' column shows 'Actions enabled' with a green circular icon. The table has columns for Name, State, Last state update, Conditions, and Actions.

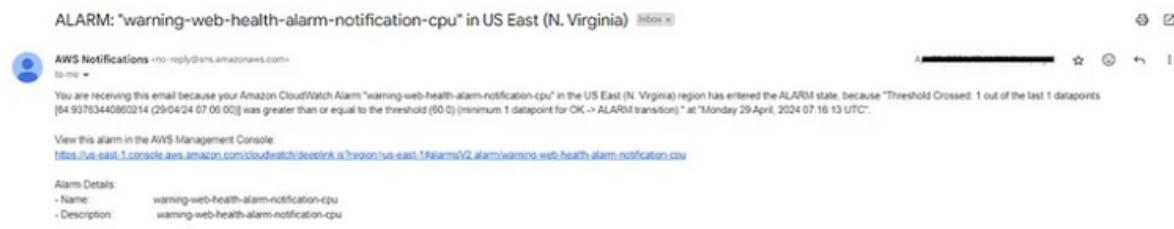
Currently, the graph fluctuates because I've executed the stress command multiple times. Remember, you need to run the stress command for at least 5 minutes or longer for it to exceed the threshold and trigger the email notification.

After running the command for 5 minutes, wait for the notification to arrive in your inbox.

Moreover, you can create a reverse alarm to monitor the instance's OK state. For instance, if the CPU utilization is below 40, it's considered OK. You can set up alarms for different thresholds and states, and you have the flexibility to create up to ten alarms under the free tier.



Email notification:



I encourage you to experiment with these alarms to familiarize yourself with their functionalities. Set alarms for various data points, both in alarm and OK states, using different conditions like greater than or less than. Once you're comfortable, remember to terminate the instance and delete the alarms before moving on to the next section, particularly the auto-scaling group section.

Auto Scaling Groups

Auto Scaling Groups (ASG) seamlessly integrates with CloudWatch to monitor specific metrics, such as CPU utilization. If a metric, like CPU utilization, exceeds a predefined threshold, ASG adjusts capacity by either adding or removing instances in the group. This adjustment is based on CloudWatch alarms, which trigger actions to maintain performance or control costs.

When creating an auto scaling group, you'll configure launch configurations or launch templates, similar to those used in EC2 instances for ELB exercises. These configurations provide the necessary information for launching instances within the ASG.

To dynamically adjust capacity based on metrics, ASG employs scaling policies. You can define these policies, such as step scaling, to add or remove instances based on workload fluctuations. For instance, you could specify that if CPU usage exceeds 60%, launch two additional instances, or if it exceeds 80%, launch four instances.

When setting up an auto scaling group, you specify parameters like minimum, desired, and maximum capacity. The minimum size ensures a baseline level of instances that cannot be terminated, while desired capacity represents the typical number of running instances. The maximum size limits the total number of instances the group can scale up to, helping to control costs.

In practice, ASG operates by creating a group powered by launch configurations. When a metric breach triggers a CloudWatch alarm, scaling policies kick in to launch or terminate instances accordingly, ensuring optimal performance and resource utilization.

