



Why You Need This Book

Reading this book about AWS (Amazon Web Services) cloud services can be beneficial for several reasons:

Understanding AWS Services: AWS offers a vast array of services for cloud computing, ranging from storage and computing to machine learning and artificial intelligence. Reading a book on AWS can help you understand the various services available, their features, and how they can be used to meet different business needs.

Learning Best Practices: Books often cover best practices for using AWS services efficiently, securely, and cost-effectively. This knowledge can help you design and implement solutions that are robust and scalable.

Preparing for Certification: If you're pursuing AWS certifications, such as AWS Certified Solutions Architect or AWS Certified Developer, studying from books can provide comprehensive coverage of the topics you need to know for the exams.

Staying Updated: Cloud computing is a rapidly evolving field, and AWS frequently introduces new services and updates existing ones. Books authored by AWS experts often provide insights into the latest trends, updates, and best practices in the AWS ecosystem.

Deep Dives and Case Studies: Some books offer in-depth explanations of specific AWS services or case studies illustrating real-world implementations. These can help deepen your understanding of how to use AWS effectively in practical scenarios.

Reference Material: Books can serve as valuable reference material that you can consult whenever you encounter challenges or need to refresh your knowledge about specific AWS services or concepts.

Overall, reading AWS cloud books can be a valuable investment in your professional development if you work with or plan to work with AWS cloud services.

About the Author:

Madhukar Reddy Vennahas earned a master's degree from California State University. He is a highly acclaimed trainer, author and solutions provider. He regularly trains students in in-house, at Vcube Software Solutions. He has an experience of more than 12+ years in industry and Training, where he has delivered more than 200 batches successfully in Devops, Devsecops, Mlops, Aiops

Though we have made taken utmost efforts to present you with this book error-free, but still, it may contain some errors or mistakes. Students are encouraged to bring if there are any mistakes or errors in this document to our notice.



Though we have made utmost efforts to present you this book error-free, it may contain some errors or mistakes. Students are encouraged to bring if there are any mistakes or errors in this document to our notice.

Few details About the Author



Author with JNTU KAKINADA Vice Chancellor (Dr. G.V.R. PRASAD RAJU)



Author with ANDHRA UNIVERSITY
Vice Chancellor (Prof.P.V.G.D. Prasad Reddy)



Author with Acharaya Nagarjuna University Vice Chancellor
(Prof. Rajasekhar, P.M.A, M.Phil)



Table of Contents

S.No.	Name	Page Nos.
1	What is AWS Cloud	6
2	Why Linux Need	
3	Linux Commands	
4	Region & Availability Zones.....	
5	Secure Remote Administration	
6	Elastic Cloud Compute	
7	Virtual Private Cloud	
8	Virtual Private Cloud Peering	
9	Transit Gateway	
10	Virtual Private Cloud Endpoint	
11	Security & Network Access Control List	
12	Elastic Block Storage	
13	Elastic File System	
14	Load Balancer	
15	Auto Scaling.....	
16	Web Application Firewall	
17	Relational Database	
18	Dynamo Database	
19	Simple Storage Service	
20	Identity Access Management	
21	Amazon Machine Image	
22	Snapshot	
23	Elastic Beam Stack	
24	Cloud Watch.....	
25	Cloud Trail	



S.No.	Name	Page Nos.
26	Route 53	
27	Cloud Front	
28	Access Certificate Manager	
29	Amplify	
30	Lambda	
31	Simple Notification Service	
32	Simple Queue Service	
33	Simple Email Service	
33	Project-1	
34	Project-2	



What is anAWS and cloud technology?

What I've found is, cloud is the future for almost every business and most of young people aren't aware much about it.

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).



This article will talk about several sub-topics related to cloud technology and amazon web services(AWS) are as follows:

- What is cloud?
- History of cloud computing.
- Which companies provide cloud services and the topmost in the market.
- What is an AWS?
- What can you do with AWS?
- Criteria and way to get a cloud job.
- Bonus: Best courses for learning cloud in AWS.

What is cloud?

Cloud, as the name might suggest, is not really cloud or something above in the air. It's simply somebody else's computer or more precisely, a server. Now, most of us don't realise that we use cloud on a daily basis, without actually knowing what it is or how are we even using with.

Let's say, you want to set-up your business online by creating websites and storing the information provided by all your users. This, in a typical early 90's scenario would require several rooms for servers and for storage of data. This also depends upon your size of business. You will also require manager, administrators, engineers and, professionals to require a manager manage and administrate the servers.



Okay, so on a smaller scale, creating a blogging platform as a part of your secondary source of income you would need an i7 core processor and proper storage which could keep your data and other information. Now, it's a lot of work because along with getting new content you would also need to manage your little baby server and organize the data on a daily basis. And also, keeping your server up 24*7? It's still a lot of work, isn't it?

History of cloud computing: The story of Cloud computing so far...

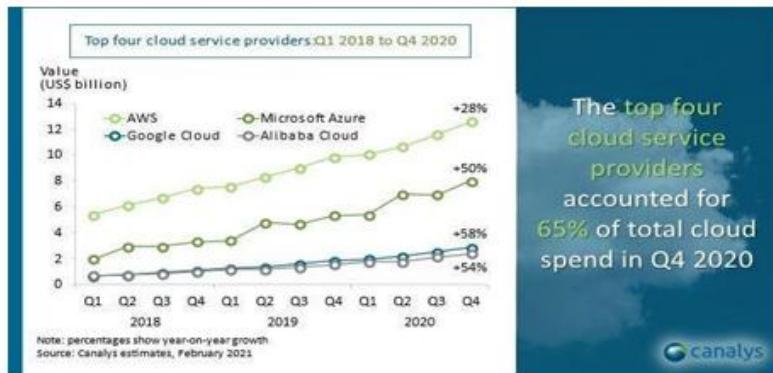
- **1999:** Salesforce.com launches CRM as a service
- **2002:** Amazon launches AWS for developers
- **2006:** AWS launches pay-per-use commercial cloud with S3 (storage) and EC2 (computer) services
- **2008:** Google launches App Engine offering developers a scalable application environment
- **2010:** Microsoft launches Azure IaaS (Beta version)
- **2011:** Apple launches iCloud and Microsoft buys Skype
- **2015:** Global Cloud industry exceeds \$100 Billion revenues
- **2016:** AWS exceeds \$12 Billion in IaaS/PaaS revenues and now offers 70 distinct Cloud services
- **2017:** Microsoft passes \$10 Billion in SaaS revenue. Salesforce is #2 SaaS player with \$8.5 Billion revenues.
- **2018:** Global Cloud IT infrastructure spend exceeds traditional IT
- **2019:** SaaS market exceeds \$110 Billion revenues.
- **2020:** Total Cloud services revenues exceed \$250 Billion.

Which companies provide cloud services and the topmost in the market?

Now, I hope you've got the idea most web hosting companies offer you. They offer you management of your platform, server, storage and professional security. So, you only have to focus on serving great content!

Cloud computing makes it easier, cheaper and faster to run state-of-the-art IT architectures in any type of company, large or small. Businesses benefit from cheaper, faster, more scalable IT resources in the Cloud and users get a better experience. A virtuous circle exists between software users and software developers in SaaS Clouds: developers can improve the software faster because they can see usage and performance data in real time. Meanwhile, users get the latest software upgrades as soon as they are released, without having to pay more or having to fiddle with clumsy downloads.

According to a report by Canalys shown in the below chart, in Q4, 2020, AWS cloud grew by 28% and Azure, Google, and Alibaba clouds grew by 50%, 58%, and 54% respectively. As of this report, AWS has 31% of the total cloud market share followed by Azure, Google, and Alibaba which have 20%, 7%, and 6% respectively. $20 + 7 + 6 = 33$ where are its mentioned as 31% above.



Here is a list of my top 10 cloud service providers:

1. Amazon Web Services (AWS)
2. Microsoft Azure
3. Google Cloud
4. Alibaba Cloud
5. IBM Cloud
6. Oracle
7. Salesforce
8. SAP
9. Rackspace Cloud
10. VMWare

The following table summarizes the top 3 key players and their offerings in the cloud computing world:

	AWS	Azure	Google Cloud
Company	AWS Inc.	Microsoft	Google
Launch year	2006	2010	2008
Geographical Regions	25	54	21
Availability Zones	78	140 (countries)	61
Key offerings	Compute, storage, database, analytics, networking, machine learning, and AI, mobile, developer tools, IoT, security, enterprise applications, blockchain.	Compute, storage, mobile, data management, messaging, media services, CDN, machine learning and AI, developer tools, security, blockchain, functions, IoT.	Compute, storage, databases, networking, big data, cloud AI, management tools, Identity and security, IoT, API Platform
Compliance Certificates	46	90	
Annual Revenue	\$33 billion	\$35 billion	\$8 billion



What is an AWS?

Amazon Web Services (AWS)



Amazon Web Services (AWS) is an Amazon company that was launched in the year 2002. AWS is the most popular cloud service provider in the world.

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 165 fully-featured services from data centers globally. This service is used by millions of customers.

AWS's revenue in the year 2018 was \$25.6 billion with a profit of \$7.2 billion. The revenue is expected to grow to \$33 billion in 2019.

AWS global availability

AWS offers the largest footprint in the market. No other cloud provider offers as many regions or Availability Zones (AZs). This includes 78 AZs within 25 geographic regions around the world. Furthermore, AWS has announced plans for 9 more AZs and three more regions in Cape Town, Jakarta, and Milan.



In simple words AWS allows you to do the following things-

- Running web and application servers in the cloud to host dynamic websites.
- Securely store all your files on the cloud so you can access them from anywhere.



- Using managed databases like MySQL, PostgreSQL, Oracle or SQL Server to store information.
- Deliver static and dynamic files quickly around the world using a Content Delivery Network (CDN).
- Send bulk email to your customers.

Compute:

- **EC2 (Elastic Compute Cloud)** — These are just the **virtual machines** in the cloud on which you have the OS level control. You can run whatever you want in them.
- **LightSail** — If you don't have any prior experience with AWS this is for you. It **automatically deploys** and manages compute, storage and networking capabilities required to run your applications.
- **ECS (Elastic Container Service)** — It is a highly scalable **container service** to allows you to run Docker containers in the cloud.
- **EKS (Elastic Container Service for Kubernetes)** — Allows you to use **Kubernetes on AWS** without installing and managing your own Kubernetes control plane. It is a relatively new service.
- **Lambda** — AWS's serverless technology that allows you to run **functions in the cloud**. It's a huge cost saver as you pay only when your functions execute.
- **Batch** — It enables you to easily and efficiently run **batch computing** workloads of any scale on AWS using Amazon EC2 and EC2 spot fleet.
- **Elastic Beanstalk** — Allows **automated deployment and provisioning** of resources like a highly scalable production website.

Storage:

- **S3 (Simple Storage Service)** — Storage service of AWS in which we can store objects like files, folders, images, documents, songs, etc. It cannot be used to install software, games or Operating System.
- **EFS (Elastic File System)** — Provides **file storage** for use with your EC2 instances. It uses NFSv4 protocol and can be used concurrently by thousands of instances.
- **Glacier** — It is an extremely low-cost **archival service** to store files for a long time like a few years or even decades.
- **Storage Gateway** — It is a virtual machine that you install on your on-premise servers. Your on-premise data can be backed up to AWS providing more durability.

Databases:

- **RDS (Relational Database Service)** — Allows you to run **relational databases** like MySQL, MariaDB, PostgreSQL, Oracle or SQL Server. These databases are fully managed by AWS like installing antivirus and patches.



Demand for AWS Jobs Outstrips Available Professionals:

In the public cloud job market, there are between six to 12 times more job postings available than there are job seekers, and 60 percent of these job postings are AWS-related. Employers in the United States, for example, say that it's quite a challenge finding professionals with cloud computing skills in general. This imbalance will continue to be the case for a long time to come.



Whether you're already an experienced IT professional seeking to take your career in a new direction or new to cloud computing (or IT, for that matter), there are several reasons why you should consider AWS. And since AWS is the leading public cloud computing service that is widely adopted by organizations both large and small, then it also follows that learning AWS has become a necessity for IT professionals who want to secure their future careers.

How to Learn AWS:

Now that you have some solid reasons why an AWS career can be beneficial, the next step is to find out how you can go about acquiring the necessary knowledge, skills, and certifications for AWS.

There is an Abundance of AWS Learning Resources:

Choose Wisely

Since AWS certifications were first introduced in 2013, a lot of resources have been made available ranging from books, manuals, courses, AWS practice exams, and AWS communities. These resources are all useful for those seeking to start and grow their career in AWS. However, choosing the right learning resource is critical since there is a lot to go through since some courses are simply better than others.

Bonus: AWS Certifications

AWS certifications are divided into four major categories — Foundational, Associate, Professional, and Specialty.



Professional

Two years of comprehensive experience designing, operating, and troubleshooting solutions using the AWS Cloud

Associate

One year of experience solving problems and implementing solutions using the AWS Cloud

Foundational

Six months of fundamental AWS Cloud and industry knowledge



Specialty

Technical AWS Cloud experience in the Specialty domain as specified in the exam guide



Choose a Career Path that Suits You Best:

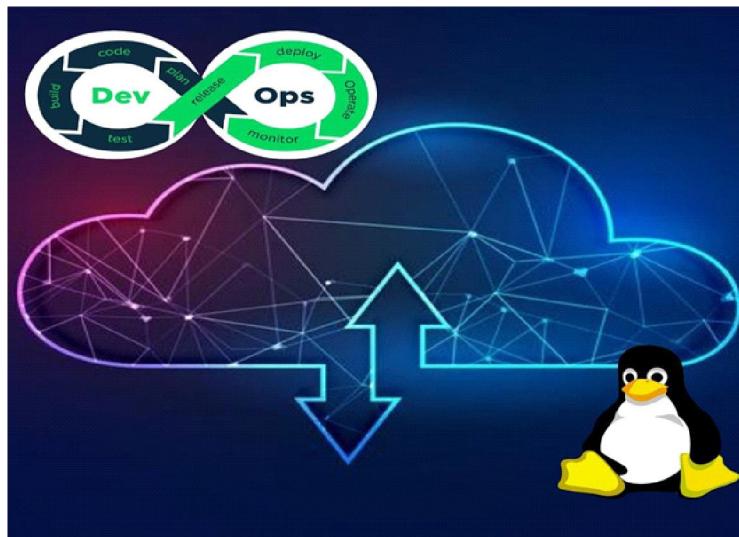
There are a lot of AWS career paths from which you can choose. The career path you want can be based on either:

- **The Role:** Such as cloud practitioner, operations, architect, and developer
- **The Solution:** Such as storage, machine learning, and AWS media services

You could also choose a specialty area on which to focus your attention and validate advanced skills in specific technical domains.



Why Linux is needed for Cloud and DevOps Professionals



Linux is widely used in the cloud and DevOps professions for several reasons:

- **Compatibility:** Linux is the dominant operating system in the cloud and DevOps ecosystem. Most cloud providers, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, offer Linux-based virtual machines and container services. Having Linux proficiency allows professionals to work seamlessly in these environments.
- **Open-source tools and technologies:** Linux is the foundation for many open-source tools and technologies commonly used in cloud and DevOps, such as Docker, Kubernetes, Ansible, Terraform, and Git. Familiarity with Linux enables professionals to effectively leverage these tools and contribute to the open-source community.
- **Command-line proficiency:** Linux provides a powerful command-line interface (CLI) that allows professionals to efficiently manage and automate tasks. Many cloud and DevOps workflows involve CLI-based operations, including provisioning and configuring cloud resources, deploying applications, and scripting automation processes.
- **Scripting and automation capabilities:** Linux offers a rich set of scripting and automation capabilities through shells like Bash and programming languages like Python. DevOps professionals often write scripts and automation code to orchestrate infrastructure, deployment pipelines, and various operational tasks.
- **Security and performance:** Linux is known for its robust security features and performance optimizations. Understanding Linux security mechanisms, file permissions, and network configurations is crucial for maintaining secure and high-performing cloud and DevOps environments.



- **Infrastructure-as-Code (IaC):** Infrastructure-as-Code is a key principle in cloud and DevOps. Linux provides the flexibility and control necessary to define infrastructure configurations as code using tools like Terraform and Ansible. These configurations can be versioned, tested, and deployed, resulting in consistent and reproducible infrastructure.
- **Troubleshooting and debugging:** In complex cloud and DevOps environments, issues can arise at various levels, from applications to infrastructure. Linux expertise allows professionals to effectively troubleshoot problems, diagnose performance bottlenecks, and debug issues at the system and application levels.
- By having a strong foundation in Linux, cloud and DevOps professionals can navigate the ecosystem, work with essential tools, automate tasks efficiently, and contribute effectively to the infrastructure and deployment processes required in these fields.



Linux Commands which are commonly used for System Admins/ Cloud & DevOps Engineers



There are several fundamental Linux commands you should be familiar with to navigate and operate Linux-based systems efficiently:

Linux Basic Commands:

- **ls:** List files and directories in the current directory.
Example: ls -l (detailed list), ls -a (including hidden files), ls -lh (human-readable file sizes).
- **cd:** Change directory.
Example: cd /path/to/directory (absolute path), cd directory (relative path), cd .. (go up one directory).
- **pwd:** Print the current working directory (shows the path of the current directory).
- **mkdir:** Create a new directory.
Example: mkdir directory_name.
- **rm:** Remove files and directories.
Example: rm file.txt (remove a file), rm -r directory (remove a directory and its contents).
- **cp:** Copy files and directories.
Example: cp file.txt destination_directory (copy a file), cp -r directory destination_directory (copy a directory and its contents).
- **mv:** Move/rename files and directories.
Example: mv file.txt new_location/file.txt (move a file), mv file.txt new_name.txt (rename a file), mv directory new_location/directory (move a directory).
- **cat:** Display the contents of a file.
Example: cat file.txt.
- **less:** View the contents of a file interactively.
Example: less file.txt.
- **head:** Display the first few lines of a file.
Example: head -n 10 file.txt (display the first 10 lines).



- **tail:** Display the last few lines of a file.
Example: tail -n 5 file.txt (display the last 5 lines).
- **grep:** Search for a pattern in files.
Example: grep “pattern” file.txt (search for a pattern in a file).
- **chmod:** Change file permissions.
Example: chmod +x script.sh (add executable permissions to a file).
- **chown:** Change the owner of a file or directory.
Example: chown user:group file.txt (change the owner and group of a file).
- **sudo:** Execute a command with superuser (administrative) privileges.
Example: sudo apt-get install package_name (install a package using the package manager).

Linux Intermediate Commands:

- **find:** Search for files and directories based on various criteria.
Example: find /path/to/search -name “*.txt” (find all files with the .txt extension).
- **grep:** Search for patterns within files.
Example: grep “pattern” file.txt (search for a pattern in a file).
- **sed:** Stream editor for modifying text.
Example: sed ‘s/foo/bar/’ file.txt (replace “foo” with “bar” in file.txt).
- **awk:** Text processing tool for extracting and manipulating data. Example: awk ‘{print \$1}’ file.txt (print the first field of each line in file.txt).
- **sort:** Sort lines of text files.
Example: sort file.txt (sort the lines in file.txt alphabetically).
- **uniq:** Remove duplicate lines from a sorted file
Example: uniq file.txt (remove duplicate lines from file.txt).
- **wc:** Word, line, character, and byte count.
Example: wc -l file.txt (count the number of lines in file.txt).
- **tar:** Archive files into a tarball (compressed file).
Example: tar -czvf archive.tar.gz files/ (create a compressed tarball of the “files” directory).
- **gzip:** Compress files.
Example: gzip file.txt (compress file.txt, creating file.txt.gz).
- **gunzip:** Decompress gzip files.
Example: gunzip file.txt.gz (decompress file.txt.gz).



- **wget:** Download files from the web.
Example: wget https://example.com/file.txt (download file.txt from a URL).
- **ssh:** Secure Shell — remotely connect to another machine over a network.
Example: ssh user@hostname (connect to a remote machine).
- **scp:** Securely copy files between hosts over a network.
Example: scp file.txt user@remote:/path/to/destination (copy file.txt to a remote machine).
- **du:** Estimate file and directory disk usage.
Example: du -sh directory (display the total size of the directory in human-readable format).
- **df:** Report file system disk space usage.
Example: df -h (display disk space usage of all mounted file systems in human-readable format).

Linux Advanced Commands:

- **rsync:** Synchronize files and directories between local and remote systems.
Example: rsync -avz source/ destination/ (synchronize the contents of the source directory to the destination directory).
- **scp:** Securely copy files between hosts over a network.
Example: scp -r user@remote:/path/to/source/ /path/to/destination/ (copy files and directories recursively between remote and local machines).
- **ssh-keygen:** Generate SSH key pairs for secure authentication.
Example: ssh-keygen -t rsa -b 4096 (generate a 4096-bit RSA key pair).
- **screen:** Create and manage multiple terminal sessions within a single SSH session.
Example: screen (start a new screen session), screen -r (resume a detached screen session).
- **top:** Monitor system processes and resource usage in real-time.
Example: top (display live process information).
- **htop:** Interactive process viewer with an enhanced UI.
Example: htop (launch htop process viewer).
- **cron:** Schedule recurring tasks or jobs.
Example: crontab -e (edit the user's crontab file), crontab -l (list the user's crontab entries).
- **systemctl:** Control and manage system services and daemons.
Example: systemctl start service_name (start a service), systemctl stop service_name (stop a service).
- **journalctl:** View and manage system logs.



Example: journalctl -u service_name (display logs for a specific service), journalctl -f (follow logs in real-time).

- **dd:** Convert and copy files and disk images.
Example: dd if=/dev/sda of=image.img bs=4M (create an image of the /dev/sda disk).
- **lsof:** List open files and processes.
Example: lsof -i :port_number (list processes using a specific port).
- **tcpdump:** Capture and analyze network traffic.
Example: tcpdump -i eth0 port 80 (capture HTTP traffic on the eth0 interface).
- **nc:** Netcat — network utility for reading/writing data across network connections.
Example: nc -l -p port_number (listen on a specific port for incoming connections).
- **strace:** Trace system calls and signals of a running program.
Example: strace -p process_id (trace system calls of a specific process).
- **chroot:** Change the root directory for a specific command or process.
Example: chroot /new_root_directory command (run a command with a different root directory).

Linux Networking Commands:

- **ifconfig:** Display or configure network interfaces.
Example: ifconfig eth0 (display information about the eth0 interface).
- **ip:** Show or manipulate routing, network devices, and addresses.
Example: ip addr show (display IP addresses of network interfaces).
- **ping:** Send ICMP echo requests to a specified network host.
Example: ping google.com (send ICMP echo requests to google.com).
- **traceroute:** Print the route packets take to a network host.
Example: traceroute google.com (trace the route to google.com).
- **netstat:** Display network connection information, routing tables, and network interface statistics.
Example: netstat -tuln (display listening ports).
- **ss:** Utility to investigate sockets.
Example: ss -tunap (display TCP and UDP sockets and associated processes).
- **dig:** DNS lookup utility for querying DNS servers.
Example: dig google.com (perform a DNS lookup for google.com).



- **host:** DNS lookup utility for querying DNS servers.
Example: host google.com (perform a DNS lookup for google.com).
- **wget:** Download files from the web.
Example: wget https://example.com/file.txt (download file.txt from a URL).
- **curl:** Command-line tool for transferring data using various protocols.
Example: curl https://example.com (retrieve the contents of a webpage).
- **ssh:** Secure Shell — remotely connect to another machine over a network.
Example: ssh user@hostname (connect to a remote machine).
- **scp:** Securely copy files between hosts over a network.
Example: scp file.txt user@remote:/path/to/destination (copy file.txt to a remote machine).
- **iptables:** Firewall administration tool for IPv4 packets.
Example: iptables -L (display the current firewall rules).
- **ip6tables:** Firewall administration tool for IPv6 packets.
Example: ip6tables -L (display the current IPv6 firewall rules).
- **route:** Show or manipulate the IP routing table.
Example: route -n (display the routing table).

Linux Performance Commands:

- **top:** Display real-time system information, including CPU usage, memory usage, and running processes.
Example: top
- **htop:** Interactive process viewer with an enhanced UI, providing detailed system monitoring.
Example: htop
- **vmstat:** Report virtual memory statistics, including CPU usage, memory utilization, and I/O statistics.
Example: vmstat
- **iostat:** Report CPU and I/O statistics for devices and partitions.
Example: iostat
- **sar:** Collect, report, or save system activity information.
Example: sar -u (display CPU usage)
- **free:** Display memory usage and statistics.
Example: free



- **ps:** Report a snapshot of the current processes, including CPU and memory usage.
Example: ps aux
- **pidstat:** Report statistics for processes, including CPU, memory, and I/O usage.
Example: pidstat
- **dstat:** Versatile resource statistics tool that combines multiple performance metrics.
Example: dstat
- **perf:** Powerful performance profiling tool for analyzing and investigating system behavior.
Example: perf record -p PID (record performance data for a specific process)
- **strace:** Trace system calls and signals of a running program.
Example: strace -p PID (trace system calls for a specific process)
- **uptime:** Display system uptime and load averages.
Example: uptime
- **lsof:** List open files and processes, useful for identifying resource usage.
Example: lsof -i (list network connections)
- **netstat:** Display network connection information, routing tables, and network interface statistics.
Example: netstat -s (display network statistics)
- **iostop:** Monitor I/O usage information of processes and disks. Example: iostop

Linux troubleshooting Commands:

- **dmesg:** Display the system's kernel ring buffer messages, which can provide information about hardware and driver issues.
Example: dmesg
- **journaldctl:** View and manage system logs, including systemd logs.
Example: journalctl
- **lsmod:** List loaded kernel modules.
Example: lsmod
- **lspci:** List PCI devices connected to the system.
Example: lspci
- **lsusb:** List USB devices connected to the system.
Example: lsusb



- **lsblk:** List information about block devices (disks).
Example: lsblk
- **fdisk:** Display or manipulate disk partition table.
Example: fdisk -l (list disk partitions)
- **blkid:** Print block device attributes, such as UUIDs and file system types.
Example: blkid
- **ifconfig:** Display or configure network interfaces.
Example: ifconfig
- **ip:** Show or manipulate routing, network devices, and addresses.
Example: ip addr show
- **ping:** Send ICMP echo requests to a specified network host.
Example: ping google.com
- **traceroute:** Print the route packets take to a network host.
Example: traceroute google.com
- **netstat:** Display network connection information, routing tables, and network interface statistics.
Example: netstat -tuln
- **ssh:** Secure Shell — remotely connect to another machine over a network.
Example: ssh user@hostname
- **sudo:** Execute a command with superuser (administrative) privileges.
Example: sudo command



AWS Regions and Availability Zones



Amazon Web Services (AWS) is renowned for its global cloud infrastructure, designed to offer high availability, fault tolerance, and scalability. At the heart of this infrastructure are AWS Regions and Availability Zones (AZs). Understanding these concepts is crucial for deploying resilient and efficient applications on AWS. This article will delve into the definitions, explanations, advantages, use cases, and real-life examples of AWS Regions and AZs.

What are AWS Regions?

AWS Regions are separate geographic areas around the world, such as North America, Europe, Asia, etc. Each Region is a collection of Availability Zones, which are isolated locations within a Region. AWS operates many Regions worldwide, allowing users to deploy their applications close to their end-users, reducing latency and improving performance.

Definition and Explanation:

- **AWS Region:** A specific geographical location hosting two or more Availability Zones.
- **Purpose:** Regions enable users to deploy applications and data across multiple locations to enhance availability, reduce latency, and comply with regulatory requirements.

What are Availability Zones?

Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other AZs. They offer the ability to operate production applications and databases that are more highly available, fault-tolerant, and scalable than would be possible from a single data center.

Definition and Explanation

- **Availability Zone:** A data center or cluster of data centers within a Region that is designed to be insulated from failures in other AZs.
- **Purpose:** AZs provide a foundation for building highly available and fault-tolerant applications by distributing them across multiple, isolated locations within a Region.



Advantages of Using AWS Regions and AZs

Enhanced Reliability and Fault Tolerance

By distributing your applications and data across multiple AZs within a Region, you can achieve higher levels of fault tolerance. In the event of an AZ failure, your application can continue to operate from other AZs, ensuring uninterrupted service.

Reduced Latency

Selecting a Region closest to your end-users can significantly reduce latency, improving the user experience for your applications.

Compliance and Data Sovereignty

Regions allow you to store data in specific geographic locations, meeting legal or regulatory requirements regarding data sovereignty.

Use Cases and Real-Life Examples

Global Web Application

A company deploying a web application for a global audience can use multiple Regions to serve content from the closest Region to the user, reducing latency and improving load times.

Disaster Recovery

By using multiple AZs within a Region or across regions, businesses can implement disaster recovery strategies that allow for rapid recovery of IT systems without data loss in the case of a disaster.

High Availability Database

Deploying a database across multiple AZs within a Region can ensure that even in the event of a complete AZ failure, the database remains available and minimizing downtime.

AWS Regions and Availability Zones are fundamental components of AWS's global infrastructure, offering significant advantages in terms of availability, fault tolerance, performance, and compliance. By strategically deploying applications and data across these geographic constructs, AWS users can ensure their systems are resilient, responsive, and compliant with regulatory requirements. Whether you're running a global application, implementing a robust disaster recovery plan, or requiring high availability for critical databases, understanding and leveraging AWS Regions and AZs is key to achieving your objectives. As of today, AWS spans 105 Availability Zones within 33 geographic regions around the world, with announced plans for 12 more Availability Zones and 4 more AWS Regions in Germany, Malaysia, New Zealand, and Thailand. Read more at : https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

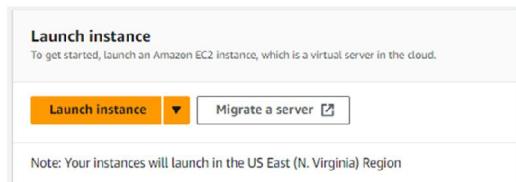


Secure Remote Administration and Troubleshooting of EC2 Instances



Step 1: Setting Up the EC2:

Sign in to your AWS account and access the EC2 Dashboard. Locate the “Launch Instances” button and click on it to proceed.



This action will direct you to a splash page where you can input settings for your new virtual machine. Assign a name to your instance and choose an appropriate AMI. For this project, ensure you select a Linux- based AMI, preferably one that is eligible for the free tier.

Name and tags info

Name Add additional tags

Application and OS Images (Amazon Machine Image) info

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

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

Recents | Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE L Search more AMIs

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI ami-0f403e3180720dd7e (64-bit (x86), uefi-preferred) / ami-0237525b5672165b3 (64-bit (Arm), uefi) Free tier eligible

Description Amazon Linux 2023 AMI 2023.3.20240304.0 x86_64 HVM kernel-6.1

Architecture 64-bit (x86) Boot mode uefi-preferred AMI ID ami-0f403e3180720dd7e Verified provider



Maintain the “Instance Type” as t2.micro, as it qualifies for the free tier and is ideal for this demonstration. Moving on to the next step will require delving into slightly more technical details.

Step 2: Creating a Keypair:

Now, we need to generate a key pair to securely establish SSH connections to our instance. Although restricting SSH access to specific IP addresses is an option, it’s not as secure as using AWS’s integrated key generator. Therefore, when you reach this section, locate the “Create new key pair” link and click on it.

▼ 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 - required

Select [Create new key pair](#)

Upon clicking the “Create new key pair” link, a popup will appear enabling you to generate a new key pair. Assign it a meaningful and easily memorable name. Ensure that you select an RSA key pair type and opt for the .pem key file format.

Create key pair [X](#)

Key pair name
Key pairs allow you to connect to your instance securely.
 The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH .ppk For use with PuTTY

⚠️ 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](#)

[Cancel](#) [Create key pair](#)

You might have observed the warning message above the “Create key pair” button. By generating a key pair, you’ll be downloading a key onto your local computer. Therefore, it’s crucial to remember where you save it and how to identify the key later when connecting to the instance.

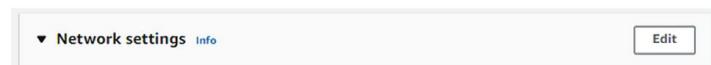
⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)



For my use, I ensured to download the .pem file it generates a key pair to remote into your EC2 instance. Be aware of where you downloaded the .pem file, you might want to place it in an easily accessible folder, as you'll need to access it later.

Step 3: Security Groups and Ports:

Next step, you'll encounter a "Network Settings" tab. Navigate to the top right-hand corner of the tab and select the "Edit" option.



Once clicked, you have the opportunity to input additional options, including the Virtual Private Cloud (VPC) where the instance resides.

▼ Network settings [Info](#)

VPC - required [Info](#)

vpc-0e84e4ec0276690ed (Default VPC) (default) ▾
172.31.0.0/16

Subnet [Info](#)

No preference ▾

Auto-assign public IP [Info](#)

Enable ▾

Firewall (security groups) [Info](#)

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

Create security group Select existing security group

Security group name - required
LUIT-SSH-sg

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](#)

LUIT-SSH-sg created 2024-03-13T01:47:04.113Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 71.190.171.227/32) Remove

Type Info	Protocol Info	Port range Info
ssh ▾	TCP	22

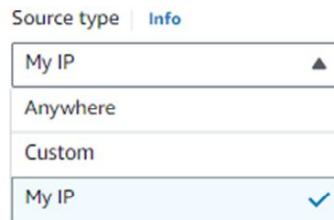
Source type [Info](#) Name [Info](#) Description - optional [Info](#)

My IP [Add CIDR, prefix list or security](#) e.g. SSH for admin desktop

If you've left the setting on the default "anything" rule with our key pair, an unauthorized individual on the internet wouldn't be able to directly log into your instance. However, they can attempt to access the instance, and over time, they may potentially breach the key pair, posing a security risk.



To mitigate this risk, we can take proactive measures by specifying a particular IP address as the sole source permitted to connect to our instance (My IP) on port 22.



My IP would be the IP address of your local PC.

AWS offers a convenient feature where it can automatically populate your IP address for the security group. This enables you to restrict access to the address associated with the device you're currently using. It's important to note that IP addresses are typically location-dependent and may change, especially if you're using a VPN. Therefore, it's essential to pay close attention to the specific CIDR (Classless Inter-Domain Routing) that is auto-populated to ensure accurate access restrictions.

Go ahead and “Launch” your instance.



That means we've successfully created an EC2 instance. Now we need to see if we can connect to it.

Step 4: SSH and Testing that Connection

Next, you'll need to navigate to the “Instances” tab located on the left - hand side. Upon clicking it, a list of instances associated with your account will appear. Locate your instance within this list, and then click on the “Instance ID” portion of the entry.

Instances (1/1) info

Find instance by name, ID or tag (case-insensitive)

Name	Instance ID	Instance state	Instance type	Status check
SSH-Project	i-01abb92c75a236299	Running	t2.micro	-

View alarms +

Public IPv4 DNS: ec2-54-196-131-9.com...

Public IPv4 IP: 54.196.131.9

Elastic IP: -

IPv6 IPs: -

Instance: i-01abb92c75a236299 (SSH-Project)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary: Info

Instance ID: i-01abb92c75a236299 (SSH-Project)

Private IP: 172.31.1.10

Public IP: 54.196.131.9 (open address)

Instance state: Running

Private IP DNS name (IPv4 only): 54.196.131.9

Private IP DNS name (IPv6 only): -

Instance type: t2.micro

VPC ID: vpc-0f8ed4ec

Subnet ID: subnet-0992a2a0

Public IP addresses:

- 54.196.131.9
- ec2-54-196-131-9.compute-1.amazonaws.com (open address)

Public IPv4 IP: 54.196.131.9

Public IPv6 IP: -

Elastic IP addresses: -

AWS Compute Optimizer findings: Click on the AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name: -

Monitoring: Disabled

AMI ID: ami-0f403e3180720007

Platform: Amazon Linux (Inferred)

Up and running.



Once the next page loads, you'll find a plethora of information about your instance. Most of this information can be disregarded for the current task. Locate and click on the "Connect" button to proceed.

The screenshot shows the AWS EC2 Instances page with the instance ID i-01abb92c75a236299 selected. The "Connect to instance" section is open, and the "SSH client" tab is active. It displays instructions for connecting via SSH, including the command ssh -i "SSH-Project.pem" ec2-user@ec2-54-196-131-9.compute-1.amazonaws.com. A note at the bottom states: "Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username."

When the next page loads, it should automatically open into the "SSH client" tab, providing all the necessary information to connect to your instance.

Now, let's open our command line. If you're using a Linux or Linux-like operating system, you'll need to adjust the permissions on your .pem key file. AWS provides the command for this, which you can copy and run in the terminal. However, since I'm on a Windows OS, I can skip this step.

Before running any commands, we need to navigate our "current working directory" to the location where our .pem file is stored. Depending on where you downloaded it, you'll need to use the cd command to navigate to the associated directory.

Once you're in the correct directory, you can copy the command from the "Connect" tab in the AWS console and paste it into your terminal. Then, execute the command. If this is your first time connecting to the instance with this particular IP address, it will prompt you to confirm the connection. Type "yes" into the command line and press Enter to proceed.

```
C:\Users\Grego\Documents\LUIT>ssh -i "SSH-Project.pem" ec2-user@ec2-54-196-131-9.compute-1.amazonaws.com
The authenticity of host 'ec2-54-196-131-9.compute-1.amazonaws.com (54.196.131.9)' can't be established.
ED25519 key fingerprint is SHA256:Dfx8v1BpJqAdWdmyFrp0QRqUfJKwSnGxc+LzFFbM1.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-196-131-9.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
#_
###_      Amazon Linux 2023
##_ \###\_
\###|      https://aws.amazon.com/linux/amazon-linux-2023
\#_ \#_ /_
\#_ \#_ /_
\#_ \#_ /_
[ec2-user@ip-172-31-31-8 ~]$
```

SSH'd.



Now we run whoami to make certain we're now connected.

```
[ec2-user@ip-172-31-31-8 ~]$ whoami  
ec2-user
```

And presto! We've deployed an EC2 instance and connected to it with SSH. Now it's safe to turn it off via the AWS Console so you don't get charged money.

Step 5: Windows Time

We are going to replicate the following functions with a Windows server.

The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Name and tags' and 'Info'. Under 'Name', the value 'Windows-RDP' is entered. Below this, there is a section titled 'Application and OS Images (Amazon Machine Image)'. It contains a search bar and a grid of icons representing different AMIs: Amazon Linux, AWS Lambda, macOS, Ubuntu, Windows (selected), Red Hat, and SUSE. A link to 'Browse more AMIs' is also present. At the bottom of this section, it says 'Free tier eligible'. The overall interface is clean and modern, typical of AWS services.

Still *t.2.micro*, if it's free it's for me.

The screenshot shows the 'Instance type' section of the AWS Lambda console. The 't2.micro' option is selected, highlighted with a blue border. To the right, it indicates 'Free tier eligible'. Below the selection, there is a summary of the instance details: Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true. It also lists base pricing for On-Demand Windows, SUSE, RHEL, and Linux instances. A note at the bottom states 'Additional costs apply for AMIs with pre-installed software'.

Generate a key pair for your Windows Instance.

The screenshot shows the 'Create key pair' dialog box. In the 'Key pair name' field, 'Windows-RDP' is typed. In the 'Key pair type' section, 'RSA' is selected. Below that, the 'Private key file format' is set to '.pem' (selected). A warning message at the bottom of the dialog box reads: '⚠ 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](#)'.

Cancel

Create key pair



Don't forget the security group rules! You don't want your EC2 instance compromised. Proceed to launch your Windows EC2 instance.

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

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, 3389, 71.190.171.227/32)

Type | [Info](#) Protocol | [Info](#) Port range | [Info](#)

Source type | [Info](#) Name | [Info](#) Description - optional | [Info](#)

 X

[Add security group rule](#)

Once the instance is up and running, click on the “Connect” button in the upper-right corner. Click on “RDP client”.

[EC2](#) > [Instances](#) > [i-02eddd44d9ea61f01](#) > Connect to instance

Connect to instance [Info](#)
Connect to your instance i-02eddd44d9ea61f01 (Windows-RDP) using any of these options

Session Manager | **RDP client** | EC2 serial console

Instance ID

Connection Type

Connect using RDP client
Download a file to use with your RDP client and retrieve your password.

Connect using Fleet Manager
To connect to the instance using Fleet Manager Remote Desktop, the SSM Agent must be installed and running on the instance. For more information, see [Working with SSM Agent](#)

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download remote desktop file](#)

When prompted, connect to your instance using the following username and password:

Public DNS

Username Info

Password | [Get password](#)

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

See a difference?

There's a difference in the “Connect” page compared to the Linux server.

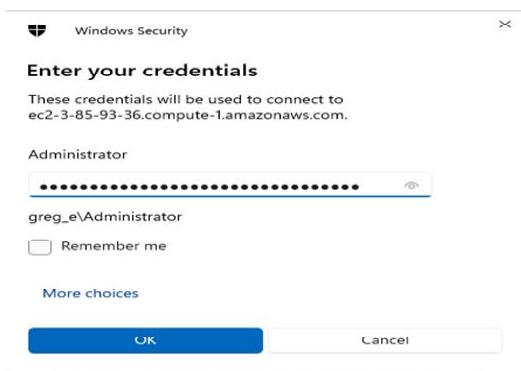


Click on the “Download remote desktop” file to install Windows RDP onto your PC.

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download remote desktop file](#)

Click on the “Get password” icon bolded in dark grey. Input your *.pem* file, which allows you to retrieve the password for your RDP session. Open your RDP application and input the generated password associated with the “Administrator” username.



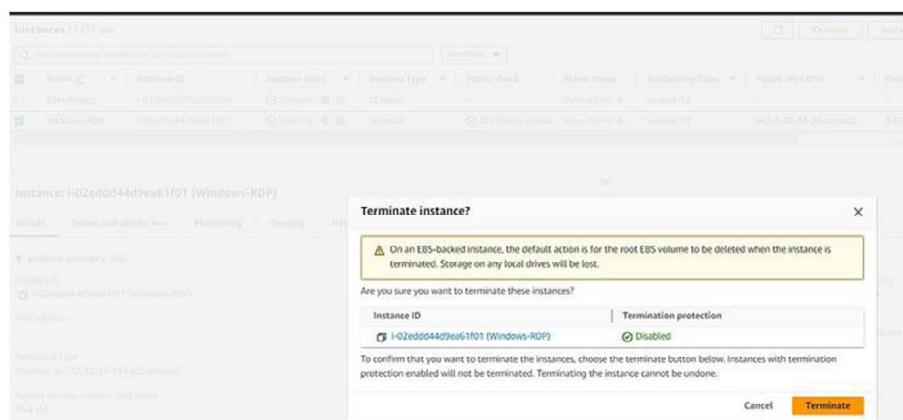
Big Yes here



Just like that, we have spun a fully operational Windows Server from the comfort of our home. This demonstrates the power and range of Cloud Computing.



Don't forget to terminate your instances, Linux and Windows, to prevent charges from occurring. Imagine you surpass the free-tier limit and get a surprise bill. Not the type of surprise you would be hoping for.





Elastic Compute Cloud - EC2

- EC2 is a web service which provides secure, resizable compute capacity in the cloud
- EC2 interface allows you to obtain & configure capacity with minimal friction
- EC2 offers the broadest and deepest compute platform with choice of processor, storage, networking, operating system, and purchase model.
- Amazon offer the fastest processor in the cloud and they are the only cloud with 400 Gbps Ethernet networking
- Amazon have the most powerful GPU instances for machine learning and graphic workloads.

Reliable, Scalable, Infrastructure on Demand:

- Increase or decrease capacity with in minutes, not hours or days
- SLA commitment of 99.99% availability for each amazon EC2 region. Each region consists of atleast 3 availability zones
- Region/AZ model is recognized by gartner as the recommended approach for running enterprise applications that require high availability.

AWS Supports 89 security standards & compliance certifications including:

- PCI-DSS
- HIPAA/HITECH
- FedRAMP
- GDPR
- FIPS
- NIST etc..

Features of Amazon EC2:

- Virtual Computing instances, known as instances
- Pre-configured templates for your instances, Known as AMI, which contains Operating System, Configuration and software.
- Various configurations of CPU, Memory, Storage, Networking Capacity for your instances, known as instance types
- Secure login information for your instances using keypairs (AWS Stores Public key, and you store the private key)
- Storage volumes for temporary data that's deleted when you stop, hibernate or terminate your instances, known as Instance store volumes



- Persistent storage volumes for your data using elastic block storage, known as amazon EBS Volumes
- Multiple physical locations for your resources, such as instance & EBS volumes, known as regions and availability zones.
- A firewall that enables you to specify the protocols, ports and source IP ranges that can reach your instance using security group
- Static IPv4 addresses for dynamic cloud computing known as Elastic IP Address

Amazon EC2 Provides the following purchasing options:

- On-Demand
- Spot Instances
- Reserved Instances
- Savings Plan

On-Demand Instances:

- You pay for compute capacity by the hour or the second depending on which instances you run
- No long-term commitment or upfront payments are needed
- You can increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use.

On Demand Instances are Recommended for

- Users that prefer the low cost and flexibility of Amazon EC2 without any upfront payment or long term commitment
- Applications with short term, spiky or unpredictable workloads that cannot be interrupted.
- Applications being developed or tested on Amazon EC2 for the first time

Spot Instances:

- Amazon EC2 spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% of on-demand price
- *Spot instances are recommended for*
- Applications that have flexible start and end times
- Applications that are only feasible at very low compute prices
- No guarantee for 24x7 uptime

Reserved Instances:

- Reserved Instances provide you with a significant discount (up to 75%) compared to on-demand instance pricing



- For applications that have steady state or predictable usage, reserved instance can provide significant savings compared to using on-demand instances

Recommended for:

- Applications with steady state usage
- Applications that may require reserved capacity
- Customers that can commit to using EC2 over a 1- or 3-year term to reduce their total computing costs

Savings Plan:

Savings plans are a flexible pricing model that offer low prices on EC2 in exchange for a commitment to a consistent amount usage for a 1- or 3-year term. Discount up to 72%

Amazon EC2 Instance Types:

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases.

Instance types comprise varying combinations of CPU, memory, storage, and networking capacity. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

- General Purpose
- Compute Optimized
- Accelerated Computing (GPU Optimized)
- Memory Optimized
- Storage Optimized

General Purpose:

General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.

Ex: Mac, T4g, T3, T3a, T2, M6g, M5, M5a, M5n, M5zn, M4, A1

Compute Optimized:

Compute Optimized instances are ideal for compute bound applications that benefit from high-performance processors. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing.

Ex: C6g, C6gn, C5, C5a, C5n, C4



Memory Optimized:

Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory.

Use case: Memory-intensive applications such as open-source databases, in-memory caches, and real time big data analytics

Ex: R6g, R5, R5a, R5b, R5n, R4, X2gd, X1e, X1, u, Z1d

Accelerated Computing:

Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating-point number calculations, graphics processing, or data pattern matching, more efficiently than is possible in software running on CPUs.

Use Case: Machine learning, high performance computing, computational fluid dynamics, computational finance, seismic analysis, speech recognition, autonomous vehicles, and drug discovery.

Ex: P4, P3, P2, Inf1, G4dn, G3, F1

Storage Optimized:

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latencies, random I/O operations per second (IOPS) to applications.

Ex: I3, I3en, D2, D3, D3en, H1

Instance Features:

Amazon EC2 instances provide a number of additional features to help you deploy, manage, and scale your applications.

- Burstable Performance instances
- Multiple Storage Options
- EBS Optimized Instances
- Cluster Networking

Burstable Performance Instances: Amazon EC2 allows you to choose between Fixed Performance Instances (e.g. M5, C5, and R5) and Burstable Performance Instances (e.g. T3). Burstable Performance Instances provide a baseline level of CPU performance with the ability to burst above the baseline.

For example, a t2.small instance receives credits continuously at a rate of 12 CPU Credits per hour. This capability provides baseline performance equivalent to 20% of a CPU core (20% x 60 mins = 12 mins). If the instance does not use the credits it receives, they are stored in its CPU Credit balance up to a maximum of 288 CPU Credits. When the t2.small instance needs to burst to more than 20% of a core, it draws from its CPU Credit balance to handle this surge automatically.



Multiple Storage Options: Amazon EC2 allows you to choose between multiple storage options based on your requirements. Amazon EBS is a durable, block-level storage volume that you can attach to a single, running Amazon EC2 instance.

Amazon EBS provides three volume types to best meet the needs of your workloads: General Purpose(SSD), Provisioned IOPS (SSD), and Magnetic.

EBS Optimized instances:

For an additional, low, hourly fee, customers can launch selected Amazon EC2 instances types as EBS-optimized instances. For M6g, M5, M4, C6g, C5, C4, R6g, P3, P2, G3, and D2 instances, this feature is enabled by default at no additional cost. EBS-optimized instances enable EC2 instances to fully use the IOPS provisioned on an EBS volume. EBS-optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 and 4,000 Megabits per second (Mbps) depending on the instance type used.

Cluster Networking:

Select EC2 instances support cluster networking when launched into a common cluster placement group. A cluster placement group provides low-latency networking between all instances in the cluster. The bandwidth an EC2 instance can utilize depends on the instance type and its networking performance specification.

EC2 Tenancy Model:

AWS offers 3 different types of tenancy model for your EC2 instances.

This relates to what underlying host your EC2 instance will reside on

- Shared Tenancy
- Dedicated Instance
- Dedicated Host

Shared Tenancy :

this option will launch your EC2 instance on any available host with the specified resources required for your selected instance type. Regardless of which other customers and users also have EC2 instances running on the same host. Means we are going to share the physical resources with other customers.

AWS implements advanced security mechanisms to prevent one EC2 instance from accessing another on the same host.

Dedicated Instance:

Dedicated instances are hosted on hardware that no other customer can access. It can only be accessed by your own AWS account. You may be required to launch your instances as a dedicated instance due to internal security policies or external compliance controls.



Dedicated instances do incur additional charges due to the fact you are preventing other customers from running EC2 instances on the same hardware and so there will likely be unused capacity remaining.

Dedicated Host:

A dedicated host is effectively the same as dedicated instances. However they offer additional visibility and control, how you can place your instances on the physical host. They also allow you to use your existing licenses, such as PA-VM license or Windows Server licenses Etc. Using dedicatedhosts gives you the ability to use the same host for a number of instances that you want to launch and align with any compliance and regulatory requirements.

Following are the list of important terms need to know before creating EC2 instances:

- Amazon Machine Image (AMI)
- Instance Type
- Network
- Subnet
- Public IP
- Elastic IP
- Private IP
- Placement Group
- Root Volume
- Security Group
- KeyPair

Amazon Machine Image:

An Amazon Machine Image (AMI) Provides the information required to launchan instance. An AMI Includes the following, one or more Elastic Block Store snapshot, a template for theroot volume of the instance (for example Operating system, software, configurations etc.)