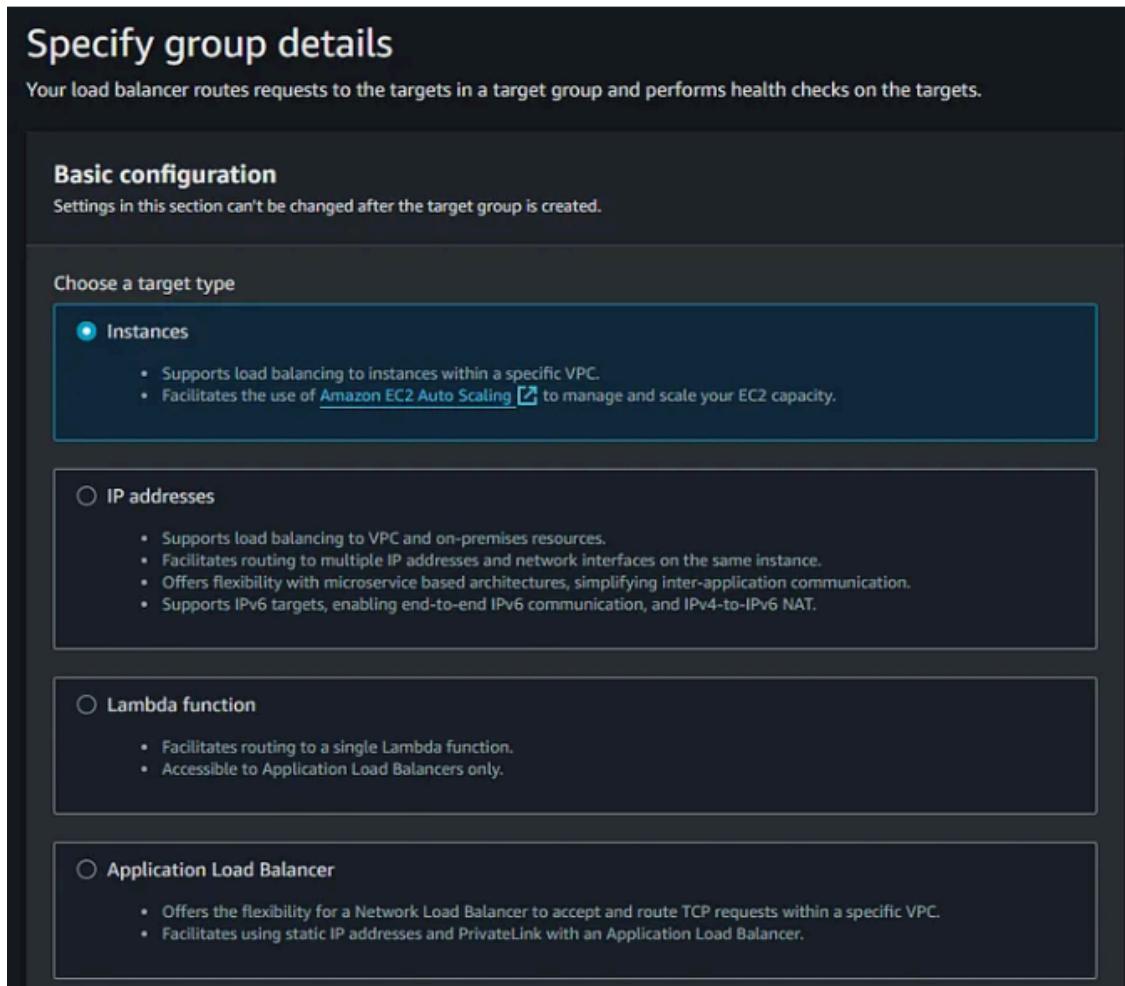
Now, let's dive into the practical implementation of these concepts.

Before diving into the auto scaling group setup, let's ensure we have all the necessary components in place.

Firstly, we'll need the launch template, which should already be created as per the instructions provided in the load balancer section. If the launch template has been removed or needs to be recreated, refer back to the load balancer section for detailed steps on how to create it.

Now, let's proceed to the Target Group setup.

- Navigate to the Target Group section.
- Click on the “Create target group” button.



Configure the target group settings, such as name. For example, let's name it “web-health-tg”.

- Keep the other configurations as default and proceed by clicking “Next”.
- Since we don't have any instances yet, select the option to create an empty target group.
- Complete the target group creation process by following the prompts.

Target group name
web-health-tg
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol : Port
Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP	80
1-65535	

IP address type
Only targets with the indicated IP address type can be registered to this target group.

- IPv4**
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
- IPv6**
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

Indeed, the Auto Scaling group will automatically add instances to this target group as needed based on your scaling policies and the metrics you've configured, ensuring seamless scalability and efficient resource management.

EC2 > Target groups						
Target groups (1) Info						
Actions Create target group						
Filter target groups						
	Name	ARN	Port	Protocol	Target type	Load balancer
	web-health-tg	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/web-health-tg/123456789012345678	80	HTTP	Instance	web-health-elb
						VPC ID
						vpc-0ba6c8b55427a214

We'll proceed by creating a load balancer for this target group. Let's select "Create load balancer" and opt for the application load balancer, maintaining consistency with our previous setup.

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer [Info](#)

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
web-health-elb
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme [Info](#)
Scheme can't be changed after the load balancer is created.

- Internet-facing**
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)
- Internal**
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type [Info](#)
Select the type of IP addresses that your subnets use.

- IPv4**
Includes only IPv4 addresses.
- Dualstack**
Includes IPv4 and IPv6 addresses.

We'll name it "health-elb" and choose the same availability zones and security groups used previously.

Network mapping [Info](#)
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

vpc-08a6c8b533427a214
IPv4 VPC CIDR: 172.31.0.0/16

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

ap-south-1a (aps1-az1)
Subnet: subnet-0b65d3e364451e724

IPv4 address: Assigned by AWS

ap-south-1b (aps1-az3)
Subnet: subnet-0be850d7e8202ce91

IPv4 address: Assigned by AWS

ap-south-1c (aps1-az2)
Subnet: subnet-0b53b17ead7f32d3b

IPv4 address: Assigned by AWS

Remember to remove the Default Security Group. Next, we'll configure it to forward traffic to the target group we just created.

Security groups [Info](#)
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

Security groups
Select up to 5 security groups

web-health-sg
sg-093b6aaaa6444d421 VPC vpc-08a6c8b533427a214

Listeners and routing [Info](#)
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80 [Remove](#)

Protocol	Port	Default action
HTTP	80	Forward to web-health-tg Target type: Instance, IPv4

[Create target group](#)

Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)
You can add up to 50 more tags.

[Add listener](#)

If you already have these resources set up, you can reuse them; otherwise, go ahead and create them now. Here's our load balancer.

The screenshot shows the AWS EC2 Load Balancers console. At the top, there is a header with the title 'Load balancers (1)'. Below the header, a table lists one item: 'web-health-elb'. The table includes columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created. The 'web-health-elb' row has a 'DNS name' of 'web-health-elb-14754805...', a 'State' of 'Provisioning...', 'VPC ID' of 'vpc-08a6c8b533427a...', '3 Availability Zones', 'Type' of 'application', and a 'Date created' of 'May 7, 2024, 01:24 (UTC+05:30)'.

Let's revisit the launch template. This template is essential as it defines the configuration for the instances that will be launched by the auto scaling group.

It's important to verify that the required AMI is available and accessible. If you've already prepared the AMI, it should be readily available for selection within the launch template settings.

The screenshot shows the AWS Amazon Machine Images (AMIs) console. At the top, there is a header with the title 'Amazon Machine Images (AMIs) (1)'. Below the header, a table lists one item: 'web-health-ami'. The table includes columns for Name, AMI name, AMI ID, Source, Owner, Visibility, and Status. The 'web-health-ami' row has a 'Name' of 'web-health-ami', an 'AMI name' of 'ami-0bf84427ca798310', a 'Source' of '323390159304/web-health-ami', an 'Owner' of '323390159304', a 'Visibility' of 'Private', and a 'Status' of 'Available'.

The launch template specifies details such as the Amazon Machine Image (AMI) to be used for the instances.

The screenshot shows the AWS Launch Templates console. At the top, there is a header with the title 'Launch Templates (1)'. Below the header, a table lists one item: 'web-health-template'. The table includes columns for Launch Template ID, Launch Template Name, Default Version, Latest Version, Create Time, and Created By. The 'web-health-template' row has a 'Launch Template ID' of 'lt-095543840386f1b9', a 'Launch Template Name' of 'web-health-template', a 'Default Version' of '1', a 'Latest Version' of '1', a 'Create Time' of '2024-05-06T20:03:14.000Z', and a 'Created By' of 'arn:aws:iam::323390159304:root'.

Now, let's navigate to the auto scaling groups section.

Let's initiate the creation process by assigning a name to our auto scaling group. For simplicity, let's name it "web-health-asg".

The screenshot shows the 'Choose launch template or configuration' step in the Auto Scaling group creation wizard. The title is 'Choose launch template or configuration' with a 'Info' link. Below the title, a note says: 'Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.' A large input field is labeled 'Name' and contains the value 'weh-health-asg'. Below the input field, a note says: 'Must be unique to this account in the current Region and no more than 255 characters.'

Next, we'll select our launch template for the auto scaling group.

You might also notice the option to use a launch configuration, which serves a similar purpose to a launch template. However, AWS now recommends using launch templates due to their added flexibility.

One notable advantage of launch templates is their editability. Unlike launch configurations, which cannot be modified once created, launch templates allow for easy edits and the creation of multiple versions. This versatility enables you to tailor your configurations to different environments or instance types seamlessly.