Instance Type:

Instance types comprise varying combinations of CPU, Memory, storage & Networkingcapacity and give you the flexibility to choose the appropriate mix of resources for your applications.

Subnet:

Subnet is a subnetwork in your virtual network of your Amazon Network. By default, there is onesubnet per availability zone.



Public IP:

A public IP is an IP Address which can be used to access internet and allow the communication over the internet. Public IP will be assigned by amazon and it is dynamic. If you stop and start your EC2 instance, The public IP will change.

Elastic IP(EIP):

Elastic IP is a kind of Fixed Public IP address which we can attach to our Instances. Elastic IP will not change if we stop & Start our EC2 instances. We need to request EIP from amazon and it will be free if we attach to any instances, if you keep this EIP unused in your account then it will be charged after initial 1st hour.

Private IP:

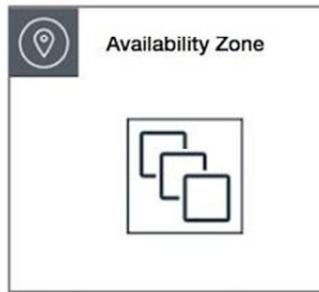
Private IP can be used to establish the communication with in the same network only, Private (internal) addresses are not routed on the Internet and no traffic can be sent to them from the Internet, Means no internet access will be available over private address.

Placement group: is a logical grouping of instances with in a single availability zone. AWS provides three types of placement groups

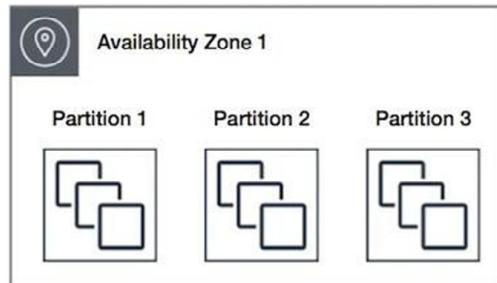
- Cluster
- Partition
- Spread

Cluster: A cluster placement group is a logical grouping of instances within a single Availability Zone. Instances in the same cluster placement group enjoy a higher per-flow throughput limit for TCP/IP traffic and are placed in the same high-bisection bandwidth segment of the network.

The following image shows instances that are placed into a cluster placement group.

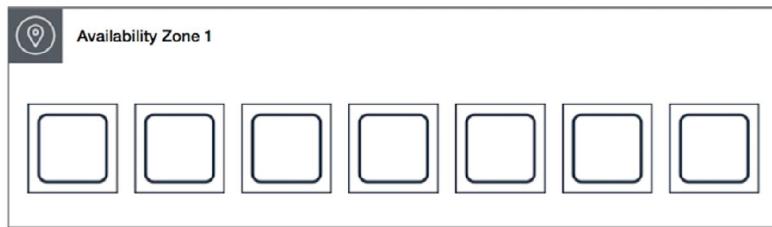


Partition Placement Group: Partition placement groups help reduce the likelihood of correlated hardware failures for your application. When using partition placement groups, Amazon EC2 divides each group into logical segments called partitions. Amazon EC2 ensures that each partition within a placement group has its own set of racks. Each rack has its own network and power source. No two partitions within a placement group share the same racks, allowing you to isolate the impact of hardware failure within your application.



Spread: A spread placement group is a group of instances that are each placed on distinct racks, with each rack having its own network and power source.

The following image shows seven instances in a single Availability Zone that are placed into a spread placement group. The seven instances are placed on seven different racks



Spread placement groups are recommended for applications that have a small number of critical instances that should be kept separate from each other.

A spread placement group can span multiple Availability Zones in the same Region. You can have a maximum of seven running instances per Availability Zone per group.

Root Volume: The storage which we used to install Operating system for instance is called as root volume (Ex: C:\ Drive). The following volume types are supported as root volumes: General purpose SSD, Provisioned IOPS SSD, Magnetic.

Security Group: A Security group acts as a virtual firewall for your instance to control incoming & Outgoing traffic. Security groups to be attached and we can attach 5 security groups to each instance.

Following are the basic characteristics of a security group:

- You can specify allow rules, but not deny rules
- You can specify separate rules for inbound and outbound traffic.
- Security group rules enable you to filter traffic based on protocols and port numbers.
- Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules.



- When you create a new security group, it has no inbound rules.
- By default, a security group includes an outbound rule that allows all outbound traffic.
- By default we can create 2500 security groups per region, can be increased upto 5000 per region
- 60 inbound and outbound rules per security group

KeyPair: Key pair is a combination of public key and private key which can be used to encrypt and decrypt the data, is a set of security credentials that you use to prove your identity when connecting to an instance.

Amazon EC2 stores the public key and user stores the private key.

Elastic Compute Cloud (EC2) Lab:

Prerequisite:

- Amazon Account access
- Putty & puttygen tools on your computer

To-do List 1:

1. Launch Windows Server 2016 EC2 instance in N.Virginia Region
 - While creating EC2 instance open RDP port in the security group from your IP only
 - Decrypt the password using keystore file which we used while creating the EC2 instance
 - Access Windows server using Remote desktop connection tool
2. Launch Amazon Linux2 EC2 instance in N.Virginia Region
 - While creating EC2 instance open SSH port in the security group from anywhere
 - Generate Private key using puttygen tool from the keypair which we used while creating the EC2 instance
 - Access Amazon Linux EC2 instance using putty tool
3. Install webserver on Amazon Linux2 EC2 and host a website
4. Create Custom AMI from the Amazon Linux EC2 in which we hosted the website.
5. Launch New EC2 instance from the custom AMI in N.Virginia Region
6. Copy the AMI from N.Virginia to Mumbai Region
7. Launch New Instance from Custom AMI in Mumbai Region
8. Share custom AMI with a specific Amazon Account & launch New EC2 instance in the otheramazon account
9. Share custom AMI with public



VIRTUAL PRIVATE CLOUD



AWS's Virtual Private Cloud (VPC) is one of its services. A VPC is a private cloud computing environment contained within a public cloud. The VPC is a virtually isolated environment made to provide a private environment according to the needs of IT companies and business requirements.

By default, you can create up to 5 VPCs. You can ask for additional VPCs using the VPC Request Limit Increase form.

An IT company hosts its products and services on servers for customers to see. They make sure no one has access to their databases or their internal codebase. That's why IT companies isolate their databases, CRM information, and internal code bases from the customers.

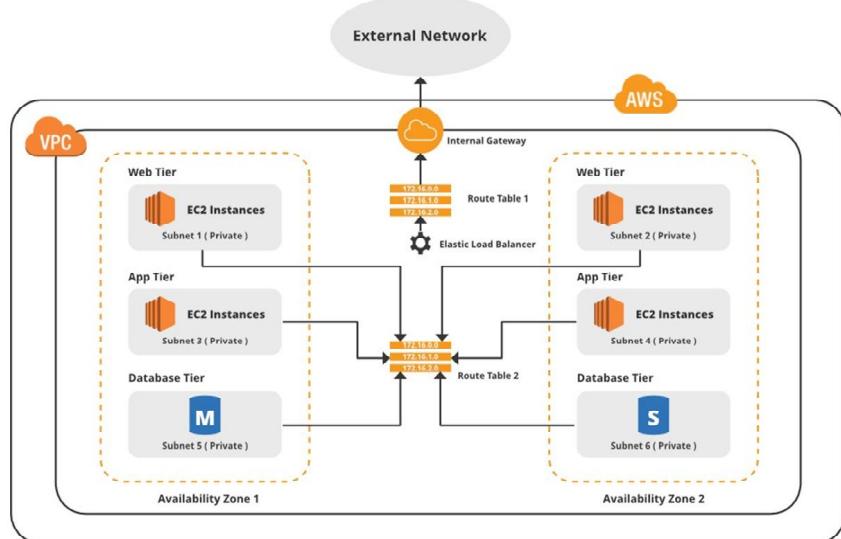
The Virtual Private Cloud consists of the following features:

- **Subnets:** A subnet is a range of IP addresses in your VPC. Subnets consist of instances which are servers, DBMSs, and CRMs. A subnet must reside in a single Availability Zone. After you add subnets, you can launch AWS resources into a specified subnet.
- **IP Addressing:** You can assign IPv4 addresses and IPv6 addresses to your VPCs and subnets. You can also bring your public IPv4 and IPv6 addresses to AWS and allocates them to resources in your VPC, such as EC2 instances, NAT gateways, and Network Load Balancers.
- **Routing:** The Route tables decide where the traffic should go. Inside subnets, the resources like instances are all connected because of the local routing table which is set already while launching instances in the subnets.
- **Gateways:** As the name suggests the gateways are meant to connect to the outside world. In this case, the AWS internet gateway will connect you to other networks like public networks, internet, and other VPCs.



- **NAT gateway:** NAT Gateways allow private subnets to connect to the Internet but it works only one way. Instances in a private subnet can connect to the network or services outside of their VPCs but external services can't initiate the network connection with those instances.
- **Endpoints:** A VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.
- **VPN connections:** Connect your VPCs to your on-premises networks using AWS Virtual Private Network (AWS VPN).
- **Peering connections:** The VPCs have the ability to connect to other VPCs using VPC peering. In this one, VPCs in certain regions can connect to another VPC that is located in other regions.

Let's see how the VPC network looks:



The description of this figure:

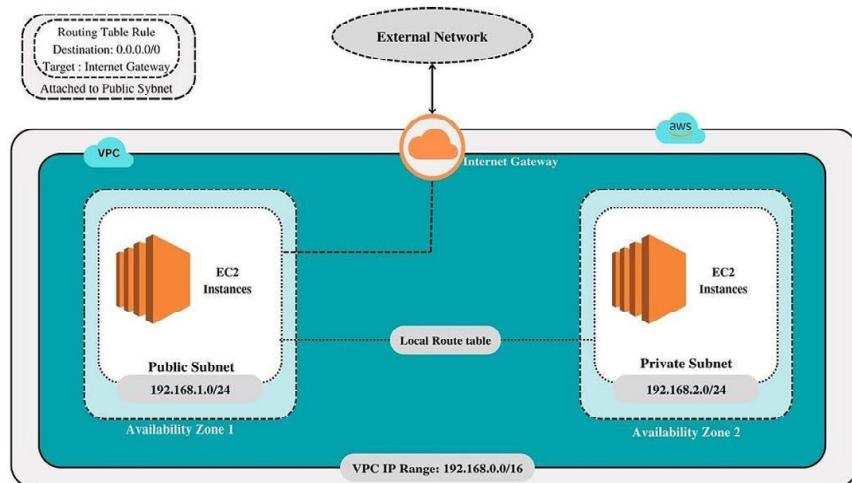
- There are two stacks of resources created in one VPC. By the looks of the figure, it is a 3-tier infrastructure because the VPC consists of a Database tier, an application tier that processes the internal codebase, and the last one is Web tier which is a presentation tier shown to the clients or Customers.
- All resources are kept private because of the security regions. We don't want our databases to be compromised by some hacker, because databases keep the most crucial data important to the customer like credit card info, IDs, and so on.



- There are two Routing tables, one routing table is the default routing table and is responsible for interconnection between subnets. And another routing table is for routing the subnet to the internet gateway which leads to the External networks.

Here are the steps for setting up a VPC in the AWS environment:

I have my own diagram to create the structure of the VPC:



So, we are going to implement above same VPC Structure. Let's start with AWS VPC services.

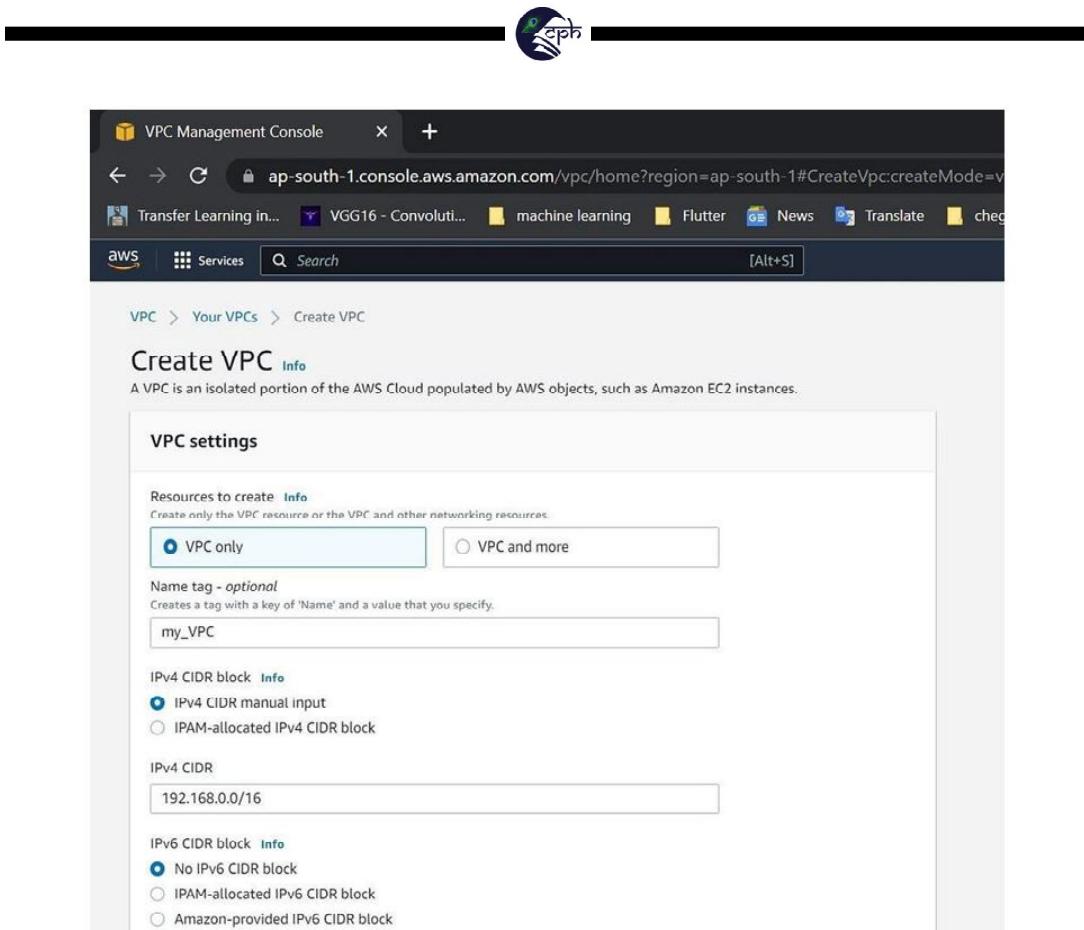
Virtual Private Cloud:

- First Create a VPC, and name it whatever you want. I name It as “my_VPC”.
- Give “IPv4 CIDR” as “192.168.0.0/16”. “CIDR” mean “Classless inter-domain Routing”. It is decided based on Netmask. Let me give you some examples:-

If the netmask is 255.255.255.0 and IP is 192.168.55.0, then IP ranges from 192.168.55.0 - 192.168.55.255 means around 256 instances can be allocated with IPs but since some IPs are reserved by other resources, Elastic Compute Engine (EC2) only can be allocated 251 IPs.

If the netmask is 255.255.0.0 and IP is 192.168.0.0, then IP ranges from 192.168.0.0 to 192.168.255.255 which means around 65,536 instances can be allocated with IPs. In this case, CIDR will be written like 192.168.0.0/16.

If the netmask is 255.0.0.0 and IP is 192.0.0.0 then CIDR will be like 192.0.0.0/8.



Subnets: Create two subnets “Public Subnet” and “Private Subnet”.

(a) Public Subnet:

- Choose “VPC ID” as “my_VPC”. Name the Subnet as “Public Subnet” and choose “Availability Zone (AZ)” as “ap-south-1a”. Since we are using Region Mumbai, there are three AZ, you can choose anyone but the next time you choose, your choice should be a different region So that your public subnet and private subnet can be isolated.
- Give “IPv4 CIDR block” as “192.168.1.0/24” and create it.

(b) Private Subnet:

Choose “VPC ID” as “my_VPC”. Name the Subnet as “Private Subnet” and choose “Availability Zone (AZ)” as “ap-south-1b”. Give “IPv4 CIDR block” as “192.168.2.0/24” and create it.

VPC ID: vpc-019ef1007703d92dc (my_VPC)

Associated VPC CIDRs: 192.168.0.0/16

Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name: Private Subnet

The name can be up to 256 characters long.

Availability Zone: Asia Pacific (Mumbai) / ap-south-1b

IPv4 CIDR block: 192.168.2.0/24

Internet Gateways: Let's create an Internet gateway.

Give name as “my_internet_gateway”. And Attach it to “my_VPC”.

Name	Internet gateway ID	State	VPC ID
my_internet_gateway	igw-05b7457cfe1a124bd	Detached	-
	igw-0ff47caebe538145a7	Attached	vpc-0a9f88fb206a5274

Details

Internet gateway ID: igw-05b7457cfe1a124bd

State: Detached

VPC ID: vpc-0a9f88fb206a5274

Owner: 964307678829



Route Tables:

- Create a new route table and name it “my_route_table”.
- Give VPC as “myVPC” and create it.

A route table specifies how packets are forwarded between the subnets within your VPC, the Internet, and your VPN connection.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.
my_route_table

VPC
The VPC to use for this route table.
vpc-019ef1007705d92dc (my_VPC)

3. **Edit route:** Select Edit routes and Add route respectively. Set Destination as 0.0.0.0/0 (means public network) and Target as internet gateway (my_internet_gateway).

The Route table we set up, allows the server to go to the internet. Here we set up the destination as 0.0.0.0/0 which means when instances are alive, they can go to the internet. The internet gateway is a router that leads to the internet. All the subnets first pass through the internet gateway to be able to connect to the internet.

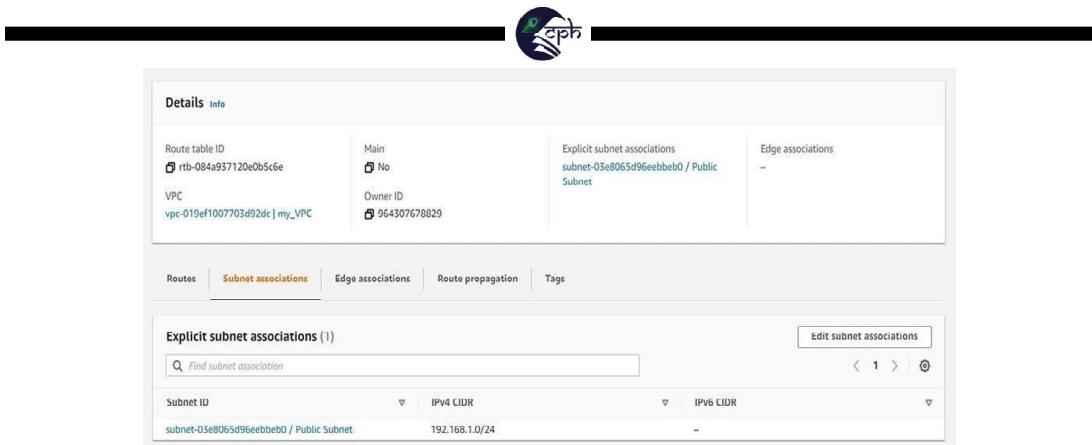
Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	igw-0567457fc1e124bd	-	No

Add route

CANCEL PREVIEW Save changes

Here is the first destination set to 192.168.0.0/16 which is the default that makes sure that all the resources in the VPC have the local connectivity

4. **Subnet association:** This is where we tell the VPC which subnet we want to attach to the route table and that will lead to the public internet. In this, we have to select the subnet we want to associate with the internet gateway.
 - (a) Click on Edit subnet associations and select “Public Subnet”, and save the subnet association.



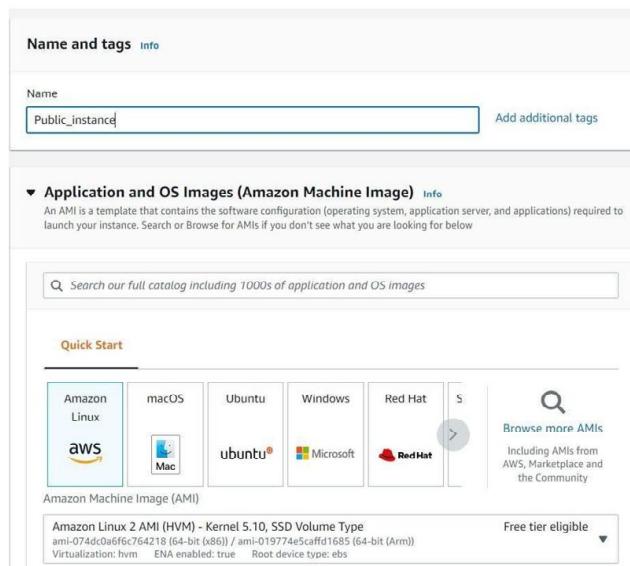
Now all the main VPC settings have been done. Let's test it now and launch an instance.

We will launch two instances. Instead of default VPC, we will use my_VPC to launch instances. The first instance will use a Public subnet and another instance will use a Private Subnet.

Launching Test Instances:

Public Instance:

1. This instance will be launched with Public Subnet. Name the instance as Publicinstance. Choose the AMI as Amazon Linux.



2. Instance Type as t2.micro (for just testing). Provide key-pair.
3. Edit the Network settings, Choose VPC as my_VPC. Choose Subnet as Public Subnet and Auto-assign public IP should be enabled.

Network settings

VPC - required Info
vpc-019ef1007703d92dc (my_VPC)
192.168.0.0/16

Subnet Info
subnet-03e8065d96ebbeb0
VPC: vpc-019ef1007703d92dc Owner: 964307678829 Availability Zone: ap-south-1a IP addresses available: 250 CIDR: 192.168.1.0/24

Auto-assign public IP Info
Enable

4. Change the Security Group rule, allow it to All traffic, and the Storage setting will be as it is.

Security group name - required
launch-wizard-4

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
launch-wizard-4 created 2022-11-22T15:47:07.778Z

Inbound security groups rules

▼ Security group rule 1 (All, All, 0.0.0.0/0)

Type Info Protocol Info Port range Info
All traffic All All

Source type Info Source Info Description - optional Info
Anywhere Add CIDR, prefix list or security e.g. SSH for admin desktop
0.0.0.0/0

5. Now launch it successfully.

Private Instance:

- Do the same as you did in the Public Instance. Name the Private instance as Private_instance and choose the subnet as Private Subnet.
- Here if you Enable Auto-assign Public IP, then there is no meaning of assigning a Public IP, because Private_instance is isolated. Keep the Public IP disabled.

Network settings

VPC - required Info
vpc-019ef1007703d92dc (my_VPC)
192.168.0.0/16

Subnet Info
subnet-0cb8d99a7e63838f
VPC: vpc-019ef1007703d92dc Owner: 964307678829 Availability Zone: ap-south-1b IP addresses available: 251 CIDR: 192.168.2.0/24

Auto-assign public IP Info
Disable



Now Launch it successfully.

Now Let's test the launched Instances. First Connect the Public_instance since Public Instance is attached to the Route table and the Route table has a route for the internet gateway to go to 0.0.0.0/0 (public network). We will able to use ssh protocol to connect to the instance through the internet.

If we ping through Public_instance to 8.8.8.8, it will work.

```
Last login: Tue Nov 22 16:04:09 2022 from ec2-13-233-177-5.ap-south-1.compute.amazonaws.com
[ec2-user@ip-192-168-1-60 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=1.33 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=1.40 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=51 time=1.41 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=51 time=1.36 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=51 time=1.36 ms
[ec2-user@ip-192-168-1-60 ~]$
```

i-07d4a5d49c4df17fa (Public_instance)
PublicIPs: 65.1.1.16 PrivateIPs: 192.168.1.60

Let's test the Privateinstance. if you try to connect it using ssh protocol, it won't work. In this case, we don't have public IP, so here it won't work.

When you connect your Public_instance, and through Public_instance you try to ping to the Private_instance's Private IP address, It will be able to ping because within the VPC there is local connectivity between the instance.

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Public_instance	i-0da5a5d49c4df17fa	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	-
Private_Instance	i-0bf55b073743eed56	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1b	-

Instance: i-0bf55b073743eed56 (Private_instance)

Details	Security	Networking	Storage	Status Checks	Monitoring	Tags
Instance ID i-0bf55b073743eed56 (Private_instance)	Public IPv4 address -	Private IPv4 addresses 192.168.2.11	Public IPv4 DNS -			
IPv6 address -	Instance state Running	Hostname type IP name: ip-192-168-2-11.ap-south-1.compute.internal	Public IPv4 DNS -			



```
Last login: Tue Nov 22 17:19:45 2022 from ec2-13-233-177-4.ap-south-1.compute.amazonaws.com
[ec2-user@ip-192-168-1-60 ~]$ [ec2-user@ip-192-168-1-60 ~]$ [ec2-user@ip-192-168-1-60 ~]$ [ec2-user@ip-192-168-1-60 ~]$ ping 192.168.2.11
PING 192.168.2.11 (192.168.2.11) 56(84) bytes of data.
64 bytes from 192.168.2.11: icmp_seq=1 ttl=255 time=0.778 ms
64 bytes from 192.168.2.11: icmp_seq=2 ttl=255 time=1.07 ms
64 bytes from 192.168.2.11: icmp_seq=3 ttl=255 time=0.785 ms
64 bytes from 192.168.2.11: icmp_seq=4 ttl=255 time=0.811 ms
```

i-07d4a5d49c4df17fa (Public_instance)
Public IPs: 65.1.1.16 Private IPs: 192.168.1.60

Whenever there is a need to access the private_instance, we can use a public_instance to connect with it. just see below:

```
[root@ip-192-168-1-60 ~]# chmod 400 my.key
[root@ip-192-168-1-60 ~]# ssh -l ec2-user -i my.key 192.168.2.11
[ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$ [ec2-user@ip-192-168-2-11 ~]$
```

So That's it, we have created our VPC in AWS.

VPC Peering across Two Region

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud.

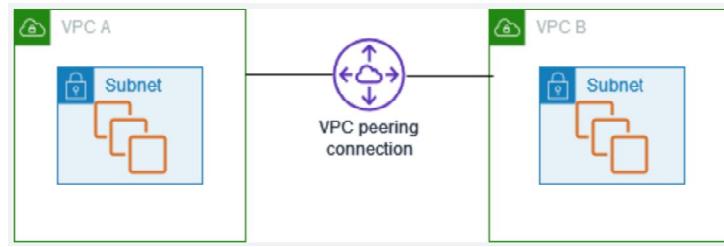
A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.

For example, if you have more than one AWS account, you can peer the VPCs across those accounts to create a file sharing network. You can also use a VPC peering connection to allow



other VPCs to access resources you have in one of your VPCs. When you establish peering relationships between VPCs across different AWS Regions, resources in the VPCs (for example, EC2 instances and Lambda functions) in different AWS Regions can communicate with each other using private IP addresses, without using a gateway, VPN connection, or network appliance.

Pricing for a VPC peering connection There is no charge to create a VPC peering connection. All data transfer over a VPC Peering connection that stays within an Availability Zone (AZ) is free. Charges apply for data transfer over a VPC Peering connections that cross Availability Zones and Regions



Creating two VPC in two different regions.and then established a connection between them and then create two EC2 and test their connectivity in both the VPC

Region 1- Virginia

* Create VPC in one region (Virginia region)

The screenshot shows the 'Create VPC' configuration page in the AWS Management Console. The 'VPC settings' section includes:

- Resources to create:** VPC only (selected)
- Name tag - optional:** VPC-1
- IPv4 CIDR block:** 10.0.0.0/16
- IPv6 CIDR block:** No IPv6 CIDR block selected
- Tenancy:** Default
- Tags:** Key: Name, Value: VPC-1



* Create subnet -1

Create subnet associated with VPC-1

2. Enter Details and click on create button

VPC
Create subnets in this VPC:
vpc-0cf9bdc5d0f0912e1 | VPC-1
Associated VPC CIDR
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1
Subnet name
Create a tag with a key of 'Name' and a value that you specify.
Subnet-1

Availability Zone info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
us-east-1a (Virginia) / us-east-1a

IPv4 CIDR block info
10.0.0.0/25

Tags - optional
Key Name Value - optional
Q. Name Q. Subnet-1 Remove

Subnets (1) info
Filter subnets
Subnet ID: subnet-0edbec569b86c92f6 Clear filters

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
Subnet-1	subnet-0edbec569b86c92f6	Available	vpc-0cf9bdc5d0f0912e1 VPC-1	10.0.0.0/25	-

Create subnet

* Create Internet gateway

1. Click on Internet Gateway
2. Enter name and click on create

VPC > Internet gateways > Create internet gateway

Create internet gateway info
An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Creates a tag with a key of 'Name' and a value that you specify.
igw-1

Tags - optional
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 Name Value - optional
Q. Name Q. igw-1 Remove

Add new tag
You can add up to 40 more tags

Create internet gateway

2. Select created internet gateway and attach to VPC-1



* Create Route table

1. Now create Route table and select VPC-1

VPC > Route tables > Create route table

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

Tags
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	Value - optional
<input type="text" value="Name"/>	<input type="text" value="Route-1"/>

Add new tag
You can add 49 more tags.

Cancel Create route table

- b. Click on Routes section and enter internet gateway at 0.0.0.0/0

VPC > Route tables > rtb-094ff0666554aaefc1 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	<input type="text" value="local"/>	<small>Active</small>	No
0.0.0.0/0	<input type="text" value="igw-095e70349474ab8d"/>	-	No

Add route Cancel Preview Save changes

- c. Associate subnet-1 into subnet association section as follows

Route-1 rtb-094ff0666554aaefc1 - - No vpc-0cf9bbc5d0f0812e1 | VPC-1 77927041369

rtb-094ff0666554aaefc1 / Route-1

Details Routes Subnet associations Edge associations Route propagation Tags

Explicit subnet associations (0)
Filter subnet association

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR

No explicit associations.
You do not have any subnet associations.

Edit subnet associations

Subnets without explicit associations (1)
Filter subnet associations

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
Subnet-1	subnet-0ebc569085c92f6	10.0.0.0/25	-	Main(rt-04ef945a53114001a)

Edit subnet associations

Available subnets (1/1)
Filter subnet associations

Selected subnets
subnet-0ebc569085c92f6 / Subnet-1 X

Cancel Save associations



* Creating EC2 Instance in Virginia region

Go to EC2 dashboard and click on launch instance

2. Edit Network setting as VPC= VPC-1
subnet = subnet-1
Auto assign public ip =enable
3. In security group SSH ,HTTP ,HTTPs should be selected
4. Click On launch Instance



5. Click on instance and edit inbound rule in security group

The screenshot shows the AWS EC2 Instances console. A specific instance, i-0473a097cccf6283 (Instance-virginia), is selected. The 'Security' tab is active. Under the 'Inbound rules' section, there is one rule listed:

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-0e26409c9c46823	22	TCP	0.0.0.0/0	launch-wizard-13

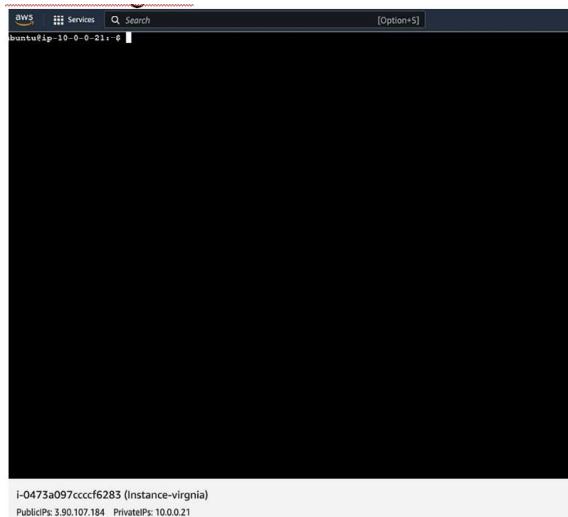
4. Allow All ICMP -IPv4 i.e ICMP protocol

The screenshot shows the AWS EC2 Security Groups console. It is navigating to the 'Edit inbound rules' page for the security group sg-0e26409c9c46823 (launch-wizard-13). The 'Inbound rules' table shows two rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0e26409c9c46823	SSH	TCP	22	Custom	Q 0.0.0.0 X
-	All ICMP - IPv4	ICMP	All	Anywhere...	Q 0.0.0.0 X

At the bottom right, there are 'Cancel', 'Preview changes', and 'Save rules' buttons.

6. connect to instance-virginia





Part II

Region 2 - Mumbai

* Create VPC in another region (Mumbai region)

The screenshot shows the AWS VPC dashboard with one VPC listed and the 'Create VPC' dialog box open.

VPC dashboard:

- Your VPCs (1) Info
- Name: `vpc-03a33fe0f6ec4d6`
- State: Available
- IPv4 CIDR: `172.31.0.0/16`
- DHCP option set: `dopt-056b3c3119c3e...`

Create VPC Dialog:

- Resources to create:** VPC only VPC and more
- Name tag - optional:** Name: `VPC-2`
- IPv4 CIDR block:** IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block Amazon-provided IPv4 CIDR block IPv6 CIDR
- IPv4 CIDR:** `192.168.0.0/16`
- IPv4 CIDR block info:** No IPv4 CIDR block IPAM-allocated IPv4 CIDR block Amazon-provided IPv4 CIDR block IPv6 CIDR owned by me
- Tenancy:** Default
- Tags:** Key: `Name`, Value: `VPC-2`

* Create subnet-2

Create subnet associated with VPC-2

2. Enter Details and click on create button

The screenshot shows the 'Subnet settings' section of the VPC creation dialog.

VPC: Associated VPC CIDRs: `192.168.0.0/16`

Subnet settings: Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1:

- Subnet name:** Name: `subnet-2`
- Availability Zone:** `Asia Pacific (Mumbai) / ap-south-1a`
- IPv4 CIDR block:** `192.168.10.0/26`
- Tags - optional:** Key: `Name`, Value: `subnet-2`



* Create Internet gateway

1. Click on Internet Gateway
2. Enter name and click on create

VPC > Internet gateways > Create internet gateway

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings	
Name tag	Creates a tag with a key of 'Name' and a value that you specify.
<input type="text" value="lgw-2"/>	
Tags - optional	
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	Value - optional
<input type="text" value="Name"/>	<input type="text" value="lgw-2"/> <input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>	
You can add 49 more tags.	
<input type="button" value="Cancel"/> <input type="button" value="Create internet gateway"/>	

3. Select created internet gateway and attach to VPC-2

VPC > Internet gateways

Internet gateways (1/2) Info

Name	Internet gateway ID	State	VPC ID	Action
-	igw-0093b6da4a8d6cfb74	Attached	vpc-03a33fc6f6c3ec4d6	<input type="button" value="View details"/> <input type="button" value="Attach to VPC"/> <input type="button" value="Detach from VPC"/> <input type="button" value="Manage tags"/> <input type="button" value="Delete internet gateway"/>
Igw-2	igw-0a61caf4420d9da4e	Detached	-	

VPC > Internet gateways > Attach to VPC (igw-0a61caf4420d9da4e)

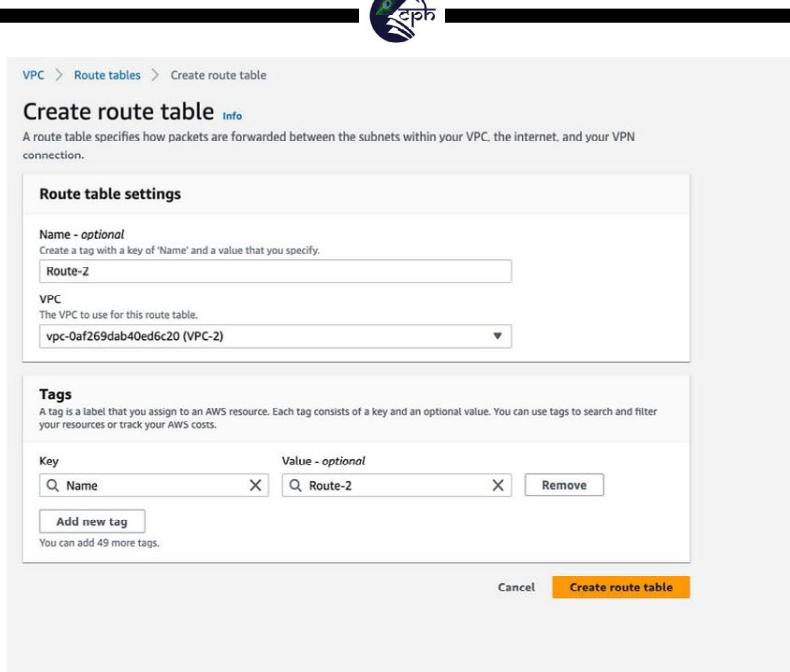
Attach to VPC (igw-0a61caf4420d9da4e) Info

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.
<input type="text" value="vpc-0af269dab40ed6c20"/> <input type="button" value="Search"/>
<input type="button" value="AWS Command Line Interface command"/>
<input type="button" value="Cancel"/> <input type="button" value="Attach internet gateway"/>

* Create Route table

- a. creates Route table and select VPC-2



- b. Click on Routes section and enter internet gateway at 0.0.0.0/0

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	igw-0x10ef4420d9de4c		No

- c. Associate public subnet into subnet association section as follows

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
Subnet-2	subnet-002917dbdf53e152c	192.168.0.0/25	-	Main (rtb-0e0679d51dab58fa0)



* Creating EC2 Instance in Mumbai region

Go to EC2 dashboard and click on launch instance

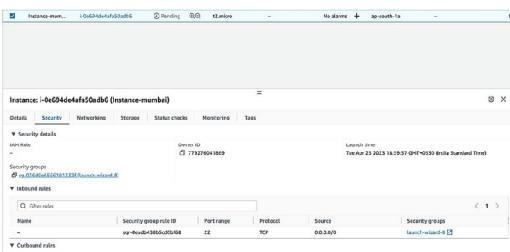
The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' step, the instance name is set to 'Instance-Mumbai'. In the 'Application and OS Images (Amazon Machine Image)' step, the selected AMI is 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type'. The 'Free tier eligible' badge is visible. On the right, the 'Summary' pane shows 1 instance, the software image as Canonical, Ubuntu, 22.04 LTS, and the instance type as t2.micro. A tooltip for the free tier is displayed.

2. Edit Network setting as VPC= VPC-2
subnet = subnet-2
Auto assign public ip =enable
3. In security group SSH ,HTTP ,HTTPs should be selected
4. Click on launch Instance

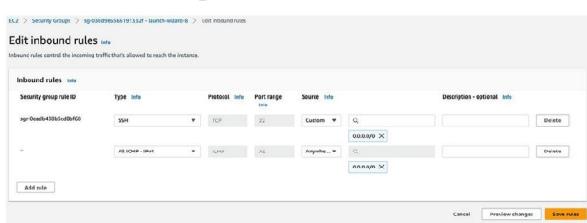
The screenshot shows the 'Launch an instance' wizard continuing through the 'Key pair (login)', 'Network settings', and 'Security group' steps. In the 'Key pair (login)' step, a new key pair is selected. In the 'Network settings' step, the VPC is set to 'vpc-0f2661a0d0e6c20 (VPC-2)', the subnet to 'subnet-095567442156970ee', and the auto-assign public IP is enabled. In the 'Security group' step, a new security group is being created with the name 'Security-wizard-8'. The 'Inbound security groups rules' section shows a rule for port 22. The 'Launch instance' button is highlighted.



5. Click on instance and edit inbound rule in security group



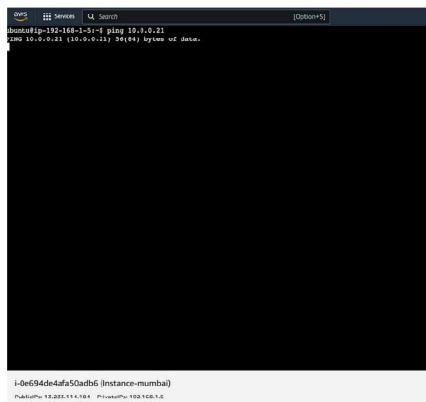
6. Allow All ICMP -IPv4 i.e ICMP protocol



7. Connect to instance-Mumbai and use command

Command: ping <private ip of Virginia region instance>

Result : As both instances are in different VPC ,they will not connect with each other



Solution for connecting Instances with Different VPC is VPC Peering

VPC Peering

- Click on peering connection
- Enter name
- Select Vpc -1 to peer with
- Select Account or Region as per your requirement

Here both VPC are in different region but with same account so select “My Account “ and “Another region”



5. Select Region name and Paste VPC ID (refer next point) of Mumbai region VPC

The screenshot shows the 'Create peering connection' wizard. In the 'Peering connection settings' step, a local VPC named 'vpc-1-2-vpc' is selected. In the 'Select another VPC to peer with' section, the 'Region' dropdown is set to 'Another Region', and a VPC named 'Asia Pacific (Mumbai) (ap-south-1)' is selected. The 'VPC ID (Acceptor)' field contains the value 'vpc-0af269dab40ed6c20'.

6. Go to Mumbai region -> Go to Vpc -> Select VPC-2 ->Copy VPC-ID and paste in above info

Name	VPC ID	Status	IP & CIDR	IP & CIDR
vpc-2	vpc-0af269dab40ed6c20	Available	10.0.0.0/16	172.31.0.0/16

7. Click on create peering connection button

The screenshot shows the 'Create peering connection' wizard in the 'Details' step. It displays the same configuration as the previous screenshot, including the local VPC 'vpc-1-2-vpc' and the remote VPC 'vpc-0af269dab40ed6c20'. At the bottom, the 'Create peering connection' button is highlighted in orange.



8. Peering connection get created

The screenshot shows the 'Peering connections' list in the AWS VPC console. There is one entry:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC
pxc-064834f51ca9a9a60	pxc-064834f51ca9a9a60	Pending acceptance	vpc-0cf9bbc5d0f0812e1	vpc-0af269db40ed6c20 / VPC-2

9. Now Go to Mumbai region and then in peering connection

10. Click on actions

11. Select “Accept request”

The screenshot shows the 'Peering connections' list. For the pending connection, an 'Actions' dropdown menu is open, and the 'Accept request' option is highlighted.

12. click on Accept request and peering connection is now active

The screenshot shows the 'Accept VPC peering connection request' dialog. It displays requester and accepter details, including CIDRs and regions. At the bottom right is a large orange 'Accept request' button.

* Edit Route Table

Go to Virginia region and Select Route-1

The screenshot shows the 'Route tables' list and the 'Routes' tab of the 'Route-1' details page. The table lists two routes:

Destination	Target	Status	Propagated
0.0.0.0/0	igw-056c170ad9474abf4	Active	No
10.0.0.0/16	local	Active	No



2. Click on Routes section and enter peering connection at 192.168.0.0/16 (VPC CIDR of vpc-2)

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-096c70a49474abf84	Active	No
192.168.0.0/16	pcc-064834f51ca9d9d60	-	No

*

-----*

Go to Mumbai region and Select Route-2

Name	Route table ID	Explicit subnet associat...	Main	VPC	Owner ID
rtd-01303e1c0653c776	rtd-01303e1c0653c776	-	Yes	vpc-00a33fd6fc3e4dd6	779276041868
Route-2	rtd-01f15cc0900d3225	subnet-093567d421569...	No	vpc-00f260aabb40e05c20 VP...	779276041869
rtd-0e057965160a5090	rtd-0e057965160a5090	-	Yes	vpc-00f260aabb40e05c20 VP...	779276041869

Details	Routes	Subnet associations	Edge associations	Route propagation	Tags
rtb-0fe91cd0f0fd3225 / Route-2					
Routes (2)					
Destination	Target	Status	Propagated		
0.0.0.0/0	igw-096c70a49474abf84	Active	No		
192.168.0.0/16	local	Active	No		

2. Click on Routes section and enter peering connection at 10.0.0.0/16 (VPC CIDR of vpc-1)

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-04f1af442059d46	Active	No
16.0.0.0/16	pcc-064834f51ca9d9d60	-	No

* Connect Both Instance and test their connectivity in both the VPC

1. Now Connect to Instance-Mumbai

Command: ping <private ip of Virginia region instance> Result:



```
AWG [ ] Review Q Search [Options] [ ] Manual [ ] Help [ ]
```

```
traceroute -l 10.0.0.21 -t 10 -N 9.9.22
PING 10.0.0.21 (10.0.0.21) 56(84) bytes of data.
4 bytes from 10.0.0.21: icmp_seq=1 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=2 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=3 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=4 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=5 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=6 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=7 ttl=64 time=192 ms
4 bytes from 10.0.0.21: icmp_seq=8 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=9 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=10 ttl=64 time=187 ms
4 bytes from 10.0.0.21: icmp_seq=11 ttl=64 time=187 ms
```

2. Now Connect to Instance-Virginia

Command : ping <private ip of Mumbai region instance> Result:

```
aws [[ services | Search | Options | S | ]]
```

N.Virginia - Sajana, machine

```
bash$ ifconfig -a | grep ens3  
ens3      Link encap:Ethernet HWaddr 00:0C:29:1A:6B:01  
          brd 00:0C:29:FF:FF:FF MTU:1500 Metric:1  
          RX packets:1066 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:1066 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:106600 (106.6 KB) TX bytes:106600 (106.6 KB)  
          Interrupt:15 Memory:00:0C:29:C0:6B:00  
  
bash$ ping 192.168.1.5  
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.  
4 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=4 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=5 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=6 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=7 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=8 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=9 ttl=64 time=188 ms  
4 bytes from 192.168.1.5: icmp_seq=10 ttl=64 time=188 ms
```



AWS TRANSIT GATEWAY

Introduction:

AWS Transit Gateway is a service that allows customers to connect their Amazon Virtual Private Clouds (VPCs) and on-premises networks to a central hub. This simplifies network management by providing a single gateway to manage network connectivity between multiple VPCs and on-premises networks.

Components of AWS Transit Gateway

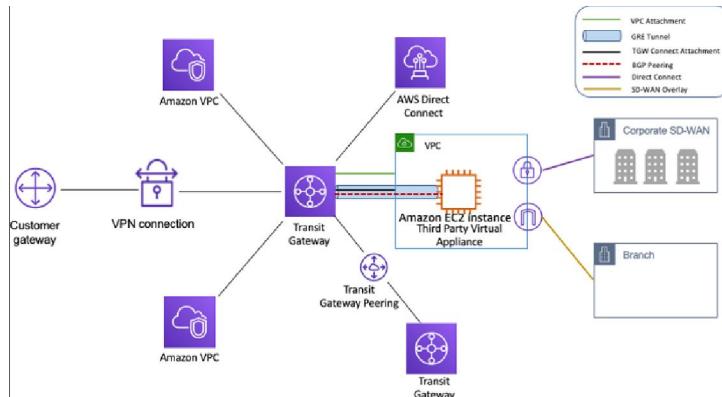
AWS Transit Gateway consists of the following components:

- **Transit Gateway:** This is the central hub that connects multiple VPCs and on-premises networks. Transit Gateway acts as a transit point for traffic between these networks.
- **Attachments:** Attachments are the connections between the Transit Gateway and VPCs or on-premises networks. VPC attachments are created by attaching a VPC to the Transit Gateway, while VPN and Direct Connect attachments are created by creating a VPN or Direct Connect connection and attaching it to the Transit Gateway.
- **Route tables:** Each attachment has its own route table, which specifies how traffic should be routed between the attachment and other attachments.

Benefits of AWS Transit Gateway

AWS Transit Gateway offers several benefits, including:

- **Simplified network management:** With Transit Gateway, customers can manage network connectivity between multiple VPCs and on-premises networks from a single gateway, reducing the need for complex networking configurations and enabling easier network management.
- **Scalability:** Transit Gateway is designed to scale as the number of attached VPCs and on-premises networks grows, making it easier to accommodate growing network traffic.
- **Improved performance:** Transit Gateway uses a highly available architecture and is designed for low-latency connections, enabling fast and reliable network performance.





Difference between AWS Transit Gateway and VPC Peering

While AWS Transit Gateway and VPC peering, both provide connectivity between VPCs, there are several key differences between the two services:

- **Scalability:** VPC peering is limited to a single VPC, while Transit Gateway can connect multiple VPCs and on-premises networks.
- **Simplified network management:** Transit Gateway provides a centralized hub for managing network connectivity between multiple VPCs and on-premises networks, while VPC peering requires customers to manage each VPC peering connection separately.
- **Routing flexibility:** Transit Gateway allows for more complex routing configurations, while VPC peering has more limited routing capabilities.

What is AWS Transit Gateway?

AWS Transit Gateway is a service that acts as a hub to connect VPCs and on-premises networks. It acts as a central routing engine that eliminates the need for each VPC to have individual connections between them.

With Transit Gateway, you only need to create connections from the VPCs, VPNs, and Direct Connect links to the Transit Gateway. Transit Gateway will then dynamically route traffic between all the connected networks.

Benefits of Using Transit Gateway

Some of the main benefits of using Transit Gateway include:

- **Simplified network topology:** No need for mesh network between Centralized connectivity configuration and no need to manage routing tables between VPCs VPCs. Just connect each VPC to the Transit Gateway
- **Scalability:** Easily scale up to tens of thousands of VPC and remote office connections.

Reduced operational complexity

- **Shared network transit:** Allows different accounts and VPCs to use the same Transit Gateway.
- **On-premises connectivity:** Connect seamlessly to on-premises data centers.

Transit Gateway Use Cases

Transit Gateway is a versatile service that can cater to a variety of use cases:

- Connect VPCs across multiple accounts and AWS Regions.
- Create a hub-and-spoke model for segmented networks.
- Share centralized internet connectivity across accounts.
- Migrate from a mesh or hub-and-spoke model to a Transit Gateway.
- Connect remote offices and data centers to AWS.



Getting Started with Transit Gateway

To start using Transit Gateway, you need to perform the following steps:

Create a Transit Gateway in a specific region

To get started with Transit Gateway, the first step is to create the Transit Gateway resource in your desired AWS region.

When creating the Transit Gateway, you need to specify a name tag so it can be easily identified. You also have the option to enable DNS support if you need resolution between your connected networks.

Some key considerations when creating the Transit Gateway

Transit Gateways are regional resources. So, you need to decide which region makes the most sense as the connectivity hub for your use case.

By default, a new Transit Gateway will be created in the default VPC for the region. You can choose to create it in a custom VPC if required.

Select the appropriate size for your Transit Gateway based on the expected network traffic volume. Sizing can be adjusted later if needed.

You can enable sharing with other accounts upon creation or do it later. Account sharing allows connections from other accounts.

Logging can be enabled to track connection activity and events. The logs will be sent to CloudWatch Logs.

Once the Transit Gateway is created, you will get an ID for it that is needed to attach VPCs and other networks. It takes some time for the Transit Gateway to be ready for use after creation.

So those are some of the key options to consider when creating your Transit Gateway in the region of your choice. The console wizard will guide you through all the necessary configuration.