As we proceed, we'll opt for the launch template, leveraging its benefits for our auto scaling group. You'll notice the option to select different versions of the launch template, providing flexibility in configuration management.

The screenshot shows the 'Launch template' configuration page. At the top, there's a 'Launch template Info' section with a dropdown menu set to 'web-health-template' and a 'Switch to launch configuration' link. Below this, there's a 'Launch template' section with a description: 'Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.' A dropdown menu shows 'web-health-template' with a 'C' button next to it. There's also a 'Create a launch template' link. The 'Version' section shows 'Default (1)' selected with a 'C' button. A 'Create a launch template version' link is available. The main configuration area has three columns:

Description	Launch template	Instance type
V1	web-health-template lt-0c93343840386f1b9	t2.micro
AMI ID	Security groups	Request Spot Instances
ami-0bfd84427ca798310	-	No
Key pair name	Security group IDs	
web-health-key	sg-093b6eaaa6444d421	

At the bottom, there's an 'Additional details' section with 'Storage (volumes)' and 'Date created' (Tue May 07 2024 01:33:14 GMT+0530 (India Standard Time)). Finally, there are 'Cancel' and 'Next' buttons at the bottom right.

Choose instance launch options:

In this step, we need to specify the availability zones where we want our instances to be launched.

You have the option to select specific zones or allow AWS to choose based on availability. It's generally recommended to choose at least two zones for redundancy and fault tolerance.

Let's proceed by selecting all zones and moving to the next step.

Network [Info](#)

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-08a6c8b533427a214
172.31.0.0/16 Default

Create a VPC [\[\]](#)

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

ap-south-1a | subnet-0b65d3e364451e724 X
172.31.32.0/20 Default

ap-south-1b | subnet-0be850d7e8202ce91 X
172.31.0.0/20 Default

Create a subnet [\[\]](#)

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

Configure advanced options

In this step, we have the option to attach our auto scaling group to an existing load balancer.

Since our website will benefit from a load balancer, especially with multiple instances, we'll choose to attach it to an existing load balancer.

Configure advanced options - *optional* [Info](#)

Choose a load balancer to distribute incoming traffic for your application across instances to make it more reliable and easily scalable. You can also set options that give you more control over health check replacements and monitoring.

Load balancing [Info](#)

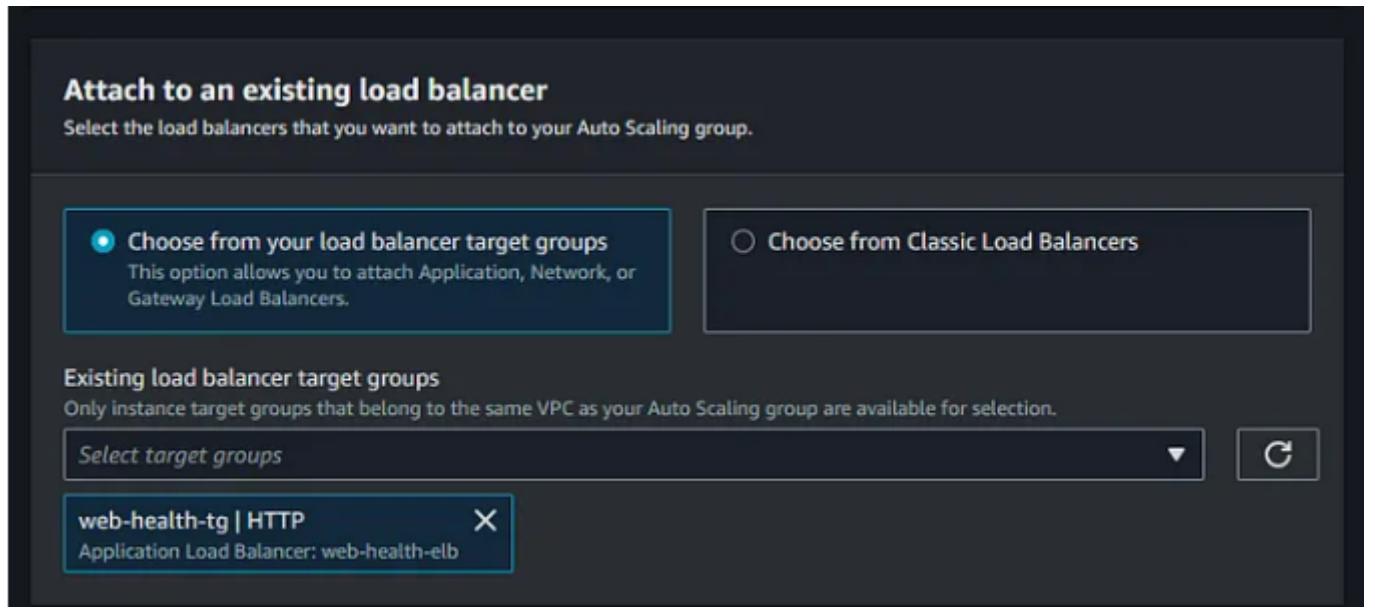
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
Choose from your existing load balancers.

Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

We'll then select the target group that we previously created. This ensures that the instances launched by our auto scaling group are registered with the specified target group, allowing the load balancer to distribute traffic effectively across these instances.



In this step, we configure the health check settings for our instances within the auto scaling group.

The primary responsibility of the auto scaling group is to maintain the desired number of healthy instances, ensuring optimal performance and reliability.

By default, the auto scaling group performs basic EC2 health checks, which include hardware and virtual machine checks. However, these checks only verify the instance's basic functionality and do not guarantee the health of your application or website.

To enable more comprehensive health checks, we opt for ELB (Elastic Load Balancer) health checks. This leverages the target group health check functionality discussed in the load balancer section.

The target group conducts health checks every 300 seconds, verifying the health of instances based on specified criteria such as port availability and process responsiveness. If an instance fails this health check, indicating a potential issue with the hosted application or service, the target group declares it as unhealthy.

Upon detecting an unhealthy instance, the auto scaling group takes corrective action by terminating the problematic instance and launching a replacement. This ensures that the overall system remains resilient and capable of handling fluctuations in workload and application health effectively.

Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

EC2 health checks

 Always enabled

Additional health check types - optional | [Info](#)

Turn on Elastic Load Balancing health checks Recommended

Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

 EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the [Load Balancer console](#) 

Health check grace period | [Info](#)

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

300 seconds

Configure group size and scaling

Now, let's proceed to the next step.

Here, we specify the capacity settings for our auto scaling group. We define the desired, minimum, and maximum number of instances that the group should maintain.

- Desired capacity: This indicates the number of instances we want to have in our auto scaling group. For example, setting it to 2 means we aim to have two instances running at all times.
- Minimum capacity: This represents the minimum number of instances that must be running in the auto scaling group, regardless of the workload or scaling policies. In our case, we set it to 1 to ensure there is always at least one instance available.
- Maximum capacity: This defines the upper limit on the number of instances that can be launched by the auto scaling group. Even if the workload demands more instances, the group will not exceed this limit. Here, we set it to 8 to cap the maximum number of instances at eight.

Configure group size and scaling - optional [Info](#)

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

Group size [Info](#)

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

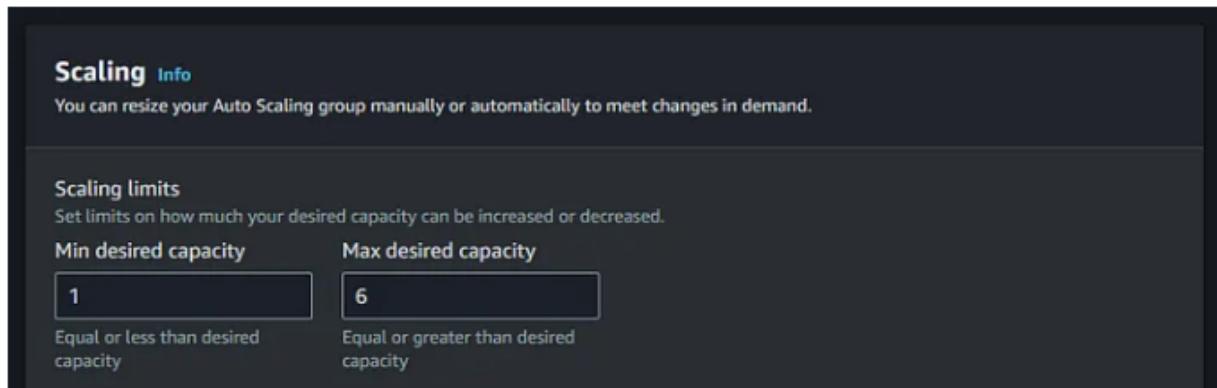
Units (number of instances) 

Desired capacity

Specify your group size.