Attach VPCs by creating Transit Gateway attachments

Once the Transit Gateway is created, the next step is to start attaching VPCs. Each VPC that needs to connect to the Transit Gateway needs to have an attachment created.

Some key points when creating VPC attachments:

You can attach VPCs from the same account as the Transit Gateway or from other accounts if account sharing is enabled.

For each VPC attachment, you need to provide the ID of the Transit Gateway, the ID of the VPC, and the subnets to associate.

An attachment propagation setting determines whether routes get automatically propagated to the VPC route table. You can enable or disable propagation.

Option to enable DNS support for private IP addresses in the VPC to be accessible across networks.



You can control access to the VPC by adding a transit gateway route table and using resource attachments.

Creating an attachment adds an entry in the VPC route table with the Transit Gateway as the target.

Attachment creation takes time to complete. The VPC can start sending traffic to the Transit Gateway once the state changes to available.

You can create multiple attachments from the same VPC for redundancy and scaling.

The Transit Gateway provides connectivity between the VPCs as soon as the attachments are created and routes propagated. So, you can build out connectivity to more VPCs incrementally.

For on-premises connectivity, create VPN or Direct Connect attachments:

The Transit Gateway allows you to connect your on-premises networks and data centers using VPN or Direct Connect links.

For VPN connectivity, you need to create an AWS Site-to-Site VPN connection from your customer gateway router to the Transit Gateway. The customer gateway can be a physical device or a software appliance.

To create the VPN attachment:

Provide the Transit Gateway ID, customer gateway ID, VPN connection ID.

Configure the inside and outside IP addresses for the VPN tunnel. Specify the AWS side ASN for BGP routing.

Enable route propagation to exchange routes between the Transit Gateway and on-premises network.

For Direct Connect connectivity, you need to link your Direct Connect connection or LAG to the Transit Gateway.

To create the Direct Connect attachment:

Specify the Direct Connect connection ID and the Transit Gateway ID. Provide the inside and outside IP addresses.

Enable BGP for propagating routes.

Specify the ASNs for the AWS and customer side.

Enable route propagation.

The attachment creation process will take some time to complete. Once available, your on-premises network will be able to connect to the VPCs and networks attached to the Transit Gateway.

You can create multiple VPN or Direct Connect attachments for high availability and failover between your data center and Transit Gateway.



Configure route tables to define traffic flow between connections

Here are some more details on configuring route tables with AWS Transit Gateway to define traffic flow between the connected networks:

Transit Gateway uses route tables to determine how traffic should flow between the VPCs, VPNs, and Direct Connect attachments.

Some key points on configuring route tables:

By default, there is a default route table that allows full communication between all attachments and VPCs.

You can create additional, custom route tables that can selectively allow or deny traffic between resources.

Route tables can be associated with VPC or VPN/Direct Connect attachments to control which networks they can communicate with.

Each route table can have multiple route table associations and propagations.

Associations determine which attachments can route traffic using the route table. Propagations automatically add routes to the associated attachments.

Routes can be manually added, for example, to route traffic for a particular VPC subnet to an internet gateway.

You can create complex network segmentation policies by leveraging multiple route tables.

For example, you can create tiers like Public, Private, Restricted and assign VPC subnets to them via route tables.

Route priorities determine which route takes effect if there are multiple routes to a destination.

By leveraging custom route tables, you can dial in fined-grained control over how traffic flows between your connected networks using Transit Gateway.

Share the Transit Gateway with other accounts

Transit Gateways can be shared with other AWS accounts to allow inter-account connectivity. Here are some key points on sharing

When you create a Transit Gateway, you can enable sharing with other accounts. This allows accounts you authorize to attach their VPCs.

To enable sharing, you need to provide the account ID or organization ARN with which the Transit Gateway will be shared.

You can share the Transit Gateway only within the same AWS organization if you have enabled Resource Access Manager.

The owner account has full control over the Transit Gateway. Shared accounts have limited privileges.

Shared accounts can view and work only with their own VPC attachments and route tables.



To simplify management, shared accounts can be provided access via IAM to work with attachments and routes.

Sharing is transitive. If Account A shares the TGW with Account B, and B shares with C, then C can also use the TGW.

For security, enable VPC route table propagation sparingly for shared accounts.

Use RAM resources to allow sharing Transit Gateways across regions.

By sharing Transit Gateways, you can significantly simplify connectivity and reduce provisioning time across different accounts in your organization. But balance the convenience with appropriate access controls.

AWS provides an easy-to-use wizard in the console to guide you through the configuration process.

Transit Gateway:

A *transit gateway* is a network transit hub that you can use to interconnect your virtual private clouds (VPCs) and on-premises networks. As your cloud infrastructure expands globally, inter-Region peering connects transit gateways together using the AWS Global Infrastructure. All network traffic between AWS data centers is automatically encrypted at the physical layer.

The problem with VPC peering & Transit VPC

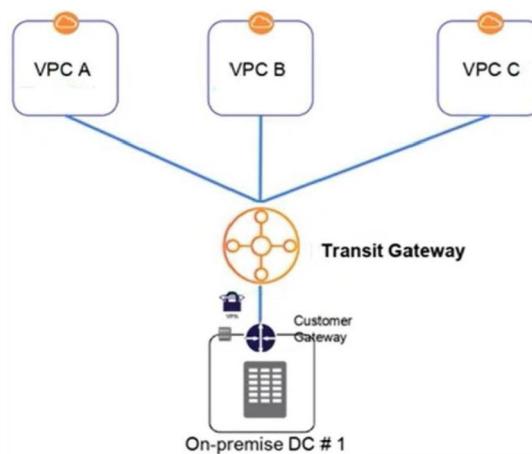
VPC Peering:

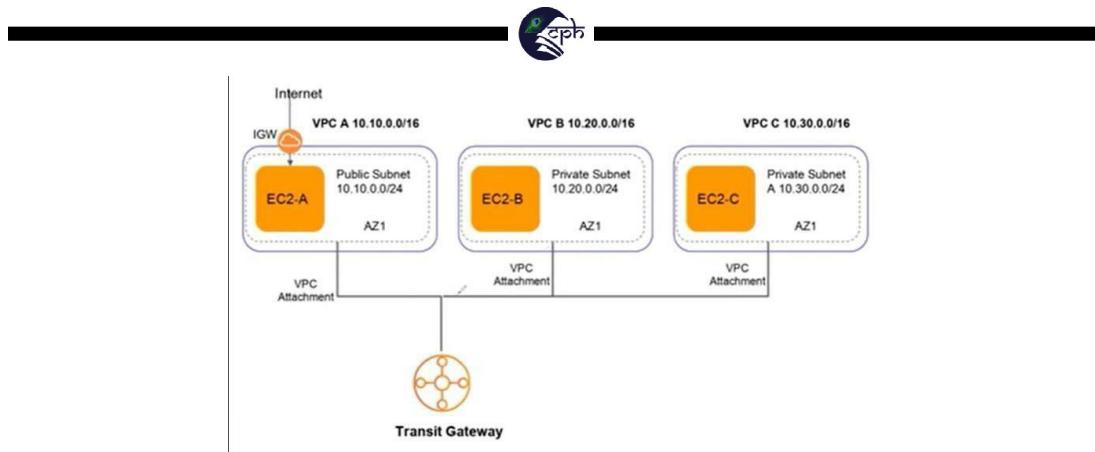
- Point-to-point connection between VPCs
- Non transitive traffic flow
- Separate connection for each VPC for on-premise VPN or Direct Connect

Transit VPC:

- Instance based (Cisco CSR 1000V)
- Additional EC2 cost
- Software Licensing cost
- Availability issues
- Bandwidth limitations of EC2

Here we're doing hands-on based on the below diagrams:





Select VPC from AWS console and Create VPC

Create VPC

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create: VPC only VPC and more

Name tag - optional: VPC A

IPv4 CIDR block: IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block
CIDR: 10.10.0.0/16

IPv6 CIDR block: No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block IPv6 CIDR owned by me

Tenancy: Default

Then click on Create VPC.

In a similar way, create other 2 CPCs naming VPC B and VPC C with 10.20.0.0/16 . 10.30.0.0/16 as IP addresses respectively.

Your VPCs (4)								
Name		VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table	Main netw
Default	vpc-0901e48909247000	Available	172.31.0.0/16	-	-	dept-029031ef655423fb...	rtd-05f161c310bc3a956	act-0f518b
VPC A	vpc-02777576ed57576	Available	10.10.0.0/16	-	-	dept-029031ef655423fb...	-	-
VPC B	vpc-001d451800988d7	Available	10.20.0.0/16	-	-	dept-029031ef655423fb...	-	-
VPC C	vpc-0119454bf19489	Available	10.30.0.0/16	-	-	dept-029031ef655423fb...	-	-

From the diagram , we can see VPC A is public and VPC B & C are private. So, we need to configure Internet Gateway for VPC A.

Click on Internet Gateway from the LHS panel and click on create Internet Gateway.

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Creates a tag with a key of 'Name' and a value that you specify.

Tags - optional
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	Value - optional
<input type="text" value="Name"/>	<input type="text" value="VPC-A-IGW"/>

Add new tag
You can add 49 more tags.

Create internet gateway

Now select the newly created IGW and select Attach to VPC from Actions

Name	Internet gateway ID	State	VPC ID	Owner
Default	igw-0290e449090f0db	Attached	vpc-0290e449090f0db	Default
VPC-A-IGW	igw-039bb0ae414457b2	Detached	-	598023471631

Select VPC A from drop down and click on Attach internet gateway

Internet gateways (1/2) Info

Actions View details Detach from VPC Manage tags Delete internet gateway

Filter internet gateways

Name	Internet gateway ID	State	VPC ID	Owner
Default	igw-0290e449090f0db	Attached	vpc-0290e449090f0db	Default
VPC-A-IGW	igw-039bb0ae414457b2	Detached	-	598023471631

Attach to VPC (igw-01443e5f8f3595425) Info

VPC
Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.

AWS Command Line Interface command

Attach internet gateway

Next, we need to create Subnets, for that click on Subnets from the LHS panel and click on create subnet for VPC A as per below

VPC

VPC ID
Create subnets in this VPC:

Associated VPC CIDRs
IPv4 CIDR blocks
10.10.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Name
Creates a tag with a key of 'Name' and a value that you specify.

Availability Zone
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 CIDR block

Tags - optional

Create subnet



Likewise , we have to create 3 subnets for each VPC. I have created with the info below:

VPC-A-Public-Subnet1 10.10.1.0/24

VPC-B-Private-Subnet1 10.20.1.0/24

VPC-C-Private-Subnet1 10.30.1.0/24

Now we need to add the route tables , Click on Route Tables from the LHS panel and click on create route table.

Route tables (4) Info							
Find resources by attribute or tag							
Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID	Actions
VPC-C-Route	rtb-009f4123d151a7f530	-	-	Yes	vpc-011a9343de191489 VPC C	598823471631	Edit Delete
VPC-B-Route	rtb-044b04ffcd4070038	-	-	Yes	vpc-002e8d51809e988d7 VPC B	598823471631	Edit Delete
Default-Route	rtb-05c161c1ccb365f6	-	-	Yes	vpc-0591e48909247ddc9 Default	598823471631	Edit Delete
VPC-A-Route	rtb-04b3d055165fd5c40	-	-	Yes	vpc-02c77757ccac5757d VPC A	598823471631	Edit Delete

Next, we have to associate the subnet with routing table. For that select VPC-A-Route -> Click on Subnet associations -> Edit subnet associations, then select VPC-A-Public-Subnet1 ->Save associations.

VPC > Route tables > rtb-04b3d055165fd5c40 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
VPC-A-Public-Subnet1	subnet-0852052e158953fe4	10.10.1.0/24	-	Main (rtb-04b3d055165fd5c40 VPC-A-Route)

Selected subnets

subnet-0852052e158953fe4 / VPC-A-Public-Subnet1

[Cancel](#) [Save associations](#)

Do the same for VPC-B-Route and VPC-C-Route.

Select the VPC-A-Route and go to Routes->Edit routes and add as per below, then click on save changes.

VPC > Route tables > rtb-04b3d055165fd5c40 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.10.0.0/16	local	Active	No
0.0.0.0/0	igw-0891bbae11e4c87a23	-	No

[Add route](#) [Preview](#) [Save changes](#)

We're all set in the VPC part , Now select EC2 from AWS console and create 2 instances as per above diagram.



EC2 > Instances > Launch an instance

Launch an instance Info

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.

Name and tags Info

Name Add additional tags

Application and OS Images (Amazon Machine Image) Info

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

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

Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Browse more AMIs Including AMIs from AWS Marketplace and the Community

Amazon Machine Image (AMI)

Configure Network Settings as below

Network settings Info

VPC - required Info
vpc-02c7757ccac5757d (VPC A)
10.10.0.0/16

Subnet Info
subnet-0837b57e158553be4 VPC: vpc-02c7757ccac5757d Owner: 598823471631 Availability Zone: us-east-2a IP addresses available: 251 CIDR: 10.10.1.0/24 Create new subnet

Auto-assign public IP Info
Enable

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
 Create security group Select existing security group

Security group name - required
VPC-A
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
VPC-A

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 105.203.73.60/32) Remove

Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>
ssh	TCP	22
Source type <small>Info</small>	Name <small>Info</small>	Description - optional <small>Info</small>
My IP	Add CIDR, prefix list or security	e.g. SSH for admin desktop
	105.203.73.60/32	X

And launch the instance.



Now create VPC-B-Private as below

VPC - required info
vpc-002c8d51809e98dd7 (VPC B)
10.20.0.0/16

Subnet info
subnet-0b7df9b79dca2085
VPC: vpc-002c8d51809e98dd7 IP Owner: 598823471631 Availability Zone: us-east-2a IP addresses available: 251 CIDR: 10.20.1.0/24

Create new subnet

Auto-assign public IP info
Disable

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

Create security group
Select existing security group

Security group name - required
VPC-B

Description - required info
VPC-B

Inbound Security Group Rules
Security group rule 1 (All, All, 10.0.0.0/8)

Type info	Protocol info	Port range info
All traffic	All	All
Source type info	Source info	Description - optional info
Custom	Q Add CIDR, prefix list or security	e.g. SSH for admin desktop
10.0.0.0/8 X		

Rest all settings are default or same as VPC-A-Public. Then create one more instance same as VPC-B-Private.

Also create one more inbound rule in VPC-A-Public as shown below

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

Inbound rules info					
Security group rule ID sg-002c8d51809e98dd7	Type info SSH	Protocol info TCP	Port range info 22	Source info Custom	Description - optional info 10.0.0.0/8 X
	All traffic	All	All	Custom	10.0.0.0/8 X

Add rule
Cancel Preview changes Save rules

Once the instances are up and running , take the ssh connection and login as root. Then from VPC-A-Public , check if the private IPs of VPC-B-Private and VPC-C-Private are reachable. It should not be reachable as below

```
[root@ip-10-10-1-23 ec2-user]# ping 10.20.1.233
PING 10.20.1.233 (10.20.1.233) 56(84) bytes of data.
^C
--- 10.20.1.233 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4130ms
[root@ip-10-10-1-23 ec2-user]# ping 10.30.1.49
PING 10.30.1.49 (10.30.1.49) 56(84) bytes of data.
^C
--- 10.30.1.49 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5192ms
[root@ip-10-10-1-23 ec2-user]#
```



Now we need to connect VPCs among each other.

Go to VPC and click on Transit Gateway.

Create Transit Gateway as shown below

VPC > Transit gateways > Create transit gateway

Create transit gateway Info

A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.

Details - optional

Name tag Creates a tag with the key set to Name and the value set to the specified string.
VPC-A-B-C-TGW

Description Info
Set the description of your transit gateway to help you identify it in the future.
connect VPCs among each other

Configure the transit gateway

Amazon side Autonomous System Number (ASN) Info
ASN

DNS support Info

VPN ECMP support Info

Default route table association Info

Default route table propagation Info

Multicast support Info

Select Transit gateway attachment from LHS panel and create Transit gateway attachment for every VPCs.

Give a name and select the transit gateway as shown below:

VPC > Transit gateway attachments > Create transit gateway attachment

Create transit gateway attachment Info

A transit gateway (TGW) is a network transit hub that interconnects attachments (VPCs and VPNs) within the same AWS account or across AWS accounts.

Details

Name tag - optional
Creates a tag with the key set to Name and the value set to the specified string.
VPC A

Transit gateway ID Info
tgw-07e5e02b7985a1b87 (VPC-A-B-C-TGW)

Attachment type Info
VPC

Then configure the attachment as below:



VPC attachment
Select and configure your VPC attachment.

DNS support [Info](#)

IPv6 support [Info](#)

Appliance Mode support [Info](#)

VPC ID
Select the VPC to attach to the transit gateway.

Subnet IDs [Info](#)
Select the subnets in which to create the transit gateway VPC attachment.

us-east-2a

us-east-2b No subnet available

us-east-2c No subnet available

Tags
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	Value - optional
<input style="border: none; width: 150px; height: 20px; border-radius: 4px; margin-right: 10px;" type="text" value="Name"/>	<input style="border: none; width: 150px; height: 20px; border-radius: 4px; margin-right: 10px;" type="text" value="VPC A"/>
Remove	
Add new tag	

You can add 49 more tags.

[Cancel](#) [Create transit gateway attachment](#)

Same way create another 2 attachments for VPC B and VPC C.

Select Transit gateway route tables from the LHS panel and go to the routes , we can see the VPC CIDR have been listed there.

Routes (5)								
	CIDR	Attachment ID	Resource ID	Resource type	Route type	Route state	Prefix list ID	
<input type="checkbox"/>	10.10.0.0/16	tgw-attach-020f102895018b928	vpc-0377757ccac5757d	VPC	Propagated	<input checked="" type="radio"/> Active	-	Actions Edit route
<input type="checkbox"/>	10.20.0.0/16	tgw-attach-09426e1aa7f8322e	vpc-0026ab518091990d7	VPC	Propagated	<input checked="" type="radio"/> Active	-	Actions Edit route
<input type="checkbox"/>	10.30.0.0/16	tgw-attach-04844505de274fbef9	vpc-011a5454aef1191493	VPC	Propagated	<input checked="" type="radio"/> Active	-	Actions Edit route

Now go to Route Tables, select VPC A and add route as shown below:

VPC > RouteTables > [rtt-5f6c80516454-49](#) > Edit routes

Edit routes

Destination	Target	State	Propagated	
10.10.0.0/16	<input style="border: none; width: 150px; height: 20px; border-radius: 4px; margin-right: 10px;" type="text" value="tgw-0000000000000000"/>	<input checked="" type="radio"/> Active	Yes	Actions Edit route
10.20.0.0/16	<input style="border: none; width: 150px; height: 20px; border-radius: 4px; margin-right: 10px;" type="text" value="tgw-0000000000000000"/>	<input checked="" type="radio"/> Active	Yes	Actions Edit route
10.30.0.0/16	<input style="border: none; width: 150px; height: 20px; border-radius: 4px; margin-right: 10px;" type="text" value="tgw-0000000000000000"/>	<input checked="" type="radio"/> Active	Yes	Actions Edit route
Add route				

[Cancel](#) [Preview](#) [Save changes](#)

Update the same for VPC B and VPC C

Now check the connectivity from VPC-A-Public , check if the private IPs of VPC-B-Private and VPC-C-Private are reachable



```
[root@ip-10-10-1-23 ec2-user]# ping 10.20.1.233
PING 10.20.1.233 (10.20.1.233) 56(84) bytes of data.
64 bytes from 10.20.1.233: icmp_seq=1 ttl=126 time=1.12 ms
64 bytes from 10.20.1.233: icmp_seq=2 ttl=126 time=0.546 ms
64 bytes from 10.20.1.233: icmp_seq=3 ttl=126 time=0.638 ms
^C
--- 10.20.1.233 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2079ms
rtt min/avg/max/mdev = 0.546/0.766/1.115/0.249 ms
[root@ip-10-10-1-23 ec2-user]# ping 10.30.1.49
PING 10.30.1.49 (10.30.1.49) 56(84) bytes of data.
64 bytes from 10.30.1.49: icmp_seq=1 ttl=126 time=0.836 ms
64 bytes from 10.30.1.49: icmp_seq=2 ttl=126 time=0.746 ms
64 bytes from 10.30.1.49: icmp_seq=3 ttl=126 time=0.768 ms
64 bytes from 10.30.1.49: icmp_seq=4 ttl=126 time=0.658 ms
^C
--- 10.30.1.49 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3097ms
rtt min/avg/max/mdev = 0.658/0.752/0.836/0.063 ms
[root@ip-10-10-1-23 ec2-user]#
```

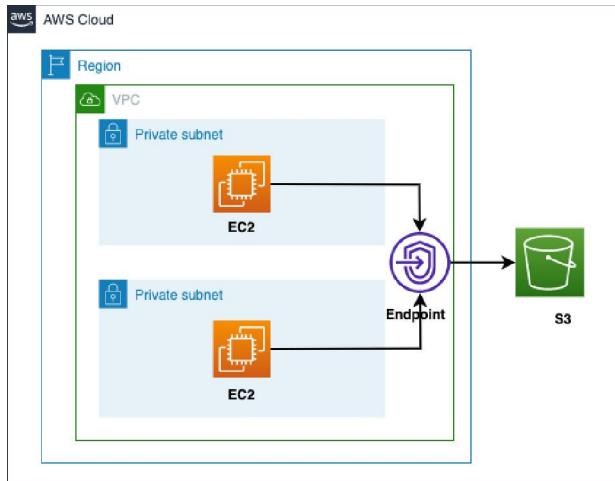
We can see it is reachable. This is how the Transit gateway works...!!!!

Conclusion:

AWS Transit Gateway simplifies cloud network architectures by acting as a hub to interconnect your VPCs, VPNs, and data centers. It eliminates complex mesh topologies and provides easy scalability, centralized management, and secure network segmentation. As your cloud footprint grows, Transit Gateway is key to maintaining a simple, efficient, and secure network topology.



CREATE VPC ENDPOINT FOR S3 BUCKET IN AWS



By default, all the communication between servers (whether local or on aws EC2-instance) and S3 is routed through the internet. Even though EC2 instances are also provided by aws, all requests from EC2 to S3 routes through the public internet. Therefore, we will be charged for all this data transmission.

AWS S3:

AWS S3 (Simple Storage Service) is one of the most well-known services being offered by aws. It provides a reliable, global and inexpensive storage option for large quantities of data. It can be used to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

Why do we need VPC Endpoint for S3:

Here VPC Endpoint for S3 comes to the rescue. VPC Endpoint for S3 provides us a secure link to access resources stored on S3 without routing through the internet. AWS doesn't charge anything for using this service.

VPC Endpoint:

VPC Endpoint for aws services enables us to privately connect our VPC to aws supported services without requiring an internet gateway, NAT device, VPN connection. Instances in our VPC do not require public IP addresses to communicate with aws services.

Types of VPC Endpoints:

1. **Interface Endpoint:** It is an elastic network interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service.



2. **Gateway Endpoint:** This type is used for connecting your VPC to AWS services over a scalable and highly available VPC endpoint. Gateway endpoints are usually associated with services that are accessed over an Internet Gateway, such as Amazon S3 and DynamoDB. *Here we will talk about S3 Vpc endpoints, which is a type of Gateway Endpoint.*

By using VPC endpoints, you can create a more isolated and secure environment for your AWS resources while still enabling them to access the necessary services without exposing them to the public internet.

Step 1: Create vpc name as “kuku-vpc” select IPV4 CIDR (10.0.0.0/16)

The screenshot shows the 'Create VPC' settings page. Under 'Resources to create', the 'VPC only' option is selected. A 'Name tag - optional' field contains 'kuku-vpc'. Under 'IPv4 CIDR block', the value '10.0.0.0/16' is entered. Other options like 'IPv4 CIDR manual input' and 'IPAM-allocated IPv4 CIDR block' are also visible.

Step 2: Create two subnet one as public another one as private, in public subnet give IPv4 CIDR as 10.0.0.0/24 and in private subnet give IPv4 CIDR as 10.0.1.0/24

<input type="checkbox"/> kuku-public-sn	subnet-0f119e610836e123	<input checked="" type="radio"/> Available	ypc-0a0416fd99acda995 kuku-vpc
<input type="checkbox"/> kuku-private-sn	subnet-0e153d039c4248119	<input checked="" type="radio"/> Available	ypc-0a0416fd99acda995 kuku-vpc
<input type="checkbox"/> public-1	subnet-062006023bb823b6	<input checked="" type="radio"/> Available	unc-01cded17ba322c0291baan

Step 3: Create internet gateway

An Internet Gateway (IGW) is a fundamental component of Amazon Web Services (AWS) networking that provides a connection between your Virtual Private Cloud (VPC) and the public internet. It allows resources within your VPC to access and communicate with services and resources on the internet and vice versa.

The screenshot shows the 'Create internet gateway' settings page. A 'Name tag' field contains 'kuku-ig'. Below it, under 'Tags - optional', there is a 'Key' field with 'Name' and a 'Value - optional' field with 'kuku-ig'. A 'Create internet gateway' button is at the bottom right.



Step 4: Attach your igw to your VPC

VPC > Internet gateways > Attach to VPC (igw-07a61e698c2e41d25) [Info](#)

Attach to VPC (igw-07a61e698c2e41d25)

VPC
Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.
Q vpc-090416fd99acda995 [X](#)

▶ AWS Command Line Interface command

Cancel **Attach internet gateway**

Step 5: create route tables

A route table is a networking component used in Amazon Web Services (AWS), particularly within Virtual Private Clouds (VPCs), to control the routing of network traffic between different subnets and destinations. A route table contains a set of rules (routes) that dictate how network traffic is directed within the VPC.

VPC > Route tables > Create route table [Info](#)

Create route table [Info](#)

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.
kuku-public-rt

VPC
The VPC to use for this route table.
vpc-090416fd99acda995 (kuku-vpc)

Tags
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	Value - optional
Q Name	Q kuku-public-rt X Remove

Add new tag

Here we create two route tables one for public and another one for private subnet.

Step 6: Subnet Association

Each subnet in a VPC must be associated with a route table. This association determines how traffic is routed for resources within that subnet.

VPC > Route tables > rtb-04145b095c69c6c979 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1 / 2)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
kuku-public-sn	subnet-01f119e610836e123	10.0.0.0/24	-	Main (rtb-082664b22182d016a)
kuku-private-sn	subnet-0e153d039e4248119	10.0.1.0/24	-	Main (rtb-082664b22182d016a)

Selected subnets

subnet-01f119e610836e123 / kuku-public-sn X

Cancel **Save associations**



Step 7: To provide internet access to resources within a subnet, you would add a default route (**0.0.0.0/0**) with the target set to an Internet Gateway (IGW). This allows traffic from the subnet to flow through the IGW to the public internet.

The screenshot shows the 'Edit routes' section of a route table. A new route is being added for destination '0.0.0.0/0' with target 'Internet Gateway' and ID 'igw-07a61e698c2e41d25'. The 'Add route' button is visible at the bottom left, and 'Save changes' is at the bottom right.

Step 8: perform similar steps for private route table and do subnet association to it.

Step 9: launch EC2 instances

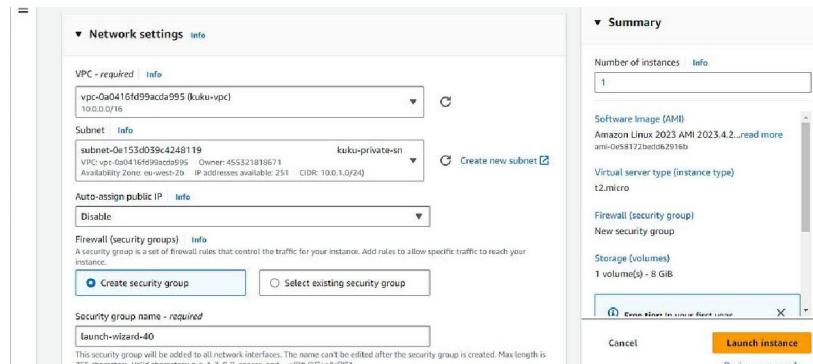
The screenshot shows the 'Launch an instance' step of the EC2 wizard. It includes fields for 'Name and tags' (name 'pub-web'), 'Application and OS Images (Amazon Machine Image)' (AMI 'Amazon Linux 2023 AMI 2023.4.2...'), and 'Summary' (1 instance, AMI 'ami-0c58172bed6d2916b', instance type 't2.micro', security group 'New security group', storage '1 volume(s) - 8 GiB').

Step 10: add network setting in EC2 instances, we select our own VPC which we created in (Step 1) and select public subnet where public IP is enable , and launch our instances

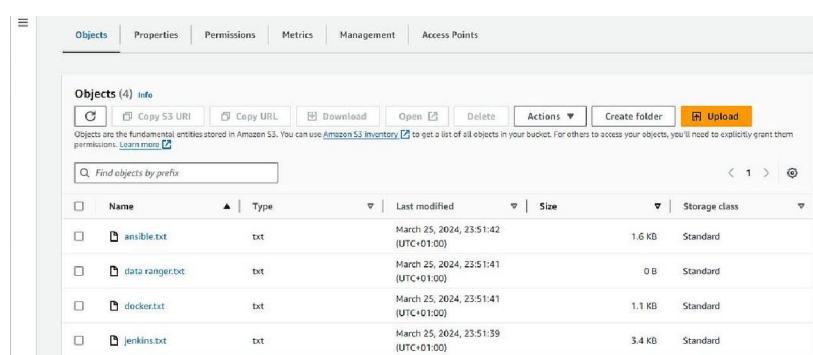
The screenshot shows the 'Network settings' step of the EC2 wizard. It includes 'VPC - required' (selected VPC 'vpc-0a0416fd99acda995 (kuku-vpc)'), 'Subnet' (selected subnet 'subnet-01f119e610836e123'), 'Auto-assign public IP' (set to 'Enable'), 'Firewall (security groups)' (selected 'Create security group' and named 'launch-wizard-39'), and 'Summary' (1 instance, AMI 'ami-0c58172bed6d2916b', instance type 't2.micro', security group 'New security group', storage '1 volume(s) - 8 GiB'). The 'Launch instance' button is highlighted at the bottom right.



Step 11: similarly, we have to create private EC2 instances where we select private subnet and public IP is disable and launch the private server.

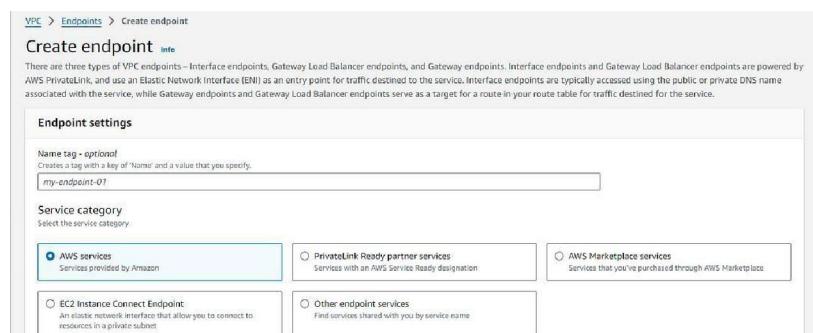


Step 12: Create S3 bucket, named as **boon123** and upload some file in it.



Our main aim is to access these files without using the internet on our private server. We have not provided a public IP. If we are able to access these files from our private server, then we have established the endpoint connection correctly.

Step 13: Create endpoint connection name as (ujjwal-endpoint), in service category we select AWS services





In services we select Gateway Endpoint, is used for connecting your VPC to AWS services over a scalable and highly available VPC endpoint. Gateway endpoints are usually associated with services that are accessed over an Internet Gateway, such as Amazon S3 and DynamoDB.

The screenshot shows the AWS Services search interface. A table lists four services: com.amazonaws.eu-west-2.s3 (Gateway), com.amazonaws.eu-west-2.s3 (Interface), com.amazonaws.eu-west-2.s3-outposts (Interface), and com.amazonaws.s3-global.accesspoint (Interface). The 'com.amazonaws.eu-west-2.s3' entry under 'Gateway' is selected. Below the table, there are two sections: 'VPC' (Select the VPC in which to create the endpoint) and 'Route tables' (Select the route table to associate with the endpoint).

We select our both route tables and give full access in policy; we established our endpoint connection.

The screenshot shows the Route tables configuration screen. It displays two route tables: 'kuku-public-rt' and 'kuku-private-rt'. Both are selected. A note at the bottom states: 'When you use an endpoint, the source IP addresses from your instances in your affected subnets for accessing the AWS service in the same region will be private IP addresses, not public IP addresses. Existing connections from your affected subnets to the AWS service that use public IP addresses may be dropped. Ensure that you don't have critical tasks running when you create or modify an endpoint.'

The screenshot shows the Route tables configuration screen, identical to the previous one, displaying the selection of 'kuku-public-rt' and 'kuku-private-rt' route tables. The note at the bottom is identical.

Step 14: First, we connect to our public server. After successful configuration on this server, we proceed to configure AWS on an Amazon Linux instance.



```
[root@ip-10-0-0-210 ec2-user]# aws configure
AWS Access Key ID [*****C000]:
AWS Secret Access Key [*****EsDa]:
Default region name [eu-west-2]:
Default output format [json]:
[root@ip-10-0-0-210 ec2-user]#
```

Step 15: After configuration we run commands for access our S3 bucket and checking the contents of a bucket. It provides a quick way to obtain an overview of the objects stored within a bucket without needing to use the AWS Management Console.

```
#aws s3 ls s3://(bucket name)
```

```
[root@ip-10-0-0-210 ec2-user]# aws s3 ls s3://boon123
2024-03-25 22:51:42      1632 ansible.txt
2024-03-25 22:51:41          0 data ranger.txt
2024-03-25 22:51:41      1082 docker.txt
2024-03-25 22:51:39      3455 jenkins.txt
[root@ip-10-0-0-210 ec2-user]#
```

Step 16: then we have to check entire S3 content from our private server we have to create a file for our private server key(.pem key) by using command then copy public key to newly created .pem file and after that we have to run command

```
vim filename.pem chmod 600 filename.pem
```

then run SSH command for private server.

```
[root@ip-10-0-0-210 ec2-user]# ssh -i "oscar.pem" ec2-user@10.0.1.230
Warning: Identity file oscar.pem not accessible: No such file or directory.
The authenticity of host '10.0.1.230 (10.0.1.230)' can't be established.
ED25519 key fingerprint is SHA256:4R0d/r2kfr80/l02scxd/rtrlVzprR2diF9aC9aM4YPIII.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.230' (ED25519) to the list of known hosts.
ec2-user@10.0.1.230: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[root@ip-10-0-0-210 ec2-user]# vim oscar.pem
[root@ip-10-0-0-210 ec2-user]#
[root@ip-10-0-0-210 ec2-user]# cat oscar.pem
[root@ip-10-0-0-210 ec2-user]# chmod 600 oscar.pem
[root@ip-10-0-0-210 ec2-user]# ssh -i "oscar.pem" ec2-user@10.0.1.230
.
.
.
Amazon Linux 2023
.
.
.
https://aws.amazon.com/linux/amazon-linux-2023
.
.
.
[ec2-user@ip-10-0-1-230 ~]$
```



Step 17: Then once again we have to configure our aws in private server

```
Warning: Permanently added '16.0.1.230' (ED25519) to the list of known hosts.
[ec2-user@i-08d6dfbdae92612db ~]$ ssh -i "oscar.pem" ec2-user@16.0.1.230
[ec2-user@i-0-0-0-230 ec2-user]$ vim oscar.pem
[ec2-user@i-0-0-0-230 ec2-user]$ chmod 400 oscar.pem
[ec2-user@i-0-0-0-230 ec2-user]$ ssh -i "oscar.pem" ec2-user@16.0.1.230
[ec2-user@i-0-0-0-230 ec2-user]$ ls
[ec2-user@i-0-0-0-230 ec2-user]$ cd /tmp/
[ec2-user@i-0-0-0-230 ec2-user]$ curl https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@i-0-0-0-230 ec2-user]$ rm -rf /tmp/*
[ec2-user@i-0-0-0-230 ec2-user]$ ls
[ec2-user@i-0-0-0-230 ec2-user]$ sudo su
[ec2-user@i-0-0-0-230 ec2-user]$ aws configure
AWS Access Key ID [None]: XXXXXXXXXX
AWS Secret Access Key [None]: XXXXXXXXXX
Default region name [None]: eu-west-2
Default output format [None]: json
[ec2-user@i-0-0-0-230 ec2-user]$
```

run same command again

```
#aws s3 ls s3://(bucket name)
```

```
aws Select Access Key [None]: XXXXXXXXXX
Default region name [None]: eu-west-2
Default output format [None]: json
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ 
[ec2-user@i-0-0-0-230 ec2-user]$ aws s3 ls
2024-03-25 22:19:48 boon123
[ec2-user@i-0-0-0-230 ec2-user]$ ^C
[ec2-user@i-0-0-0-230 ec2-user]$ aws s3 ls s3://boon123
2024-03-25 22:51:42      1632 ansible.txt
2024-03-25 22:51:41        0 data ranger.txt
2024-03-25 22:51:41     1082 docker.txt
2024-03-25 22:51:39    3455 jenkins.txt
[ec2-user@i-0-0-0-230 ec2-user]$
```

We can access the files present in our S3 bucket from a private server without using the internet, by established endpoint connection

Conclusion

In conclusion, utilizing an AWS S3 VPC endpoint offers a **secure** and efficient means of accessing S3 buckets from within an Amazon Virtual Private Cloud (VPC). By establishing a direct and private connection between resources in the VPC and S3 without traversing the public internet, VPC endpoints enhance security and **reduce latency**. This setup ensures that data transfers to and from S3 remain within the AWS network, mitigating exposure to potential security threats and optimizing performance. Implementing S3 VPC endpoints is therefore a recommended best practice for organizations seeking to maximize the security and efficiency of their AWS infrastructure.



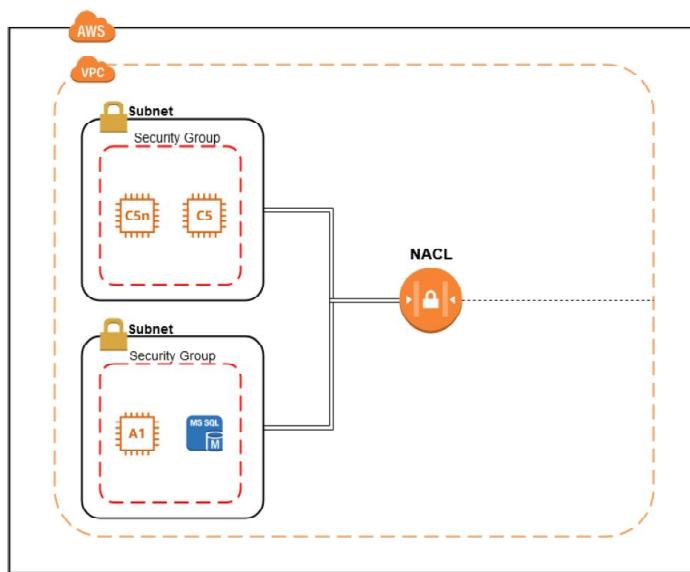
Security Groups and NACL

Security Groups and Network Access Control Lists in AWS and to understand when to use them and when not to.

Let's start with the basic definitions

Security Group: Security Group is a stateful firewall to the instances. Here stateful means, security group keeps a track of the State. Operates at the instance level.

Network Access Control List: NACL is stateless, it won't keep any track of the state. Operates at Subnet level.



Security Group and NACL Basic Architecture in AWS

Security Group:

Security Group is a stateful firewall which can be associated with Instances. Security Group acts like a Firewall to Instance or Instances. Security Group will always have a hidden Implicit Deny in both Inbound and Outbound Rules. So, we can only allow something explicitly, but not deny something explicitly in Security Groups.

Default Security Group:

By default, a Security Group is like:

When we talk about the default Security Group, there are two things to discuss — *AWS created Default SG, User Created Default SG*.

AWS creates a default SG when it creates a default VPC — in this security group they will add an inbound rule which says all Instances in this Security Group can talk to each other.



Any Security Group created by a User explicitly, wouldn't contain this Inbound Rule which would allow communication between the Instances, we should explicitly add it if required.

Both in the *AWS created SG* and *User Created Custom SG*, the Outbound Rules would be the same — which allows ALL TRAFFIC out.

We cannot add a Deny Rule, both in Inbound and Outbound Rules as there's a hidden default Implicit Deny Rule in Security Groups. All we can do is allow which is required, everything else which isn't allowed by us is blocked.

A default security group that is created by default in the default VPC by AWS looks like this

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	sg-07b788f2f4981b831 (default)	-

Default Security Group Inbound Rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-

Default Security Group Outbound Rules.

Security Group Features:

There are two main features which will make Security Groups different from NACLs —

- **Stateful Firewall**
- **Connection Tracking**



Stateful Firewall:

Stateful means – maintain the state of connection so that you introduce yourself only once, not every time you start talking – think TCP session, once established, they start talking till one of them says Finish or Reset.

The reason why a Security Group is called a Stateful Firewall is – Security Group basically maintains the State of a connection, meaning – *if an instance sends a request , the response traffic from outside is allowed back irrespective of the inbound rules, and vice versa.*

Example: If my security group inbound rule allows NO TRAFFIC and outbound rule allows ALL TRAFFIC and I visit a website on my instance, the response from the WebServer back to my instance will be allowed even though the inbound rule denies everything.

Security Group achieves this by leveraging something known as Connection Tracking which we will be discussing shortly.

Connection Tracking:

Security Groups use connection tracking to keep track of connection information that flows in and out of an instance, this information includes — IP address, port number and some other information(for some specific protocols).

Security group needs to track any connection only in this case — *if there's no inbound/outbound rule that allows everything*. Let's say we have allowed all traffic from outside and all traffic to outside, it need not track anything because, whatever comes and goes should be allowed.

Security Group Rule Fields:

Inbound rules				
Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	0	Custom	
<button>Add rule</button> <button>Delete</button>				

Outbound rules				
Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	0.0.0.0/0
<button>Add rule</button> <button>Delete</button>				

Editing Security Group Inbound & Outbound Rules

Type: Type of Traffic which can be TCP, UDP, ICMP. Type field provides the well-used protocols, when selected it auto fills the Protocol field. You may also select a Custom Protocol Rule, which allows you to select the Protocol field from a wide range of Protocols.



Protocol: As mentioned already, if you select a custom protocol Rule in Type field, you can select a protocol from the available protocol list.

Port Range: You can specify a single port or a range of ports like this 5000–6000.

Source [Inbound Rules only]: Can be custom — a single IP address or an entire CIDR block, anywhere — 0.0.0.0/0 in case of IPv4, My IP Address — AWS auto-detects your public IP address. Destination can only be mentioned in outbound rule.

Destination [Outbound Rules only]: Can be Custom — a single IP address or an entire CIDR block, anywhere — 0.0.0.0/0 in case of IPv4, My IP Address — AWS auto-detects your Public IP address. Source can only be mentioned in Inbound Rule.

Description: This field is optional. You can add a description which helps you to keep a track of which rule is for what.

NACL — Network Access Control List:

NACLs are stateless firewalls which work at Subnet Level, meaning NACLs act like a Firewall to an entire subnet or subnets. A default NACL allows everything both Inbound and Outbound Traffic. Unlike Security Groups, in NACLs we have to explicitly tell what to deny in Inbound and outbound rules. There's no implicit deny in NACL.

Default NACL:

By default, a NACL is like:

When we create a VPC, a default NACL will be created which will allow all Inbound Traffic and Outbound Traffic. If we don't associate a Subnet to NACL, the default NACL in that VPC will be associated to that subnet. A default NACL looks like this —

The screenshot shows the AWS CloudFormation interface for managing Network ACLs. It displays two tables: 'Inbound Rules' and 'Outbound Rules'. Both tables have the following columns: Rule #, Type, Protocol, Port Range, Source/Destination, and Allow / Deny.

Inbound Rules:

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	All Traffic	All	All	0.0.0.0/0	ALLOW
*	All Traffic	All	All	0.0.0.0/0	DENY

Outbound Rules:

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
100	All Traffic	All	All	0.0.0.0/0	ALLOW
*	All Traffic	All	All	0.0.0.0/0	DENY



NACL Features:

Statelessness:

Unlike security groups, NACL doesn't maintain any track of connections which makes it completely stateless, meaning — if some traffic is allowed in NACL Inbound Rule, the response outbound traffic is not allowed by default unless specified in the Outbound Rules.

NACL Rule Fields:

The screenshot shows two pages of the AWS Network ACLs interface:

Editing NACL Inbound Rules

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	All Traffic	All	All	0.0.0.0/0	ALLOW

Editing NACL Outbound Rules

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
100	All Traffic	All	All	0.0.0.0/0	ALLOW

Rule Number: Rules are evaluated starting with the lowest numbered rule. If a rule matches, it gets executed without checking for any other higher numbered rules.

Type: Type of a Traffic which can be TCP, UDP, ICMP. Type field provides the well-used protocols, when selected it auto fills the Protocol field. You may also select a custom protocol rule, which allows you to select the protocol field from a wide range of protocols.

Protocol: As mentioned already, if you select a Custom Protocol Rule in Type field, you can select a Protocol from the available Protocol List. **Port Range** — You can specify a single port or a range of ports like this 5000–6000.

Source [Inbound Rules only]: Can be a Single IP Address or an entire CIDR block. Destination can only be mentioned in Outbound Rule.

Destination[Outbound Rules only]: Can be a Single IP Address or an entire CIDR block. Source can only be mentioned in Inbound Rule. **Allow/Deny** — Specifies whether to allow or deny traffic.

Security Group and NACL Key Differences:

SG and NACL Differences

Use Case:

I will give an example to make you understand when to use Security Group and when to use NACL —



Let's say you have allowed SSH Access of an Instance to a user in Dev Team and he's connected to it and actively accessing it and for some reason(realizing that the user is involved in some malicious activity) you wanted to remove his SSH access.

In this case you have two choices —

1. Remove SSH inbound allow rule of that user in the Security Group inbound rule.
2. Add an NACL Rule explicitly denying traffic from his IP address. If you go with the first one, he would not lose his SSH connection, this is due to the connection tracking behavior of Security Groups. If you go with the latter choice, NACL would immediately block his connection.

So, in this case, it's better to use a NACL Deny Rule rather than deleting a Security Group allow Rule.

NACL & SG Default Quota:

NACL :

- NACLs Per VPC — 200
- Rules per NACL — 20

Key Points:

Single NACL can be associated with multiple subnets, however single subnet cannot be associated with multiple NACLs at same time as there can be multiple deny rules which contradict each other.

Security Groups:

- VPC Security Groups per Region — 2500
- Rules Per Security Group — 60 Inbound and 60 Outbound.

Key Points:

Single security group can be associated to multiple instances and unlike NACL, multiple Instances can be associated with multiple security groups as there cannot be explicit deny rules which can contradict each other here.

These quota limits are the default ones, if you want to increase the limit you can request AWS to do so. Some quota limits in the VPC are strict and cannot be increased.



Elastic Block Store

Amazon Elastic Block Store (EBS): Reliable Block Storage



In today's digital age, businesses rely heavily on data storage solutions to store and manage their critical information. One such solution that has gained significant popularity is Amazon Elastic Block Store (EBS). EBS is a cloud-based block storage service offered by Amazon Web Services (AWS) that provides secure and reliable storage for your business's data.

Understanding block storage and its importance for businesses

Before delving into the benefits and features of Amazon EBS, it's crucial to understand what block storage is and why it is essential for businesses. Block storage is a type of data storage that allows data to be stored and retrieved in fixed-sized blocks or chunks. This method provides businesses with more flexibility and control over their data, as it allows for direct access to specific data blocks.

Block storage is especially crucial for businesses that deal with large amounts of data or require high-performance storage solutions. It enables faster data access, efficient data replication, and increased storage capacity. With block storage, businesses can ensure the integrity and availability of their data, which is vital for their day-to-day operations.

Benefits of using Amazon EBS for block storage:

Amazon EBS offers several benefits that make it an ideal choice for businesses looking for secure and dependable block storage.

Firstly, EBS provides high durability and availability. It automatically replicates data within a specific Availability Zone (AZ), ensuring that your data remains safe and accessible even in the event of a hardware failure. This level of redundancy guarantees business continuity and minimizes the risk of data loss.

Secondly, EBS allows for easy scalability. Businesses can increase or decrease their storage capacity as per their requirements without any disruption to their operations. This flexibility enables businesses to adapt to changing storage needs, whether it's handling increased data volumes or downsizing storage requirements.



Furthermore, Amazon EBS offers snapshot capabilities, allowing businesses to create point-in-time copies of their volumes. These snapshots can be used for backup, disaster recovery, or even to create new volumes in different regions, providing an additional layer of data protection.

Key features and capabilities of Amazon EBS

Amazon EBS comes with a range of features and capabilities that enhance its functionality and make it a powerful storage solution for businesses.

One key feature is the ability to choose between different volume types based on your workload requirements. EBS offers General Purpose SSD (gp2), Provisioned IOPS SSD (io1), Throughput Optimized HDD (st1), and Cold HDD (sc1) volume types. Each type is optimized for specific use cases, ensuring that businesses can select the most suitable option for their storage needs.

Another important capability of Amazon EBS is its support for Elastic Volumes. Elastic Volumes allows businesses to adjust the size, performance, and type of their EBS volumes without interrupting their EC2 instances. This feature enables businesses to optimize their storage resources and adapt to changing workload demands seamlessly.

Additionally, Amazon EBS provides encryption at rest, ensuring that your data is protected from unauthorized access. By leveraging AWS Key Management Service (KMS), businesses can encrypt their EBS volumes and manage encryption keys securely. This feature is particularly crucial for businesses that handle sensitive or confidential data.

Types of Amazon EBS volumes and their use cases

Amazon EBS offers different types of volumes, each designed to cater to specific use cases. Understanding these volume types can help businesses make informed decisions when selecting the most appropriate storage solution.

General purpose SSD (gp2) volumes are suitable for a wide range of workloads, including boot volumes, small to medium-sized databases, and development/test environments. These volumes provide a balance of price and performance, making them a popular choice for many businesses.

Provisioned IOPS SSD (io1) volumes are designed for applications that require consistently high performance and low latency. Use cases include large databases, data warehousing, and applications with high transaction rates. io1 volumes allow businesses to specify the desired level of input/output operations per second (IOPS), providing predictable performance for critical workloads.

Throughput Optimized HDD (st1) volumes are ideal for frequently accessed, large sequential workloads, such as log processing, big data, and data warehouses. These volumes deliver high throughput and low-cost storage, making them cost-effective solutions for data-intensive applications.

Cold HDD (sc1) volumes are designed for infrequently accessed workloads, such as backups and disaster recovery. These volumes offer the lowest cost per gigabyte and are suitable for data that does not require frequent access.



By understanding the characteristics and use cases of each volume type, businesses can optimize their storage infrastructure and ensure optimal performance and cost-efficiency.