2

Understanding the distinction between desired, minimum, and maximum capacities is crucial. While desired capacity determines the ideal number of instances, minimum capacity ensures a baseline level of availability, and maximum capacity prevents excessive scaling beyond a predetermined threshold.

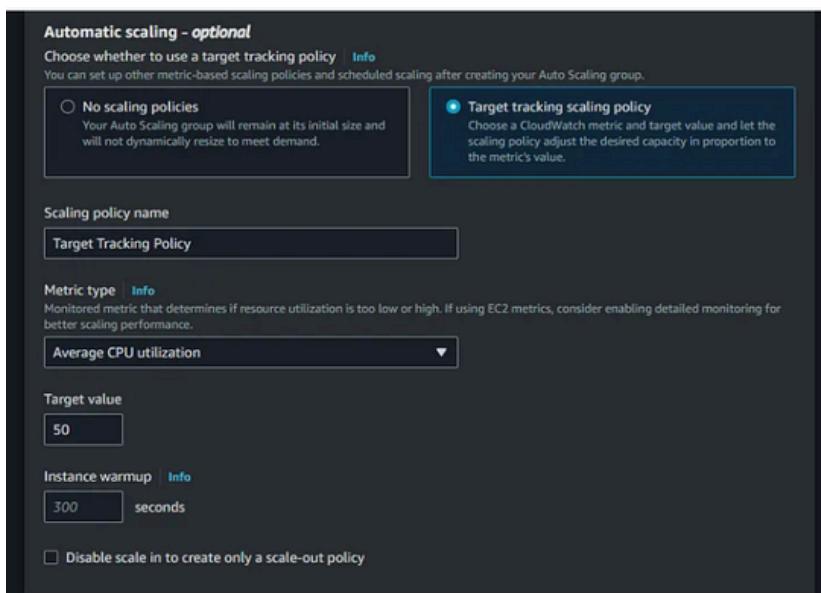


Next, we delve into scaling policies, which dictate how the auto scaling group responds to changes in workload or system metrics.

- **Scaling policies:** These policies govern when and how the auto scaling group should scale out (add instances) or scale in (remove instances) based on predefined conditions. We can choose from various scaling policy types, such as target tracking scaling policies or simple scaling policies.

For instance, we opt for a target tracking scaling policy based on CPU utilization. If the combined CPU utilization across instances exceeds 50%, the auto scaling group will add instances to handle the increased workload. Conversely, if CPU utilization drops below 50%, instances may be removed to optimize resource usage.

Additionally, we have the option to enable scaling protection, which safeguards newly launched instances from immediate termination. This feature can be beneficial in scenarios where instances require time to initialize or synchronize data.



By configuring these capacity and scaling policy settings, we ensure that our auto scaling group operates efficiently, maintaining optimal performance while dynamically adjusting to changing demand.

The screenshot shows the 'Instance maintenance policy' configuration step in the AWS Auto Scaling wizard. It includes sections for 'Mixed behavior', 'Prioritize availability', 'Control costs', and 'Flexible' policies, with 'No policy' selected. It also includes an 'Instance scale-in protection' section with an unchecked checkbox for enabling protection. Navigation buttons at the bottom are 'Cancel', 'Skip to review' (disabled), 'Previous', and 'Next'.

Add notifications

Now, let's proceed to the next step.

Here, we have the option to configure event notifications for various actions performed by the auto scaling group.

- **Notification settings:** We can specify a notification topic to receive notifications for specific events related to the auto scaling group's lifecycle. These events include instance launches, terminations, failures to launch, and failures to terminate.

By defining a notification topic, we ensure that relevant stakeholders or automated systems are promptly informed of any changes or issues affecting the auto scaling group's operation. This facilitates proactive monitoring and troubleshooting, allowing us to maintain the reliability and availability of our infrastructure.

After configuring the notification settings as needed, we can proceed to the next stage of the auto scaling group setup process.

Add notifications - optional Info

Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.

▼ Notification 1

[Remove](#)

SNS Topic

Choose an SNS topic to use to send notifications

[Create a topic](#)

Event types

Notify subscribers whenever instances

- Launch
- Terminate
- Fail to launch
- Fail to terminate

[Add notification](#)

[Cancel](#)

[Skip to review](#)

[Previous](#)

[Next](#)

Add tags

Let's add tags for identification purposes.

- Tag Name: "web-health"

We'll keep the naming simple and generic since the number of instances can vary dynamically based on scaling policies.

Add tags - optional Info

Add tags to help you search, filter, and track your Auto Scaling group across AWS. You can also choose to automatically add these tags to instances when they are launched.

ⓘ You can optionally choose to add tags to instances (and their attached EBS volumes) by specifying tags in your launch template. We recommend caution, however, because the tag values for instances from your launch template will be overridden if there are any duplicate keys specified for the Auto Scaling group. X

Tags (1)

Key

Name

Value - optional

web-health

Tag new instances



[Remove](#)

[Add tag](#)

49 remaining

[Cancel](#)

[Previous](#)

[Next](#)

Now, let's proceed by clicking on "Next" and review all the provided information before finalizing the creation of the auto scaling group. Once verified, we can proceed with the creation process.

Currently, we don't have any instances running. As per our configuration, the desired capacity is set to two instances. So, the auto scaling group will start creating instances to meet this desired capacity.

You can observe that two instances are being created to fulfill the desired capacity. Additionally, there's a scaling policy in place. If the CPU utilization drops below 50%, it will terminate instances accordingly. However, it will always maintain a minimum of one instance.

In the instance management section, you'll find all the instances within the auto scaling group. This section is particularly useful for troubleshooting purposes. Here, you can perform various actions such as detaching an instance from the auto scaling group, setting it to standby mode, or bringing it back into service.

The screenshot shows the AWS Auto Scaling console for the 'web-health-asg' group. The 'Activity' tab is selected. In the 'Activity notifications (0)' section, there is a 'Create notification' button. Below it, the 'Activity history (2)' section lists two entries:

Status	Description	Cause	Start Time	End Time
Not yet in service	Launching a new EC2 instance: i-0da9603fada3a0bf	At 2024-05-06T20:19:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-06T20:19:42Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 May 07, 01:49:44 AM +05:30	
Not yet in service	Launching a new EC2 instance: i-0bb1b6954340e2bb6	At 2024-05-06T20:19:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-06T20:19:42Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 May 07, 01:49:44 AM +05:30	

You have the option to set scale-in protection for instances. This means that these instances won't be terminated by the auto-scaling process. This feature is particularly useful during troubleshooting or maintenance activities when you don't want instances to be terminated unexpectedly.

We'll wait for a few minutes to ensure that all instances are in a healthy state. Once they are marked as healthy, we'll explore some additional options. Currently, both instances are running and healthy. You should also see them updated in the target group.

The screenshot shows the 'Instance management' tab in the AWS Auto Scaling console for the 'web-health-asg' group. The 'Instances (2)' section lists two instances:

Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template/la...	Availability Zone	Health status
i-0da9603fada3a0bf	InService	t2.micro	-	web-health-template	ap-south-1a	Healthy
i-0bb1b6954340e2bb6	InService	t2.micro	-	web-health-template	ap-south-1b	Healthy

An 'Actions' dropdown menu is open for the first instance, showing options like 'Detach', 'Set to Standby', 'Set to InService', 'Instance scale-in protection', and 'Remove scale-in protection'.

When managing instances within an auto-scaling group, it's crucial to remember that these instances are dynamic. They are automatically created and deleted by the auto-scaling group based on the defined policies and metrics. Therefore, manual changes to instances should be avoided to prevent unexpected behavior.

Any changes to instances should be made through the launch template. This includes changes such as instance type, security group, application upgrades, or software installations. If you need to make changes, create a new launch template or a new version of the existing launch template with the desired configurations.

To apply changes to instances, you can edit the auto-scaling group and select the new launch template. Once the changes are saved, you must initiate an instance refresh to apply the changes to existing instances. During instance refresh, instances are terminated and recreated based on the updated launch template.