How to create and attach an Amazon EBS volume to an EC2 instance

Creating and attaching an Amazon EBS volume to an EC2 instance is a straightforward process.

To create a new EBS volume, you can navigate to the EC2 dashboard in the AWS Management Console and click on “Volumes” in the left-hand menu. From there, you can choose the volume type, size, and other parameters based on your requirements. Once the volume is created, you can attach it to an EC2 instance by selecting the instance and clicking on “Actions” and then “Attach Volume.”

Alternatively, you can use the AWS Command Line Interface (CLI) or AWS SDKs to create and attach EBS volumes programmatically. This method is particularly useful for businesses that have automated deployment processes or require a high level of customization.

When attaching an EBS volume to an EC2 instance, it’s essential to ensure that the instance is in the same Availability Zone as the volume. This ensures optimal performance and availability.

Best Practices for securing and optimizing Amazon EBS

Securing and optimizing Amazon EBS is crucial to ensure the integrity and performance of your storage infrastructure. Here are some best practices to consider:

- **Implement encryption at rest:** Enable encryption for your EBS volumes using AWS KMS. This ensures that your data is protected from unauthorized access, even if the physical volume is compromised.
- **Regularly back up your EBS volumes:** Take regular snapshots of your EBS volumes to create backups. These snapshots can be used for disaster recovery or to create new volumes in different regions, providing an additional layer of data protection.
- **Monitor and optimize performance:** Utilize Amazon CloudWatch to monitor the performance of your EBS volumes. By tracking metrics such as volume latency and throughput, you can identify bottlenecks and optimize your storage configuration for better performance.
- **Use Elastic Volumes to adjust storage capacity:** Leverage Elastic Volumes to resize your EBS volumes as needed. This allows you to scale your storage resources based on actual usage, reducing costs and optimizing performance.
- **Implement access controls:** Use AWS Identity and Access Management (IAM) to manage user access and permissions for your EBS volumes. By granting least privilege access, you can ensure that only authorized users can modify or access your storage resources.

By following these best practices, businesses can enhance the security, performance, and cost-effectiveness of their Amazon EBS implementation.



Monitoring and managing Amazon EBS performance

Monitoring and managing the performance of your Amazon EBS volumes is essential to ensure optimal storage performance and identify any potential issues. Amazon CloudWatch provides a range of metrics and alarms that can help you monitor the health and performance of your EBS volumes.

Some key metrics to monitor include volume read/write operations, volume latency, and volume throughput. By tracking these metrics, you can identify any performance bottlenecks and take appropriate actions to optimize your storage configuration.

In addition to monitoring, Amazon EBS provides features such as Elastic Volumes and Enhanced Monitoring that allow you to proactively manage and optimize your storage resources. Elastic volumes enables you to adjust the size and performance of your volumes without interrupting your EC2 instances, providing flexibility and cost optimization.

Enhanced monitoring provides additional insights into the performance of your EBS volumes, allowing you to fine-tune your storage configuration for optimal performance.

Backup and disaster recovery strategies with Amazon EBS

Backup and disaster recovery are critical aspects of any business's data storage strategy. Amazon EBS offers several features and capabilities that enable businesses to implement robust backup and disaster recovery strategies.

One such feature is the ability to create snapshots of your EBS volumes. Snapshots are point-in-time copies of your volumes that can be used to create new volumes or restore existing volumes in the event of data loss or system failure. By regularly taking snapshots and storing them in different regions, businesses can ensure data availability and recovery in case of a disaster.

In addition to snapshots, businesses can also leverage AWS services such as Amazon S3 and AWS storage gateway for long-term data backup and archiving. By integrating EBS with these services, businesses can create cost-effective and scalable backup solutions that meet their specific requirements.

It's important to note that a comprehensive backup and disaster recovery strategy should include regular testing and validation of the recovery process. By periodically restoring snapshots and verifying data integrity, businesses can ensure the effectiveness of their backup and recovery procedures.

Case studies: Real-world examples of businesses benefiting from Amazon EBS

To illustrate the real-world benefits of Amazon EBS, let's take a look at a couple of case studies:

- Company XYZ, a leading e-commerce platform, relies on Amazon EBS to store and manage its extensive product catalog. By utilizing Provisioned IOPS SSD (io1) volumes, Company XYZ ensures fast and predictable performance for its database-intensive workloads. The ability to scale storage capacity seamlessly has enabled the company to handle increased traffic and data volumes without impacting user experience.



- Company ABC, a healthcare provider, uses Amazon EBS to store sensitive patient data securely. By leveraging encryption at rest and regular snapshots, Company ABC ensures data privacy and enables quick recovery in case of a system failure. The high durability and availability of EBS have provided the company with peace of mind, knowing that their critical patient information is protected and accessible at all times.

These case studies highlight how Amazon EBS has helped businesses across various industries achieve secure and dependable block storage solutions.

Conclusion: Why Amazon EBS is the ideal solution for secure and dependable block storage

In conclusion, Amazon Elastic Block Store (EBS) is a powerful and versatile block storage solution that offers businesses the security, reliability, and scalability they need to drive their success. With its range of volume types, robust features, and seamless integration with other AWS services, EBS provides businesses with the flexibility and control they require for their data storage needs.

By understanding the benefits, features, and best practices associated with Amazon EBS, businesses can leverage this solution to optimize their storage infrastructure, enhance data security, and ensure high-performance storage for their critical workloads.

So, if you're looking for a secure and dependable block storage solution that can drive your business success, look no further than Amazon EBS. Take advantage of its capabilities, implement best practices, and unlock the full potential of your data storage infrastructure.



Elastic File System

Mount Elastic File System (EFS) on EC2



Well, you've come to the right place! In this guide, we'll go through the steps to create an Elastic File System, we'll launch and configure two Amazon EC2 Instances ,we'll practice mounting the EFS to both instances by logging into each instance via SSH authentication and we'll practice sharing files between two instances.

Introduction

What's Amazon Elastic File System (EFS) ?

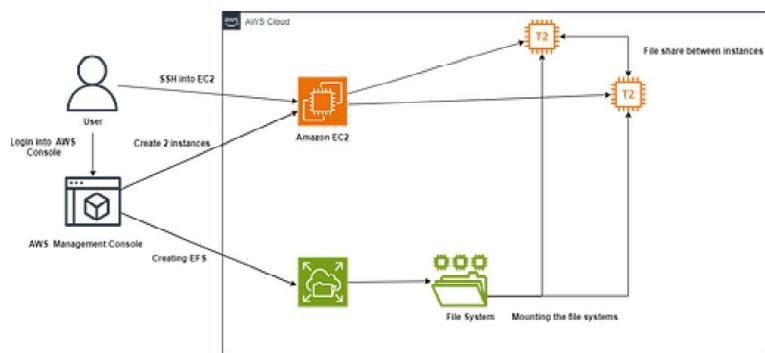
- Amazon EFS is a fully managed, scalable file storage service designed to provide shared access to files across multiple Amazon EC2 instances.
- It is particularly useful for applications and workloads that require shared file storage in a cloud environment.

Key features and aspects of Amazon EFS:

- **Scalability:** Amazon EFS can scale automatically as your storage needs grow, accommodating varying workloads without requiring you to provision additional capacity.
- **Shared File Storage:** EFS allows multiple EC2 instances to access the same file system concurrently, providing a simple and scalable solution for applications that require shared access to files.
- **Performance:** It is designed to deliver low-latency performance, suitable for a wide range of applications, including big data analytics, media processing, and content management.
- **Ease of Use:** EFS is easy to set up and manage, eliminating the need for manual intervention in capacity planning or performance tuning.
- **Compatibility:** It supports the Network File System version 4 (NFSv4) protocol, making it compatible with Linux-based EC2 instances.
- **Security:** EFS supports encryption of data at rest and in transit, helping you maintain the security of your file system.
- **Lifecycle Management:** You can manage the lifecycle of your files with the EFS Infrequent Access storage class, which provides a lower-cost storage option for files that are accessed less frequently.



Architecture Diagram



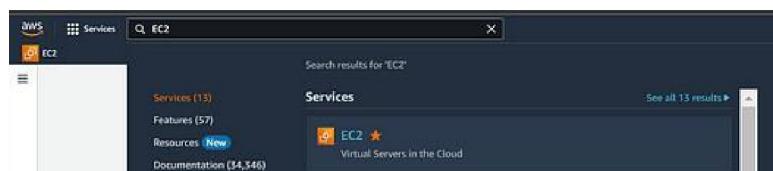
Task Steps

Step 1: Sign in to AWS Management Console

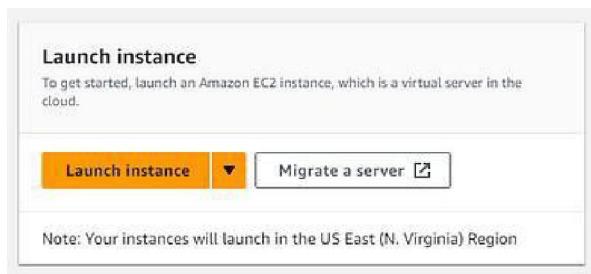
- On the AWS sign-in page, enter your credentials to log in to your AWS account and click on the Sign in button.
- Once Signed In to the AWS Management Console, Make the default AWS Region as US East (N. Virginia) us-east-1

Step 2: Launching two EC2 Instances

- Make sure you are in the N. Virginia Region.
- Navigate to the Services menu in the top, then click on EC2 in the Compute section.



- Click on **Instances** from the left side bar and then click on **Launch instances**.



- Number of Instances:** Enter 2 on the right side under summary



5. **Name:** Enter *MyEC2*
6. **For Amazon Machine Image (AMI):** Search for **Amazon Linux 2 AMI** in the search box and click on the **Select** button.

▼ 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.

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

Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Li [Browse more AMIs](#)
Including AMIs from AWS Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0c0b74d29acd0cd9f7 (64-bit (x86)) / ami-095889fa7a7b9da4e (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs Free tier eligible

Description
Amazon Linux 2 Kernel 5.10 AMI 2.0.20240109.0 x86_64 HVM gp2

Architecture AMI ID Verified provider

7. For Instance Type: select *t2.micro*

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

Instance type

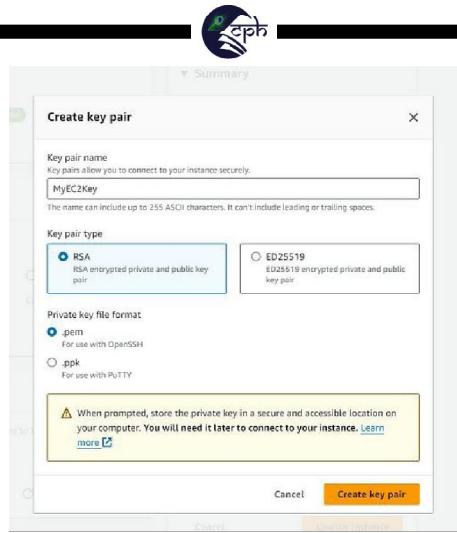
t2.micro Free tier eligible

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

All generations [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

8. For **Key pair:** Select **Create a new key pair** Button
 - **Key pair name:** MyEC2Key
 - **Key pair type:** RSA
 - **Private key file format:** .pem
9. Select **Create key pair** Button.

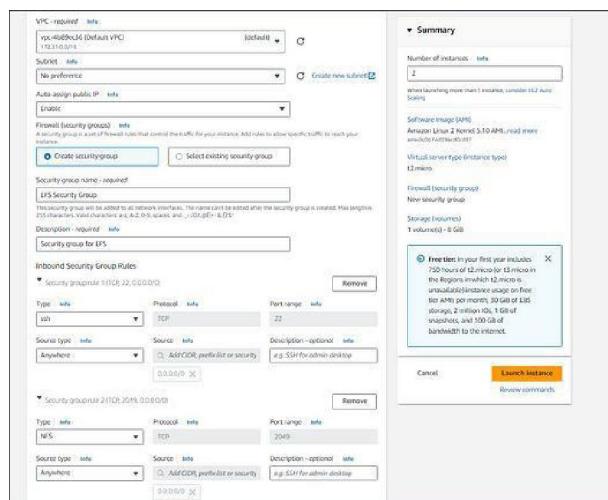


10. In Network Settings Click on Edit:

- Auto-assign public IP: Enable
- Select Create new Security group
- Security Group Name: Enter *EFS Security Group*
- To add SSH:
- Choose Type: SSH
- Source: Anywhere

For NFS:

- Click on Add security group rule
- Choose Type: NFS
- Source: Anywhere





11. Keep Rest thing **Default** and Click on **Launch Instance** Button.
12. Select **View all Instances** to View Instance you Created.
13. **Launch Status:** Your instance is now launching. Click on the **instance ID** and wait until the **initialization status changes to Running**.
14. Click on each instance and enter a name as **MyEC2-1** and **MyEC2-2**.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
MyEC2-1	i-0b440e7e0d8ffca3f8	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-55-173-156-210.x... 55.173.154	55.173.154
MyEC2-2	i-0f11e912d41b5a56c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-3-84-2-0.x... 3.84.2.0	3.84.2.0

15. Take note of the IPv4 Public IP Addresses of the EC2 instances and save them for later.

Step 3: Creating an Elastic File System

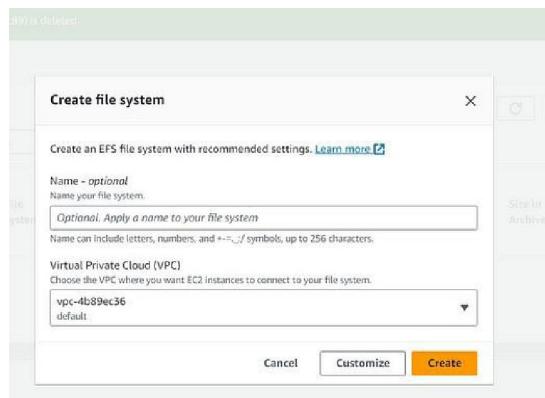
Navigate to EFS by clicking on the Services menu at the top. Click on EFS in the Storage section.



2. Click on Create file system



3. Click on Customize button.





4. Enter the details below, Type the **Name** as **EFS-Demo** and make sure **default VPC** and **default Regional** options are selected.
5. Uncheck the option of **Enable automated backups**

File system settings

General

Name – optional
Name your file system.

File system type
Choose to either store data across multiple Availability Zones or within a single Availability Zone. [Learn more](#)

Regional
Offers the highest levels of availability and durability by storing file system data across multiple Availability Zones within an AWS Region.

One Zone
Provides continuous availability to data within a single Availability Zone within an AWS Region.

Automatic Backups
Automatically backup your file system data with AWS Backup using recommended settings. Additional pricing applies. [Learn more](#)

Enable automatic backups

Lifecycle management

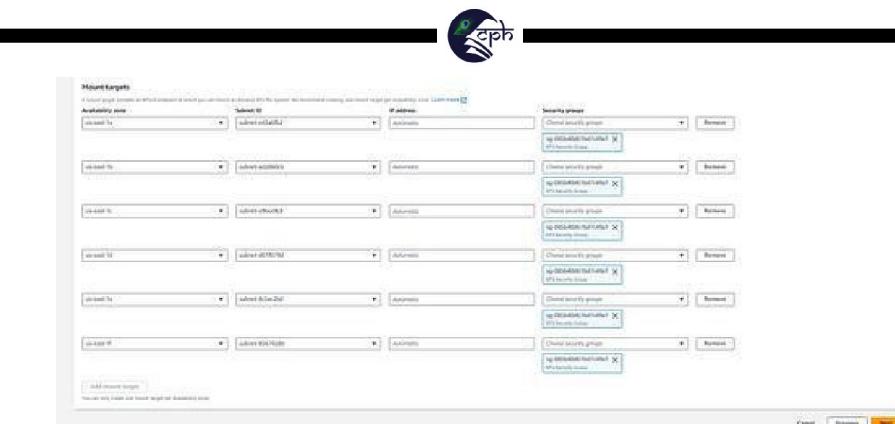
6. Leave everything by **default** and click on the **Next** button present below.
7. Network Access:

VPC:

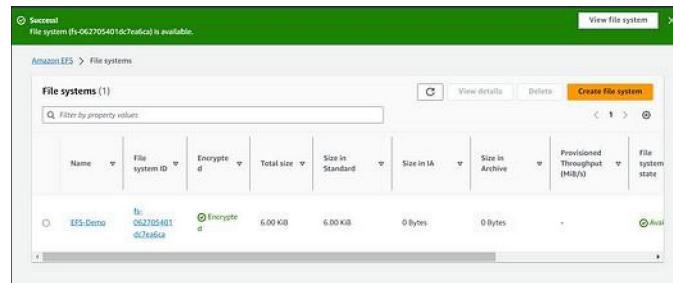
- An Amazon EFS file system is accessed by EC2 instances running inside one of your VPCs.
- Choose the same VPC you selected while launching the EC2 instance (leave as default).

Mount Targets:

- Instances connect to a file system by using a network interface called a mount target. Each mount target has an IP address, which we assign automatically or you can specify.
- We will select all the Availability Zones (AZ's) so that the EC2 instances across your VPC can access the file system.
- Select all the Availability Zones, and in the Security Groups, select EFS Security Group instead of the default value.
- Make sure you remove the default security group and select the EFS Security Group, otherwise you will get an error in further steps.
- Click on **Next** button.

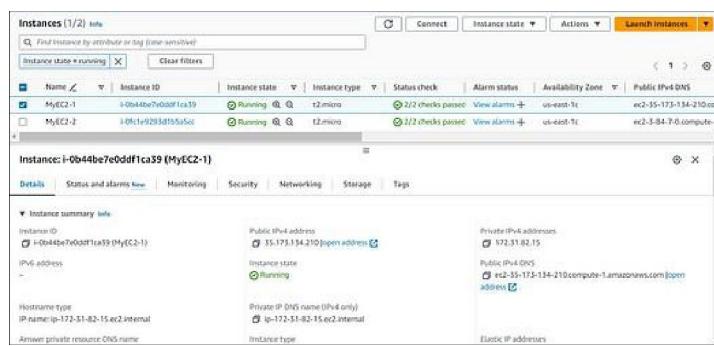


8. File system policy — optional let it be optional only. Click on Next button.
9. Review and Create: Review the configuration below before proceeding to create your file system. Click on Create button.
10. Congratulations on creating the EFS File system, it's time to mount your EC2 Instance with the EFS File system.



Step 4: Mount the File System to MyEC2-1 Instance

- Select the MyEC2-1 Instance and copy the IPv4 Public IP.
- SSH into the EC2 Instance
- Once instance is launched, Select EC2 Instance Connect option and click on Connect button.(Keep everything else as default)





- A new tab will open in the browser where you can execute the CLI Commands.

3. Switch to **root user**

```
sudo -s
```

4. Run the updates using the following command:

```
yum -y update
```

5. Install the NFS client as amazon-efs-utils.

```
yum install -y amazon-efs-utils
```

6. Create a directory by the name **efs**

```
mkdir efs
```

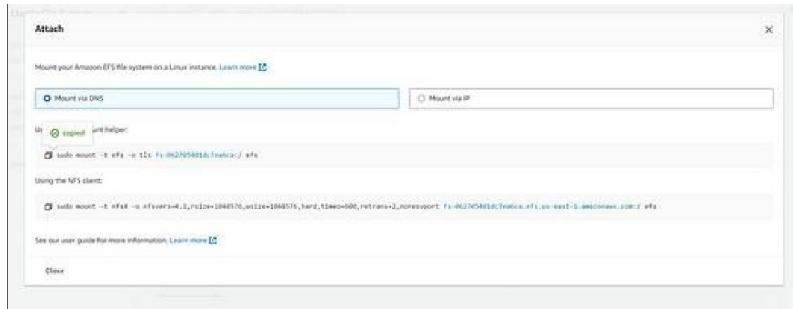
7. We have to mount the file system in this directory.

8. To do so, navigate to the AWS console and click on **the created file system**. On the top-right corner, click on View details then click on Attach

File systems (1)								
Name	File system ID	Encrypted	Total size	Size in Standard	Size in IA	Size in Archive	Provisioned Throughput (MiB/s)	File system state
EFS-Demo	fs-062705401dc7ea6ca	Encrypted	6.00 KB	6.00 KB	0 Bytes	0 Bytes	-	Available

EFS-Demo (fs-062705401dc7ea6ca)	
General	Edit
Performance mode	Automatic backups
General Purpose	Enabled
Throughput mode	Encrypted
Elastic	b2f15689-900f-4eef-925e-6f40a8285d88 (aws/elasticfilesystem)
Lifecycle management	File system state
Transition into Infrequent Access (IA): 30 day(s) since last access	Available
Transition into Archive: 90 day(s) since last access	DNS name
Transition into Standard: None	fs-062705401dc7ea6ca.efs.us-east-1.amazonaws.com
Availability zone	
Regional	

- Copy the command of Using the **EFS mount helper**



9. To display information for all currently-mounted file systems, we'll use the command bellow:

```
df -h
```

```
[root@ip-172-31-82-15 ec2-user]# sudo mount -t efs -o tls fs-062705401dc7ea6ca:/ efs
[root@ip-172-31-82-15 ec2-user]# df -h
Filesystem      Size   Used  Available  Mounted on
/devtmpfs       468M    0B   468M   /dev
tmpfs          477M    0B   477M   /dev/shm
tmpfs          477M   524K  476M   /run
tmpfs          477M    0B   477M   /sys/fs/cgroup
/dev/xvda1     8.0G  1.7G  6.4G   21%   /
tmpfs          96M    0B   96M   /run/user/0
tmpfs          96M    0B   96M   /run/user/1000
127.0.0.1:/    8.0B   0B   8.0B   /home/ec2-user/efs
[root@ip-172-31-82-15 ec2-user]#
```

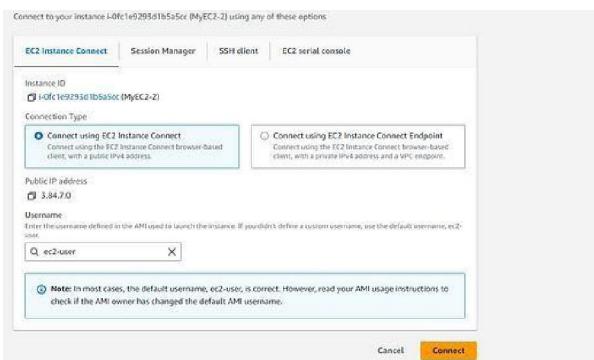
10. Create a directory in our current location:

```
mkdir aws
```

```
[root@ip-172-31-82-15 ec2-user]# mkdir aws
[root@ip-172-31-82-15 ec2-user]# ls
aws  efs
[root@ip-172-31-82-15 ec2-user]#
```

Step 5: Mount the File System to MyEC2-2 Instance

- Select the MyEC2-2 Instance and copy the IPv4 Public IP.
- SSH into the EC2 Instance
- Select EC2 Instance Connect option and click on Connect button.(Keep everything else as default)



- A new tab will open in the browser where you can execute the CLI Commands.



3. Switch to **root user**

```
sudo -s
```

4. Run the updates using the following command:

```
yum -y update
```

5. Install the NFS client as amazon-efs-utils.

```
yum -y install amazon-efs-utils
```

6. Create a directory with the name **efs**

```
mkdir efs
```

7. We have to mount the file system in this directory.

8. To do so, navigate to the AWS console and click on **the created file system**. On the top-right corner, click on **Attach**.

- Copy the command of Using the EFS mount helper into the CLI.
- To display information for all currently mounted file systems, we'll use the command:

```
df -h
```

```
[root@ip-172-31-94-52 ec2-user]# sudo mount -t efs -o tls fs-06270540dc7ea6ca:/ efs
[root@ip-172-31-94-52 ec2-user]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        468M   0B  468M  0% /dev
tmpfs          477M   0B  477M  0% /dev/shm
tmpfs          477M  524K  476M  1% /run
tmpfs          477M   0B  477M  0% /sys/fs/cgroup
/dev/xvda1     8.0G  1.7G  6.4G  21% /
tmpfs          968M   0B  968M  0% /run/user/0
tmpfs          968M   0B  968M  0% /run/user/1000
127.0.0.1:/    8.0G   0B  8.0G  0% /home/ec2-user/efs
[root@ip-172-31-94-52 ec2-user]#
```

Step 6: Testing the File System

1. SSH into both instances in a side-by-side view on your machine, if possible.

2. Switch to root user

```
sudo -s
```

3. Navigate to the **efs** directory in both the servers using the command

```
cd efs
```

4. Create a file in any one server.

```
touch hello.txt
```

5. Check the file using the command

```
ls -ltr
```

6. Now go to the other server and give the command

```
cd efs
```



7. You can see the file created on this server as well. This proves that our EFS is working.
8. You can try creating files (touch command) or directories (mkdir command) on other servers to continue to grow the EFS implementation.

```
[root@ip-172-31-94-52 ec2-user]# sudo -s
[root@ip-172-31-94-52 ec2-user]# cd efs
[root@ip-172-31-94-52 efs]# touch hello.txt
[root@ip-172-31-94-52 efs]# ls -ltr
total 4
-rw-r--r-- 1 root root 0 Jan 12 13:12 hello.txt
[root@ip-172-31-94-52 efs]#
```

```
[root@ip-172-31-82-15 ec2-user]# sudo -s
[root@ip-172-31-82-15 ec2-user]# cd efs
[root@ip-172-31-82-15 efs]# ls -ltr
total 4
-rw-r--r-- 1 root root 0 Jan 12 13:12 hello.txt
```



Demystifying AWS Load Balancers: Understanding Elastic, Application, Network, and Gateway Load Balancers

In the realm of cloud computing, load balancing plays a crucial role in distributing incoming traffic across multiple targets to ensure high availability, fault tolerance, and scalability of applications. In the high-traffic world of cloud applications, ensuring smooth operation and optimal performance requires a skilled conductor — the load balancer. AWS offers a robust suite of load balancers, each catering to specific needs. Amazon Web Services (AWS) offers a suite of load balancers tailored to different use cases and requirements. In this blog, we'll delve into the distinctions between AWS Elastic Load Balancer (ELB), Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GWLB), exploring their features, examples, and dissimilarities. Additionally, we'll shed light on the flow hash algorithm used by AWS load balancers to route traffic efficiently.

The Balancing Act: What They Do

At their core, all these load balancers perform the same essential function: distributing incoming traffic across a pool of resources, ensuring no single server gets overwhelmed. This enhances application availability and responsiveness for your users.

1. Elastic Load Balancer (ELB):

- **Description:** AWS Elastic Load Balancer (ELB) is the original load balancer service offered by AWS, providing basic traffic distribution across multiple targets within a single AWS region. It is a simple and cost-effective way to distribute traffic across multiple EC2 instances. ELB supports both HTTP and TCP traffic.
- **Example:** Distributing incoming traffic across multiple EC2 instances running web servers to ensure high availability and fault tolerance for a web application.

Features:

- Simple to configure and manage
- Supports HTTP and TCP traffic
- Can be used to distribute traffic across multiple EC2 instances
- Offers a variety of features, including health checks, sticky sessions, and SSL termination.

Use Cases:

- Distributing traffic to web servers
- Load balancing for TCP applications, such as databases and mail servers
- Providing SSL termination for web applications



2. Application Load Balancer (ALB):

- **Description:** AWS Application Load Balancer (ALB) operates at the application layer (Layer 7) of the OSI model, enabling advanced routing and content-based routing capabilities. ALB is a newer type of load balancer that is designed for modern applications. It offers a number of features that are not available in ELB, such as support for HTTP/2, WebSockets, and container-based applications. It's inspecting incoming requests based on factors like HTTP headers, path, or cookies. This allows for intelligent routing based on application logic. For instance, an ALB can direct traffic to a specific server based on the user's location or the type of request.
- **Example:** Routing traffic based on URL paths or hostnames to different backend services, such as directing /api requests to a set of API servers and /app requests to web servers.

Features:

- Supports HTTP/2, WebSockets, and container-based applications
- Offers a variety of features, including health checks, sticky sessions, and SSL termination
- Can be used to distribute traffic across multiple EC2 instances, containers, and Lambda functions

Use Cases:

- Load balancing for web applications
- Distributing traffic to microservices
- Load balancing for container-based applications

3. Network Load Balancer (NLB):

- **Description:** AWS Network Load Balancer (NLB) operates at the transport layer (Layer 4) of the OSI model, offering ultra-low latency and high throughput for TCP and UDP traffic. NLB is a high-performance load balancer that is designed for use with TCP applications. It offers very low latency and high throughput. This prioritizes speed and efficiency, making it ideal for high-volume, low-latency applications like gaming servers or chat platforms.
- **Example:** Load balancing traffic for TCP-based services such as databases, FTP servers, and gaming applications that require high performance and minimal overhead. NLB is ideal for applications that require low latency, such as gaming, financial trading, and video streaming.

Features:

- Very low latency and high throughput
- Supports TCP traffic
- Can be used to distribute traffic across multiple EC2 instances
- Offers a variety of features, including health checks and sticky sessions



Use Cases:

- Load balancing for TCP applications, such as gaming, financial trading, and video streaming
- Distributing traffic to EC2 instances that are running in a VPC.

4. Gateway Load Balancer (GLB):

- **Description:** The GLB is a versatile player, operating across layers 3 (network layer) and 7 (application layer). It acts as a central gateway for managing virtual appliances like firewalls or intrusion detection systems. It balances traffic across these appliances while maintaining secure communication through VPC endpoints. AWS Gateway Load Balancer (GLB) is designed for deploying, scaling, and managing third-party virtual appliances such as firewalls, intrusion detection systems (IDS), and encryption appliances. GLB is a load balancer that is designed for use with VPC endpoints. It allows you to load balance traffic to endpoints in a private VPC. GLB is ideal for applications that require access to private resources, such as databases and internal APIs.
- **Example:** Deploying a third-party firewall appliance to inspect and filter traffic between VPCs or between on-premises networks and the AWS cloud.

Features:

- Load balances traffic to endpoints in a private VPC
- Supports HTTP and TCP traffic
- Can be used to distribute traffic across multiple endpoints
- Offers a variety of features, including health checks and sticky sessions

Use Cases:

- Load balancing for applications that require access to private resources
- Distributing traffic to endpoints in a private VPC.

Similarities: A United Front

- **High Availability:** All load balancers ensure that even if individual instances fail, traffic seamlessly flows to healthy ones, keeping your application up and running.
- **Scalability:** They automatically adjust to traffic fluctuations, scaling resources up or down as needed.
- **Health Monitoring:** They constantly monitor the health of target instances and remove unhealthy ones from the pool.

Dissimilarities:

- **Layer of Operation:** ALB operates at Layer 7 (application layer), allowing for content-based routing, while NLB operates at Layer 4 (transport layer), focusing on routing traffic based on IP addresses and ports.



- **Performance Characteristics:** NLB offers ultra-low latency and high throughput for TCP and UDP traffic, making it ideal for high-performance applications, whereas ALB provides advanced routing features and supports WebSocket and HTTP/2 protocols.
- **Use Cases:** ALB is suitable for modern application architectures, microservices, and container-based environments, while NLB is preferred for TCP-based workloads requiring high performance and minimal overhead.
- **Routing Intelligence:** ALBs excel in application-level routing, while NLBs prioritize speed and efficiency.
- **Supported Protocols:** ALBs handle HTTP/HTTPS traffic, while NLBs work with TCP/UDP protocols.
- **Virtual Appliance Management:** GLBs are specifically designed for managing and scaling virtual appliances.

The following table summarizes the key dissimilarities between the four types of AWS load balancers:

Feature	ELB	ALB	NLB	GLB
Supported traffic	HTTP, TCP	HTTP/2, WebSockets, TCP	TCP	HTTP, TCP
Layer	Application	Application	Network	Application
Latency	Medium	Low	Very low	Medium
Throughput	Medium	High	Very high	Medium
Features	Health checks, sticky sessions, SSL termination, support for HTTP/2 and WebSockets	Health checks, sticky sessions, SSL termination, support for HTTP/2 and WebSockets	Health checks, sticky sessions	Health checks, sticky sessions
Use cases	Web servers, TCP applications	Web applications, microservices, container-based applications	TCP applications with low latency requirements	Applications that require access to private resources

dissimilarities between the four types of AWS load balancers

Flow Hash Algorithm: The flow hash algorithm is used by AWS load balancers to distribute incoming traffic across multiple targets while maintaining session affinity for stateful protocols. The flow hash algorithm calculates a hash value based on specific attributes of each incoming request, such as source IP address, destination IP address, source port, destination port, and protocol. This hash value is then used to determine which target receives the incoming request. The flow hash algorithm takes into account the source IP address, destination IP address, and destination port of each request. This ensures that requests from the same client are always sent to the same target.

The flow hash algorithm is a very effective way to distribute traffic evenly across multiple targets. It is also very efficient, as it does not require any additional overhead.

It takes a portion of the data flow (like source and destination IP addresses, ports) and generates a hash value. Based on this hash, the load balancer directs traffic to a specific instance. This ensures even distribution and prevents overloading individual instances.



Examples

Example 1: Load balancing a web application

You can use an ALB to load balance traffic to a web application that is running on multiple EC2 instances. The ALB will distribute traffic evenly across the instances and will ensure that requests from the same client are always sent to the same instance.

Example 2: Load balancing a TCP application

You can use an NLB to load balance traffic to a TCP application that is running on multiple EC2 instances. The NLB will provide very low latency and high throughput, making it ideal for applications that require low latency, such as gaming, financial trading, and video streaming.

Example 3: Load balancing traffic to a private VPC

You can use a GLB to load balance traffic to a private VPC. This allows you to load balance traffic to endpoints in a private VPC, such as databases and internal APIs.

Choosing the Right Load Balancer: It All Depends

Selecting the optimal load balancer hinges on your application's unique requirements:

- **ALB:** Ideal for web applications requiring intelligent routing based on application logic.
- **NLB:** Perfect for high-performance applications that prioritize speed and low latency.
- **GLB:** The go-to choice for managing and scaling virtual appliances within your network.

Conclusion: AWS offers a range of load balancing options, each tailored to different use cases and requirements. By understanding the distinctions between Elastic Load Balancer (ELB), Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GWLB), you can choose the right load balancing solution to optimize the performance, availability, and scalability of your applications in the AWS cloud. Additionally, the flow hash algorithm employed by AWS load balancers ensures efficient traffic distribution while maintaining session affinity, further enhancing the reliability and performance of your application deployments.

When choosing a load balancer, it is important to consider the following factors:

- The type of traffic that you need to load balance
- The latency and throughput requirements of your application
- The features that you need

By considering these factors, you can choose the right load balancer for your application and ensure that your traffic is distributed evenly and efficiently.



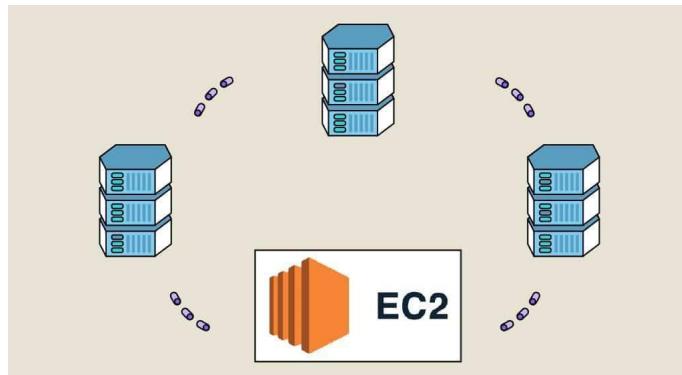
Create AWS Application Load Balancer (ALB)

If you are looking for a comprehensive guide on setting up an AWS Application Load Balancer (ALB) with two EC2 instances, displaying their IP addresses using a bash script, and demonstrating the load balancer's functionality, then you're in the right place!

In this step-by-step guide, we will take you through the entire process, starting with the basics and leading you through configuring the load balancer, setting up the instances, and testing the load balancer's functionality.

By the end of this guide, you will clearly understand how to set up an Application Load Balancer on AWS and use it to distribute traffic across multiple instances.

Set Up EC2 Instances



Launch EC2 Instances:

- Go to the AWS Management Console and navigate to the EC2 dashboard.
- Click on “Launch Instance” and choose an Amazon Machine Image (AMI) of your choice (e.g., Amazon Linux 2).
- Select an instance type, configure instance details, such as storage, tags
- Configure security group (allow HTTP and SSH), and review.
- Go to Advanced Settings and in user data add the following bash script to display the IP address of the instance:

```
#!/bin/bash
# install httpd (Amazon Linux 2) yum update -y
yum install -y httpd systemctl start httpd systemctl enable httpd
echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html
```

- Create instance



- Repeat this process to launch another EC2 instance.

Instances (1 / 2) Info								
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
hitc-ec2-demo-01	i-0576a031fb9dceba	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-52-91-115-127.co...	52.91.115.127
hitc-ec2-demo-02	i-093253b4d26d8d9e7	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-52-70-55-60.com...	52.70.55.60

Once the instances are ready copy their IP address and paste them into your Internet Browser to test it's working:

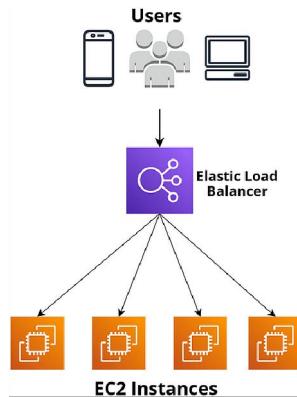
← → ⌂ 52.91.115.127
Hello World from ip-172-31-90-108.ec2.internal
hitc-ec2-demo-01

← → ⌂ 52.70.55.60
Hello World from ip-172-31-80-188.ec2.internal
hitc-ec2-demo-02

Assign Elastic IPs (Optional but recommended):

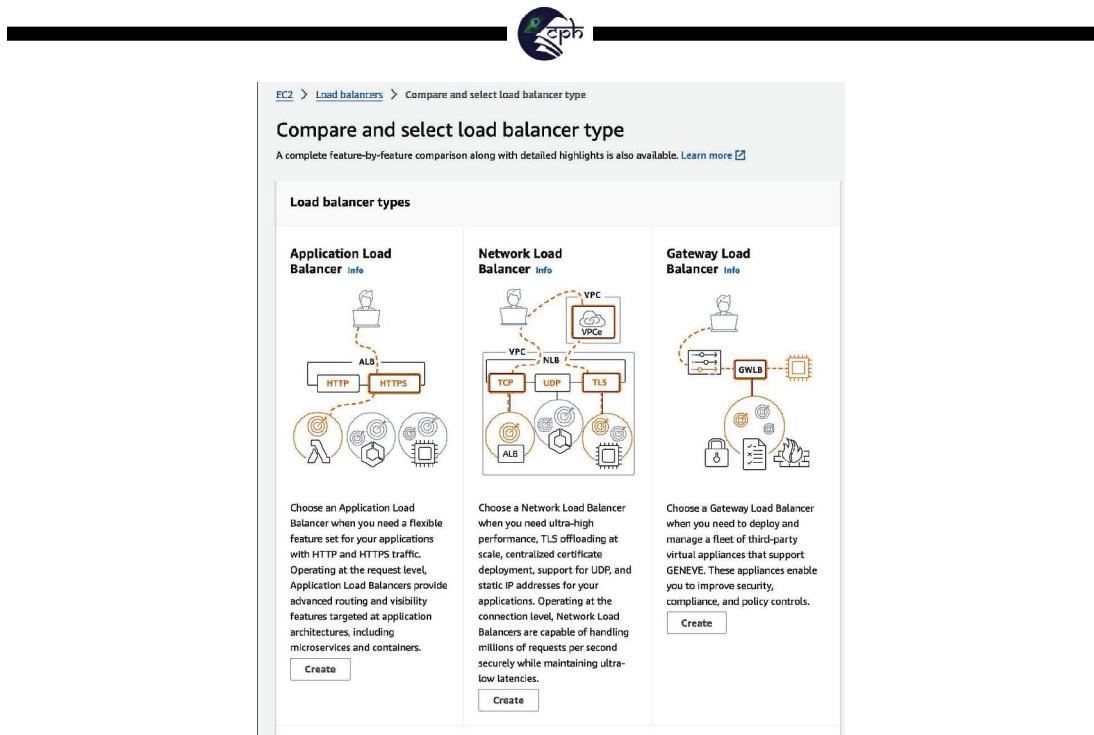
- Go to the “Elastic IPs” section in the EC2 dashboard.
- Allocate and associate an Elastic IP to each of your instances. This ensures that your instances have static public IP addresses.

Set Up Load Balancer



Create a Load Balancer:

- Go to the EC2 dashboard and click on “Load Balancers” under the “Load Balancing” section.
- Click on “Create Load Balancer” and choose “Application Load Balancer”.



- Configure load balancer name such as hitc-alb demo
- Scheme should be set to Internet-Facing
- IP Address Type to IPv4
- Network Mapping — select first 3 AZs in your selected region (e.g. us-east-1a, us-east-1b, us-east-1c)

The screenshot shows the 'Network mapping' configuration page for a load balancer. It includes sections for VPC and Mappings.

VPC Info: Selects a VPC (vpc-09f751bc0cd4d8fb1, IPv4: 172.31.0.0/16).

Mappings: Selects Availability Zones (us-east-1a, us-east-1b, us-east-1c) and their corresponding subnets and IPv4 addresses.

Availability Zone	Subnet	IPv4 address
us-east-1a (use1-az2)	subnet-09f32519aae759fa2	Assigned by AWS
us-east-1b (use1-az6)	subnet-01c5cd4d5d13a27c1a	Assigned by AWS
us-east-1c (use1-az1)	subnet-0b010afab5fc894e6	Assigned by AWS



Security group — Click on the link **Create a new security group** for ALB with the following config

The screenshot shows the 'Create security group' wizard. The 'Basic details' tab is selected, displaying fields for Security group name (hitc-sg-alb), Description (Allow HTTP into ALB), and VPC (vpc-059751bc4d668afb1). Below are two tabs: 'Inbound rules' and 'Outbound rules'. The 'Inbound rules' tab shows one rule: Type: HTTP, Protocol: TCP, Port range: 80, Source: Anywhere (IPv4) 0.0.0.0/0. The 'Outbound rules' tab shows one rule: Type: All traffic, Protocol: All, Port range: All, Destination: Gateway.

The screenshot shows the 'Security groups' page. It lists several security groups, with 'hitc-sg-alb' selected. Other listed groups include 'launch-wizard-1', 'default', and another 'hitc-sg-alb' entry.

Refresh and add newly created hitc-sg-alb

- In Listeners and routing click on **Create target group**. Target type should be set to Instances as we have 2 EC2 instances. Target group name could be hitc-tg-alb. Protocol set to HTTP. IP Address type should be set to IPv4 and Protocol version to HTTP1. Lastly, health checks should be set to HTTP. Click Next.
- In Register Targets, select both EC2 instances and click on the button **Include as pending** below, and then register the targets.

The screenshot shows the 'Register targets' step of the wizard. It displays a table of available instances: 'hitc-ec2-demo-01' and 'hitc-ec2-demo-02', both running and associated with 'launch-wizard-1'. Below the table, a note says '0 selected'. A dropdown for 'Ports for the selected instances' is set to '80'. A button 'Include as pending below' is visible. At the bottom is a 'Review targets' section.



In the next window we can check that the targets have been successfully registered.

The screenshot shows a CloudWatch Metrics Insights query results page. The query is:

```
aws_lambda.list_functions --region us-east-1 | jq .Functions[?FunctionName == "hitc-tg-alb"] | jq .FunctionArn
```

The results table has the following columns:

Function Name	ARN
hitc-tg-alb	arn:aws:lambda:us-east-1:105418075664:function:hitc-tg-alb

Below the table, there is a section titled "Distribution of targets by Availability Zone (AZ)" with a table showing target counts across three AZs: us-east-1d, us-east-1e, and us-east-1f.

Go back to your ALB setup, refresh and add the newly created target group

The screenshot shows the "Listeners and routing" section of the Load Balancers console. A new target group named "hitc-tg-alb" is being created under the "Listener HTTP:80" configuration. The "Default action" dropdown is set to "Select a target group" and the "Forward to" field is empty. The "Create target group" button is highlighted.

Click Create Load Balancer

The screenshot shows the "Load balancers" section of the Load Balancers console. It lists one load balancer named "hitc-alb-demo" with the ARN "arn:aws:elasticloadbalancing:us-east-1:111499920000:loadbalancer/hitc-alb-demo-1111499920000". The table includes columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created.

Wait a few moments until the Provisioning of the Load Balancer is completed

The screenshot shows the "Resource map" for the application load balancer "hitc-alb-demo". The map visualizes the relationships between the load balancer and its associated resources, such as subnets and security groups. The "Actions" menu at the top right includes options like "Create load balancer" and "Give feedback".



Then go back to Target Group that we previously created and check the status, that both registered targets are showing **Healthy Status**

Instance ID	Name	Port	Zone	Health status	Health status details	Launch time	Anomaly detection result
I-09325bb0ed216dc8bd7	htc-ec2-dene-02	80	us-east-1d	Healthy	-	March 20...	Normal
I-057ca031f994ebaf	htc-ec2-dene-01	80	us-east-1d	Healthy	-	March 20...	Normal

Test Load Balancer

- Select your load balancer, and copy its DNS name.
- Paste the DNS name into your browser.
- You should see the same page as before, but this time, the IP addresses displayed should be those of the instances as served by the load balancer.



Congratulations! You have successfully set up an AWS Application Load Balancer with two EC2 instances, displayed their IP addresses using a bash script, and demonstrated the load balancer's functionality.

Closing Thoughts

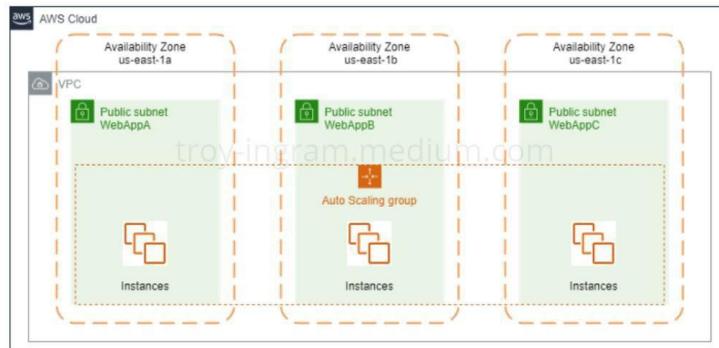
As you wrap up, it's important to know how to master the AWS Application Load Balancer (ALB) if you want to optimise your cloud infrastructure for scalability, reliability, and performance. This tutorial will teach you how to set up an ALB, configure EC2 instances, and use its features to distribute traffic efficiently.

It's important to regularly review and optimise your configurations to adapt to changing demands and ensure peak performance as you continue your journey with AWS and cloud computing. Experiment with different load balancing strategies, monitor your resources, and stay updated with best practices to stay ahead in this dynamic field.



AUTO SCALING

Achieve High Availability with AWS Auto Scaling



Use Case:

Your company has a web server that has a volatile level of traffic. Your company has to ensure that the webservers are always available and currently have a fixed amount of instances to guarantee that even at a max CPU Utilization, the web server will be able to perform. The problem is that when the traffic is low, the unused web servers are unnecessarily costing the company money. The current way of having a fixed number of instances also presents a problem if for some reason a web server goes down and a new one has to be manually spun up.

To solve this issue for our made-up ABC Company, we will create an Auto Scaling Group with a policy to scale in or out depending on demand with a minimum of 2 instances and a maximum of 5. One policy will scale out if CPU Utilization goes over 80% and the other will scale in if CPU Utilization goes under 40%.

Prerequisites

- Multi Availability Zone VPC with public subnets.
- A Webserver security group that allows inbound HTTP from 0.0.0.0/0 and SSH from your ip.

Launch Template

- Navigate to EC2 Dashboard.
- Click **Launch Templates** then click **Create launch template**.





3. Add a **Launch template name** and **Template version description**

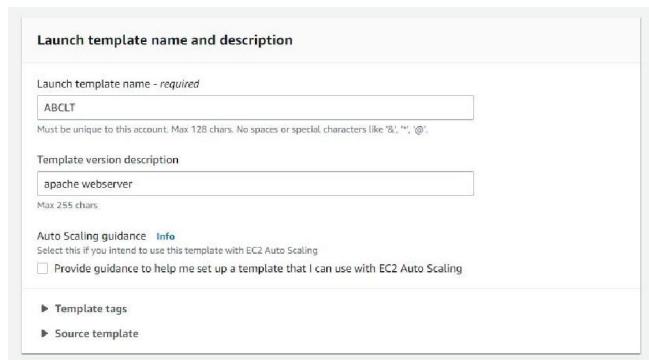
Launch template name and description

Launch template name - required
ABCLT
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '^', '@'.

Template version description
apache webserver
Max 255 chars

Auto Scaling guidance [Info](#)
Select this if you intend to use this template with EC2 Auto Scaling
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags
► Source template



4. For AMI select **Amazon Linux 2 AMI (HVM)**.

Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

▼ Amazon machine image (AMI) [Info](#)

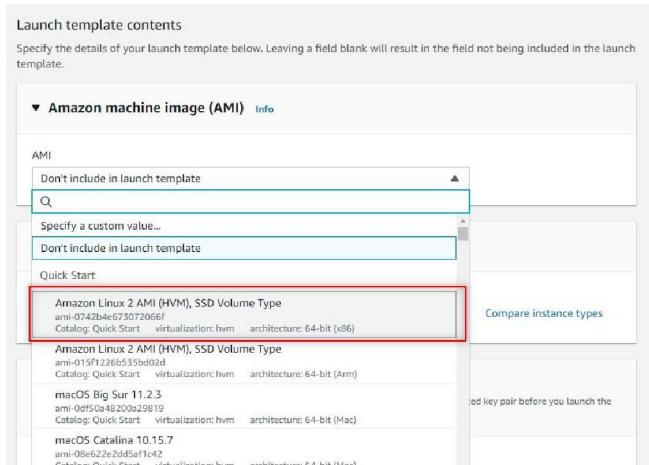
AMI
Don't include in launch template
Q
Specify a custom value...
Don't include in launch template

Quick Start

Amazon Linux 2 AMI (HVM), SSD Volume Type
ami-0742b4e67307206f6
Catalog: Quick Start virtualization: hvm architecture: 64-bit (x86)
Amazon Linux 2 AMI (HVM), SSD Volume Type
ami-01f1226b535bd02d
Catalog: Quick Start virtualization: hvm architecture: 64-bit (Arm)

macOS Big Sur 11.2.3
ami-0ef5c4820029619
Catalog: Quick Start virtualization: hvm architecture: 64-bit (Mac)

macOS Catalina 10.15.7
ami-08e522e2dd5af1c42
Catalog: Quick Start virtualization: hvm architecture: 64-bit (Mac)



5. For Instance type select **t2.micro**.

▼ Instance type [Info](#)

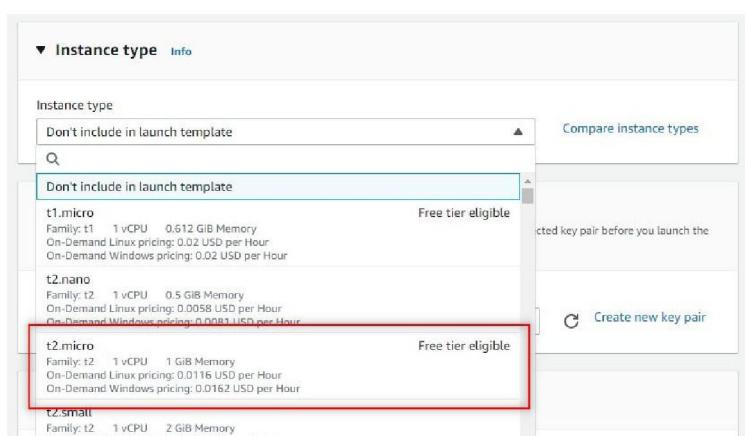
Instance type
Don't include in launch template
Q
Don't include in launch template

t1.micro
Family: t1 1 vCPU 0.612 GiB Memory
On-Demand Linux pricing: 0.02 USD per Hour
On-Demand Windows pricing: 0.02 USD per Hour
Free tier eligible

t2.nano
Family: t2 1 vCPU 0.5 GiB Memory
On-Demand Linux pricing: 0.0058 USD per Hour
On-Demand Windows pricing: 0.0083 USD per Hour

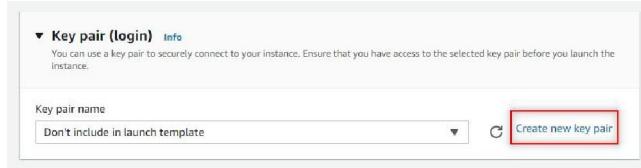
t2.micro
Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0116 USD per Hour
On-Demand Windows pricing: 0.0162 USD per Hour
Free tier eligible

t2.small
Family: t2 1 vCPU 2 GiB Memory

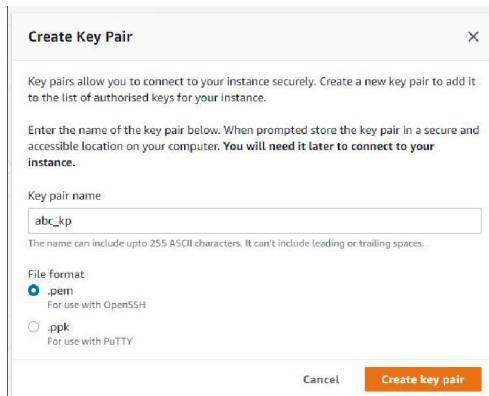




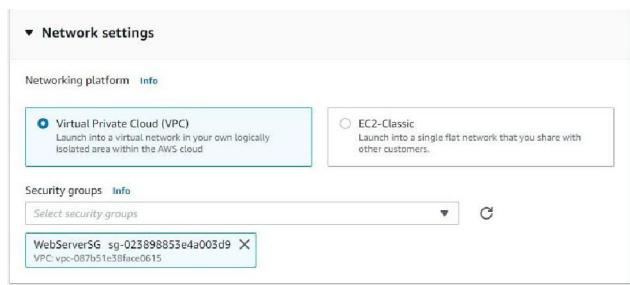
6. Since we plan to SSH into an instance later, you will need to select a key pair. You can use an existing Key pair or you can create a new one. To create a new one click **Create new key pair**.



7. A **Create Key Pair** dialog box should pop up. Enter a **Key pair name**. Select a **File format** and click **Create key pair**.



8. Back on the Create launch template page under Network settings select your **WebServer Security Group**.



9. Scroll down to **Advanced details** and click to expand.

10. Scroll down to User data. Copy and paste the following into the User data field:

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
```



This code will be run when each instance from the template boots up. It is updating patches, installing and starting an Apache webserver.

11. Click **Create launch template**.

Auto Scaling Group

1. Navigate to EC2 and select **Auto Scaling Groups** on the left side.



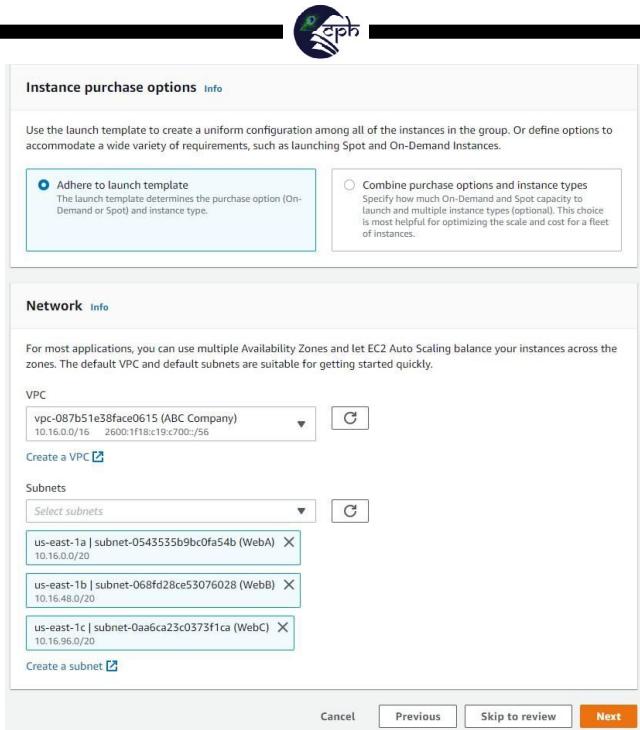
2. Click **Create Auto Scaling group**.
3. Enter a name for **Auto Scaling group name** and select our **Launch template** from the drop down.

Choose launch template or configuration Info

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.

Name		
Auto Scaling group name Enter a name to identify the group. <input type="text" value="ABCASG"/> 		
Launch template <small>Info</small> Switch to launch configuration		
Launch template Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups. <input type="text" value="ABCLT"/> 		
Version <input type="button" value="Default (1)"/> 		
Create a launch template version [Create]		
Description apache webserver	Launch template ABCLT 	Instance type t2.micro

4. Review the information and click **Next**.
5. On the **Configure Settings** page select **Adhere to launch template** for **Instance purchase options**.
6. For **Network**:
 - **VPC**: Select the VPC you'd like to launch the instances in. Be sure to select the VPC in which your Security Group is associated
 - **Subnets**: Select all public subnets in your VPC in your web layer (if you have a tiered architecture). In my custom VPC for ABC Company, I have 3 public subnets and have chosen all 3 for high availability.



7. Click **Next**.
8. On the **Configure advanced options** page select **No load balancer** and keep the default **Health checks**. Then click **Next**.
9. On the **Configure group size and scaling policies** page:

Desired capacity: 2

 - **Minimum capacity:** 2
 - **Maximum capacity:** 5

This will ensure that we always have at least 2 instances running and up to 5 if CPU Utilization gets too high.
10. For **Scaling policies** select **None** and click **Next**.
11. I don't plan to add **notifications** or **tags** so click **Skip to review**. Feel free to add either if you'd like, but it's not necessary for this project.
12. Click **Create auto scaling group**.
13. If you navigate to the Activity tab you will see that two instances have been created to meet our minimum capacity.

The screenshot shows the AWS CloudWatch interface for Auto Scaling groups. The top navigation bar includes 'EC2', 'Auto Scaling groups', and 'ABCASG'. Below this, there are tabs for 'Details', 'Activity' (which is selected), 'Automatic scaling', 'Instance management', 'Monitoring', and 'Instance refresh'. The 'Activity notifications (0)' section has a search bar and a 'Create notification' button. The 'Activity history (2)' section lists two successful events:

- Launching a new EC2 instance - At 2021-04-15T10:05:17Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2021-04-15T10:05:40Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.
- Launching a new EC2 instance - At 2021-04-15T10:05:12Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2021-04-15T10:05:40Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.

CloudWatch Alarms

We will need to create two CloudWatch alarms that will trigger our Scaling Policies if they go into an alarm state.

1. Navigate to **CloudWatch**. Click **Alarms** then **Create Alarm**.

The screenshot shows the CloudWatch Alarms page. On the left, there's a sidebar with 'Metrics' (502), 'Alarms' (1), 'Logs' (0), 'Metrics Insights' (0), and 'CloudWatch Metrics' (0). The main area shows a list of alarms with columns for 'Name', 'Last updated', 'State', and 'Actions'. A prominent orange 'Create alarm' button is at the bottom right.

2. Click **Select metric**.
3. Click **EC2**.

The screenshot shows the CloudWatch Metrics page. It displays a grid of metrics categorized by service: Backup (28), Billing (22), EBS (131), EC2 (255, highlighted with a red box), NATGateway (68), RDS (88), SNS (4), and Usage (6). A 'Graphed metrics' button is located at the top right.

4. Click **By auto scaling group**
5. If you have multiple auto scaling groups then you can use search to find the one, we created by typing our auto scaling group name. Then select the one for **CPU Utilization**.

The screenshot shows a modal dialog titled 'Metrics (17) info'. It lists metrics under 'AutoScalingGroupName (17)' for 'ABCASG'. The 'CPUUtilization' metric is selected and highlighted with a blue border. Other metrics listed include StatusCheckFailed_System, StatusCheckFailed_Instance, CPUCreditUsage, NetworkIn, and NetworkOut. A 'Graph search' and 'Graphed metrics (1)' button are at the top right, and 'Cancel' and 'Select metric' buttons are at the bottom right.



6. Click **Select metric**.
7. Keep all **metric** defaults.
8. For the Conditions section:
 - **Threshold type:** Static
 - **Whenever <alarm name> is:** Greater/Equal than: 80

Conditions

Threshold type

Static
Use a value as a threshold

Anomaly detection
Use a band as a threshold

Whenever CPUUtilizationGreaterThan80 is...
Define the alarm condition.

Greater
> threshold

Greater/Equal
>= threshold

Lower/Equal
<= threshold

Lower
< threshold

than...
Define the threshold value.

Must be a number

► Additional configuration

Cancel **Next**

9. click **Next**.
10. You can add a notification if you'd like but I'm going to click **Remove**. One can always be added later.

Configure actions

Notification

Alarm state trigger
Define the alarm state that will trigger this action.

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.

Remove

Select an SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic

Use topic ARN

Send a notification to...

⚠ Required
Only email lists for this account are available.

Add notification



11. Name our alarm and add a description then click **Next**.

Add name and description

Name and description
Alarm name Simple-Scaling-AlarmHigh-AddCapacity
Alarm description - optional Add capacity if <u>cpuutilization</u> is over 80%.
Up to 1024 characters (43/1024)
<input type="button" value="Cancel"/> <input type="button" value="Previous"/> <input type="button" value="Next"/>

12. Review then click **Create alarm**.

13. Repeat steps 1–12 for our scale in alarm with the following changes:

Step 8: Lower/Equal 40

Step 11: Alarm name: Simple-Scaling-AlarmLow-SubtractCapacity

Note: When you first create your alarms the Status will say insufficient Add Scaling Policy
Navigate to **Auto Scaling Groups** and click our newly created auto scaling group.

Auto Scaling groups (1)								
<input type="button" value="Create an Auto Scaling group"/>								
<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Actions"/>								
<input type="checkbox"/>	Name	ABCASG	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
		ABCLT Version Default		2	-	2	2	5
							us-east-1a, us-east-1b, us-east-1c	

2. Select the **Auto scaling** tab and click **Add policy**.

Automatic scaling					
<input type="button" value="Add policy"/>					
No scaling policies are currently specified					
<input type="button" value="Add policy"/>					

3. For **Policy type** select **Simple scaling**. For **CloudWatch alarm** select the **AddCapacity** alarm and for **Take the action** select **Add 1 capacity units**.



EC2 > Auto Scaling groups > ABCASG

Create scaling policy

Policy type: Simple scaling

Scaling policy name: AddCapacity

CloudWatch alarm: Simple-Scaling-AlarmHigh-AddCapacity

Take the action: Add 1 capacity units

And then wait: 300 seconds

Create

4. Repeat steps 1–3 but to remove capacity.

EC2 > Auto Scaling groups > ABCASG

Create scaling policy

Policy type: Simple scaling

Scaling policy name: Subtract Capacity

CloudWatch alarm: Simple-Scaling-AlarmLow-SubtractCapacity

Take the action: Remove 1 capacity units

And then wait: 300 seconds

Create

5. You should now see two policies added to the Auto Scaling Group.

The screenshot shows the AWS EC2 Auto Scaling Groups interface. The 'Activity' tab is selected. Under 'Activity notifications (0)', there is a search bar and a 'Create notification' button. Below it, under 'On instance action', there is a 'Send to' section and a note stating 'No notifications are currently specified'. Under 'Activity history (2)', there is a table with columns: Status, Description, Cause, Start time, and End time. Two entries are listed:

Status	Description	Cause	Start time	End time
Successful	Launching a new EC2 instance - i-037680016825bda64	At 2021-04-15T10:05:17Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2021-04-15T10:05:40Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2021 April 15, 06:05:42 AM -04:00	2021 April 15, 06:05:45 AM -04:00
Successful	Launching a new EC2 instance - i-0c93fe7bf51e7062c	At 2021-04-15T10:05:17Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2021-04-15T10:05:40Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2021 April 15, 06:05:42 AM -04:00	2021 April 15, 06:05:45 AM -04:00

Testing

Let's first test to make sure our apache web server is running.

- Navigate to the **EC2 Dashboard**.
- Click **Instances**.
- Select one of our instances.
- In the **details** tab, in the **Instance summary** section copy the **Public IPv4 DNS** address and paste it into a new tab.

The screenshot shows the AWS EC2 Instance details page for instance i-0c93fe7bf51e7062c. The 'Details' tab is selected. In the 'Instance summary' section, the 'Public IPv4 DNS' field contains the value 'ec2-54-221-128-81.compute-1.amazonaws.com' with a red box around it. Other fields shown include 'Instance ID' (i-0c93fe7bf51e7062c), 'Instance state' (Running), and 'Instance type' (t2.micro).

You should see your Test Page.

The screenshot shows a browser window displaying the Apache test page. The URL is 'http://ec2-54-221-128-81.compute-1.amazonaws.com'. The page content includes a red header bar with 'Test Page', a message for website administrators, and a footer with the Apache logo.

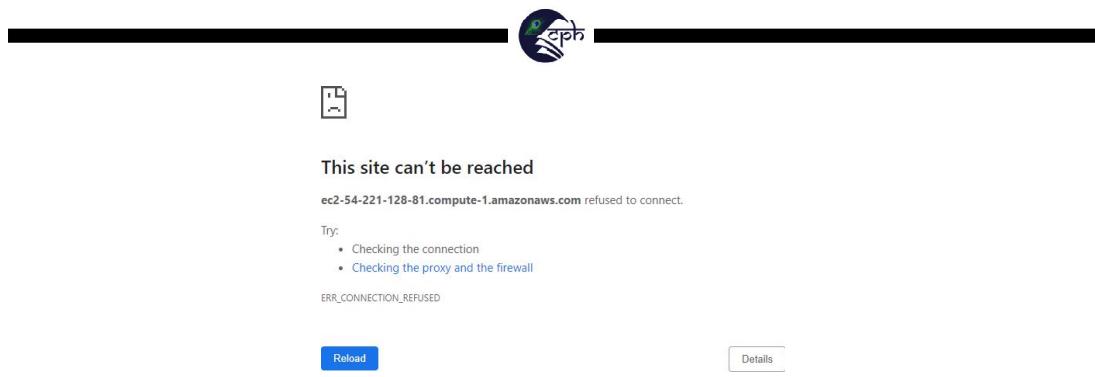
If you are the member of the general public:
The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you suspect the site is not the administrator of the website you have seen this page instead of the page you expected, you should send them an email (preferably, mail sent to the name "webmaster" and directed to the website's domain should reach the website's owner).

For example, if you experienced problems while visiting `www.example.com`, you should send e-mail to "`webmaster@example.com`".

Success!

DO NOT, I REPEAT DO NOT click the **open address link.** When you click this link, it will open the link in a new tab which on the surface seems like what you would want to do, however please learn from my mistake. When you click this link, it will open up the IPv4 DNS in https and since we have not set up https, we will get an error. I spent an hour troubleshooting what the possible issue could be before realizing my mistake.



A valuable learning experiences

Let's test to see if our Auto Scaling Group works if an instance was to fail and go below our minimum capacity.

- Navigate to the **EC2 Dashboard**.
- Click **Instances**.
- Select one of our instances. Click Instance state and click Terminate instance.

Instances (1/2) Info							
Filter instances		Instance state ▲					
Instance state: running X		Clear filters					
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available	v4 D
-	i-0376800168255b464	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	Terminate instance
-	i-0c93f67bf51e7052c	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	652-34-221-1

4. After some time you should see a new instance created.

Instances (2) Info							
Filter instances		Instance state ▲					
Instance state: running X		Clear filters					
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available	v4 D
-	i-0e0a3fd1a4d14d04d	Running	t2.micro		Initializing		

5. You can navigate to the IPv4 DNS for the new instance to verify the apache web server is working.

Let's test CPU Utilization

- SSH into both of your running instances.
- Make sure to run them both around the same time. It will do no good to run the commands on one instance, wait 10 minutes, then run the command on the other instance.

Run the following:

```
sudo amazon-linux-extras install epel -y
sudo yum install -y stress
```

3. Then run the following:

```
stress --cpu 12 --timeout 600
```



Note: It took some time before both instances were maxed out. You may want to play with upping the cpu number if needed.

3. Once our alarm status changes to **In alarm**, we should see our Auto Scaling Group launch a new instance.
5. Navigate to **Auto Scaling groups**, select our group and review the **Activity history**.

The screenshot shows the AWS Auto Scaling Groups activity history. A single entry is highlighted:

Status	Description	Cause	Start Time	End Time
Provisioning	Launching a new EC2 instance: i-02e6fd7a6d84a6ec	Scaling activity triggered policy AddCapacity changing the desired capacity from 2 to 3. At 2021-04-15T12:13:46Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.	2021-04-15, 08:12:48 AM	-0400

6. You should also now see three instances when you navigate to **Instances**.

The screenshot shows the AWS Instances page. Three instances are listed, all in the 'running' state:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
i-02e6fd7a6d84a6ec	i-02e6fd7a6d84a6ec	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-52-235-241-228.co...	3.235.241.228	-
i-0ef3af3f1a4d1404d	i-0ef3af3f1a4d1404d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1d	ec2-3-94-208-146.con...	3.94.208.146	-
i-0e0fd101454e7082c	i-0e0fd101454e7082c	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-54-221-128-81.co...	54.221.128.81	-

7. You can either wait until our commands stop running or you can cancel in the terminal with **Ctrl + C**.
8. Once CPU Utilization goes below 40%, we should see a scale in action triggered by our other alarm. Navigate back to **Auto Scaling groups**.
9. Select the **Activity** tab and note that the Auto Scaling group terminated and instance.

The screenshot shows the AWS Auto Scaling Groups Activity tab. A single entry is highlighted:

Status	Description	Cause
Successful	Terminating EC2 instance: i-0e0a3fd1a4d14d04d	At 2021-04-15T12:21:20Z a monitor alarm Simple-Scaling-AlarmLow-SubtractCapacity in state ALARM triggered policy SubtractCapacity changing the desired capacity from 3 to 2. At 2021-04-15T12:21:25Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2021-04-15T12:21:25Z instance i-0e0a3fd1a4d14d04d was selected for termination.



AWS Web Application Firewall

There are **many security threats** that exist today in a typical enterprise distributed application.

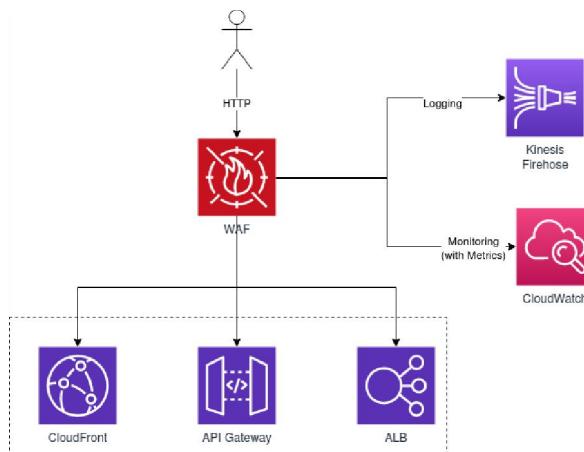
- **DDoS:** Flood Attacks (SYN Floods, UDP Floods, ICMP Floods, HTTP Floods, DNS Query Floods), Reflection Attacks
- **Application Vulnerabilities:** SQL Injections, Cross Site Scripting (XSS), Open Web Application Security Project (OWASP), Common Vulnerabilities and Exposures (CVE)
- **Bad Bots:** Crawlers, Content Scrapers, Scanners and Probes

Out of these, AWS WAF can be used to handle security threats such as SQL injections, Cross Site Scripting (XSS) in a typical web application.

The web application HTTP requests, can be routed via AWS WAF and then will be forwarded to either one of the AWS services.

- AWS CloudFront (A Global Service)
- AWS API Gateway (A Regional Service)
- AWS Application Load Balancer (A Regional Service)

Logging and Monitoring of WAF are handled by *Kinesis Firehose* and *CloudWatch* respectively.



Web ACL

When WAF associating any of the above three AWS services, it associates with a Web ACL. A Web ACL is a fundamental component of WAF, which defines a set of rules for any of these services (See Figure 2).

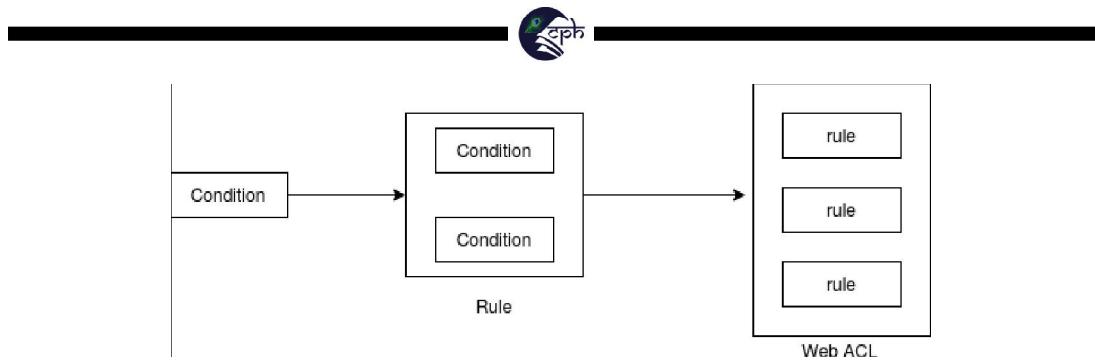


Figure 2 — Conditions, Rules and Web ACLs

As mentioned, a Web ACL is a collection of **rules**. A rule is a collection of **conditions** (See Figure 3).

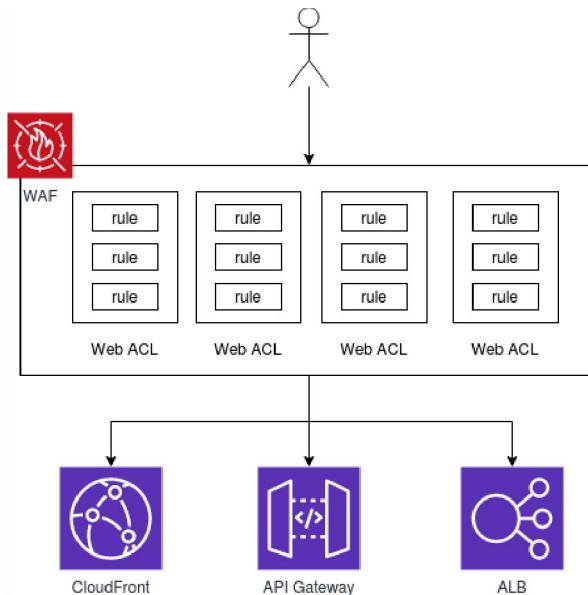


Figure 3- WAF with Web ACLs

How to create a Web ACL in WAF?

In order to demonstrate the WAF capability, it is always good to go through a simple scenario that can showcase its capability. Here, I am going to block a CloudFront distribution, which I created some time ago.

So, if you are trying this out, please make sure you have one of the services (CloudFront, API Gateway or ALB) is created already before trying this out.

Task 1: Describe a Web ACL and associate it to AWS resources

Go to AWS WAF → Web ACLs → Click *Create Web ACL* button (See Figure 4).

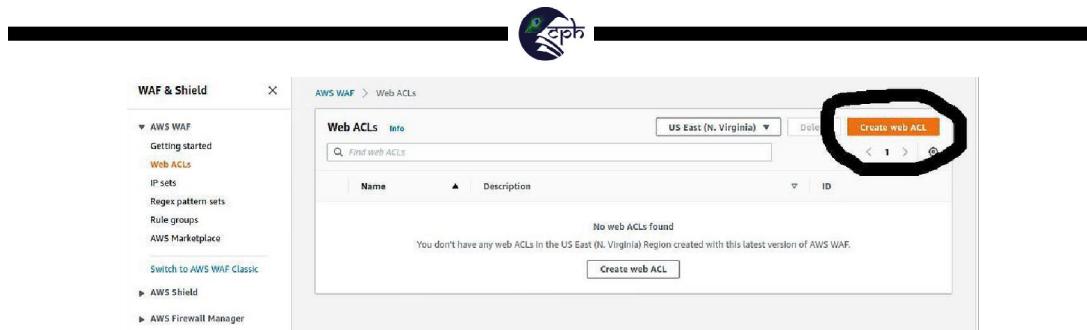


Figure 4

Give a name to Web ACL and associate a Resource Type to it. Here we are associate a CloudFront distribution (See Figure 5), which I have already created before. You can attach this to not only CloudFront but ALB and API Gateway as well.

This is a screenshot of the 'Step 1: Describe web ACL and associate it to AWS resources' screen. It has five steps listed on the left: Step 1 (current), Step 2, Step 3, Step 4, and Step 5. The main area is titled 'Web ACL details' and contains fields for 'Name' (set to 'Test-WebACL'), 'Description - optional' (empty), 'CloudWatch metric name' (set to 'Test-WebACL'), and 'Resource type' (radio button selected for 'CloudFront distributions'). Below these is a 'Region' dropdown set to 'Global (CloudFront)'. There is also a note at the bottom: 'There are no resources to display.'

Figure 5

Click *Add AWS Resources* button to associate the CloudFront Distribution that you created before (See Figure 6).

This is a screenshot of the 'Add AWS resources' dialog box. It has a 'Resource type' section with a radio button selected for 'CloudFront Distribution'. Below this is a search bar and a table with one row, 'E7C7LJ972N1QW - mithum.com', which has a checked checkbox next to it. At the bottom are 'Cancel' and 'Add' buttons.

Figure 6



Click *Next* button and you will get another page to add your rules to Web ACL. We will skip this for the moment allowing us to do it at a later stage.

Select *Allow* for Web ACL Action as well.

Leave *Set Rule Priority* as it is and click *Next*.

Leave *Configure Metrics* and click *Next*.

Finally review your selections and click *Create Web ACL* button.

The above will create a Web ACL without any rules. You can go back to Web ACL link and you will see the below. Make sure not to select a region and select *Global (CloudFront)* in the top drop down to see your created Web ACL (See Figure 7).

Web ACLs		Info	Global (CloudFront) ▾	Delete	Create web ACL
		Find web ACLs	< 1 > ⌂		
Name	Description	ID			
Test-WebACL	-	d588bd8c-0fdf-4b3d-b8bd-86f7bdc69e2e			

Figure 7

However, even if you see a created Web ACL, CloudFront propagation for this update will take a bit of time. You can see it if you visit the CloudFront console page. Give a little bit of time finish the CloudFront propagation before you start the next step.

Task 2: Add a Condition to block my IP address

Go to AWS WAF → IP Sets → Click *Create IP Set* button.

Select IPV4 and give your IP address with /32 as the postfix. If you are not sure how to get your network's public IP, you may type "What is my IP" on Google. It is that simple (See Figure 8).

The screenshot shows the 'Create IP set' interface in the AWS WAF console. The 'IP set details' section includes:

- IP set name:** BlockMyIP
- Description (optional):** (empty)
- Region:** Global (CloudFront)
- IP version:** IPv4 (selected)
- IP addresses:** 128.199.220.1/32

Figure 8



Task 3: Add a Rule to the created condition

In order to create a rule, you need to create a Rule Group.

Go to AWS WAF → Rule Group → Click *Create Rule Groups* button (See Figure 9)

The screenshot shows the 'Describe rule group' step of the AWS WAF Rule Group creation wizard. The 'Name' field is set to 'MyRuleGroup'. The 'Description - optional' field contains a single character. The 'CloudWatch metric name' field is also set to 'MyRuleGroup'. Under the 'Region' section, 'Global (CloudFront)' is selected. On the left sidebar, 'Step 1: Describe rule group' is highlighted, followed by 'Step 2: Add rules and set capacity', 'Step 3: Set rule priority', and 'Step 4: Review and create rule group'.

Figure 9

Click *Next* → Click *Add Rule* button → Set the following parameters to create a Rule
Rule Name → *MyRule*

If a Request → Select *Matches the requirement*

Statement (Inspect) → Select *Originates from an IP Address In* Statement (IP Set) →
Select the IP Set that you created in Task 2 Action → Select *Block*

Click *Next*

Select the *Rule Priority*. This is not required here since you have only one rule.

Finally review your selections and click *Create Rule Group* to confirm your rule settings.

Task 4: Add the created Rule Group / Rule to the Web ACL

Go to AWS WAF → Web ACL → Select the *Web ACL* that you have
created → Click *Rules* tab (See Figure 10).

The screenshot shows the 'Rules' tab for the 'Test-WebACL' in the AWS WAF. The interface includes a search bar and buttons for 'Edit', 'Delete', and 'Add rules'. There are no results listed under the 'Rules' table. The left sidebar shows the navigation path: AWS WAF > Web ACLs > Test-WebACL. A success message at the top states: 'You successfully updated the web ACL Test-WebACL.'

Figure 10

You can see the Web ACL still does not have its rules attached.

Click *Add Rules* button drop down → Select *Add my own rules and rule groups*

Figure 11

Give a name for the rule that you are specifying here (See Figure 11). [P.Note: I strongly feel the new WAF UI has some issues related its fields. This is a good example of having to define Rule name twice.

Once under the Rules Group and once under Web ACL rule attachments.]

Select the Rules Group that you created from the drop down and click *Add rule* button and then click *Save*.

Now you can see the added rule is attached to the Web ACL.

Now it is time to browse the web URL that you have blocked for your IP. If all fine, it will be similar to below screen (See Figure 12).

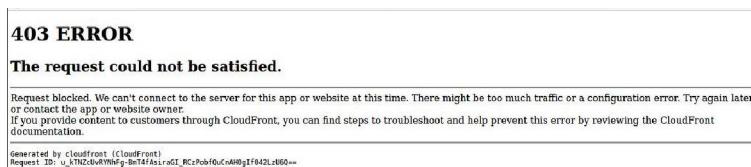


Figure 12

If you want to remove the blocking, you can go to the Web ACL and delete the related Rule and try the web link again. After a few refresh attempts, you will get your site back.



Amazon Web Application Firewall (AWS WAF): Web Security for AWS Users



What is an AWS WAF ?

AWS Web Application Firewall (WAF) is a firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources. AWS WAF gives you control over how traffic reaches your applications by enabling you to create security rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that filter out specific traffic patterns you define. You can deploy AWS WAF on either Amazon CloudFront as part of your CDN solution or the Application Load Balancer (ALB) that fronts your web servers or origin servers running on EC2.

Key Features of AWS WAF

- Customizable Rules:** Create rules to filter traffic based on conditions like IP addresses, HTTP headers, and body contents.
- Real-Time Metrics and Logging:** Monitor web traffic and get real-time metrics and logs for in-depth analysis.
- Integration with AWS Services:** Seamlessly integrates with services like Amazon CloudFront and Application Load Balancer.

Compare AWS Security Groups, NACL, AWS WAF, Network Firewall

AWS Security Groups vs NACLs vs WAF vs Network Firewall

Security Groups		NACLs		WAF	Network Firewall
Protection	Instance Level	Subnet Level	Endpoint Level (ALB, API Gateway, CloudFront)	VPC Level	
State	Stateful	Stateless	Stateless	Stateful & Stateless	
OSI Layer	Layer 3/4	Layer 3/4	Layer 7	Layer 3-7	
Flows	Ingress & Egress at instance level	Ingress & Egress at subnet level	Ingress from Internet to Endpoint level	Ingress & Egress at VPC level	
Features	IP, Port, Protocol Filtering	IP, Port, Protocol Filtering	Application Layer Filtering	Stateless/ACL L3 rules, Stateful/L4 rules, IPS-IDS/L7 rules, FCDN filtering, Protocol detection, Large IP block/allow lists	



How Amazon Web Application Firewall (WAF) Works ?

The working of WAF in AWS mentioned below.

- **AWS Firewall Manager:** It Manages multiple AWS Web Application Firewall Deployments.
- **AWS WAF:** Protect deployed applications from common web exploits.
- **Create a Policy:** Now you can build your rules using the visual rule builder.
- **Block Filter:** Block filters protect against exploits and vulnerability attacks.
- **Monitor:** Use Amazon CloudWatch for incoming traffic metrics & Amazon Kinesis Firehose for request details, then tune rules based on metrics and log data.

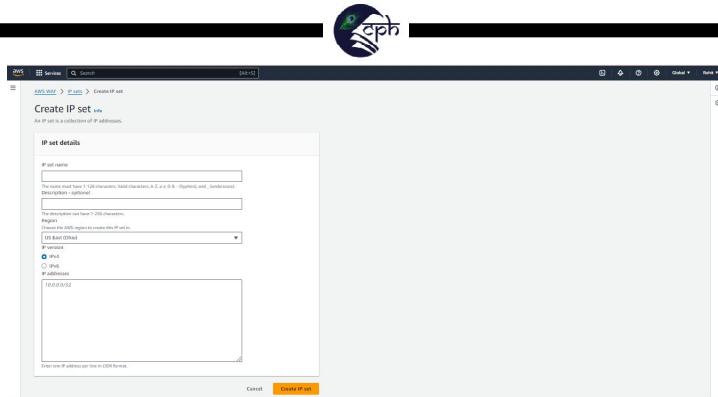
WAF AWS monitors all incoming and outgoing web requests forwarded to API Gateway, Amazon CloudFront, and Application Load Balancer.

Now let's get started with WAF and create web ACL in some steps.

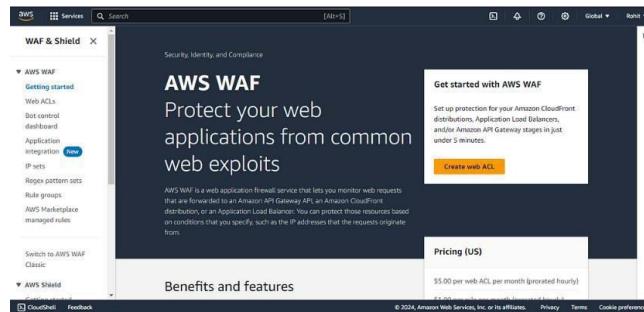
Step 1: Create web ACL:

First, sign up for an AWS account, then go to AWS Console and search for Web Application Firewall. You will land on the WAF home page.

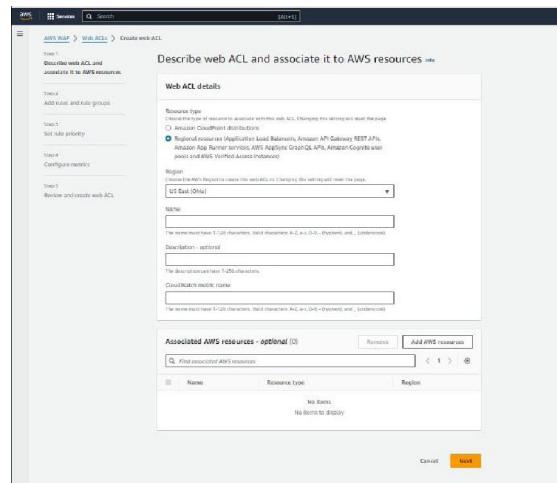
Step 2: In the next step you need to create the IP Set to deny the application access. Click on IP Set then select Create IP Set then add the IP list then click on Create IP set which needs to block the access of the application. The IP that we have added to the list does not access the application over the Internet.

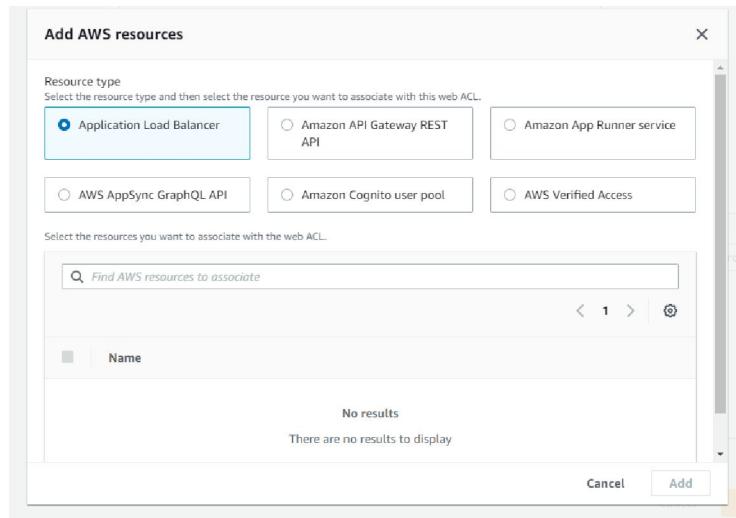


Step 3: Create web ACL: Open a new tab of the browser then go to AWS Console and search for Web Application Firewall. You will land on the WAF home page, and choose to **Create Web ACL**.



Step 4: Give a Name: Type the name you want to use to identify this web ACL. After that, enter Description if you want (optional), add the AWS resources (Application Load Balancer, Amazon API Gateway REST API, Amazon App Runner service, AWS AppSync GraphQL API, Amazon Cognito user pool, AWS Verified Access), and then hit **Next**.





Step 3: Add your Own rules and rule group: In the next step, you need to add rules and rule groups. Click on Add my own rules and rule groups. You will land on a new page to Rules type then select IP Set and choose the IP set which is created in Step2 and click on the add rule option mentioned in the below snapshot.

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add rules and rule groups info

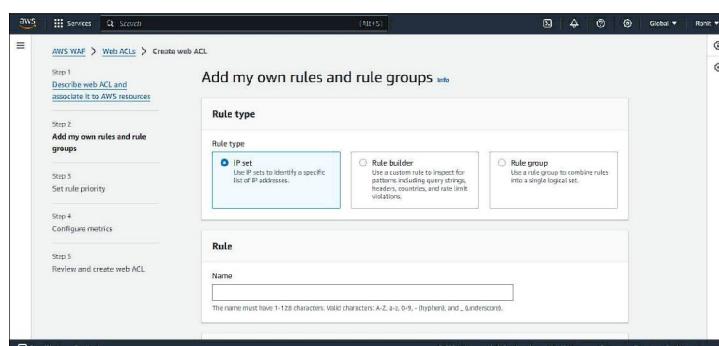
A rule defines attack patterns to look for in web requests and the action to take when a request matches the patterns. Rule groups are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Name	Capacity	Action
No rules.		You don't have any rules added.

Web ACL capacity units (WCUs) used by your web ACL

The WCUs used by the web ACL will be less than or equal to the sum of the capacities for all of the rules in the web ACL.

The total WCUs for a web ACL can't exceed 5000. Using over 5000 WCUs affects your costs. [AWS WAF Pricing](#)



Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add my own rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add my own rules and rule groups info

Rule type

IP set Use IP sets to identify a specific list of IP addresses.

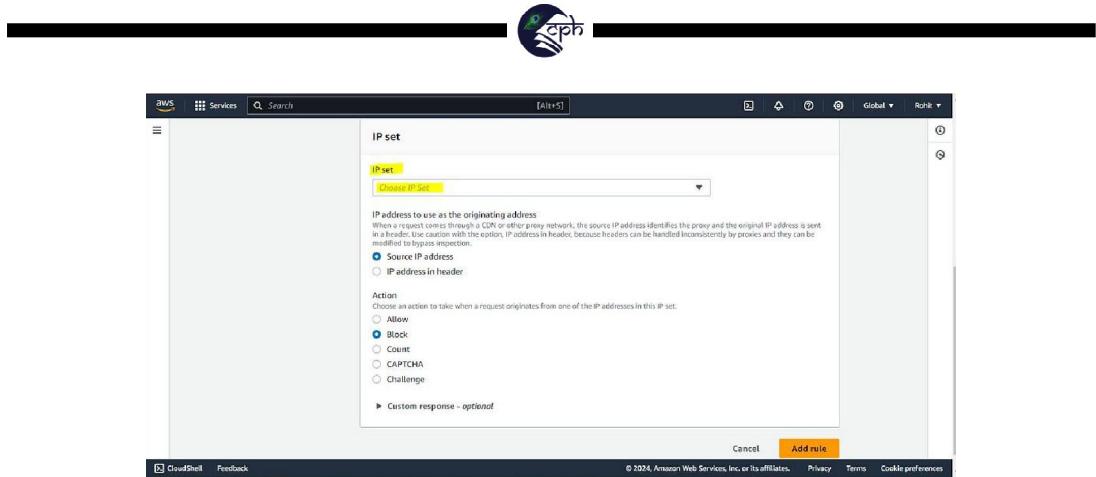
Rule builder Use a custom rule to inspect for patterns, including query strings, headers, metrics, and rate limit violations.

Rule group Use a rule group to combine rules into a single logical set.

Rule

Name: 142

The name must have 1-128 characters. Valid characters: A-Z, a-z, 0-9, -, (hyphen), and _ (underscore).



Step 4: Once the rule is created then Select Rule and click on the Next

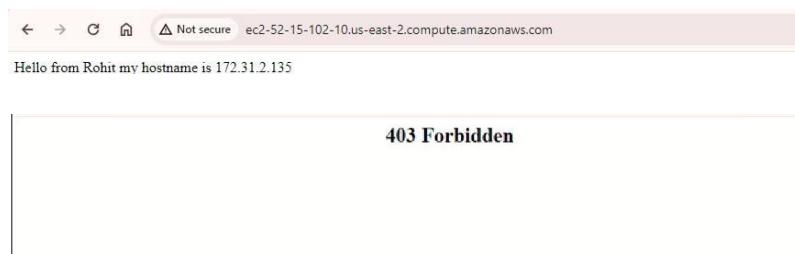
Step 5: Configure CloudWatch Metrics

Step 6: Review Web ACL Configuration: In the final step, check all the rules and hit Create Web ACL.

Finally, a message will pop up **You Successfully created web ACL: ACL-name**



Then test the application access on the internet, The IP added in the IP set that is blocked will get 403 Forbidden, and all other users will access the application.



Best Practices for Using AWS WAF

- Regularly Update Rules:** Keep your rule sets updated to protect against the latest vulnerabilities and threats.
- Testing and Validation:** Regularly test new rules in a non-production environment to validate their efficacy and minimize false positives.
- Layered Security:** Combine AWS WAF with other AWS security services for a comprehensive security strategy.

Estimating Costs

The cost of AWS WAF can vary depending on the scale of your deployment, ranging from a few dollars per month for small deployments to several thousand dollars per month for large-scale deployments. AWS WAF pricing is based on the number of web requests processed and the number of security rules that are used.

Example of cost for our example (1 Web ACL with a few managed rules):

\$5.00 per web ACL per month (prorated hourly) * 1 web ACL = \$5.00

\$1.00 per rule per month (prorated hourly) * 5 rules = \$1.00

\$0.60 per million requests processed * 1 (we will assume 1 million request) = \$0.60

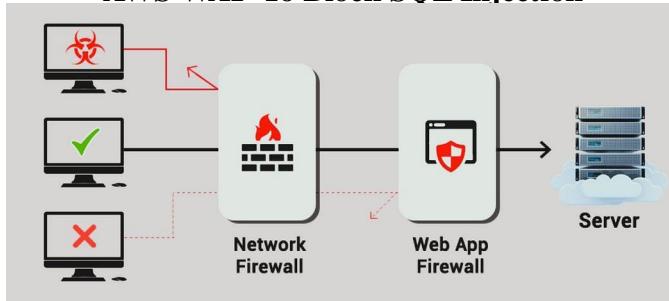
\$0.10 per alarm metric * 1 alarm = \$0.10

Total: \$6.70 per month

Conclusion

AWS Web Application Firewall provides a managed solution to protect your web applications and APIs against common exploits and vulnerabilities. By leveraging WAF's advanced rulesets and integration with services like Application Load Balancer, you can effectively filter malicious web traffic while allowing legitimate users access. With customizable rules, real-time metrics, and easy association with AWS resources, WAF is a robust web application firewall to secure your workloads in the cloud. Carefully monitor your WAF to fine-tune rules and maximize threat protection. Using AWS WAF can improve your overall security posture in the cloud.

AWS WAF-To Block SQL Injection



Implementing AWS WAF with ALB to block SQL Injection, Geo Location and Query string

AWS WAF helps you protect against common web exploits and bots that can affect availability, compromise security, and reduce performance.

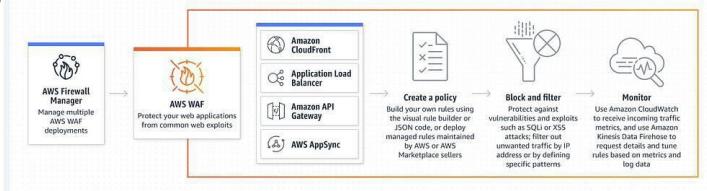


Image taken from amazon.com



Q1. What is AWS WAF?

Ans :

AWS WAF is a web application firewall that helps you to protect your web applications against common web exploits that might affect availability and compromise security.

It gives you control over how traffic reaches your applications by enabling you to create **security rules** that block common attack patterns like **SQL injection** and **cross-site scripting**.

It only allows the request to reach the server based on the rules or patterns you define. AWS WAF also allows us to review rules and customize them to prevent new attacks from reaching the server.

Q2. What is the difference between a firewall and a WAF?

Ans:

A **WAF** protects web applications by targeting Hypertext Transfer Protocol (**HTTP**) traffic. This differs from a standard firewall, which provides a barrier between external and internal network traffic. A WAF sits between external users and web applications to analyze all HTTP communication.

Thus, **firewall** is usually associated with protection of only the network and transport layers (**layers 3 and 4**). However, a web application firewall (**WAF**) provides protection to **layer 7**.

Q3. Explain how AWS WAF work. And how it integrates with CloudFront and CloudWatch?

Ans:

AWS WAF gives a developer the ability to **customize security rules** to allow, block or monitor Web requests. **Amazon**

CloudFront (AWS' content delivery network) receives a request from an end user and forwards that request to AWS WAF for inspection.

AWS WAF then responds to either **block** or **allow** the request. A developer can also use AWS WAF's integration with CloudFront to apply protection to sites that are hosted outside of AWS.

Developers **create rules in AWS WAF** that can include placing limitations on certain IP addresses, HTTP headers and URI strings. AWS WAF rules can prevent common Web attacks, such as **SQL injection** and **cross-site scripting** (XSS), which look to exploit vulnerabilities in a site or application. Rules take roughly one minute to activate, and a developer can track the effectiveness of those rules by viewing **real-time** metrics in **Amazon CloudWatch** or through sampled Web requests stored in the AWS WAF API or AWS Management Console. These metrics include IP addresses, geo locations and URIs for each request.

Q4. Can I use AWS WAF to protect web sites not hosted in AWS?

Ans:

Yes, AWS WAF is integrated with Amazon CloudFront, which **supports custom origins** outside of AWS.



Q5. How is Amazon WAF priced?

Ans:

The cost of WAF is only for **what you use**. The pricing is based on **how many rules** you deploy and how many web requests your application receives.

There are no upfront commitments. AWS WAF charges are in addition to *Amazon CloudFront pricing*, the *Application Load Balancer (ALB) pricing*, *Amazon API Gateway pricing*, and/or *AWS AppSync pricing*.

Q6. Can I use Managed Rules along with my existing AWS WAF rules?

Ans:

Yes, you can use Managed Rules along with your custom AWS

WAF rules. You can add Managed Rules to your existing AWS WAF web ACL to which you might have already added your own rules.

The number of rules inside a **Managed Rule does not count towards your limit**. However, each Managed Rule added to your web ACL will count as 1 rule.

Q7. What services does AWS WAF support?

Ans:

AWS WAF can be deployed on Amazon **CloudFront**, the Application Load Balancer (**ALB**), Amazon **API Gateway**, and **AWS AppSync**.

As part of Amazon CloudFront it can be part of your Content Distribution Network (CDN) protecting your resources and content at the Edge locations. As part of the Application Load Balancer, it can protect your origin web servers running behind the ALBs. As part of Amazon API Gateway, it can help secure and protect your REST APIs. As part of AWS AppSync, it can help secure and protect your GraphQL APIs.

Q8. What is Elastic Load Balancing or ELB?

Ans:

ELB is a service that automatically distributes incoming application traffic and scales resources to meet traffic demands. It helps in adjusting capacity according to incoming application and network traffic.

It can be enabled within a single availability zone or across multiple availability zones to maintain consistent application performance.

ELB offers features like:

- Detection of unhealthy EC2 instances.
- Spreading EC2 instances across healthy channels only.
- Centralized management of SSL certificates.

- Optional public key authentication.
- Support for both IPv4 and IPv6.
- ELB accepts incoming traffic from clients and routes requests to its registered targets.
- When an unhealthy target or instance is detected, ELB stops routing traffic to it and resumes only when the instance is healthy again.
- ELB monitors the health of its registered targets and ensures that the traffic is routed only to healthy instances.
- ELB's are configured to accept incoming traffic by specifying one or more **listeners**.

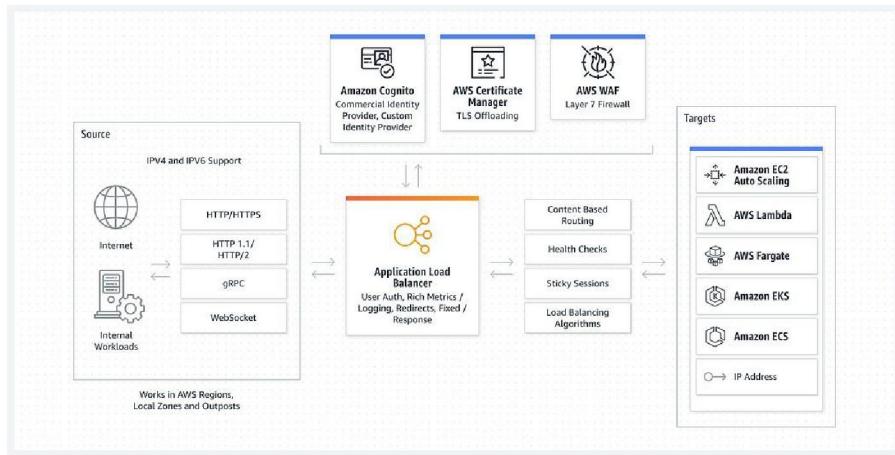


Image taken from amazon.com

Q9. How do I decide which load balancer to select for my application?

Ans:

Elastic Load Balancing (ELB) supports four types of load balancers:

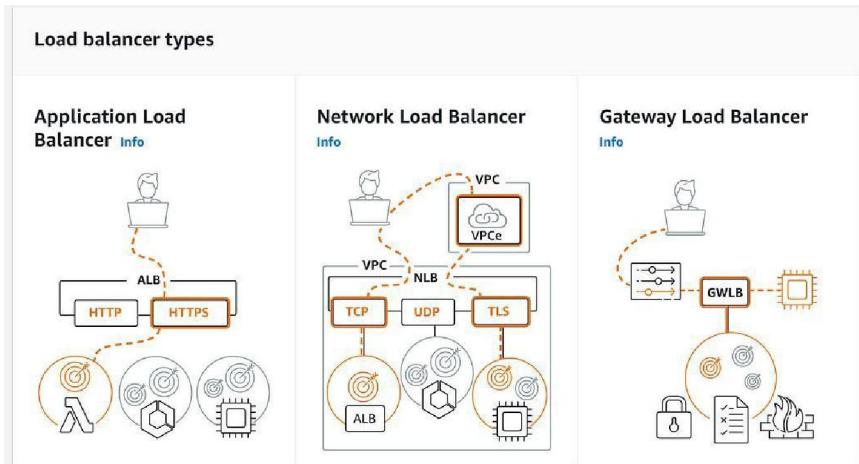
- Application Load Balancers.
- Network Load Balancers.
- Gateway Load Balancers.
- Classic Load Balancers.

You can select the appropriate load balancer based on your application needs. If you need to load balance HTTP requests, we recommend you use the **Application Load Balancer (ALB)**.

For network/transport protocols (layer4 — TCP, UDP) load balancing, and for extreme performance/low latency applications we recommend using **Network Load Balancer**.

If your application is built within the Amazon Elastic Compute Cloud (Amazon EC2) Classic network, you should use **Classic Load Balancer**.

If you need to deploy and run third-party virtual appliances, you can use **Gateway Load Balancer**.



Q10. What are listeners in ELB?

Ans:

A listener is a **process that checks for connection requests**. Listeners are configured with a protocol and port number from the client to the ELB and vice-versa i.e., back from ELB to the client.

Q11. How are load Balancers configured?

Ans:

Each load balancer is configured differently.

For **Application and Network Load Balancers**, you register targets in target groups and route traffic to target groups.

Gateway Load Balancers use Gateway Load Balancer endpoints to securely exchange traffic across VPC boundaries.

For **Classic Load Balancers**, you register instances with the load balancer.

AWS recommends users to work with Application Load Balancer to use **multiple Availability Zones** because if one availability zone fails, the load balancer can continue to route traffic to the next available one. We can have our load balancer be either internal or internet-facing.

The **nodes** of an internet-facing load balancer have **Public IP addresses**, and the DNS name is publicly resolvable to the Public IP addresses of the nodes.

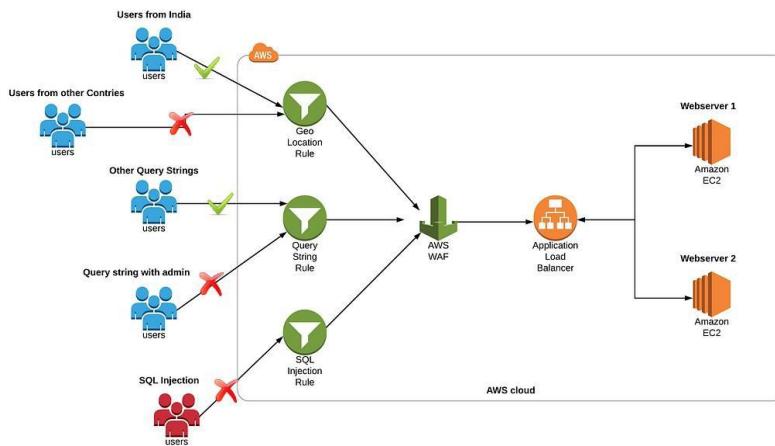
Due to the point above, internet-facing load balancers can route requests from clients over the Internet.

The **nodes** of an internal load balancer have only **Private IP addresses**, and the DNS name is publicly resolvable to the Private IP addresses of the nodes.

Due to the point above, internal load balancers can only route requests from clients with access to the VPC for the load balancer.

Note: Both internet-facing and internal load balancers route requests to your **targets** using Private IP addresses.

Implement



Task:

- Sign in to AWS Management Console. Launch First EC2 Instance (MyEC2Server1).
- Launch Second EC2 Instance (MyEC2Server2).
- Create a Target Group (MyWAFTargetGroup)
- Create an Application Load Balancer (MyWAFLoadBalancer).
- Test Load Balancer DNS.
- Create AWS WAF Web ACL (MyWAFWebAcl).
- Test Load Balancer DNS.

Solution:

Task 1: Sign in to AWS Management Console and launch First EC2 Instance

In this task, we are going to launch the first EC2 instance (MyEC2Server1) by providing the required configurations like name, AMI selection, security group, instance type and other settings.

Furthermore, we will provide the user data as well.



1) Goto Services menu in the top left, then click on EC2 in the Compute section. Navigate to Instances from the left side menu and click on Launch Instances button.

2) Enter/select the required details:

✓ Name : Enter **MyEC2Server1**

✓ Amazon Machine Image (AMI) : select **Amazon Linux 2 AMI**

✓ Instance Type : Select **t2.micro**

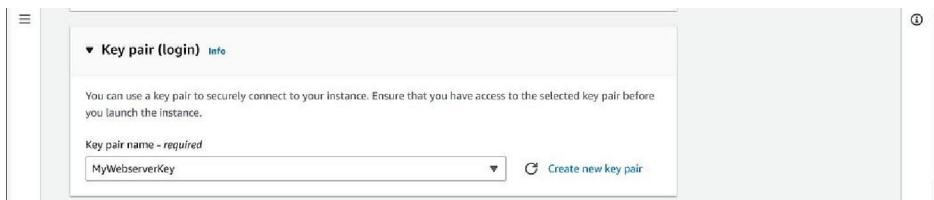
✓ Under the Key Pair (login) section : Click on Create new key pair hyp

Key pair name: **MyWebserverKey**

Key pair type: **RSA**

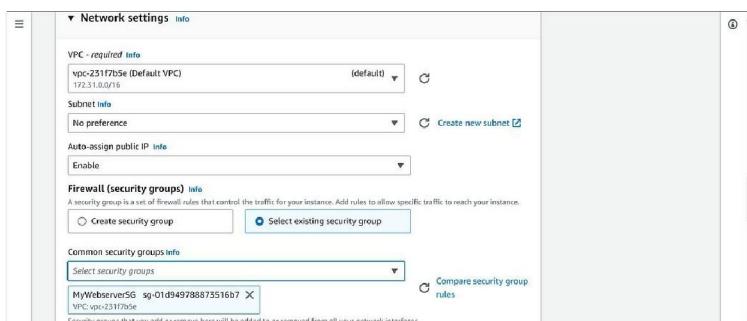
Private key file format: **.pem** or **.ppk**

Click on **Create key pair** and then select the created key pair from the drop-down.



✓ Under the **Network Settings** section :

Click on Edit button. Auto-assign public IP: select **Enable**



✓ Firewall (security groups) : Select **Create a new security group**

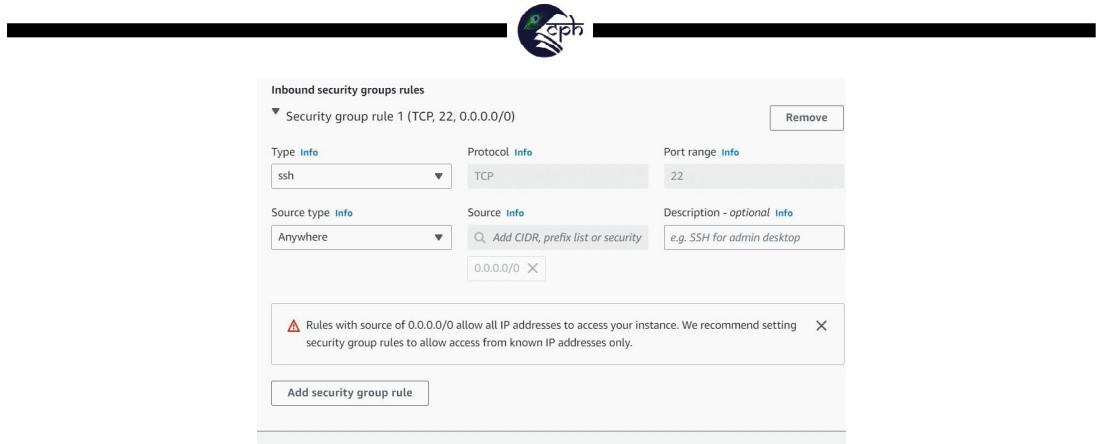
Security group name : Enter **MyWebserverSG**

Description : Enter **My EC2 Security Group**

To add SSH:Choose Type: **SSH**

Source: **Anywhere** (From ALL IP addresses accessible).

Similarly add For **HTTP** and **HTTPS**, click on **Add security group** rule.



- ✓ Under the **Advanced details** section :

Under the **User data**: copy and paste the following script to create an HTML page served by an Apache HTTPD web server.