The screenshot shows the 'Start instance refresh' configuration page for an Auto Scaling group named 'weh-health-asg'. The top navigation bar includes 'EC2 > Auto Scaling groups > weh-health-asg > Start instance refresh'. The main section is titled 'Start instance refresh' with a 'Info' link. A note states: 'An instance refresh performs a rolling update, replacing all or some instances.' Below this is the 'Availability settings' section, which explains the use of instance replacement method and instance warmup to control capacity and availability. The amount of capacity available will vary between the specified minimum and maximum values. The 'Instance replacement method' section contains three options: 'Prioritize availability' (radio button), 'Control costs' (radio button, selected), and 'Flexible' (radio button). The 'Control costs' option is described as terminating and launching instances at the same time, allowing for temporary availability reduction. The 'Flexible' option is described as setting custom minimum and maximum capacity values. The 'Set healthy percentage' section allows setting the minimum percentage of desired capacity that must be healthy. The current range is set from 90% to 100%, covering 2 instances. A note indicates that this range is currently within the group's scaling limits.

When initiating an instance refresh, ensure that you specify a minimum healthy percentage. This ensures that a certain percentage of instances remain healthy during the refresh process, preventing disruptions to your application. It's recommended to set a higher healthy percentage to maintain application stability.

Once changes are applied and instance refresh is initiated, monitor the auto-scaling group activity to ensure that new instances are launched successfully and the application remains available. Avoid manual termination of instances unless necessary for troubleshooting or maintenance purposes.

Please note – instance: i-02e53d55300f6659a I stopped purposefully.

Activity history (6)					
Status	Description	Cause	Start time	End time	
Not yet in service	Launching a new EC2 instance: i-02e53d55300f6659a	At 2024-05-06T20:27:42Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 May 07, 01:57:45 AM +05:30		
Connection draining in progress	Terminating EC2 instance: i-02e53d55300f6659a - Waiting For ELB Connection Draining.	At 2024-05-06T20:27:41Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2024 May 07, 01:57:41 AM +05:30		
Successful	Launching a new EC2 instance: i-02e53d55300f6659a	At 2024-05-06T20:25:36Z an instance was launched in response to an unhealthy instance needing to be replaced.	2024 May 07, 01:55:38 AM +05:30	2024 May 07, 01:56:10 AM +05:30	
Connection draining in progress	Terminating EC2 instance: i-0db3b0954540e2bb - Waiting For ELB Connection Draining.	At 2024-05-06T20:25:36Z an instance was taken out of service in response to an ELB system health check failure.	2024 May 07, 01:55:36 AM +05:30		
Successful	Launching a new EC2 instance: i-0ca960e3fade2aa8f	At 2024-05-06T20:19:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-06T20:19:42Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 May 07, 01:49:44 AM +05:30	2024 May 07, 01:50:16 AM +05:30	
Successful	Launching a new EC2 instance: i-0db3b0954540e2bb	At 2024-05-06T20:19:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-06T20:19:42Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2024 May 07, 01:49:44 AM +05:30	2024 May 07, 01:50:16 AM +05:30	

We can see 2 instances are running:

Instances (2) Info										
Find instance by attribute or tag (case-sensitive)										
Running										
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP	
web-health	i-0db3b0954540e2bb	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-235-69-162.ap...	13.235.69.162	-	
web-health	i-0ca960e3fade2aa8f	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-3-110-214-146.ap...	3.110.214.146	-	

By following these best practices, you can effectively manage instances within your auto-scaling group while ensuring the reliability and scalability of your application.

In conclusion, auto-scaling groups in AWS offer a powerful solution for managing the scalability and availability of your application infrastructure. By dynamically adjusting the number of EC2 instances based on predefined policies and metrics, auto-scaling groups enable you to efficiently handle fluctuating traffic loads while maintaining optimal performance and cost-effectiveness.

Key points to remember include:

1. Dynamic Nature: Auto-scaling groups automatically create and terminate EC2 instances based on configured scaling policies, ensuring that your application can handle varying levels of demand without manual intervention.
2. Launch Templates: Changes to instance configurations should be made through launch templates, allowing for easy management of instance settings such as instance type, security groups, and software configurations.
3. Instance Refresh: When updating configurations, initiate an instance refresh to apply changes to existing instances. Specify a minimum healthy percentage to ensure application stability during the refresh process.
4. Avoid Manual Changes: Manual changes to instances within auto-scaling groups should be avoided to prevent conflicts with scaling policies and unexpected behavior. Instead, utilize launch templates for any necessary updates.
5. Monitoring and Maintenance: Regularly monitor the auto-scaling group activity to ensure that instances are being created and terminated as expected. Conduct troubleshooting and maintenance activities carefully to avoid disruptions to your application.

We're concluding our discussion on EC2 concepts.....