```
#!/bin/bash
sudo su
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "<html><h1>Welcome to My Server 1 </h1></html>" >>
/var/www/html/index.html
```

Keep everything else as **default** and click on the **Launch instance** button.

Task 2: Launch Second EC2 Instances (MyEC2Server2)

Launch the second **EC2** instance similar to previous one with slight variation, by providing the required configurations like name, AMI selection, security group, instance type and other settings. Furthermore, here too we will provide the user data as well. Here,

- ✓ Name : Enter **MyEC2Server2**
- ✓ Instance Type : Select **t2.micro**
- ✓ Key Pair (login) section : Select **MyWebserverKey** from the list.
- ✓ Under the Network Settings section : Click on **Edit** button Auto-assign public IP: select **Enable**

Firewall (security groups) : Select existing security group **MyWebserverSG**

- ✓ Under the Advanced details section :

Under the **User data**: copy and paste the following script to create an HTML page served by Apache httpd web server:



```
#!/bin/bash
sudo su
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "<html><h1>Welcome to Whizlabs Server 2 </h1></html>" >>
/var/www/html/index.html
```

Keep everything else as **default**, and click on the **Launch instance** button.

Instances (2) Info							
	Name	Instance ID	Instance state	Instanc...	Status check	Alarm status	Availabi...
<input type="text"/> Find Instance by attribute or tag (case-sensitive)							
<input type="checkbox"/>	MyEC2Server1	i-080f7e3a6af14a332	Running	t2.micro	2/2 checks p	No alarms +	us-east-1b
<input type="checkbox"/>	MyEC2Server2	i-0ce19c5d7fda57303	Running	t2.micro	Initializing	No alarms +	us-east-1b

Task 3: Create a Target Group (MyWAFTargetGroup)

In this task, we are going to create a target group for the load balancer and will add the target instances so that the load balancer can distribute the traffic among these instances.

1. In the EC2 console, navigate to Target groups in the left-side panel under Load Balancer in the Load Balancing section.
2. Click on Create target group button on the top right corner.

Target groups Info							
<input type="text"/> Find resources by attribute or tag							
	Name	ARN	Port	Protocol	Target type	Load balancer	
No target groups							
You don't have any target groups in us-east-1							
Create target group							

3. Enter basic configuration:
 - ✓ Choose a target type : Select Instances



EC2 > Target groups > Create target group

Step 1
Specify group details

Step 2
Register targets

Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling [?] to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and non-VPC resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice-based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

✓ Target group name : Enter **MyWAFTargetGroup**

Protocol : Select **HTTP**

Port : Enter **80**

✓ Health Checks:

Health check protocol : Select **HTTP**

Target group name
MyWAFTargetGroup

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol : Port

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 \[?\]](#)

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

Default VPC
vpc-231fb75e
IPv4: 172.31.0.0/16

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

3 **2-10**

Unhealthy threshold

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

2 **2-10**

Timeout

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

5 **seconds** **2-120**

Interval

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

6 **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").

200



✓ Leave everything as **default** and click on **Next** button.

✓ Register targets:

- Select the two instances we have created
i.e. **MyEC2Server1** and **MyEC2Server2**.
- Click on **Include as pending** below and scroll down.

Review targets and click on **Create target group** button.

Remove	Health status	Instance ID	Name	Port	State	Security groups	Zone	Subnet
X	Pending	i-048434e99d77038	MyEC2server1	80	Running	MyWebserverSG	us-east-1c	subnet-00000000
X	Pending	i-02ead111eb23b1d54	MyEC2server2	80	Running	MyWebserverSG	us-east-1d	subnet-00000001

2 pending Cancel Previous Create target group

Your Target group has been successfully created.

Successfully created the target group: **MyWAFTargetGroup**.

EC2 > Target groups > MyWAFTargetGroup

MyWAFTargetGroup

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
0	0	0	0	0	0

Task 4: Create an Application Load Balancer (MyWAFLoadBalancer)

In this task, we are going to create an Application Load balancer by providing the required configurations like name, target group etc.

- 1) In the EC2 console, navigate to Load Balancers in the left-side panel under Load Balancing. Click on Create Load Balancer at the top-left to create a new load balancer for our web servers.

EC2 > Load balancers

Load balancers

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

Name	DNS name	State	VPC ID	Availability Zones
No resources to display				



On the next screen, choose **Application Load Balancer** since we are testing the high availability of the web application and click on **Create** button.

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 Elastic Load Balancing works

1. Clients make requests to your application.
2. The listeners in your load balancer receive requests matching the protocol and port that you configure.
3. The receiving listener evaluates the incoming request against the rules you specify, and if applicable, routes the request to the appropriate target group. You can use an HTTPS listener to offload the work of TLS encryption and decryption to your load balancer.
4. Healthy targets in one or more target groups receive traffic based on the load balancing algorithm, and the routing rules you specify in the listener.

- 2) Enter the required Basic configuration:

Load balancer name: Enter **MyWAFLoadBalancer**

Scheme: Select **Internet-facing**

IP address type: Choose **Ipv4**

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

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
Recommended for internal load balancers.
 Dualstack
Includes IPv4 and IPv6 addresses.

Network mapping:

VPC : Select **Default**

Mappings : Check All Availability Zones

Security groups: Select an existing security group

i.e. **MyWebserverSG** from the drop-down menu.

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

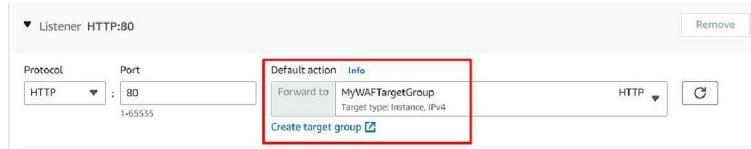
default sg-e6f0d3fe X
VPC: vpc-25911458

Listeners and routing:

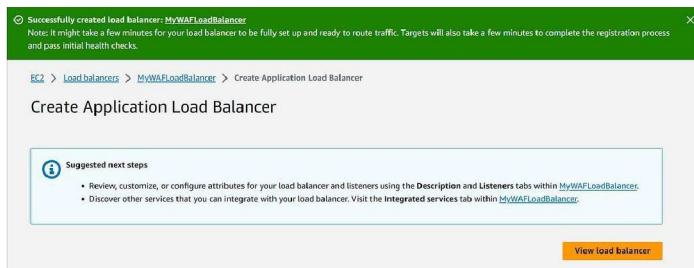
Protocol : Select **HTTP**

Port : Enter **80**

Default action : Select **MyWAFTargetGroup** from the drop down menu



Leave everything as default and click on Create load balancer button.

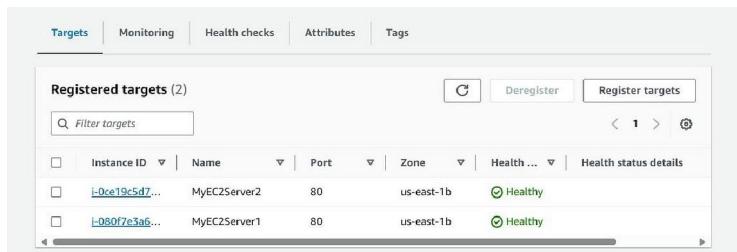


You have successfully created Application Load Balancer.

Task 5: Test Load Balancer DNS

In this task, we will test the working of load balancer by **copying the DNS to the browser** and find out whether it is able to distribute the traffic or not.

1. Now navigate to the **Target Groups** from the left side menu under Load balancing.
2. Click on the **MyWAFTargetGroup** Target group name.
3. Now select the **Targets** tab and wait till both the targets become **Healthy** (Important).



Now again navigate to Load Balancers from the left side menu under Load balancing. Select the **MyWAFLoadBalancer** Load Balancer and copy the DNS name under Description tab.



The screenshot shows the AWS Elastic Load Balancing console. At the top, there is a header with the text "Load balancers (1/1)" and a "Create load balancer" button. Below the header is a search bar labeled "Filter load balancers". A table lists one load balancer: "MyWAFLoadBalancer" (Status: Active, VPC ID: vpc-251f7..., Availability Zones: 6). The table has columns for Name, DNS name, State, VPC ID, and Availability Zones. Below the table, a modal window titled "Load balancer: MyWAFLoadBalancer" displays the ARN and the DNS name "MyWAFLoadBalancer-1202768545.us-east-1.elb.amazonaws.com". A tooltip indicates that the DNS name has been copied.

1. Copy the DNS name of the ELB and enter the address in the browser.
You should see index.html page content of Web Server 1 or Web Server 2

Two browser windows are shown side-by-side. Both windows have the URL "mywafloadbalancer-1202768545.us-east-1.elb.amazonaws.com" in the address bar. The first window displays the text "Welcome to My Server 1" and the second window displays "Welcome to My Server 2".

Now Refresh the page a few times. You will observe that the index pages change each time you refresh.

Note: The ELB will equally divide the incoming traffic to both servers in a Round Robin manner.

2. Test SQL Injection :

- Along with the ELB DNS add the following URL parameter: `/product?item=securitynumber'+OR+1=1` —
- Syntax : `http://<ELB DNS>/product?item=securitynumber'+OR+1=1` —

You will be able to see output similar to below.

A browser window shows a "Not Found" error page. The URL in the address bar is "mywafloadbalancer-1202768545.us-east-1.elb.amazonaws.com/product?item=securitynumber'+OR+1=1". The page content says "Not Found" and "The requested URL was not found on this server."

Here the SQL Injection went inside the server and since we only have an index page, the server doesn't know how to solve the URL that is why you got Not Found page.

3. Test Query String Parameter :

- Along with the ELB DNS add the following URL parameter: `?admin=123456`
- Syntax : `http://<ELB DNS>/?admin=123456`



You will be able to see output similar to below.



Here also the Query string went inside the server and the server always passes the query string inside and it is resolved by the code that you write. Here the query string is passed and there is no code to resolve the this but it won't throw any error it just becomes an unused value. so you got a response back.

Task 6: Create AWS WAF Web ACL (MyWAFWebAcl)

In this task , we are going to create an AWS WAF Web ACL where we will add some customized rules for location restriction, query strings and

1. Navigate to **WAF** by clicking on the Services menu in the top, then click on **WAF & Shield** in the **Security, Identity & Compliance** section.
2. On the left side menu, select **Web ACL's** and then click on **Create web ACL** button.



3. **Describe web ACL and associate it to AWS resources :**

Name : Enter **MyWAFWebAcl**

Description : Enter **WAF for SQL Injection, Geo location and Query String parameters**

CloudWatch metric name : Automatically selects the WAF name, so no changes required.

Resource type : Select **Regional resources**

Region : Select current region from the dropdown.

Associated AWS resources : Click on the Add AWS resources button.

- Resource type : Select **Application Load Balancer**
- Select **MyWAFLoadBalancer** Load balancer from the list.



AWS WAF > Web ACLs > Create web ACL

Step 1
Describe web ACL and associate it to AWS resources

Step 2
Add rules and rule groups

Step 3
Set rule priority

Step 4
Configure metrics

Step 5
Review and create web ACL

Describe web ACL and associate it to AWS resources [Info](#)

Web ACL details

Resource type
Choose the type of resource to associate with this web ACL. Changing this setting will reset the page.
 Amazon CloudFront distributions
 Regional resources (Application Load Balancers, Amazon API Gateway REST APIs, Amazon App Runner services, AWS AppSync GraphQL APIs, Amazon Cognito user pools and AWS Verified Access Instances)

Region
Choose the AWS Region to create this web ACL in. Changing this setting will reset the page.
US East (N. Virginia)

Name
MyWAFWebAcl
The name must have 1-128 characters. Valid characters: A-Z, a-z, 0-9, -, (hyphen), and _ (underscore).

Resource type
Choose the type of resource to associate with this web ACL. Changing this setting will reset the page.
 Amazon CloudFront distributions
 Regional resources (Application Load Balancers, Amazon API Gateway REST APIs, Amazon App Runner services, AWS AppSync GraphQL APIs, Amazon Cognito user pools and AWS Verified Access Instances)

Region
Choose the AWS Region to create this web ACL in. Changing this setting will reset the page.
US East (N. Virginia)

Name
MyWAFWebAcl
The name must have 1-128 characters. Valid characters: A-Z, a-z, 0-9, -, (hyphen), and _ (underscore).

Description - optional
WAF for SQL injection, Geo location and Query String parameters
The description can have 1-256 characters.

CloudWatch metric name
MyWAFWebAcl
The name must have 1-128 characters. Valid characters: A-Z, a-z, 0-9, -, (hyphen), and _ (underscore).

Associated AWS resources - optional (1)		
Remove Add AWS resources		
<input type="text"/> Find associated AWS resources < 1 > @		
Name	Resource type	Region
<input type="checkbox"/> MyWAFLoadBalancer	Application Load Balancer	US East (N. Virginia)

Add AWS resources

Resource type
Select the resource type and then select the resource you want to associate with this web ACL.

Application Load Balancer Amazon API Gateway REST API Amazon App Runner service

AWS AppSync GraphQL API Amazon Cognito user pool AWS Verified Access

Select the resources you want to associate with the web ACL.

Find AWS resources to associate [<](#) [1](#) [>](#) [@](#)

Name
 MyWAFLoadBalancer

[Cancel](#) [Add](#)



Now click on the **Add** button. Click on the **Next** button.

Add rules and rule groups : Here we will be adding three rules.

Rule 1

- Under Rules, click on Add rules and then select **Add my own rules and rule groups**.
 - Rule type : Select **Rule builder**
 - Name : Enter **GeoLocationRestriction**
 - Type : Select **Regular** type
 - If a request : Select **Doesn't match the statement (NOT)**
 - Inspect : Select **Originates from a country in**
 - Country codes : Select <Your Country> In this example we select India-IN
 - IP address to use to determine the country of origin : Select Source IP address
- Note : You can also select multiple countries also.
- Under Then : Action Select **Block**. Click on **Add rule**.

Here we are only allowing requests to come from India and all the requests that come from other countries will be blocked.

AWS WAF > Web ACLs > Create web ACL

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add rules and rule groups Info

A rule defines attack patterns to look for in web requests and the action to take when a request matches the patterns. Rule groups are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Name	Capacity

Rules (0)

If a request matches a rule, take the corresponding action. The rules are prioritized in descending order of rule priority.

Add rules ▾

Add managed rule groups

Add my own rules and rule groups

No rules.
You don't have any rules added.

Describe web ACL and associate it to AWS resources

Step 2: Add my own rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add my own rules and rule groups Info

Rule type

IP set Use an IP set to identify a specific list of IP addresses.

Rule builder Use a custom rule to inspect for patterns including query strings, headers, countries, and rate limit violations.

Rule group Use a rule group to combine rules into a single logical set.

Rule builder

You can use the JSON editor for complex statement nesting, for example to nest two OR statements inside an AND statement. The visual editor handles one level of nesting. For web ACLs and rule groups with complex nesting, the visual editor is disabled.

Rule visual editor Rule JSON editor

Rule

Name: GeoLocationRestriction

Validate



If a request doesn't match the statement (NOT)

Statement

Inspect

Originates from a country in

Country codes

Choose country codes

India - IN X

IP address to use to determine the country of origin
When a request comes through a CDN or other proxy network, the source IP address identifies the proxy and the original IP address is sent in a header. Use caution with the option, IP address in header, because headers can be handled inconsistently by proxies and they can be modified to bypass inspection.

Source IP address

IP address in header

Then

Action

Choose an action to take when a request matches the statements above.

Allow

Block

Count

CAPTCHA

Challenge

► Custom response - optional

► Add label - optional
Add labels to requests that match this rule. Rules that are evaluated later in the same web ACL can reference the labels that this rule adds.

Cancel Add rule

Rule 2

- Under **Rules**, click on **Add rules** and then select **Add my own rules and rule groups**.
- Rule type : Select **Rule builder**
- Name : Enter **QueryStringRestriction**
- Type : Select **Regular type**
- If a request : **Select matches the statement**
- Inspect : Select **Query string**
- Match type : Select **Contains string**
- String to match : Enter **admin**
- Text transformation : Leave as default.
- Under Then : Action Select **Block**.
- Click on Add rules.

Anytime in the request URL contains a query string as admin WAF will block that request.



If a request matches the statement

Statement

Inspect

Query string

Match type

Contains string

String to match

admin

Text transformation

AWS WAF applies all transformations to the request before evaluating it. If multiple text transformations are added, then text transformations are applied in the order presented below with the top of the list being applied first.

None

Add text transformation

You can add up to 10 text transformations.

AWS WAF > Web ACLs > Create web ACL

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add rules and rule groups Info

A rule defines attack patterns to look for in web requests and the action to take when a request matches the patterns. Rule groups are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Rules (2)

Name	Capacity	Action
GeoLocationRestriction	1	Block
QueryStringRestriction	10	Block

Rule 3

- Under **Rules**, click on **Add rules** and then select **Add managed rule groups**.
- It will take a few minutes to load the page. It lists all the rules which are managed by AWS.
- Click on AWS managed rule groups.
- Scroll down to SQL database and enable the corresponding Add to web ACL button.

AWS WAF > Web ACLs > Create web ACL

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add rules and rule groups Info

A rule defines attack patterns to look for in web requests and the action to take when a request matches the patterns. Rule groups are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Rules (0)

If a request matches a rule, take the corresponding action. The rules are prioritized in descending order of rule priority.

Add managed rule groups

No rules.

You don't have any rules added.

Web ACL capacity units (WCUs) used by your web ACL

The WCUs used by the web ACL will be less than or equal to the sum of the capacity for all of the rules in the web ACL.

AWS WAF > Web ACLs > Create web ACL

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add managed rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Add managed rule groups Info

Managed rule groups are created and maintained for you by AWS and AWS Marketplace sellers. Any fees that a managed rule group provider charges for using a managed rule group are in addition to the standard service charges for AWS WAF. AWS WAF Pricing

AWS managed rule groups

Cloudbric Corp. managed rule groups

Cyber Security Cloud Inc. managed rule groups

F5 managed rule groups



SQL database

Contains rules that allow you to block request patterns associated with exploitation of SQL databases, like SQL injection attacks. This can help prevent remote injection of unauthorized queries. [Learn More](#)

[Add to web ACL](#)

[Edit](#)

200

Scroll down to the end and click on **Add rules** button.

Now we have 3 rules added.

Rules (3)		
If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.		
<input type="checkbox"/>	Name	Capacity Action
<input type="checkbox"/>	GeoLocationRestriction	1 Block
<input type="checkbox"/>	QueryStringRestriction	10 Block
<input type="checkbox"/>	AWS-AWSManagedRulesSQLRuleSet	200 Use rule actions

Under **Default web ACL** action for requests that don't match any rules, Default action Select **Allow**. Click on the **Next** button.

- Set rule priority:
- No changes required, leave as default. Note : You can move the rules based on your priority.
- Click on the **Next** button.

Configure metrics:

- Leave it as default. Click on the **Next** button.

Review and create web ACL :

- Review the configuration done, scroll to the end and click on **Create web ACL** button.

AWS WAF > Web ACLs > Create web ACL

Step 1: Describe web ACL and associate it to AWS resources

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Set rule priority

If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.

Name	Capacity	Action
GeoLocationRestriction	1	Block
QueryStringRestriction	10	Block
AWS-AWSManagedRulesSQLRuleSet	200	Use rule actions

[Cancel](#) [Previous](#) **Next**

Step 2: Add rules and rule groups

Step 3: Set rule priority

Step 4: Configure metrics

Step 5: Review and create web ACL

Amazon CloudWatch metrics

CloudWatch metrics allow you to monitor web requests, web ACLs, and rules.

Rules	CloudWatch metric name
GeoLocationRestriction	GeoLocationRestriction
QueryStringRestriction	QueryStringRestriction
AWS-AWSManagedRulesSQLRuleSet	AWS-AWSManagedRulesSQLRuleSet

Request sampling options

If you disable request sampling, you can't view requests that match your web ACL rules.

Options

Enable sampled requests

Disable sampled requests

Enable sampled requests with exclusions

It will take a few seconds to create the Web ACL.

Web ACL created.

Note: I explicitly associated as it failed, later after creation of WAF ACLs.

Task 7: Test Load Balancer DNS

- Again navigate to Load Balancers from the left side menu under Load balancing. Select the **MyWAFLoadBalancer** Load Balancer and copy the DNS name under **Description** tab.

Copy the DNS name of the **ELB** and enter the address in the browser. You should see index.html page content of Web Server 1 or Web Server 2.

Now Refresh the page a few times. You will observe that the index pages change each time you refresh. thus, ELB is working fine. *Note: The ELB will equally divide the incoming traffic to both servers in a Round Robin manner.*



2. Test SQL Injection

- Along with the ELB DNS add the following URL parameter: `/product?item=securitynumber' + OR+1=1 —`
- Syntax : **http://<ELB DNS>/product?item=securitynumber'+OR+1=1 —**

You will be able to see the below output. Unlike **Page Not found** error before.



Here **SQL Injection is blocked by WAF** before it goes inside the server. İ%

3. Test Query String Parameter

- Along with the ELB DNS add the following URL parameter: `?admin=123456`
- **Syntax : http://<ELB DNS>/?admin=123456**

You will be able to see the below output.



Here also the **Query string which contains admin is blocked by WAF** before it could go inside the server.

Do you know?

WAF can offer protection against Distributed Denial of Service (DDoS) attacks by analyzing traffic patterns, detecting abnormal behaviour, and mitigating the impact of such attacks.



RDS (Relational Database Service)

We dive into the heart of relational databases on the cloud with Amazon RDS. In this session, we'll explore the fundamentals of RDS, its benefits, and how to get started with launching and configuring RDS instances. Additionally, we'll walk through the process of connecting to RDS instances from EC2 instances.



Introduction to RDS and managed database services:

Let's delve deeper into the Introduction to RDS and managed database services.

Introduction to RDS (Relational Database Service)

Amazon Relational Database Service (RDS) is a fully managed database service provided by AWS. It allows users to set up, operate, and scale relational databases in the cloud without worrying about the underlying infrastructure. RDS automates routine database tasks such as hardware provisioning, database setup, patching, and backups, enabling developers to focus more on their applications and less on database management.

Key Features of RDS:

- Multiple Database Engine Support:** RDS supports various popular database engines, including MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora. This flexibility allows users to choose the engine that best suits their application requirements.
- Automated Backups and Point-in-Time Recovery:** RDS automatically takes backups of your databases according to the retention period you specify. It also enables point-in-time recovery, allowing you to restore your database to any specific point within the retention period.
- High Availability and Replication:** RDS provides high availability features such as Multi-AZ (Availability Zone) deployments and Read Replicas. Multi-AZ deployments replicate your database synchronously across multiple Availability Zones to ensure data durability and fault tolerance, while Read Replicas enable you to offload read traffic from the primary database instance, improving performance and scalability.
- Security and Compliance:** RDS offers several security features to help you secure your databases, including network isolation using Amazon VPC, encryption at rest using AWS KMS (Key Management Service), and SSL encryption for data in transit. RDS also supports database authentication mechanisms such as IAM database authentication and traditional username/password authentication.



5. **Scalability and Performance:** With RDS, you can easily scale your database instance vertically (by increasing instance size) or horizontally (by adding Read Replicas). RDS also provides performance monitoring metrics and tools to help you optimize database performance.

Managed Database Services:

Managed database services like RDS abstract much of the operational overhead associated with traditional database management. They offer the following benefits:

1. **Simplified Database Administration:** Managed database services handle routine database administration tasks such as provisioning, patching, backups, and monitoring, freeing up developers and DBAs to focus on application development and business logic.
2. **High Availability and Reliability:** Managed services typically offer built-in high availability features such as automated failover, data replication, and backup/restore capabilities, ensuring that your databases are highly available and reliable.
3. **Scalability and Performance:** Managed services make it easy to scale your databases up or down based on demand. They often provide tools and features for performance optimization and monitoring, helping you maintain optimal database performance.
4. **Security and Compliance:** Managed database services offer robust security features and compliance certifications to help you meet your security and regulatory requirements. They handle security patching, encryption, access control, and auditing, reducing the risk of security breaches and data loss.

Launching RDS instances and configuring parameters:

These are fundamental steps in setting up relational databases using Amazon RDS. Here's a detailed guide on how to launch RDS instances and configure parameters:

Launching RDS Instances:

1. **Navigate to the RDS Console:** Log in to your AWS Management Console and navigate to the RDS service.
2. **Click on “Create database”:** In the RDS dashboard, click on the “Create database” button to initiate the instance creation process.
3. **Choose Engine and Version:** Select the database engine you want to use (e.g., MySQL, PostgreSQL, Oracle, SQL Server) and choose the version that best suits your application requirements.
4. **Specify Instance Details:**
 - **DB Instance Class** Choose the appropriate instance class based on your compute and memory requirements.



- **Multi-AZ Deployment:** Optionally, select multi-AZ deployment for high availability and redundancy across multiple Availability Zones.
- **Storage Type and Allocated Storage:** Select the storage type (e.g., General Purpose SSD, Provisioned IOPS SSD) and specify the allocated storage space for your database.
- **DB Instance Identifier:** Provide a unique identifier for your RDS instance.

5. Configure Advanced Settings:

- **Network & Security:** Choose the Virtual Private Cloud (VPC) where you want to launch your RDS instance. Configure the subnet group and specify security groups to control inbound and outbound traffic.
- **Database Options:** Set the database name, port, and parameter group (optional).
- **Backup:** Configure automated backups and specify the retention period for backup storage.
- **Monitoring:** Enable enhanced monitoring to collect additional metrics for your RDS instance.
- **Maintenance:** Specify the preferred maintenance window for applying patches and updates to your database instance.

6. Add Database Authentication and Encryption:

- Set the master username and password for your RDS instance.
- Enable encryption at rest using AWS Key Management Service (KMS) for enhanced data security.

7. Review and Launch:

- Double-check all the configurations you've made for your RDS instance.
- Click on the “Create database” button to launch your RDS instance.

Configuring Parameters:

Once you've launched your RDS instance, you may need to configure additional parameters based on your application requirements. Here's how you can configure parameters for your RDS instance:

1. **Parameter Groups:** RDS parameter groups contain configuration settings that govern the behavior of your database instance. You can create custom parameter groups or use default parameter groups provided by AWS.
2. **Modify Parameters:**
 - Navigate to the RDS dashboard and select your RDS instance.



- In the “Configuration” tab, click on “Modify” to change parameter settings.
- You can modify various parameters such as database engine settings, memory allocation, logging options, and performance parameters.

3. Apply Changes:

- After modifying parameters, review the changes and click on “Apply immediately” or choose a maintenance window for applying changes.
- AWS RDS will apply the parameter changes to your database instance without causing downtime.

4. Monitor Performance:

Monitor the performance of your RDS instance after applying parameter changes to ensure that your database is operating optimally.

Connecting to RDS instances from EC2:

It is a common scenario in cloud-based applications where you might have your application servers (EC2 instances) accessing databases hosted on RDS. Here's a step-by-step guide on how to connect to RDS instances from EC2 instances:

Prerequisites:

- **Both RDS and EC2 instances must be in the same VPC:** Ensure that your RDS instance and EC2 instance are deployed within the same Virtual Private Cloud (VPC) to enable network communication between them.
- **Security Group Configuration:** Configure the security group associated with your RDS instance to allow inbound traffic from the security group associated with your EC2 instance on the appropriate database port (e.g., 3306 for MySQL, 5432 for PostgreSQL).

Steps to Connect to RDS from EC2:

1. Identify RDS Endpoint and Port:

- Log in to the AWS Management Console and navigate to the RDS service.
- Select your RDS instance and note down the endpoint (e.g., my-database-instance.abcdefg123.us-west-2.rds.amazonaws.com) and the port number (default is 3306 for MySQL, 5432 for PostgreSQL).

2. Install Database Client on EC2 Instance:

- Connect to your EC2 instance using SSH.
- Install the appropriate database client for your RDS database engine (e.g., MySQL client for MySQL databases, PostgreSQL client for PostgreSQL databases). You can install these clients using package managers like apt (for Ubuntu) or yum (for Amazon Linux).



2. Connect Using Command Line:

Once the database client is installed, you can connect to your RDS instance from the command line using the following syntax:

For MySQL:

```
mysql -h <RDS_endpoint> -P <port> -u <username> -p
```

For PostgreSQL:

```
psql -h <RDS_endpoint> -p <port> -U <username> -d <database_name>
```

Replace **<RDS_endpoint>, <port>,**

<username>, and <database_name> with your actual RDS endpoint, port, username, and database name respectively.

3. Provide Credentials:

When you run the command, you'll be prompted to enter the password for the specified user. Enter the password associated with the username you provided.

4. Verification:

- After successfully connecting, you'll be presented with a database prompt or console, indicating that you're connected to your RDS instance from your EC2 instance.
- You can now execute SQL queries, perform database operations, and interact with your RDS database as needed from your EC2 instance.

Below are examples covering the topics mentioned earlier: launching an RDS instance, configuring parameters, and connecting to the RDS instance from an EC2 instance using MySQL as the database engine.

Example: Launching an RDS Instance (Using AWS CLI)

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.t2.micro \
    --engine mysql \
    --allocated-storage 20 \
    --master-username mymasteruser \
    --master-user-password mymasterpassword \
    --availability-zone us-west-2a \
    --backup-retention-period 7 \
    --port 3306 \
    --multi-az \
    --engine-version 5.7
```

Example: Configuring Parameters (Using AWS CLI)

```
aws rds modify-db-parameter-group \
```



```
—db-parameter-group-name mydbparametergroup \
—parameters "ParameterName=innodb_buffer_pool_size,ParameterValue=67108864,ApplyMethod=immediate" \
"ParameterName=max_connections,ParameterValue=200,ApplyMethod=immediate"
```

Example: Connecting to RDS Instance from EC2 (Using MySQL Client)

1. Install MySQL Client on EC2 instance:

For Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get install mysql-client -y
```

For Amazon Linux:

```
sudo yum update
sudo yum install mysql -y
```

2. Connect to RDS Instance:

For MySQL:

```
mysql -h mydbinstance.abcdefg123.us-west-2.rds.amazonaws.com -u mymasteruser -p
```

You'll be prompted to enter the password for the master user

(‘**mymasteruser**’). After entering the password, you'll be connected to the MySQL prompt where you can execute SQL queries.

These examples demonstrate the basic steps for launching an RDS instance, configuring parameters, and connecting to the RDS instance from an EC2 instance using MySQL as the database engine. Make sure to replace placeholders like ‘**mydbinstance**’, ‘**mymasteruser**’, ‘**mymasterpassword**’, and ‘**abcdefg123.us-west-2**’ with your actual values.

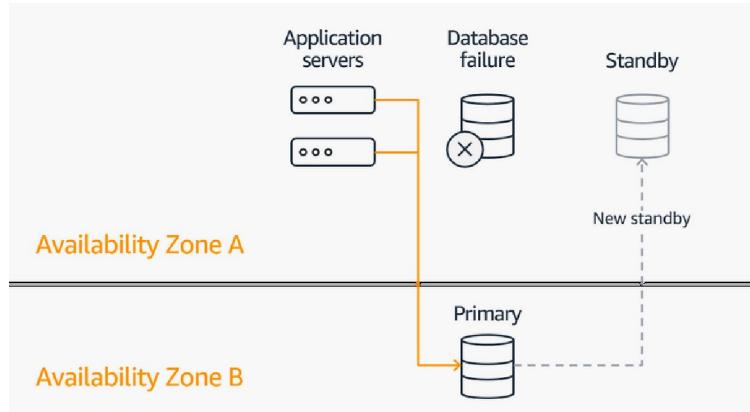
Conclusion:

Amazon RDS, from launching instances to connecting them from EC2. Managed database services like RDS empower developers to focus on building applications while offloading the heavy lifting of database management to AWS.

Deploying multi-AZ RDS instances with Read Capabilities

What is Multi AZ RDS?

When using AWS RDS, we can deploy instances in multiple availability zones which would increase the availability of the instances in case of a disaster and would also help in the architecture being more reliable and Fault tolerant.



In the diagram shown above, a multi-AZ RDS is deployed in Availability Zones A and B. When there is a database failure in AZ A, the DNS would switch to AZ B instance which would have all the data as multi-AZ instances are synchronously replicated which means any update to the first instance would replicate the same update to the second instance in a synchronized fashion.

Thus, the application is always able to access the database in case of a disaster/database failure.

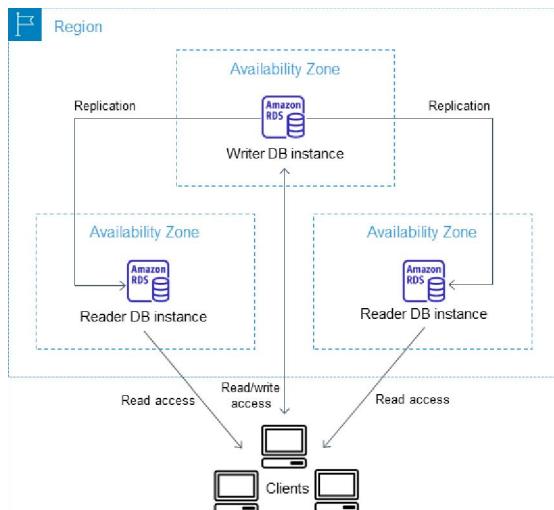
Problem Statement:

Although we have high availability of the application due to instances deployed in Multi AZ, the standby instances do not accept any read traffic, which means that even though we have the instances in different AZ's, we can't perform any read/write operations on them. Any read/write operations have to be done on the primary instance which would synchronously replicate the same to the standby instances which can also lead to performance bottlenecks on the primary instances as it would accept the read, write operations and replicate the same to the standby instances in different availability zones. Also, in order to perform any such, read operations on the DB other than the writer node, we would need to provision Read Replicas which have an additional cost associated as well.

Note: We can have instances in multiple AZ's and have an order/priority assigned to them, which means in case of the disaster the DNS would switch to the instances based on the order/priority assigned.

Solution:

- AWS has introduced a new cluster option for the RDS service which deploys the instances in Multi AZ fashion while also allowing the instances to accept Read Only traffic. Thus, resolving the problem statement as discussed earlier.
- The Instances are created in a Cluster fashion and while the communication with the primary instances take place with the cluster endpoint, we also have instances and Reader endpoints to redirect traffic to the other instances and utilize the read capabilities of these instances in order to ensure the Multi AZ instances can accept read traffic. Since the Replication between the Primary and the other standby instances is synchronous, at any point in time the standby instances would have the same set of changes/updates as the primary instance.



Note 1: This option is available only for Amazon RDS for MySQL version 8.0.26 and PostgreSQL version 13.4 and in the US East (N. Virginia), US West (Oregon), and EU (Ireland) Regions. This setup is available only for R6gd or M6gd instance types.

Note 2: The SLA(Service Level Agreement) for this cluster setup does match the SLA for the RDS service provided by AWS and hence this setup is ideal for Development and Test environments but not Production environments which require the same SLA for RDS.

Deployment Steps:

Prerequisites:

- A Valid AWS Account with permissions to create, Read and Write access to RDS.
- The VPC has at least one subnet in each of the three Availability Zones in the Region where you want to deploy your DB cluster. This configuration ensures that your DB cluster always has at least two readable standby DB instances available for failover, in the unlikely event of an Availability Zone failure.



- The Setup may result in cost for the DB, hence ensure you don't have any billing related issues for the account.
- An EC2 instance to connect to the DB or a local machine.

Procedure:

DB Creation and Setup

1. Go to the AWS console, search for RDS and select Create Database
2. Select the Dev/Test Template and select Multi AZ DB Cluster- preview option

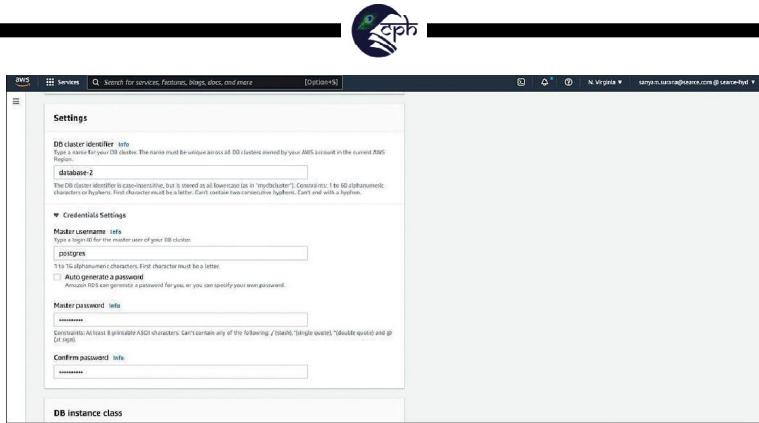
Note: Only available in US East (N. Virginia), US West (Oregon), and EU (Ireland) Regions

3. Select the checkbox to acknowledge the SLA for the DB cluster.

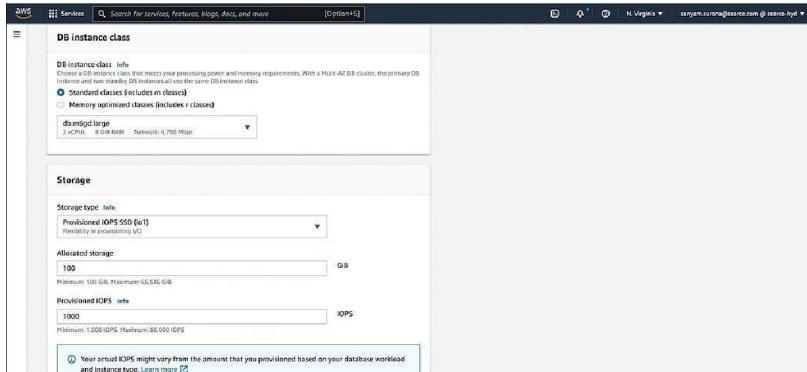
The screenshot shows the 'Engine options' step of the AWS RDS 'Create Database' wizard. It displays a grid of engine icons and names. PostgreSQL is selected. The 'Templates' section shows 'Production' and 'Dev/Test' options, with 'Dev/Test' being selected. A note states: 'This instance is intended for development use outside of a production environment.'

The screenshot shows the 'Availability and durability' step of the AWS RDS 'Create Database' wizard. It lists deployment options: Multi-AZ DB Cluster - preview (selected), Multi-AZ DB instance, and Single DB instance. A note states: 'Multi-AZ DB clusters are now available in preview.' A checkbox at the bottom accepts the service agreement for Multi-AZ DB clusters.

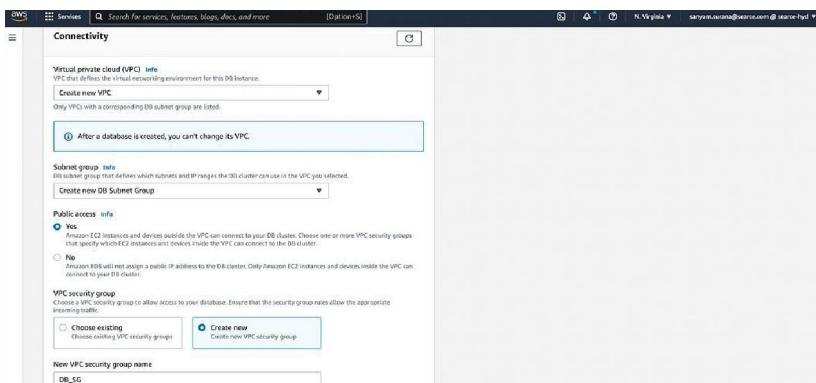
4. Enter the required parameters such as DB name, Db username and Password.



We have selected the m6gd instance type and allocated space as 100 Gb and Iops as 1000 as it's the minimum requirement to save costs.



5. For the purpose of this Blog, we have enabled Public access to the DB, but it is highly recommended that you do not enable public access as it's a more secure approach to protect the DB.
7. The DB uses 5432 port to communicate and hence we can either use an existing security group which has the port enabled or create a new Security Group.





8. We can go to additional configuration for more detailed information and configuration settings

The screenshot shows the AWS RDS console for a PostgreSQL database cluster named 'Info'. Under 'Database authentication', 'Password authentication' is selected. In the 'Additional configuration' section, options like 'Encryption enabled', 'Backup enabled', 'Performance Insights disabled', and 'Enhanced Monitoring disabled' are listed. Under 'Database options', the 'Initial database name' is set to 'Info' (not supported for Multi-AZ DB cluster). The 'DB cluster parameter group' is 'default.postgres13' and the 'Option group' is 'Info' (not supported for Multi-AZ DB cluster).

This screenshot is identical to the one above, showing the AWS RDS console for the same PostgreSQL database cluster 'Info'. It displays the same configuration settings for database authentication, additional configurations, and database options.

9. We have disabled automated backups and encryption for the purpose of this Blog, but it is highly recommended to enable these for more security, reliability and disaster recovery.

The screenshot shows the AWS RDS console for a PostgreSQL database cluster. Under 'DB cluster parameter group' 'default.postgres13', the 'Option group' is 'Info' (not supported for Multi-AZ DB cluster). In the 'Backup' section, 'Enable automated backups' is unchecked. Under 'Encryption', 'Enable encryption' is checked with a note about choosing a master key. In the 'Performance insights' section, 'Enable Performance Insights for all instances in the cluster' is unchecked. Under 'Monitoring', 'Enable Enhanced monitoring for all instances in the cluster' is checked. In the 'Log exports' section, 'PostgreSQL log' and 'Upgrade log' are selected. At the bottom, there are links for 'Feedback', 'English (US)', 'N. Virginia', 'sanyam.aurana@searce-hyd', and 'Cookie preferences'.



10. Let the other settings be default, else we can change them according to our preference and click create Database.

The screenshot shows the 'Set maintenance preferences' step of the AWS RDS Create Database wizard. It includes the following configuration:

- IAM role:** Select 'Amazon RDS service-linked role'.
- Log exports:** Select 'PostgreSQL log' and 'Upgrade log'.
- IAM role:** Select 'Amazon RDS service-linked role'.
- Maintenance:** Select 'Auto minor version upgrade'.
- Maintenance window:** Select 'No preference'.
- Deletion protection:** Select 'Enable deletion protection'.

A note at the bottom states: "Ensure that general, slow query, and audit logs are turned on. Error logs are enabled by default. Learn more".

11. The DB takes a couple of Minutes to create and once created the results will be similar to as shown below.
12. Once created we can see cluster endpoint names for both the reader node and writer node. We can select individual instances and view their endpoints as well

The screenshot shows the AWS RDS Dashboard. On the left, there's a sidebar with links like Dashboard, Databases, Query Editor, etc. The main area displays a table of databases:

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU
database-2	Multi-AZ DB cluster	PostgreSQL	us-east-1	3 instances	Modifying	-
database-2-instance-1	Writer instance	PostgreSQL	us-east-1b	db.m6g.large	Available	6.01%
database-2-instance-2	Reader instance	PostgreSQL	us-east-1a	db.m6g.large	Available	7.44%
database-2-instance-3	Reader instance	PostgreSQL	us-east-1d	db.r6g.large	Available	7.65%

Below the table, there's a section for Endpoints:

Endpoint name	Status	Type	Port
database-2.cluster-0bnpqyogiu.us-east-1.efs.amazonaws.com	Available	Writer instance	5432
database-2.cluster-0bnpqyogiu.us-east-1.efs.amazonaws.com	Available	Reader instance	5432

The screenshot shows the AWS Management Console for the Amazon RDS service. A specific Multi-AZ DB cluster named "database-2" is selected. The cluster contains three instances: "database-2-instance-1" (Writer instance, PostgreSQL, us-east-1a, db.m6gd.large), "database-2-instance-2" (Reader instance, PostgreSQL, us-east-1a, db.m6gd.large), and "database-2-instance-3" (Reader instance, PostgreSQL, us-east-1d, db.m6gd.large). The "Connectivity & security" tab is active, displaying information such as the Endpoint (database-2-instance-1.us-nayyudqlauj-us-east-1.rds.amazonaws.com), Subnet groups (us-east-1b), VPC (EU_1st_VPC (vpc-0be9f7c89a55f15e4)), and Subnet group (defn-1t-vpc-0fe72e79a5f1544). The "Networking" section shows the Availability Zone (us-east-1b) and the "Security" section lists VPC security groups (DB_SG (sg-02253bfaf0a672903d1)) and Certificate authority (rdca-2019). The "Logs & events", "Configuration", "Maintenance & backups", and "Tags" tabs are also visible.

13. Connect to the local machine or EC2 instance which already has the psql package installed . For the purpose of this blog, we have used a linux 2 AMI which is free tier eligible and installed the psql package on it for connecting to the PostgreSQL DB created.

Using Cluster Writer Endpoint

In the below screenshot, we are using the Cluster Writer endpoint to connect to the cluster and perform write operations on the DB

```
[root@ip-172-31-124-124 bin]# pgsql --host "database-1.cluster-cchnopvqgkq.us-east-1.rds.amazonaws.com" --port 5432 --username postgres --dbname postgres
Password for user "postgres":  
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.  
postgres>
```

Create a Database, connect to the database and insert records into it.



```
test124-> select * from team;
ERROR:  syntax error at or near "/"
LINE 1: /dt
               ^
test124=> select * from team;
   sno | first_name | last_name
   ---+-----+-----
     1 | john      | doe
     2 | stephen   | doe
     3 | karim     | bulla
     4 | garchu    | pedro
(4 rows)
```

Connect To Cluster Reader Endpoint

Connect to the Cluster Reader endpoint and query the records.

```
[root@ip-172-31-12-1 ~]# psql -F t -Fc t -U postgres -c "SELECT * FROM team;" -h 172.31.12.1 -p 5432 -v ON_ERROR_STOP=1
Password for user postgres:
psql (13.3, server 13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres> \dt
Did not find any relations.
postgres> \q
Deltas only is on.

postgres> \c test124
You are now connected to database "test124" as user "postgres".
test124=> select * from team;
   sno | first_name | last_name
   ---+-----+-----
     1 | john      | doe
     2 | stephen   | doe
     3 | karim     | bulla
     4 | garchu    | pedro
(4 rows)

test124=> \q
```

If we try to write records to the reader endpoint, we will get an error as it is a reader endpoint and not a writer endpoint, we can write records only via the writer endpoint.

```
[root@ip-172-31-12-1 ~]# psql -F t -Fc t -U postgres -c "SELECT * FROM team;" -h 172.31.12.1 -p 5432 -v ON_ERROR_STOP=1
Password for user postgres:
psql (13.3, server 13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres> \dt
Did not find any relations.
postgres> \q
Deltas only is on.

postgres> \c test124
You are now connected to database "test124" as user "postgres".
test124=> select * from team;
   sno | first_name | last_name
   ---+-----+-----
     1 | john      | doe
     2 | stephen   | doe
     3 | karim     | bulla
     4 | garchu    | pedro
(4 rows)

postgres> \connect test124
psql (13.3, server 13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
You are now connected to database "test124" as user "postgres".
test124=> insert into team(sno,first_name,last_name) values('1','john','doe');
psql: error: ERROR:  cannot execute INSERT in a read-only transaction
test124=>
```

Connect to Individual Instance Endpoints

Let's connect to the individual endpoints of the instances and perform the read and write operations.

Note: In the current configuration, **instance 1 is the writer node while instance 2 and 3 are reader nodes**. We can perform read and write operations on instance 1 but only read operations on instance 2 and 3



Let's connect to Instance 2 using the reader endpoint and query the table.

Amazon RDS Available

Dashboard Databases Query Editor Performance insights Snapshots Automated backups Reserved instances Proxies

Subnet groups Parameter groups Option groups Custom Availability Zones Custom engine versions

Events Event subscriptions

Recommendations Certificate update

Feedback Log in (US) x

Services Q ec2 X

N Viruses sanyam.sureja@searce.com [Logout] Available

database-2-instance-1 Writer instance PostgreSQL us-east-1b db.mgd.large Available 12.64%

database-2-instance-2 Reader instance PostgreSQL us-east-1a db.mgd.large Available 12.79%

database-2-instance-3 Reader instance PostgreSQL us-east-1d db.mgd.large Available 14.18%

Connectivity & security Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity & security

Endpoint & port

Endpoint database-2-instance-2.bnqapuqeu.us-east-1.rds.amazonaws.com

Port 5432

Networking

Availability Zone us-east-1a

VPC DB_Test_VPC (vpc-0fe37c89a55f15d4)

Subnet group default-vpc-0beef37c89a55f15d4

Subnets subnet-01139203dc5ca107
subnet-080261300276039f7
subnet-0723597c043a15fd
subnet-0538df0bf9799a1005
subnet-01ff50a6db28093d7
subnet-01d1d3a2c7028f86c46

Security

VPC security groups DB_SG (ip-02-3e9f6fe67293d1) Active

Public accessibility No

Certificate authority rds-ca-2019

Certificate authority date August 22, 2024, 10:58 (UTC+10:38)

© 2024, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

PostgreSQL user postgres:
host all 127.0.0.1, ::1 5432
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

```
postgres=> \connect team124
FATAL:  database "team124" does not exist
Previous connection kept
postgres=> \l
List of databases
  Name  | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | rdsadmin=CTC/rdsadmin+
rdsadmin | rdsadmin | UTF8 | en_US.UTF-8 | en_US.UTF-8 | rdsreader=TC/rdsadmin+
template | rdsadmin | UTF8 | en_US.UTF-8 | en_US.UTF-8 | rdsreader=TC/rdsadmin+
template | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/postgres+
test123 | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | rdsreader=TC/postgres
test124 | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | rdsreader=TC/postgres
(6 rows)

postgres=> \connect test124
FATAL:  host all 127.0.0.1, ::1 5432
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
You must specify a database in the command-line.
postgres=> \select * from team;
test124=> \select * from team;
ERROR: syntax error at or near "select"
LINE 1: \select * from team;
          ^
postgres=> \select * from team;
and i first_name | last_name
-----+-----+-----+
1 | john | doe
2 | stephen | doe
3 | paris | bulla
4 | gretchen | pedro
(4 rows)

test124=>
```

Let's connect to the 3rd instance and query the table.

```
[root@ip-172-31-0-124 ~]# psql --host "database-2-instance-1.chnchnyqoqeu.us-east-1.rds.amazonaws.com" --port 5432 --username postgres --dbname postgres
Password for user postgres:
psql (13.6, server 13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \connect test124
You are now connected to database "test124" as user "postgres".
test124=> \select * from team;
   id   | first_name | last_name
-----+-----+-----+
     1  |    john    |    doe
     2  |   stephen  |    lee
     3  |    karen    |    bulla
     4  |    gazchu  |    pedro
(4 rows)
```



4. Let's connect to instance 1 which is the writer instance and write records to it.

```

[rocksdb@ip-10-0-124-112 ~]$ psql --host database-1-instance-1.ckbjpygugiu.us-east-1.rds.amazonaws.com --port 5432 --username postgres --dbname postgres
psql (13.5, server 13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

test124> \c
You are now connected to database "postgres" as user "postgres".
test124> insert into testname(first_name,last_name) values('e','harry'),('jones');
test124> select * from testname;
testname | first_name | last_name
-----+-----+-----
1 | John | doe
2 | Karin | bella
3 | gregu | pedro
4 | Harry | jones
(4 rows)

test124>

```

Rebooting Setup

In case the instances are rebooting, we can connect to the other instances for read operations

```

[rocksdb@ip-10-0-124-112 ~]$ psql --host database-2-instance-2.ckbjpygugiu.us-east-1.rds.amazonaws.com --port 5432 --username postgres --dbname postgres
Password for user postgres:
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
TestDB>

test124> \c
You are now connected to database "postgres" as user "postgres".
test124> insert into testname(first_name,last_name) values('e','harry'),('jones');
test124> select * from testname;
testname | first_name | last_name
-----+-----+-----
1 | John | doe
2 | Karin | bella
3 | gregu | pedro
4 | Harry | jones
(4 rows)

test124>

```



Performing Failover:

Let's perform a manual failover on the DB and check the results.

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
database-2	Multi-AZ DB cluster	PostgreSQL	us-east-1	5 instances	Pending over	-	
database-2-instance-1	Writer instance	PostgreSQL	us-east-1a	db.m6g.large	Available	12.50%	2 Connections
database-2-instance-2	Reader instance	PostgreSQL	us-east-1a	db.m6g.large	Available	12.00%	0 Connections
database-2-instance-3	Reader instance	PostgreSQL	us-east-1d	db.m6g.large	Available	13.50%	0 Connections

3. If we view carefully, after the failover, Node 3 became the writer instance while node 1 and node 2 became the reader instances.

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
database-2	Multi-AZ DB cluster	PostgreSQL	us-east-1	5 instances	Available	-	no
database-2-instance-1	Writer instance	PostgreSQL	us-east-1d	db.m6g.large	Available	12.54%	1 Connections
database-2-instance-2	Reader instance	PostgreSQL	us-east-1b	db.m6g.large	Available	6.00%	0 Connections
database-2-instance-3	Reader instance	PostgreSQL	us-east-1a	db.m6g.large	Available	11.47%	0 Connections
database-2-instance-4	Reader instance	PostgreSQL	us-east-1c	db.m6g.large	Available	12.00%	0 Connections

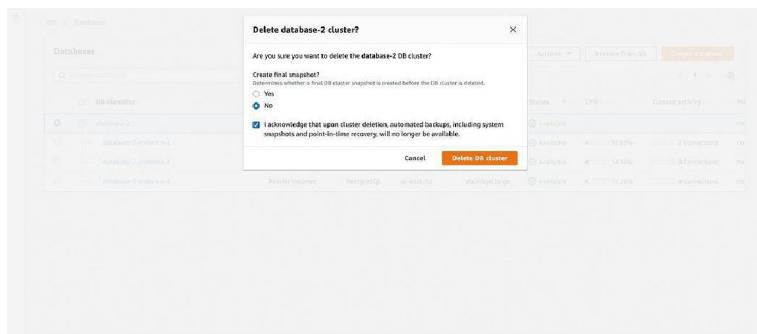
4. Let's perform the failover again and check the results.

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
database-2	Multi-AZ DB cluster	PostgreSQL	us-east-1	5 instances	Pending over	-	
database-2-instance-1	Writer instance	PostgreSQL	us-east-1d	db.m6g.large	Available	12.50%	3 Connections
database-2-instance-2	Reader instance	PostgreSQL	us-east-1b	db.m6g.large	Available	12.00%	0 Connections
database-2-instance-3	Reader instance	PostgreSQL	us-east-1a	db.m6g.large	Available	11.52%	0 Connections

5. Now instance 2 became the writer node while instance 1 and instance 3 became the reader node

Deleting Cluster

Once done, we can delete the cluster and terminate the Ec2 instances as well in order to save costs.



Conclusion:

The Multi AZ RDS with Read capability provides the best of both worlds, High Availability and Read capabilities. This also ensures we don't require separate Read Replica just to offset the read traffic which can be taken care of by the standby nodes, thus ensuring cost saving, greater reliability and availability of the Database Layer. Since the Nodes are highly available, even in case of a failover or disaster, our DB remains safe and completely operational. *AWS is expected to roll out this feature to other DB engines as well and hopefully the SLA would also be amended to ensure this solution is ideal for the Production environments as well, which would serve as a Boon for the database layer in various applications and complex environments.*

Amazon DynamoDB



Quick brush up of DynamoDB

DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services (AWS). It is designed for applications that require low latency, high scalability, and seamless scalability of throughput and storage.

Key features of DynamoDB include:

- **NoSQL Database:**

DynamoDB is a NoSQL database, which means it doesn't adhere to the traditional relational database model. It provides a flexible schema, allowing you to store and retrieve unstructured, semi-structured, and structured data.

- **Fully Managed Service:**

DynamoDB is a fully managed service provided by AWS. This means that AWS handles the underlying infrastructure management, such as provisioning, scaling, replication, backups, and patching, allowing you to focus on building applications rather than managing the database infrastructure.

- **Scalability and Performance:**

DynamoDB offers seamless scalability for both read and write capacity. It automatically distributes data and traffic across multiple servers, allowing the database to handle high volumes of requests and support massive workloads. You can scale up or down based on demand without any downtime.

- **Flexible Data Model:**

DynamoDB uses a key-value data model with support for composite primary keys. Each item (row) in the database is uniquely identified by its primary key, which can be a single attribute or a combination of attributes. This flexible data model enables efficient access patterns for different types of queries.

- **High Availability and Durability:**

DynamoDB replicates data across multiple Availability Zones (AZs) within a region to ensure high availability and durability. It automatically synchronously replicates data across three AZs, providing fault tolerance and data redundancy.



- **Performance at Any Scale:**

DynamoDB offers consistent single-digit millisecond latency for both read and write operations, regardless of the data volume or throughput requirements. It achieves this by using SSD storage and a distributed architecture optimized for low-latency performance.

- **Security and Fine-Grained Access Control:**

DynamoDB provides several security features, including encryption at rest and in transit, fine-grained access control through AWS Identity and Access Management (IAM), and integration with AWS Identity and Access Management Service (IAM roles).

- **Integration with AWS Ecosystem:**

DynamoDB seamlessly integrates with other AWS services, allowing you to build powerful and scalable applications. It can be easily combined with services like AWS Lambda, Amazon S3, Amazon Redshift, and others to create complete data-driven solutions.

Do check out the **DynamoDB 101 : An introduction to**

DynamoDB article if you are new to DynamoDB. That will help you get a quick grasp on what the use cases of something like DynamoDB are.

If you are stuck in choosing a database, the article **SQL vs NoSQL: Choosing the Right Database Model for Your Business Needs** will help you get started on the right path.

Architectural components of DynamoDB

For a clearer understanding, let's follow an example and try to base our thinking on that. Let's take an example of a Payments System. The use-case is that we are designing the database for the same. Let's say the system supports 2 modes of Payments:

- Wallet Payments
- Card Payments

Apart from this whenever there is a transaction done with an amount greater than 100\$, then we need to trigger a notification after 3 days into the payments service that propagates the same to the Rewards Service which then issues a coupon to the user.

I am listing out the components here. We will be going in depth into each component after that. Please go through them sequentially as you would require previous context to understand the latest.

1. Tables
2. Items
 - Time To Live (TTL)
3. Attributes



4. Primary Key
 - Partition Key (Hash Key)
 - Sort Key (Range Key)
5. Secondary Indexes
 - Local Secondary Index (LSI)
 - Global Secondary Index (GSI)
6. Streams
7. DynamoDB Accelerator (DAX)

Let's understand these components one by one with the help of the example. I will be providing examples of these components to their counterpart examples in SQL databases.

1. Tables

Tables are the fundamental storage units that hold data records. The counterpart for a table in the SQL world is also a table.

According to our example, the **Table** would be a **Transactions table** that would essentially record all the data w.r.t. transactions done by the system like transaction data, transaction status history data etc

2. Items

- **Item** is a single data record in a table. Each item in a table can be uniquely identified by the stated **Primary Key (Simple Primary Key or Composite Primary Key)** of the table.
- According to our example, in your **Transactions** table, an item would be a particular Transaction's data or Transaction's history record.
- The counterpart for an **item** in the SQL world is a **record** or a **row**.
- DynamoDB provides a **TTL** feature that allows you to automatically delete expired items from a table. By specifying a TTL attribute for a data record, DynamoDB will automatically remove the item from the table when the TTL value expires.
- Now, this would be helpful in fulfilling the use-case presented in our example where we would need to trigger a notification 3 days post having a successful transaction over 100\$. We simply set the appropriate TTL when the required conditions are met and we are done.

3. Attributes

Attributes are pieces of data attached to a single item. It can be of different data types, such as string, number, Boolean, binary, or complex types like lists, sets, or maps. Attributes are not required to be predefined, and each item can have different attributes.



An **attribute** is comparable to a **column** in the SQL world. In our example a few attributes can be the following:

- **transaction_id:** This attribute represents the unique identifier of each transaction. It could be a string or a number.
- **transaction_date:** This attribute stores the timestamp or date when the transaction occurred. It could be stored as a string or in a specific date/time format.
- **user_id:** This attribute represents the identifier of the customer associated with the transaction. It could be a string or a number, depending on how you identify customers.
- **amount:** This attribute stores the monetary value of the transaction. It could be represented as a number or a string, depending on your application's requirements.

4. Primary Key

Primary key is a unique identifier that is used to uniquely identify each item (row) within a table. The primary key is essential for efficient data retrieval and storage in DynamoDB. It consists of one or two attributes:

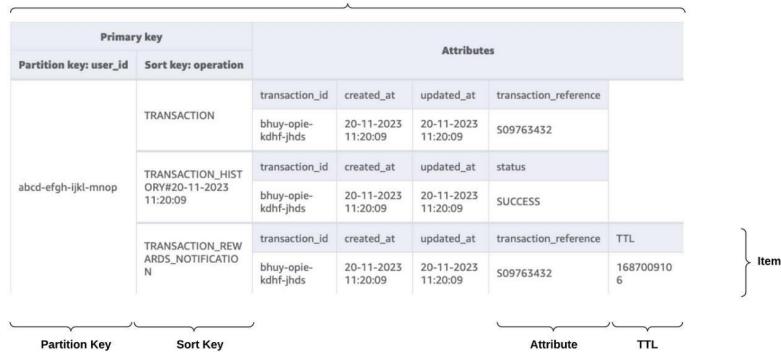
- **Partition Key (Hash Key):**

The partition key is a single attribute that determines the partition or physical storage location of the item. DynamoDB uses the partition key's value to distribute the data across multiple partitions for scalability and performance. Each partition key value must be unique within the table. In our case, the Transactions table can have a Partition Key of **user_id**.

- **Sort Key (Range Key):**

The sort key is an optional attribute that, when combined with the partition key, creates a composite primary key. The sort key allows you to further refine the ordering of items within a partition. It helps in performing range queries and sorting items based on the sort key's value. The combination of the partition key and the sort key must be unique within the table. Examples of sort keys include timestamps, dates, or any attribute that provides a meaningful ordering of items within a partition. In our case, the Transactions table can have a Sort Key depicting the type of item it is. As an example, one of the item can have a Sort Key with value **TRANSACTION_HISTORY** that essentially stores the status updates of the transaction. Similarly, something like **TRANSACTION_REWARDS_NOTIFICATION** can cater to our requirement of sending a notification of a specific transaction to the Rewards system.

Table



Quick display of the Architectural components of DynamoDB

5. Secondary Index

The primary key uniquely identifies an item in a table, and you may make queries against the table using the primary key. However, sometimes you have additional access patterns that would be inefficient with your primary key. DynamoDB has secondary indexes to enable these additional access patterns. DynamoDB supports two kinds of secondary index:

Local Secondary Index

Local Secondary Index (LSI) is an additional index that you can create on a table. LSI allows you to define a different sort key for the index while keeping the same partition key as the table. They are useful when you have different access patterns or when you want to query data based on attributes other than the primary key. They provide more flexibility in data retrieval without the need to create a separate table or duplicate data.

- Same Partition Key:** The partition key for the LSI is the same as the base table's partition key. This means that the LSI partitions the data in the same way as the base table. There are a few caveats however, which are listed down as follows:
- Subset of Attributes:** When creating an LSI, you can specify a subset of attributes from the base table to include in the index. These attributes become part of the index and can be projected into the index's key schema or as non-key attributes.
- Query Flexibility:** With an LSI, you can perform queries that utilize the LSI's partition key and sort key. This allows you to efficiently retrieve a subset of data based on specific query requirements without scanning the entire table.
- Eventual Consistency:** Unlike the primary index (base table), LSIs only support eventual consistency for read operations. This means that after a write operation, the index may not immediately reflect the updated data.
- Write Performance:** When you modify data in a table with LSIs, DynamoDB needs to update the base table as well as all the corresponding LSIs. This can impact the write performance compared to a table without any secondary indexes.



However, keep in mind that the number of LSIs you can create per table is limited, and the provisioned throughput is shared between the base table and all its LSIs.

For example, if there is a requirement to query all transactions for a specific customer within a particular date range. Then, you can utilize a composite LSI of the type + created_at attribute for achieving that.

Primary key		Attributes				
Partition key: user_id	Sort key: LSI_1	operation	transaction_id	created_at	updated_at	transaction_reference
abcd-efgh-ijkl-mnop	TRANSACTION#12-12-2023T11:20:09	TRANSACTION#jfuf-wter-mjui-gsfe	jfuf-wter-mjui-gsfe	12-12-2023T11:20:09	12-12-2023T11:20:09	539769481
	TRANSACTION#20-11-2023 11:20:09	TRANSACTION#bhuy-opie-kdhf-jhds	bhuy-opie-kdhf-jhds	20-11-2023T11:20:09	20-11-2023T11:20:09	S09763432
	TRANSACTION_HISTORY#20-11-2023T11:20:09	TRANSACTION_HISTORY#bhuy-opie-kdhf-jhds#20-11-2023T11:20:09	bhuy-opie-kdhf-jhds	20-11-2023T11:20:09	20-11-2023T11:20:09	SUCCESS
	TRANSACTION_REWARDS_NOTIFICATION#20-11-2023T11:20:09	TRANSACTION_REWARDS_NOTIFICATION#bhuy-opie-kdhf-jhds	bhuy-opie-kdhf-jhds	20-11-2023T11:20:09	20-11-2023T11:20:09	S09763432
						1687009106

Local Secondary Index with a hybrid of type + created_at

In this query, **user_id** represents the specific user you want to query, and the sort key contains the range of dates you are interested in. By utilizing the LSI, DynamoDB can efficiently retrieve the transactions that match the specified **user_id** and fall within the given date range.

The LSI in this example helps you perform targeted queries on transactions based on the customer ID and transaction date, without having to scan the entire table. It provides an alternative access pattern for retrieving transaction data and can improve query performance for specific use cases.

Global Secondary Index: It is an additional index that you can create on a table. It provides an alternative way to query the data within the table using attributes other than the primary key. GSI allows you to define a different partition key and sort key from the base table, providing alternative access patterns and query capabilities.

Different Partition and Sort Key: Unlike Local Secondary Indexes (LSIs), GSIs allow you to define a different partition key and sort key from the base table. This means you can partition and sort the data in the GSI independently of the base table's primary key.



Query Flexibility: With a GSI, you can perform queries based on the GSI's partition key and sort key. This allows you to efficiently retrieve a subset of data based on specific query requirements without scanning the entire table.

Projection: When creating a GSI, you can choose to project a subset of attributes from the base table into the index. These projected attributes can be included in the index's key schema or as non-key attributes, allowing you to retrieve a subset of data directly from the GSI without accessing the base table.

Separate Provisioned Throughput: GSIs have their own provisioned throughput capacity, independent of the base table. This means you can specify different read and write capacity units for the GSI, enabling you to optimize performance based on the expected query workload.

Eventual Consistency: Similar to LSIs, GSIs only support eventual consistency for read operations. After a write operation, the index may not immediately reflect the updated data.

However, keep in mind that creating GSIs can consume additional storage and provisioned throughput, so careful consideration is needed to optimize the indexing strategy based on your application's requirements.

Let's introduce the field called **payment_method** which we mentioned earlier. To provide an alternative access pattern based on payment methods, you can create a GSI on the transaction table with the following attributes:

- Partition Key:** A different attribute representing the payment method, `payment_method` being that attribute here.
- Sort Key:** The `transaction_id`, which uniquely identifies each transaction.

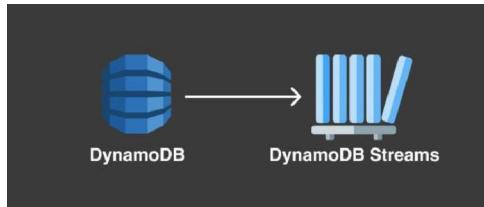
Primary key		Attributes						
Partition key: <code>payment_method</code>	Sort key: <code>transaction_id</code>	user_id	operation	created_at	updated_at	transaction_reference	LSI_1	
CARD	bhuy-opie-kdhf-jhds	abcd-efgh-ijkl-mnop	TRANSACTION#bhuy-opie-kdhf-jhds	20-11-2023T11:20:09	20-11-2023T11:20:09	S09763432	TRANSACTION#20-11-2023 11:20:09	
	bhuy-opie-kdhf-jhds	abcd-efgh-ijkl-mnop	TRANSACTION_ON_REWARDS_NOTIFICATION#bhuy-opie-kdhf-jhds	20-11-2023T11:20:09	20-11-2023T11:20:09	S09763432	1667009106	TRANSACTION_ON_REWARDS_NOTIFICATION#20-11-2023T11:20:09
WALLET	jfuf-wter-mjui-gsfe	abcd-efgh-ijkl-mnop	TRANSACTION#jfuf-wter-mjui-gsfe	12-12-2023T11:20:09	12-12-2023T11:20:09	S39769481	TRANSACTION#12-12-2023T11:20:09	

Global Secondary Index: PartitionKey — `payment_method`, SortKey `transaction_id`

The GSI in this example helps you perform targeted queries on transactions based on the payment method, without having to scan the entire table. It provides an alternative access pattern for retrieving transaction data and can improve query performance for specific use cases.



6. Streams



DynamoDB Streams is a feature of Amazon DynamoDB that provides a time-ordered sequence of item-level modifications made to a table. It captures a stream of events that represent changes to the data in a DynamoDB table, including inserts, updates, and deletions. Each event in the stream contains information about the modified item, such as its primary key, attribute values before and after the modification, and a timestamp indicating when the modification occurred.

Here are some key features and use cases of DynamoDB Streams:

- **Capture Changes:**

DynamoDB Streams captures changes happening in real-time as modifications are made to the table. It provides a durable and reliable way to track and react to changes in your data. This use-case is often referred to as **CDC** or **Change Data Capture**.

- **Integration and Event-Driven Architecture:**

Streams can be used to integrate DynamoDB with other AWS services or external systems. By processing the stream events, you can trigger actions or update downstream systems based on changes in the DynamoDB table.

- **Change History and Audit Logs:**

Streams provide a complete history of changes made to a DynamoDB table, allowing you to maintain an audit trail of all modifications. You can use the stream to review past changes, investigate issues, or perform compliance and security checks.

- **Data Synchronization:**

Streams can be used for replicating data across multiple DynamoDB tables or databases. By consuming the stream and applying the changes to other destinations, you can keep different data stores synchronized in near real-time.

- **Data Transformation and Analytics:**

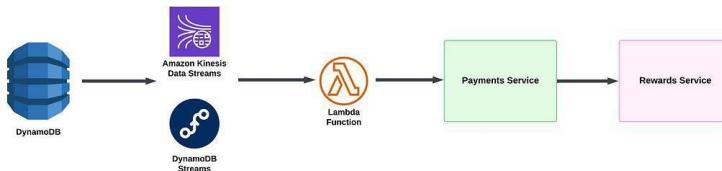
By processing the stream events, you can transform the data, aggregate information, and perform real-time analytics or generate derived datasets for reporting purposes.

- **Cross-Region Replication:**

Streams can be utilized to replicate data from one DynamoDB table to another in a different AWS region. This helps in creating disaster recovery setups or distributing read traffic across regions.

To consume DynamoDB Streams, you can use AWS Lambda, which allows you to write code that runs in response to stream events. You can also use AWS services like **Amazon Kinesis Data Streams** or custom applications to process and react to the stream events.

Remember our use-case that whenever there is a transaction done with an amount greater than 100\$, then we need to trigger a notification after 3 days into the payments service that propagates the same to the **Rewards Service** which then issues a coupon to the user. **DDB Streams** along with **TTL** can help us achieve this.



Flow diagram for the Rewards Notification DDB Streams use-case

7. DynamoDB Accelerator (DAX)

DynamoDB Accelerator is an in-memory caching service provided by Amazon Web Services (AWS) specifically designed for DynamoDB.

DAX improves the performance of DynamoDB by caching frequently accessed data in memory, reducing the need to access the underlying DynamoDB tables for every request. It provides low-latency read access to the cached data, resulting in faster response times and reduced database load.

Here are some key features and benefits of DAX:

- **In-Memory Caching:**

DAX caches frequently accessed data from DynamoDB tables in memory. This eliminates the need for repeated reads from the database, resulting in reduced latency and improved response times for read-intensive workloads.

- **Seamless Integration:**

DAX is fully compatible with DynamoDB and integrates seamlessly with existing DynamoDB applications. You can simply point your application to the DAX endpoint, and it will automatically route read requests to the DAX cluster.

- **High Performance:** DAX delivers microsecond latency for read operations, enabling applications to handle millions of requests per second with consistent, predictable performance.

- **Automatic Data Management:**

DAX automatically manages the cache and keeps it in sync with the underlying DynamoDB tables. It handles invalidations and updates to ensure data consistency between the cache and the database.



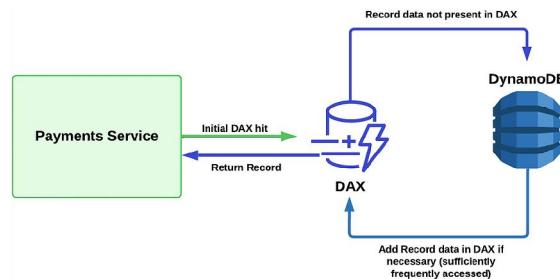
- **Scalability and Availability:**

DAX is a managed service that provides scalability and high availability. It automatically scales the cache capacity based on the workload, and data is distributed across multiple nodes to ensure durability and fault tolerance.

- **Cost Optimization:**

By reducing the number of read operations on the underlying DynamoDB tables, DAX can help lower the cost of running read-intensive applications by reducing provisioned throughput and minimizing the number of DynamoDB read capacity units required.

DAX is particularly useful for applications with high read traffic, such as Payment Recon systems, real-time analytics, gaming leader-boards, session stores etc. It improves the overall performance and efficiency of DynamoDB, providing a seamless caching layer that enhances the speed and scalability of your applications without sacrificing data consistency.



DynamoDB and its features

How to create a simple table, add data, scan and query the data, delete data, and delete the table.

What is DynamoDB?

DynamoDB is a fully managed NoSQL serverless database offered by AWS. DynamoDB is a great fit for mobile, web, gaming, ad tech, and IoT applications where scalability, throughput, and reliable performance are key considerations.

Features of DynamoDB:

- **NoSQL database:**

Managed and Serverless: It has a flexible schema which allows you to have many different attributes for one item. We can easily adapt business requirement change without having to refine the table schema. It also supports key-value and document data models.

- **Managed and Scalable service:**

As it is a managed service, you don't need to worry about the provisioning, managing, maintaining and operating the underlying servers. It is pay-as-go service which scales to zero and automatically scales tables to adjust the capacity to match your workload for you.



- **DynamoDB Global Table:**

With Global tables you can read and writes can access data locally in the selected Regions to get single-digit millisecond read and write performance. global tables automatically scale capacity to accommodate your multi-Region workloads which improves your applications multi- region resiliency.

- **DynamoDB Streams:**

It captures data modification events i.e. create, update, or delete items in a table near real time. Each record has a unique sequence number which is used for ordering

- **Secondary Indexes:**

DynamoDB provides fast access to items in a table by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available, to allow efficient access to data with attributes other than the primary key.

DynamoDB provides 2 kinds of Indexes: Global Secondary Index and Local Secondary Index.

- **DynamoDB Accelerator (DAX):**

Built-in caching service for DynamoDB which is fully managed and highly available. It can serve millions of requests per second with micro- seconds response time for read-heavy workload.

- **On-Demand Capacity Mode:**

Allows user to scale seamlessly without capacity planning. It ensures optimal performance and cost efficiency for fluctuating workloads.

- **Point-in-time Recovery:**

Enables users to restore their data to any second within a 35 days of retention period, protecting against accidental data loss. It provides peace of mind by allowing effortless recovery from user errors or malicious actions, ensuring data integrity and availability.

- **Encryption at Rest:**

By default, DynamoDB encrypts all data at rest using AWS KMS (Key Management Service), providing an additional layer of protection against unauthorized access.

- **Built-in support for ACID transactions:**

DynamoDB guarantees ACID (Atomicity, Consistency, Isolation, Durability) properties for transactions, ensuring reliable and predictable data operations. Amazon DynamoDB provides native, server-side transactions, simplifying the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables.

Login into your AWS Management Console account and search for DynamoDB

Creating a NoSQL table:

Click on Create Table



The screenshot shows the Amazon DynamoDB homepage. On the left, there's a sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main content area features a large heading "Amazon DynamoDB" and sub-headings "A fast and flexible NoSQL database service for any scale". It includes a brief description of what DynamoDB is, a "Get started" button with a "Create table" option, a "How it works" section with a video thumbnail, and a "Pricing" section.

Give the table name (AWS Learners), partition key (LearnerName) and sort key (Certifications)

The screenshot shows the "Create table" wizard in the AWS DynamoDB console. The "Table details" step is active. The "Table name" field contains "AWSLearners". The "Partition key" section shows "LearnerName" as the primary key type, which is a string. The "Sort key (optional)" section shows "Certifications" as the secondary key type, also a string.

For table Settings, select Customize settings (To enable auto-scaling for our table). DynamoDB auto scaling will change the read and write capacity of your table based on request volume. Rest of the settings should be remained as it is, just add the json policy.

The screenshot shows the "Table settings" page in the AWS DynamoDB console. The "Customize settings" option is selected, indicating that auto-scaling is enabled. Other settings visible include "Table class" (DynamoDB Standard selected), "Capacity calculator", and "Read/write capacity settings".



Resource-based policy for table Info

The resource-based policy, written in JSON, helps manage access to this DynamoDB table.

```
1 | [
2 |   "Version": "2012-10-17",
3 |   "Statement": [
4 |     {
5 |       "Sid": "Statement1",
6 |       "Effect": "Allow",
7 |       "Principal": {
8 |         "AWS": "arn:aws:iam::654654167899:user/Ruchika"
9 |       },
10 |      "Action": "dynamodb:...",
11 |      "Resource": "arn:aws:dynamodb:us-east-1:654654167899:table/AWSLearners"
12 |    }
13 |  ]
14 |]
```

JSON Ln 1, Col 1

Add data to the NoSQL table

Click on Create Item

DynamoDB > Explore items > AWSLearners

Tables (1) AWSLearners

Any key Any key value

Find items by value name

Completed: Read capacity units consumed: 0.5

Items returned (1)

LearnerName (String) Certifications (String)

Ruchika AWS Certified Cloud Practitioner

Create item

Attributes

Attribute name Value Type

LearnerName (String) Empty value String

Certifications (String) Empty value String

Add new attribute

Cancel Create item

Enter the value for LearnerName and Certification

DynamoDB > Explore items > AWSLearners

Tables (1) AWSLearners

Any key Any key value

Find items by value name

Completed: Read capacity units consumed: 0.5

Items returned (2)

LearnerName (String) Certifications (String)

Kunal AWS Security Specialist

Ruchika AWS Certified Cloud Practitioner



Query the NoSQL Data

Search for data in the table using query operations. In DynamoDB, query operations are efficient and use keys to find data.

DynamoDB > Explore items > AWSLearners

DynamoDB > Explore items > AWSLearners

LearnerName	Certifications
Ruchika	AWS Certified Cloud Practitioner
Ruchika	AWS Security Specialist
Keyur	AWS Solution Architect
Keyur	AWS SysOps Certification
Jay	AWS DevOps Certification

Deleting an existing item

Here I am deleting the item named Jay

DynamoDB > Explore items > AWSLearners

AWSLearners

Completed. Read capacity units consumed: 0.5

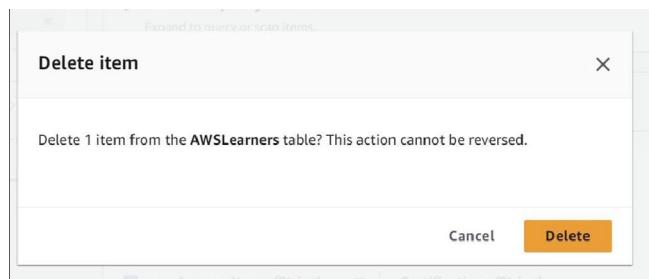
Items returned (1/5)

LearnerName	Certifications
Ruchika	AWS Certified Cloud Practitioner
Ruchika	AWS Security Specialist
Keyur	AWS Solution Architect
Keyur	AWS SysOps Certification
Jay	AWS DevOps Certification

Actions ▾

- Edit item
- Duplicate item
- Delete items**

Download selected items to CSV
Download results to CSV

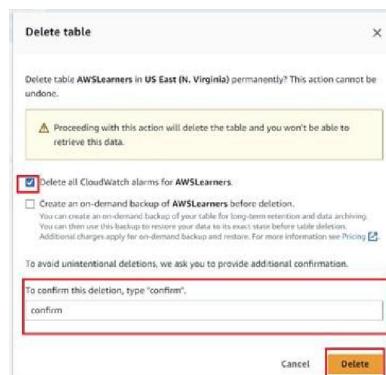


Delete a NoSQL Table:

Deleting the entire AWSLearner table.

The first screenshot shows the AWS DynamoDB 'Tables' page with the 'AWSLearners' table selected. The second screenshot shows the same page after the deletion has been submitted, with the table status now listed as 'Deleting'.

Type confirm to delete entire table





So, we created our first DynamoDB table, added items to the table, and then queried the table to find the items we wanted. Also, learned how to visually manage your DynamoDB tables and items through the AWS Management Console.

SIMPLE STORAGE SERVICE



Amazon S3 (Simple Storage Service) is an industry-leading object storage service. You can securely store any objects such as files, images, videos, and payloads. S3 is the same as drives that you use daily basis such as Google Drive and Microsoft OneDrive. It is so powerful and reliable that can scale indefinitely. Its ease of use has garnered widespread admiration among engineers, cementing its status as a favored service within the tech community.

In S3, you can sidestep complex setups entirely. All you need is just provide valid credentials. You can create an IAM user for the app with an access key. However, relying solely on a permanent access key isn't considered best practice. A preferable approach involves deploying the app on EC2 or ECS and utilizing IAM roles. This method generates temporary tokens and manages rotation automatically, enhancing security.

Novices often misconstrue S3 as a database, when in fact, it lacks the typical characteristics of a database technology. However, it frequently complements databases in usage scenarios. For instance, storing actual images for posts in S3 while retaining object keys in the database as pointers to these images.

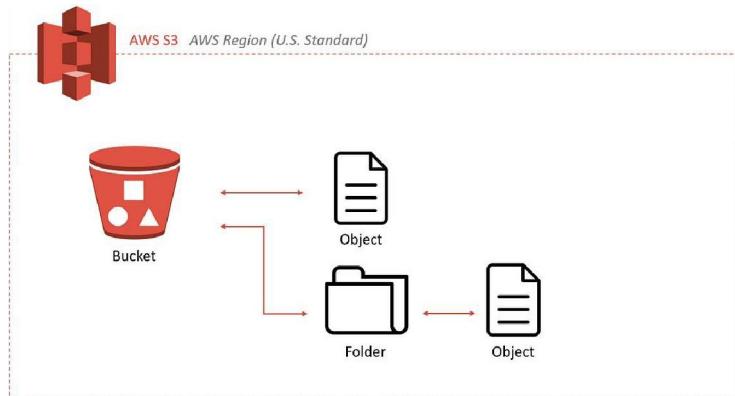
Many organizations in the USA use S3 as a datalake. A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. On top of the S3 datalake, you can utilize analytical services like Athena and Glue to analyze organizational data and extract insights without impacting the business apps.

S3 Concepts:

- **Buckets** are containers for objects stored in Amazon S3. Every object is contained in a bucket. Think of it as a folder for organizing files.
- **Objects** are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. Maximum object size of 5TB. You can store an unlimited number of objects.
- **Keys** are the unique identifiers for an object within a bucket. Every object in a bucket has exactly one key that you can use later for object retrieval.

If you enable **versioning** in the bucket, it doesn't overwrite the existing object if the key already exists. Instead, it adds enumerated versions.

The combination of a bucket, key, and version ID uniquely identifies each object. When you create an S3 bucket, you have to select the region where it belongs. When you go to the S3 console, you will see all buckets on one screen. That illustrates the S3 bucket is a regional service but the S3 namespace is global. The best practice is to select the region that is physically closest to you, to reduce transfer latency. You can create a folder in an S3 bucket to organize your data. Data engineers use S3 folders for achieving data partitioning by date in the S3 data lake.



S3 Permissions

S3 permissions allow you to have granular control over who can view, access, and use specific buckets and objects. Permissions functionality can be found on the bucket and object levels. There are 3 types of permissions in S3:

- **Identity-based** — IAM user or role has permission to access S3. For example, A developer who has S3FullAccess or an EC2 instance with an IAM role that has S3FullAccess policy.
- **Resource-based** — You can write the S3 bucket policy where the principal tag defines who can access it. For instance, the bucket policy can allow the public to get all objects for static website hosting.
- **Access Control List (ACL)** — Sets policies to both a bucket and an object. With identity-based and resource-based policies, you cannot assign a policy to an object.

Storage Classes

A storage class represents the classification assigned to each object. Each storage class has varying attributes that dictate:

- Storage cost
- Object availability — It is the percent (%) over a one-year time period that a file stored in S3 will be accessible.



- Object durability — It is the percent (%) over a one-year time period that a file stored in S3 will not be lost. S3 provides 11 Nines durability which means if you store 1 million objects in S3 for 10 million years, you would expect to lose 1 file. There's a higher likelihood of an asteroid destroying Earth within a million years.
- Frequency of access

Storage classes are:

- S3 Standard for frequently accessed data. The default option and expensive.
- S3 Express One Zone for your most frequently accessed data. It provides consistent single-digit millisecond latency. Good for data-intensive apps that require fast runtime.
- S3 Standard-Infrequent Access (S3 Standard-IA) and S3 One Zone- Infrequent Access (S3 One Zone-IA) for less frequently accessed data.
- The storage cost is cheaper but charges more when retrieving data.
- S3 Glacier Instant Retrieval for archive data that needs immediate access. Decent pricing for object storage.
- S3 Glacier Flexible Retrieval (formerly S3 Glacier) for rarely accessed long-term data that does not require immediate access. Data retrieval could take minutes to hours.
- Amazon S3 Glacier Deep Archive (S3 Glacier Deep Archive) for long-term archive and digital preservation with retrieval in hours at the lowest cost storage in the cloud.
- S3 Intelligent-Tiering for automatic cost savings for data with unknown or changing access patterns.

You can set lifecycle configurations that automate the migration of an object's storage class to a different storage class, or its deletion, based on specified time intervals for cost optimization. For example, you can set the following rule:

- I'll be regularly using a work file for the next 30 days, so please keep it in the standard class during this time.
- After this initial period, I'll only need to access the file once a week for the subsequent 60 days. Therefore, after 30 days, please transfer the file from the standard class to the infrequently accessed class.
- Following this, I anticipate no further need to access the file after a total of 90 days, but I'd like to retain it for archival purposes. Accordingly, after 90 days, please move the file from the infrequently accessed class to Glacier.

Whenever possible, it's advisable to establish lifecycle configurations as part of best practices for cost efficiency. A lifecycle policy can be applied to the entire bucket, specific folders, and objects.



S3 Static web hosting

One standout feature of S3 is its capability for static web hosting, which I've utilized for hosting my side projects. What's particularly remarkable about hosting a website on S3 is its cost-effectiveness — it's essentially free. Setting up static web hosting with S3 is straightforward, requiring just three simple steps:

- Make the bucket public. And make objects public by writing bucket policy.
- Enable static web hosting in the bucket.
- Build the front-end app for example in React. Then drag and drop the build files. After these steps, if you get a 403 (Forbidden) error, make sure the bucket policy allows public reads as I stated in Step 1.
- Another trick I do with the S3 static web hosting feature is that redirecting requests to another website in a serverless and cost-effective way without needing a launch a proxy server and configure tools like Nginx. When you type “fb.com”, it redirects you to “facebook.com”.

That functionality is achievable through the static website hosting feature as illustrated below.

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting
 Disable
 Enable

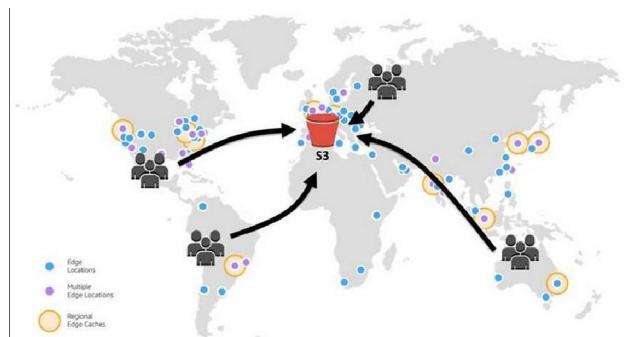
Hosting type
 Host a static website
Use the bucket endpoint as the web address. [Learn more](#)
 Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#)

Host name

Target bucket website address or personal domain

Protocol - *Optional*
 none
 http
 https

To optimize your website's performance globally and bolster security, consider leveraging the CDN (Content Delivery Network) service, CloudFront, on top of the website hosted on S3. It's advisable to implement Origin Access Identity (OAI) as a best practice, as this secures the website assets stored in Amazon S3. With OAI, direct public access to the bucket is restricted, ensuring that the public can only access the website through CloudFront. Additionally, CloudFront aids in cost reduction by minimizing the number of requests to S3. It's important to note that every individual request in S3 incurs a charge, highlighting the general cost implications associated with cloud-based services.



Clients directly accessing a static website hosted in Europe



Clients accessing a static website deploy through the CDN (CloudFront)

CloudFront is a versatile and widely utilized service. Here are the key aspects:

- Blacklisting every IP address of a country would be impractical and labor-intensive. Instead, use CloudFront for blacklisting countries.
- CloudFront reduces S3 cost. Because the content is delivered from the edge server and there are no API calls against S3.
- You can run business logic code and small functions with Lambda@Edge. For example, inspect headers of requests and pass the requests down only a valid token is present.
- You can implement an advanced caching behavior based on query strings and headers and serve private content through the CDN.
- It helps with video streaming.

You can do a path-based routing. For example, if the request has “us” in the path, then forward it to the origin in the US region. If it is “ap”, then forward it to Asia Pacific, etc.

Presigned URL

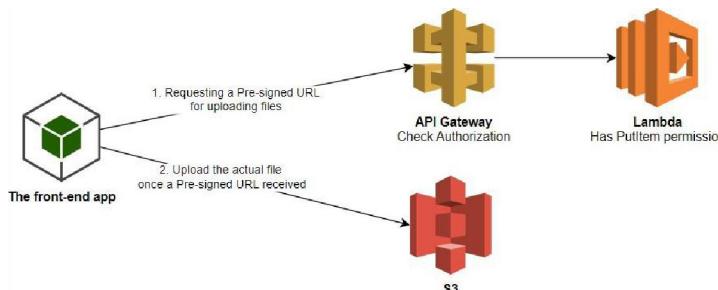
The S3-presigned URL stands out as a crucial feature, widely utilized in practical applications. It allows clients to download and upload an object to the bucket with a temporary URL. For

instance, consider a book- selling application where customers need immediate access to their purchased books. In such cases, leveraging S3-presigned URLs allows you to generate temporary links, granting customers access to download their books promptly after purchase.

Another practical application of S3-presigned URLs arises when uploading large files to an S3 bucket through the API Gateway. Typically, API Gateways serve as the entry point for RESTful endpoints. However, API Gateway has a limitation of handling requests up to 10 MB. I encountered this issue while dealing with an endpoint for blobs.

The endpoint functioned properly for certain files but failed for others, leading to extensive debugging efforts. Initially, I couldn't pinpoint whether the problem lay with the API Gateway or S3 configuration, as everything appeared correct. Eventually, I discovered the 10 MB request limit of API Gateway, which was causing the failure. To address this, I implemented a solution using pre-signed URLs. Instead of directly storing files from the API Gateway to S3, I introduced an additional step.

Initially, the request goes to the API Gateway, where authorization for file storage is verified. If authorized, the API Gateway returns a presigned URL. Subsequently, the client application makes another call to store the file using the provided URL.



S3 Gateway endpoint

All resources within the AWS cloud are integrated via the AWS Global Network, as they operate within the same AWS network and infrastructure. These endpoints facilitate communication between instances within a Virtual Private Cloud (VPC) and various AWS services, enabling efficient interaction across different accounts within the AWS ecosystem. AWS resources in your account can be connected to the resources in my AWS account through VPC endpoints.

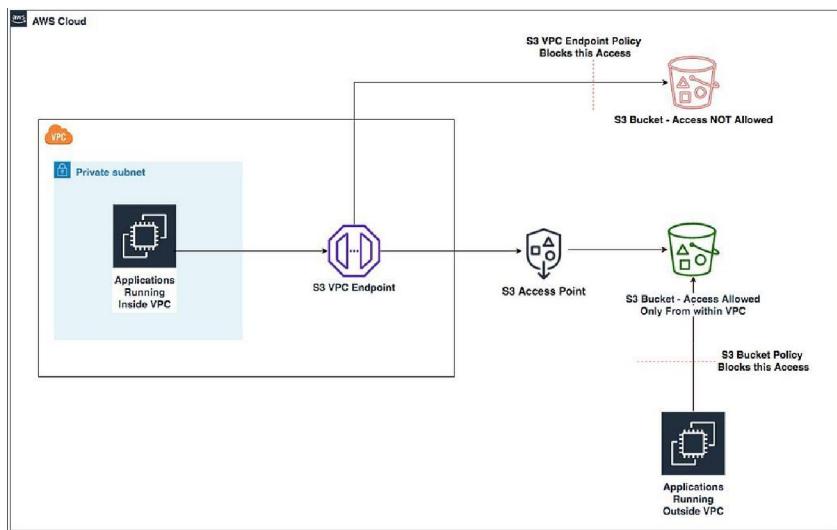
When I was working on a multi-account initiative, we frequently encountered the need to link resources across various accounts. VPC endpoints come in handy for this purpose. When I required connectivity between resources in different accounts, I simply submitted a ticket to our DevOps team requesting the creation of a VPC endpoint. This streamlined the process, enabling seamless connectivity to resources in different AWS accounts. Although VPC endpoints offer practical solutions, it's essential to note that they incur additional charges.



Therefore, it's advisable to decommission VPC endpoints once their use is no longer required, as a best practice.

S3 is a publicly accessible service available for anyone to use. To connect to S3, you need a valid token and an internet connection.

However, in certain scenarios, EC2 servers may not have internet connectivity due to stringent security reasons. Nonetheless, these servers can still establish connections with S3 using a VPC endpoint, specifically the S3 Gateway endpoint. You can configure which resources can access the S3 bucket through the VPC endpoint by writing an S3 bucket policy.

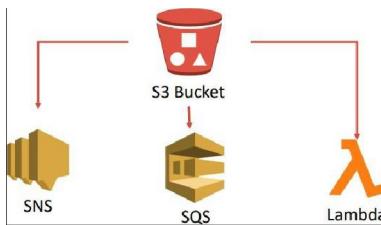


S3 Event notification

You can use the Amazon S3 Event Notifications feature to receive notifications when certain events happen in your S3 buckets such as a new object being created, or an object getting removed. It is a way to achieve a modern Event-Driven Architecture that emphasizes asynchrony and loose-couple which helps achieve better responsiveness.

When users upload profile pictures, synchronously generating a thumbnail could take seconds. However, employing an Event-Driven approach with S3 Event Notifications could significantly reduce latency from seconds to milliseconds. Here's how it works: After the user uploads the profile picture, it's stored in S3. Subsequently, S3 event notifications initiate a new object-created event, triggering a Lambda function. This Lambda function asynchronously generates the thumbnails, optimizing the process for faster response times.

The S3 Event notification can trigger SNS, SQS, and Lambda. When setting up the event notification, make sure you have the right resource-based policy set on the destination or SNS/SQS.



S3 Global Replication

A seasoned software architect shared with me a strategy for architecting global applications, S3 Global Replication. This feature boasts impressive capabilities, allowing data replication between regions in a matter of minutes. The replication enables automatic, asynchronous copying of objects across Amazon S3 buckets between different accounts and regions. An object may be replicated to a single destination bucket or multiple destination buckets.

When to use the global replication:

- Data redundancy — If you need to maintain multiple copies of your data in the same, or different AWS Regions, or across different accounts. S3 Replication powers your global content distribution needs, compliant storage needs, and data sharing across accounts. Replica copies are identical to the source data, that retain all metadata, such as the original object creation time, ACLs, and version IDs.
- Replicate objects to more cost-effective storage classes — You can use S3 Replication to put objects into S3 Glacier, S3 Glacier Deep Archive, or another storage class in the destination buckets.

Maintain object copies under a different account

S3 Cross-Region Replication (CRR) is used to copy objects across Amazon S3 buckets in different AWS Regions. When to use Cross- Region Replication:

- Meet compliance requirements — Although Amazon S3 stores your data across multiple geographically distant Availability Zones by default, compliance requirements might dictate that you store data at even greater distances (regions).
- Minimize latency — If your customers are in two geographic locations, you can minimize latency in accessing objects by maintaining object copies in AWS Regions that are geographically closer to your users.
- Increase operational efficiency — If you have compute clusters in two different AWS Regions that analyze the same set of objects, you might choose to maintain object copies in those Regions.

Same-Region Replication (SRR) is used to copy objects across Amazon S3 buckets in the same AWS Region. SRR can help you do the following:

- Aggregate logs into a single bucket — If you store logs in multiple buckets or across multiple accounts, you can easily replicate logs into a single, in-region bucket. This allows for simpler processing of logs in a single location.



- Configure live replication between production and test accounts — If you or your customers have production and test accounts that use the same data, you can replicate objects between those multiple accounts, while maintaining object metadata.

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock provides two ways to manage object retention:

- Retention period — Specifies a fixed period of time during which an object remains locked. During this period, your object is WORM-protected and can't be overwritten or deleted.
- Legal hold — Provides the same protection as a retention period, but it has no expiration date. Instead, a legal hold remains in place until you explicitly remove it.

Object Lock works only in versioned buckets.

Multipart Upload

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation. It can make your app faster. But adds complexity as well. For example, you have to provide more parameters in the API and there are edge cases such as what happens when there are incomplete files.

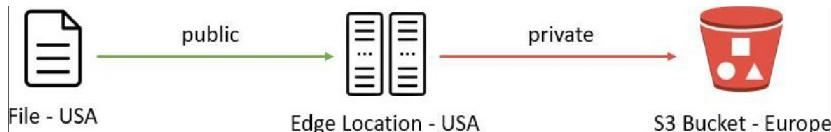
Benefits of multi-part upload :

- Improved throughput — You can upload parts in parallel to improve throughput.
- Quick recovery from any network issues — Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- Pause and resume object uploads — You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- Begin an upload before you know the final object size — You can upload an object as you are creating it.

S3 Transfer Acceleration

Amazon S3 Transfer Acceleration offers a significant boost in content transfers. Users operating web or mobile applications with a broad user base or those hosted far from their S3 bucket may encounter prolonged and fluctuating upload and download speeds over the internet. S3 Transfer Acceleration (S3TA) addresses these challenges by minimizing variability in internet routing, congestion, and speeds that typically impact transfers. It effectively reduces the perceived

distance to S3 for remote applications by leveraging Amazon CloudFront's Edge Locations and AWS backbone networks, along with network protocol optimizations, S3TA enhances transfer performance, ensuring smoother and faster data transfers.



S3 Encryption

In our database, we had a column storing PII (Personal Identifiable Information) data, which required encryption for security purposes.

Rather than investing significant effort and writing extensive code to handle this, we opted for a solution with minimal effort. Leveraging AWS S3 encryption support, we securely stored PII data payloads in an S3 bucket to comply with regulations. Then we stored the corresponding S3 object keys in database tables, effectively keeping sensitive information out of the database.

There are 4 methods of encrypting objects in S3:

- **SSE-S3:** Encrypts S3 objects using keys handled & managed by AWS. Out-of-box feature. Nothing to do on your side. All objects stored in S3 are encrypted. Therefore, your app complies with regulations without any effort from you.
- **SSE-KMS:** AWS Key Management Service (KMS) is a designated service for encryption. You can opt out of the default S3 encryption and utilize encryption features provided by KMS in your objects in S3.
- **SSE-C:** In a highly regulated environment, you must maintain your own encryption key. S3 supports that with SSE-C.

Client-Side Encryption: Encrypt files before storing them in S3 in your client app.

S3 Pricing

I had been using Google Drive for storing my images and videos for 7+ years. I paid \$20 yearly to get 100 GB storage. It had been red for days. Then, I downloaded all my media and uploaded them to Amazon S3.

Since I did not access my files frequently, I picked the archival class (Glacier Instant Retrieval) which costs \$0.004 per GB per month. So, the yearly cost to store my memories reduced from \$20 to \$4.8 which is a 75% cost reduction.

Despite the numerous impressive features I wrote in this blog, S3 remains one of the pricier services. I know a case where a company switched their file storage service to use S3 instead of on-premise servers. Then the cost was overwhelming and they switched back to the on-premise servers. S3 cost can be bearable if you configure the storage classes correctly. Storage price is much cheaper when using an IA (Infrequently Accessed) storage class. But if the object is retrieved frequently, it costs more.



Pricing varies by region like all other services. There are many factors it charges for:

- Storage — How much data you stored
- Requests & data retrieval — Number of requests against the bucket
- Data transfer — It is a hidden cost. It is like a tax when running apps in the cloud.
- Management & Analytics (if you enable)
- Global Replication (if you enable)

There are Requester Pays buckets, the requester pays the cost of the request and the data downloaded from the bucket. The bucket owner always pays the cost of storing data.

The API call to delete an object is free of charge, but obtaining the object prior to deletion incurs costs. Retrieving 1 million objects, for instance, can amount to approximately \$5 depending on the region. Deleting or replicating 1 million objects may take around 30 minutes. These figures are sourced from another medium blog, emphasizing the importance of vigilance and monitoring of S3 and cloud costs. It's crucial to be mindful of various factors that contribute to charges.

Additionally, it's worth noting that the exact timing of object deletion is uncertain, as it's handled asynchronously by AWS. It's possible that deletions occur once daily, possibly during nighttime, but this remains unknown.

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive**
Designed for durability	99.999999999% (11 9's)					
Designed for availability	99.99%	99.9%	99.9%	99.5%	N/A	N/A
Availability SLA	99.9%	99%	99%	99%	N/A	N/A
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours

IAM Programmatic access and AWS CLI

IAM Programmatic access

In order to access your AWS account from a terminal or system, you can use AWS Access keys and AWS Secret Access keys.

AWS CLI

The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.



The AWS CLI v2 offers several new features including improved installers, new configuration options such as AWS IAM Identity Center (successor to AWS SSO), and various interactive features.

Task-01

Create AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY from AWS Console.

- Log in to your AWS Management Console.
- Click on your username in the top right corner of the console and select “Security Credentials” from the drop-down menu.

The screenshot shows the AWS Management Console with the IAM service selected. A modal window titled "Introducing the new Security Credentials experience" is displayed, stating: "We've redesigned the Security Credentials experience to make it easier to use. Let us know what you think." It includes a warning message: "MFA not activated for root user" with a note: "The root user for this account does not have multi-factor authentication (MFA) activated. Activate MFA to improve security for this account." There is a "Assign MFA" button. Below the modal, the "Account details" section shows the account name as "Sayali Shewale", email address as "sayali.shewale.sx@gmail.com", and AWS account ID as "683613011377".

Click on the “Access keys (access key ID and secret access key)” section.

The screenshot shows the IAM service page with the "Access keys" section open. A message at the top states: "Use access keys to make programmatic calls to APIs from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive)." Below this is a "Create access key" button. The main table header includes columns for "Access key ID", "Created on", "Access key last used", "Region last used", "Service last used", and "Status". A note below the table says: "As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more." A "Create access key" button is also present here. The bottom section is for "CloudFront key pairs" with a "Create CloudFront key pair" button.



Click on “Create Access Key.”

The screenshot shows the AWS IAM 'Create access key' interface. Step 1: Alternatives to root user access keys. A warning message states: "Root user access keys are not recommended. We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice." Below it says: "Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. Learn More." A checkbox at the bottom left says: "I understand creating a root access key is not a best practice, but I still want to create one." A large orange 'Create access key' button is at the bottom right.

Your access key ID and secret access key will be displayed. Make sure to download the CSV file with your access key information and store it in a secure location.

The screenshot shows the AWS IAM 'Retrieve access key' interface. Step 2: Retrieve access key. It displays the generated Access key (AKIAZKRS1SKYHEPFP2FV) and Secret access key (XXXXXXXXXXXXXX). Below it is a section titled 'Access key best practices' with the following bullet points: "Never store your access key in plain text, in a code repository, or in code.", "Disable or delete access key when no longer needed.", "Enable least privilege permissions.", "Rotate access keys regularly." A 'Download .csv file' button is at the bottom left, and an orange 'Done' button is at the bottom right.

The screenshot shows the AWS IAM 'Identity and Access Management (IAM)' dashboard. It includes sections for 'Access keys' (with one listed: AKIAZKRS1SKYHEPFP2FV) and 'CloudFront key pairs' (with none listed). The sidebar contains links for User management, Roles, Policies, Identity providers, Account settings, Access reports, Access analyzer, Archive rules, Analyses, and Settings.

Task-02

Setup and install AWS CLI and configure your account credentials

Install the AWS CLI by following the instructions for your operating system: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html>



AWS Documentation > AWS Command Line Interface > User Guide for Version 2

AWS Command Line Interface

User Guide for Version 2

Windows

Install and update requirements

- We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.
- Admin rights to install software

Install or update the AWS CLI

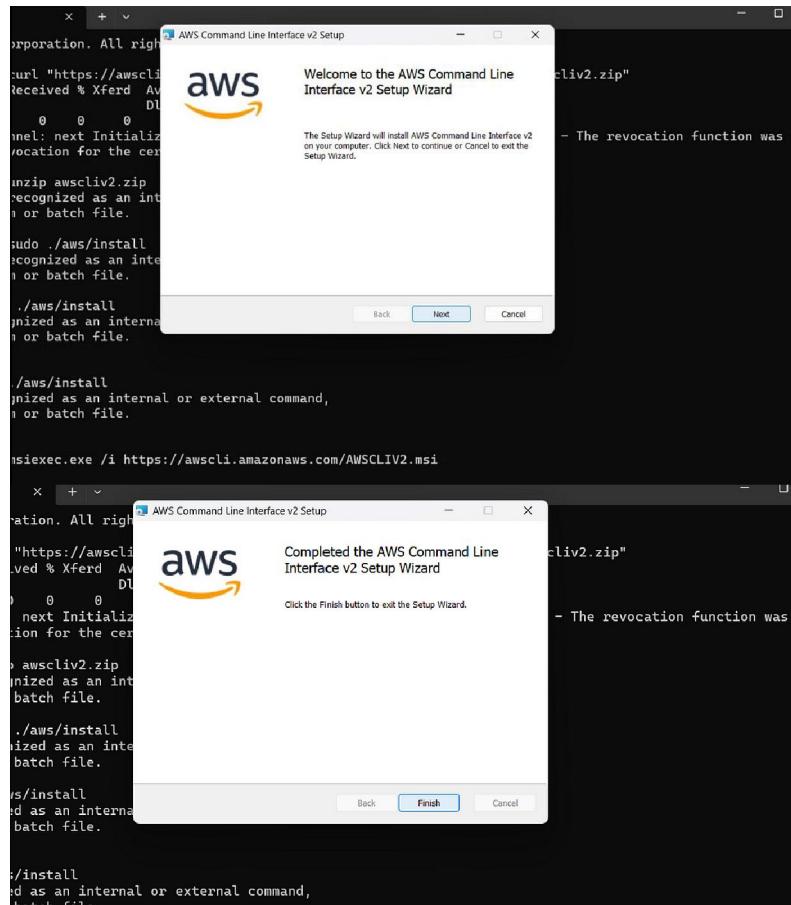
To update your current installation of AWS CLI on Windows, download a new installer each time you update to overwrite previous versions. AWS CLI is updated regularly. To see when the latest version was released, see the AWS CLI version 2 Changelog on GitHub.

- Download and run the AWS CLI MSI installer for Windows (64-bit):
<https://awscli.amazonaws.com/AWSCLIV2.msi>
Alternatively, you can run the msieexec command to run the MSI installer.

```
C:\> msieexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

For various parameters that can be used with msieexec, see [msieexec](#) on the Microsoft Docs website. For example, you can use the /qn flag for a silent installation.

```
C:\Users\sayal>
C:\Users\sayal>msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```





Check aws-cli version

```
C:\Users\sayal>aws --version
aws-cli/2.10.1 Python/3.9.11 Windows/10 exe/AMD64 prompt/off
```

Once you have installed the AWS CLI, open a terminal or command prompt and run the following command to configure your account credentials:

You will be prompted to enter your AWS Access Key ID and Secret Access Key. Copy and paste access key and secret key from downloaded csv file. You will also be prompted to enter your default region and output format. Choose the region that is closest to your location and select a suitable output format.

```
C:\Users\sayal>aws configure
AWS Access Key ID [*****6MVP]: AKIAZ6K5T5KYXE7PF2FY
AWS Secret Access Key [*****Lro+]: sLOejYCKVEcjctS6a8HTD7MalX5m+zIe5Twx7x0Y
Default region name [ap-south-1]:
Default output format [json]:
```

Once you have entered your credentials and configured your default settings, you can test that the CLI is working by running the following command:

```
C:\Users\sayal>
C:\Users\sayal>aws s3 ls
2023-02-18 09:33:55 myawsdevopsdemobucket
```

This command should list the contents of your default S3 bucket. You have now set up and installed the AWS CLI and configured your account credentials.

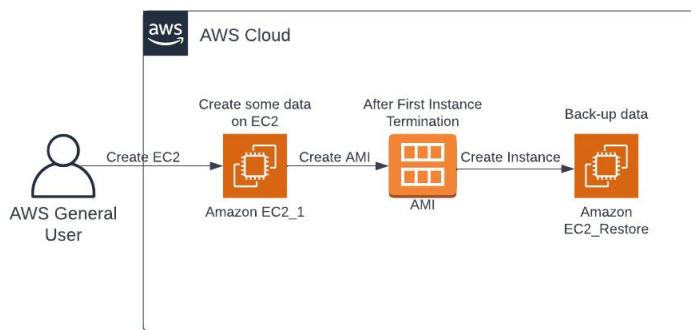
Data Preservation Strategies for EC2 Instances: Safeguarding Your Information Before Destruction

We will explore how to preserve data in our EC2 instance before it gets destroyed.

Use Case :

Imagine you are responsible for an e-commerce website hosted on an EC2 instance. You decide to upgrade your application to a newer version. Before initiating the upgrade, you need to preserve customer data, transaction records, and other critical information stored on the instance to ensure a seamless transition and avoid any data loss.

Overview :



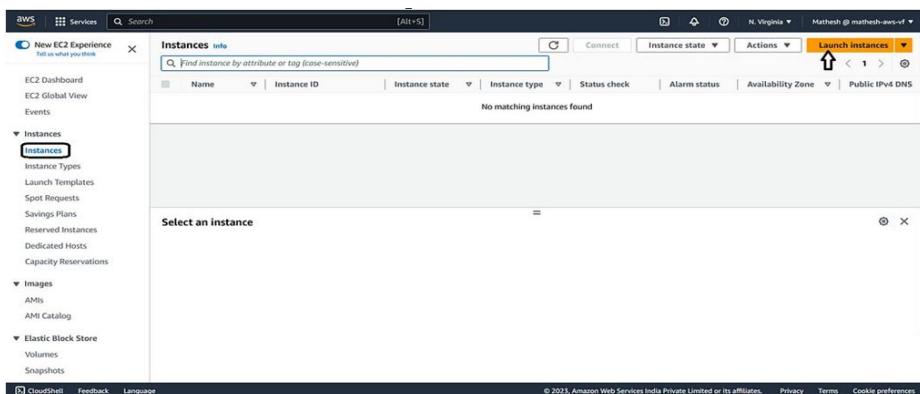
Workflow Diagram

Steps :

Step 1:

Creating the EC2 Instance :

- Navigate to the EC2 console.
- Follow the Outlines steps below.



Launch an instance

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.

Name and tags

Name: Instance-A

Application and OS Images (Amazon Machine Image)

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

Recent AMIs: Amazon Linux (AMI), macOS, Ubuntu, Windows, Red Hat, SUSE Linux Enterprise Server

Virtual server type (instance type)

t2.micro

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month. 20 GiB of EBS

Launch instance

You can Choose Your Preferred AMIs

Browse more AMIs Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI ami-01c647eace872fc02 (64-bit (x86)) / ami-01c647eace872fc03 (64-bit (Arm)) Virtualisation: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 AMI 2023.1.20230906.1 x86_64 HVM kernel-6.1

Architecture 64-bit (x86) **AMI ID** ami-01c647eace872fc02 **Verified provider**

Free tier eligible

Virtual server type (instance type)

t2.micro

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month. 20 GiB of EBS

Launch instance

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: 0.0116 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour Additional costs apply for AMIs with pre-installed software

All generations

Compare instance types

Key pair (login)

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 - required: aws demo

You can choose your key pair Create new key pair

Network settings

Edit

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.1.2... read more ami-01c647eace872fc02

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month. 20 GiB of EBS

Launch instance

AWS Services Search [Alt+S] N. Virginia Mathesh @ mathesh-aws-vf

New EC2 Experience Tell us what you think

EC2 Dashboard EC2 Global View Events

Instances Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations

Images AMIs AMI Catalog

Elastic Block Store Volumes Snapshots

CloudShell Feedback Language

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance: i-01e75c9a115749622 (Instance-A)

Details Security Networking Storage Status checks Monitoring Tags

Instance ID: i-01e75c9a115749622 (Instance-A) Public IPv4 address: 174.129.127.8 | open address Private IPv4 address: [REDACTED]

IPv4 address: - Instance state: Running Public IPv4 DNS: ec2-174-129-127-8.compute-1.amazonaws.com | open address

Hostname type: IP name: ip-172-31-47-122.ec2.internal Private IP DNS name (IPv4 only): ip-172-31-47-122.ec2.internal

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S] N. Virginia Mathesh @ mathesh-aws-vf

Connect to instance info

Connect to your instance i-01e75c9a115749622 (Instance-A) using any of these options

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID: i-01e75c9a115749622 (Instance-A)

Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address: 174.129.127.8

User name: Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.
ec2-user

Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel Connect

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Now, I am going to insert some Data in the Instance

aws Services Search [Alt+S] N. Virginia Mathesh @ mathesh-aws-vf

Amazon Linux 2023 https://aws.amazon.com/linux/amazon-linux-2023

```
[ec2-user@ip-172-31-47-122 ~]$ cat > MyFile.txt
Hello Guys
Iam Mathesh
[ec2-user@ip-172-31-47-122 ~]$ ls
MyFile.txt
[ec2-user@ip-172-31-47-122 ~]$ cat MyFile.txt
Hello Guys
Iam Mathesh
[ec2-user@ip-172-31-47-122 ~]$
```

i-01e75c9a115749622 (Instance-A)

PublicIPs: 174.129.127.8 PrivateIPs: [REDACTED]

Create any file using cat command and use ls command to check whether the file is created

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences



Step 2 :

Creating AMI from Instance :

Now, I am going to create AMI from our Instance.

Steps :

Instance (1/1) Info

Actions ▾ **Launch instances ▾**

- Connect
- View details
- Manage instance state
- Instance settings
- Networking
- Security
- Image and templates**
- Monitor and troubleshoot

Create image

Create template from instance

Launch more like this

Instance: i-01e75c9a115749622 (Instance-A)

Details **Security** **Networking** **Storage** **Status checks** **Monitoring** **Tags**

Instance summary

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Private IPv4 addresses

Public IPv4 address 174.129.127.8 | [open address](#)

Public IPv6 DNS ec2-174-129-127-8.compute-1.amazonaws.com | [open address](#)

Instance ID i-01e75c9a115749622 (Instance-A)

IP address -

Hostname type IP example: ip-172-31-47-172.ec2.internal

Private IP DNS name (IPv6 only) ip-172-31-47-172.ec2.internal

Create image

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Image name backup-image

Image description - optional For Backup Instance

No reboot **Enable**

Instance volumes

Storage type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/...	Create new snapshot fr...	8	EBS General Purpose S...	3000		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

Add volume

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional

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.

Tag image and snapshots together Tag the image and the snapshots with the same tag.

Tag image and snapshots separately Tag the image and the snapshots with different tags.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Create image

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences



Step 3 :

Terminating the Instance :

Now , I am going to Terminate the Instance and after that I am going to create a new instance to retrieve data.

Steps :

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Actions
Instance-B	i-0fb8e849d01120e5	Running	t2.micro	2/2 checks	Stop instance Start instance Reboot instance Terminate instance
Instance-A	i-01e75c9a115749622	Running	t2.micro	2/2 checks	Terminate instance

Instance: i-01e75c9a115749622 (Instance-A)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary

Instance ID: i-01e75c9a115749622 (Instance-A)	Public IPv4 address: 174.129.127.8 [open address]	Private IPv4 addresses:
IPv6 address: -	Instance state: Running	Public IPv4 DNS: ec2-174-129-127-8.compute-1.amazonaws.com [open address]
Hostname type: IP name: in 177.31.47.127.ec2.internal	Private IP DNS name (IPv4 only): in 177.31.47.127.ec2.internal	

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Instance-B	i-0fb8e849d01120e5	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-156-89-50
Instance-A	i-01e75c9a115749622	Terminated	t2.micro	-	No alarms	us-east-1b	-

Select an instance

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Step 4 :

Banking-up and Restore the Data :

Now, I am going to Create a new instance to retrieve data.



Steps :

Amazon Machine Images (AMIs) (1/1) Info

Owned by me | Find AMI by attribute or tag

Name	AMI ID	AMI name	Source	Owner	Visibility
ami-040d1ad7838d21593	backup-image	804937851364/backup-image	804937851364	Mathesh @ mathesh-aws-vf	Private

Wait for 5 mins for the AMI status to be Available

AMI ID: ami-040d1ad7838d21593

Details | Permissions | Storage | Tags

AMI ID ami-040d1ad7838d21593	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name backup-image	Owner account ID 804937851364	Architecture x86_64	Usage operation RunInstances
Root device name /dev/xvda	Status Available	Source 804937851364/backup-image	Virtualization type hvm
Volumes	State reason	Creation date Sun Sep 10 2023 19:55:10 GMT+0530	Kernel ID -
Snapshots	Boot mode uefi-preferred		

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Launch an instance

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.

Name and tags

Name
Instance-A-Backup

Application and OS Images (Amazon Machine Image)

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 for your full catalog including 1000s of application and OS images

AMI from catalog | Recents | My AMIs | Quick Start

Amazon Machine Image (AMI)
backup-image
ami-040d1ad7838d21593

Browse more AMIs

© Free tier in your first year includes 750 hours of t2.micro or t3.micro in the Regions in which t2.micro is unavailable instance usage on free tier AMI use monthly 25.000 of ECR

Cancel | Launch Instance | Review commands

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Application and OS Images (Amazon Machine Image)

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 for your full catalog including 1000s of application and OS images

AMI from catalog | Recents | My AMIs | Quick Start

Amazon Machine Image (AMI)
backup-image
ami-040d1ad7838d21593

Here we can see that it automatically choose AMI from catalog

Browse more AMIs

Published | Architecture | Virtualization | Root device type | ENA Enabled

2023-09-10T14: x86_64 | hvm | ebs | Yes

© Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMI use monthly 25.000 of ECR

Cancel | Launch Instance | Review commands

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences



AWS Services Search [Alt+S] N. Virginia Mathesh @ mathesh-aws-vf

Instance type info

Instance type t2.micro Family: t2 1 vCPU 1 GiB Memory Current generation: true Free tier eligible On-Demand Windows base pricing: 0.0162 USD per Hour On-Demand Linux base pricing: 0.0162 USD per Hour On-Demand RHEL base pricing: 0.0176 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour Additional costs apply for AMIs with pre-installed software

All generations Compare instance types

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 - required aws demo Create new key pair

Network settings info

Summary

Number of instances info 1

Software Image (AMI) For Backup Instance ami-060d7ad7386d1595

Virtual server type (instance type) t2.micro

Firewall (security group) New security group

Storage (volumes) 1 volume(s) - 8 GiB

Leave the other settings as it is and Launch the Instance

Launch Instance

Cancel Review commands

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

New EC2 Experience Tell us what you think

EC2 Dashboard

EC2 Global View

Events

Instances Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images AMIs

AM Catalog

Elastic Block Store Volumes

Snapshots

CloudShell Feedback Language

Services Search [Alt+S] N. Virginia Mathesh @ mathesh-aws-vf

Instances (1/2) info Connect Instance state Actions Launch Instances

Find instance by attribute or tag (case-sensitive)

Instance state running Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Instance-B	i-0fbdb494d01120e5	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-156-89-50
Instance-A-Backup	i-075f0ccf3dc041401	Running	t2.micro	Initializing	No alarms	us-east-1b	ec2-3-88-136-55-0

Instance: i-075f0ccf3dc041401 (Instance-A-Backup)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-075f0ccf3dc041401 (Instance-A-Backup)	3.88.156.55 open address	172.31.45.219

IPv6 address Instance state Running

Hostname type Public IP DNS name (IPv4 only)

IP name is 172.31.45.219.ec2.internal

Private IP address 172.31.45.219.ec2.internal

Public IPv4 DNS ec2-3-88-136-55.compute-1.amazonaws.com | open address

CloudShell Feedback Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Connect to instance info

Connect to your instance i-075f0ccf3dc041401 (Instance-A-Backup) using any of these options

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID i-075f0ccf3dc041401 (Instance-A-Backup)

Connection type

Connect using EC2 Instance Connect Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

Connect using EC2 Instance Connect Endpoint Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address 3.88.156.55

User name Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, root.

ec2-user May be your user name will be root, change it to ec2-user

Note: In most cases, the default user name, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel Connect

CloudShell Feedback Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

The screenshot shows a terminal window in AWS CloudShell with the following content:

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

last login: Sun Sep 10 14:13:20 2023 from 10.206.107.27
[ec2-user@ip-172-31-45-219 ~]$ ls
MyFile.txt
[ec2-user@ip-172-31-45-219 ~]$ 

```

Below the terminal, a message box says: "Now, we can see that our data is backed-up".

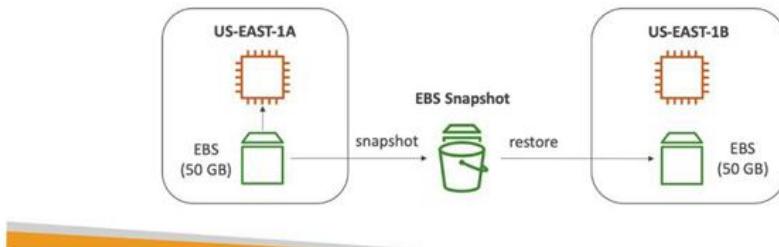
We successfully backed-up and restored the Old Data.

AWS EBS Snapshots

- **EBS** is the network storage drive and can be connected with one EC2 instance at a time.
EBS cannot be connected from one availability zone to another.
- **Snapshots** are the backups of the EBS instance and can be restored in any other availability zone as a copy (with the same data).

EBS Snapshots

- Make a backup (snapshot) of your EBS volume at a point in time
- Not necessary to detach volume to do snapshot, but recommended
- Can copy snapshots across AZ or Region



NOTE: If we want to connect EBS storage from one availability zone to another directly. It will not happen, rather we take screenshots of the EBS and recreate another EBS in different EBS

In this article, we will see below three ideas:

- Way to create EBS snapshots
- Move to some other availability zones



c. Way to use snapshot and restore a new drive

A. Way to create EBS snapshots:

- Search EC2 instance:

The screenshot shows the AWS Services search results for 'EC2'. The 'EC2' service is highlighted with a red circle. Other services listed include EC2 Image Builder and Marketplace.

- Go to the Storage tab of the EC2 instance → Click the volume id

The screenshot shows the AWS EC2 Instances page with two instances listed. The 'Storage' tab is selected for the first instance, 'My first instance'. It shows two EBS volumes: one attached at /dev/vda (Volume ID: vol-08020c496a36cb6d6) and another at /dev/efd (Volume ID: vol-0194b1da6e9e5e976). A red arrow points to the Volume ID of the second volume.

Here we will see a list of EBS storage. Right Click on the storage whose backup/snapshot to create → Create Snapshot

The screenshot shows the AWS Create Volume page for a new EBS volume. A context menu is open over the volume 'vol-0194b1da6e9e5e976'. The 'Create Snapshot' option is highlighted with a red circle. Other options in the menu include Modify Volume, Create Snapshot Lifecycle Policy, Delete Volume, Detach Volume, Force Detach Volume, Change Auto-Enable IO Setting, and Add/Edit Tags.



Write the description of the snapshot you are making. Click on the Create Snapshot button

The screenshot shows the 'Create Snapshot' interface for an EBS volume. The volume ID is 'vol-0194b1da5e0e5e976'. The 'Description' field contains 'Snapshot'. The 'Encrypted' option is set to 'Not Encrypted'. Below the fields, there is a section for tags with a note: 'This resource currently has no tags' and a link to 'Add Tag'. At the bottom right, there is a 'Create Snapshot' button, which is circled in red.

The screenshot shows the same 'Create Snapshot' interface after the process is completed. A green box displays the message 'Create Snapshot Request Succeeded' with the snapshot ID 'snap-007e1c799d8890450'. At the bottom right, there is a 'Close' button, which is circled in red.

A snapshot is created. Click on the snapshot link on left. You will find your snapshot there.

Image of the EBS drive is ready (i.e. backup ready)

The screenshot shows the AWS EC2 Dashboard. On the left, there is a navigation menu with sections like 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances', 'Images', and 'Elastic Block Store'. Under 'Elastic Block Store', the 'Snapshots' link is highlighted with a red circle. The main content area shows a table of snapshots, with one entry visible: 'snap-007e1c799d8890450', '2 GiB', and 'Snapshot'. Below the table, there is a detailed view for the selected snapshot, showing its description, permissions, and tags. The progress bar indicates 100% completion.

B. Move to some other availability zones:

- Regular EBS storage cannot be connected to other availability zones or regions. This is possible with the help of a snapshot.
- So, for connecting it to different EC2 instances, we can now convert the EBS snapshot to a different region or availability zone.

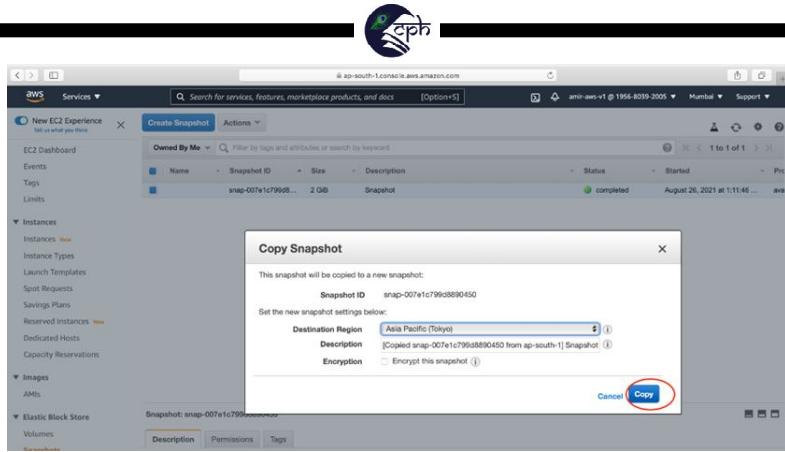
NOTE: One region can have many availability zones

Right-click on the snapshot → Copy

The screenshot shows the AWS EC2 console with the 'New EC2 Experience' selected. On the left, there's a sidebar with 'Instances' and 'Elastic Block Store' sections. In the main area, a snapshot named 'Snapshot' is listed with a status of 'completed'. A context menu is open over this snapshot, with the 'Copy' option highlighted by a red circle. Other options in the menu include 'Delete', 'Create Volume', 'Manage Fast Snapshot Restore', 'Create Image', 'Modify Permissions', and 'Add/Edit Tags'.

Choose your region region and click **Copy** button

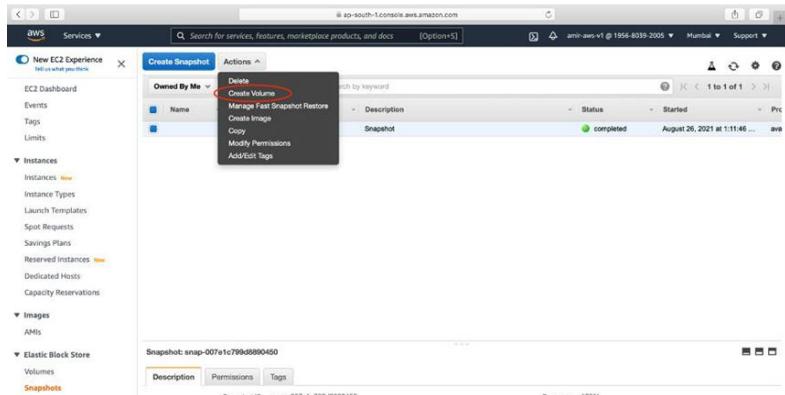
The screenshot shows the 'Copy Snapshot' dialog box. It displays the message: 'This snapshot will be copied to a new snapshot. Snapshots can be connected to any new region. This was not possible in direct EBS instance connection.' Below this, it says 'Set the new snapshot settings below'. The 'Destination Region' dropdown is set to 'Asia Pacific (Tokyo)', which is highlighted with a red arrow. Other options in the dropdown include 'Asia Pacific (Osaka)', 'Asia Pacific (Mumbai)', 'Asia Pacific (Singapore)', 'Asia Pacific (Sydney)', 'Europe (Frankfurt)', 'Europe (Paris)', 'Europe (Stockholm)', 'Europe (Ireland)', 'Europe (London)', 'South America (Sao Paulo)', 'US East (N. Virginia)', 'US East (Ohio)', 'US West (N. California)', and 'US West (Oregon)'. At the bottom right of the dialog box is a 'Copy' button.



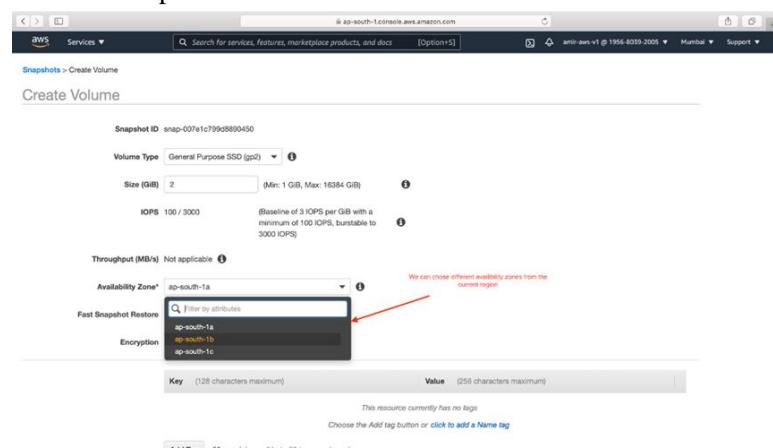
A new copy of the snapshot will be created with your new region.

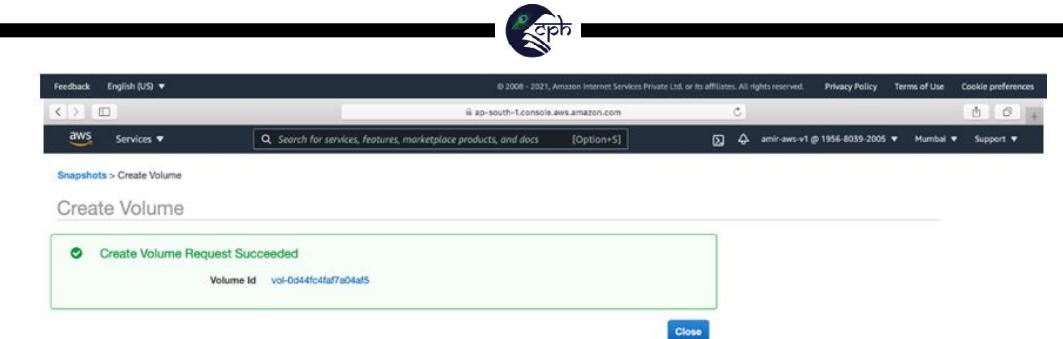
C. Restore a new EBS drive from Snapshot:

The snapshot creates an EBS drive exactly the copy of the backed-up drive.



You can change the availability zone of the drive from the below selection, the rest of the details will be the same as previous. Click Create Volume.





Now, when we go to volumes on left, we see a new EBS storage volume is set up.

Name	Volume ID	Size	Volume Type	IOPS	Throughput	Snapshot	Created	Availability Zone	State
vol-0d44fc4fa7a04af5	2 GB	gp2	100	-	snap-007e1c7...	August 26, 2021 at ...	ap-south-1a	available	
New EBS Ac...	vol-0154bd1...	2 GB	gp2	100	-		August 25, 2021 at ...	ap-south-1a	in-use
My first Insta...	vol-0db0b04...	8 GB	gp2	100	-	snap-0bdcc50...	August 17, 2021 at ...	ap-south-1a	in-use
Ec2Instance-1	vol-0bc22e7...	8 GB	gp2	100	-	snap-0d3528...	April 12, 2020 at 6:0...	ap-south-1a	in-use

Closing thoughts:

- In this article, we have understood how EBS snapshots work in AWS and to restore the EBS volume drive from Snapshots.
- Keeping a backup image of the drive is always safe. Additionally, EBS can be used to move in another region as well (the copy of the storage drive)

Elastic Beanstalk: Advantages and Drawbacks

First, let's start with the basics: According to the [AWS](#) site, “Elastic Beanstalk makes it even easier for developers to quickly deploy and manage applications in the AWS cloud. Developers simply upload their application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.”

Advantages:

Elastic Beanstalk’s main benefits include timesaving server configuration, powerful customization, and a cost-effective price point.



Lightning-Fast Configuration with Automation

Elastic Beanstalk automates the setup, configuration, and provisioning of other AWS services like EC2, RDS, and Elastic Load Balancing to create a web service. You can log into your AWS management console, and have a new site up and accessible in less than an hour. Elastic Beanstalk also creates a fairly standard configuration for a modern Rails application.

This automation can save precious time by handling all the things that need to be completed for a production app (configuring log file rotations, nginx config files, puma service configuration, Linux package installation, ruby installation, load balancer configuration, and database setup). There are more services like Heroku, Engine Yard, and others that do this as well, but in general, we found Elastic Beanstalk was on par and had a good standard setup, especially when you factor in its price. These are some of the specifics on the configuration:

- **Support:** AWS Elastic Beanstalk supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications.
- **Rails Servers:** For Rails, you can run either a Passenger or a Puma application stack - we've only used Puma so far, and the servers will be configured by Elastic Beanstalk to run a Puma process (with Nginx used as a reverse proxy in front of it) and a reasonable server configuration for log files, directory security, and user security.
- **SQL server:** This is configured through Amazon RDS, but at its heart is a EC2 server with a database running on it. We've used both postgres and Mysql.
- **AWS Elastic Load Balancer:** Running and with the correct configuration for the Rails servers.
- **Security:** New AWS security groups and policies are created, along with permissions for these services to securely talk to each other. All the servers are configured so they can only talk to and have permissions for what they need. For example, your Rails servers have just one port open specifically for the load balancers, and nothing can talk to your DB server, except for your Rails servers. This is fantastic, because it can be hard to do correctly on your own.
- **Default Configuration:** ENV variables on the Rails servers that securely set secrets needed for Rails to run — like Database endpoint, username, and password. This is also helpful, because figuring out a secure way to do this on your own can be a pain and is easy to get wrong.
- **Custom Configuration:** An easy and secure way (either in the Elastic Beanstalk UI or through the AWS command line tools) to set custom ENV variables. For example, you can set your Mailchimp account and password and have it accessible to your running Rails code as an ENV variable.
- **Monitoring:** Basic monitoring of your servers through Cloudwatch.
- **Deployment:** Easy deployment of new versions through AWS CLI. Once configured, you run 'eb deploy' from the root of your git repository, and the deploy just works. This was also easy to integrate with Codeship, a Continuous Integration service that we used.



Elastic Beanstalk's automated configuration helps avoid mistakes that happen from missing small details when you try to DIY. These are great boilerplate specs that you typically look for when using a service like this because they can be pretty tricky to get right on your own.

Powerful Customization

With Elastic Beanstalk, it's all under your control. Everything that is created is just an AWS service - so you can look at EC2, see the new instances, and ssh into them. You can update your database config file. You can update the security group for all of the machines, so that, for example, the entire application is only accessible from your office IP address.

While at some level this is a similar service to Heroku, it also gives you more low-level access and control. Customization is more complicated, but much more flexible and powerful than using Heroku. For example, when we wanted to add sidekiq, it wasn't straightforward (but possible) - that's another post for another time.

Price and Flexibility

Elastic Beanstalk's price and flexibility are great. The platform itself is nothing, so there is no extra charge on top of the AWS services you're using. Also, because you can pick your instance size, as well as easily add more front-end servers to the load balancer, you can easily match

your server needs to your service load. Elastic Beanstalk even has auto-scaling functionality built in, which we never used, but would be a great way to save money for larger applications by only bringing up extra servers when needed. Overall, our costs on a application with a similar size and scope we built during the same timeframe was 400% higher on Heroku vs. Elastic Beanstalk. Although each application is different, it's a good ballpark comparison.

Drawbacks

Some of the biggest pains with Elastic Beanstalk include unreliable deployments, lack of transparency and documentation around stack and application upgrades, and an overall lack of clear documentation.

Unreliable Deployment

We do a lot of deployments — we have a continuous integration setup via Codeship, so with every commit (or merged pull request), if tests pass, we deploy a new version. We practice small, incremental changes and strive to be as nimble as possible. Some days, we might deploy 10 times.

Over the last year or so of using Elastic Beanstalk, our deploys have failed five or six times. When the failure happens, we get no indication why, and further deployments will fail as well. On the positive side, this didn't result in downtime for us. We simply couldn't deploy, and if we tried again it would fail.

Each time, we needed to troubleshoot and fix on our own. We found and tried multiple solutions, such as terminating the instance that had the deployment issue, and let Elastic Beanstalk recover. Sometimes, we could ssh into the stuck machine, kill a process that was part of the eb deploy, the machine would recover. But overall, we didn't know what failed, and it's never a good thing to not be sure that your machine is in a good state.



Considering we have done over 1000 deployments; this isn't a high failure rate. It never hit us at a critical time, but what if this happened when we were trying to do a hotfix for a performance issue that was crippling our site? Or, what if we had larger sites with more than two or three front-end servers? Would this increase our deployment failure rate? For the two applications we have done, we decided that the risk of this happening was small and that it didn't warrant switching to a new service. For some applications this would not be an acceptable risk.

Deployment Speed

Deployments would take five minutes at least, and sometimes stretch to 15, for a site with just two front-ends. With more servers, deployments could take even longer. This might not seem like much, but we have setup other Rails environments where deployment can be done in one or two minutes. And this can be critical if you are trying to be responsive in real-time.

Attempts have been made to improve the Elastic Beanstalk deployment process, and a good summary to start with is this [post](#) from HE:labs. We may try some things from there in the future.

Stack Upgrades

Elastic Beanstalk comes out with new stack versions all the time — but we have zero information on what has changed. No release notes, no blog post, not even a forum post. Sometimes, it's obvious — the version of Ruby or Puma will change. But other times, it's just an upgrade.

We've done several upgrades, and sometimes it goes smoothly, and sometimes it takes a week.

Old Application Versions

Another thing we learned is to occasionally delete old application versions. With every deploy, Elastic Beanstalk archives the old application version in an S3 bucket. However, if there are 500 old versions, further deploys fail. You can delete them through the Elastic

Beanstalk UI, but this caught us off guard multiple times. Although this seems like a small problem, I really don't like it when deployments fail.

All of these problems are an indication of Elastic Beanstalk's general lack of transparency. We had to figure out a lot of these issues on our own and through blog posts and internet searches. On the plus side, we had complete transparency into EC2 instances, database, and other services created, so we were free to learn on our own. And while stack upgrades and failed deployments are the most clear moments of pain, in general they were indicators of the types of things that you have to learn on your own.

Summary

Elastic Beanstalk helps us to easily deploy updates to our Rails application while also leveraging Amazon's powerful infrastructure. Enhancing our deployment process with containers — like Docker — will add even more versatility. Thanks to the fine-grain control offered by Elastic Beanstalk we get to choose technologies that work best for us. Ultimately, we found the most helpful thing about Elastic Beanstalk to be that its automation features let us easily deploy updates to our Rails application. While it's certainly not a perfect tool, if you're looking to reduce system operations and just focus on what you're developing, Elastic Beanstalk is a solid choice.



Adding a custom domain for the AWS Elastic Beanstalk application using Route 53.



The main objective of this article is to deploy a simple web application to the AWS cloud platform. Our plan is to deploy the application to Elastic Beanstalk.

Prerequisites to Proceed Further

1. Valid node application

- We will be uploading a Node application to EBS in this demo.
- Other deployment environments are also available with EBS, but this course is explicitly focused on a Node application.

If you don't have a Node application available, you can access the following GitHub project:
<https://github.com/Venn1991/node-typescript-boilerplate-mongoose-Public>

2. Valid AWS account

Of course, you need to have a valid AWS account available

Deploying Applications to Elastic Beanstalk:

Let's first look at the application we need to deploy. We have a simple Node project that renders a sample user page. Given below is the package.json file for the sample project.

```
{  
  "name": "node-typescript-boilerplate",  
  "version": "1.0.0",  
  "description": "",  
  "main": "src/index.ts",  
  "scripts": {  
    "watch": "tsc -w",  
    "dev": "nodemon dist/index.js",  
    "start": "node dist/index.js",  
    "build": "tsc",  
    "build:lib": "tsc"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {
```



```
  "@types/jsonwebtoken": "^8.5.8",
  "aws-sdk": "^2.1167.0",
  "bcrypt": "^5.0.1",
  "chalk": "^4.1.2",
  "dotenv": "^16.0.1",
  "express": "^4.18.1",
  "express-validator": "^6.14.2",
  "joi": "^17.6.0",
  "jsonwebtoken": "^8.5.1",
  "lodash": "^4.17.21",
  "moment": "^2.29.3",
  "mongoose": "^6.4.4",
  "multer": "^1.4.5-lts.1",
  "multer-s3": "^3.0.1",
  "nodemailer": "^6.7.5",
  "ts-node": "^10.8.2"
},
"devDependencies": {
  "@types/aws-sdk": "^2.7.0",
  "@types/bcrypt": "^5.0.0",
  "@types/express": "^4.17.13",
  "@types/lodash": "^4.14.182",
  "@types/multer-s3": "^3.0.0",
  "typescript": "^4.7.4"
}
}
```

}

In the package file specified above, we need to ensure that the scripts tag contains a start script. This script will be used to bootstrap the application on the Elastic Beanstalk server. In the sample code, we have configured the start script as node dist/index.js.

Our app is ready, so let's create the Stretch Bean Establishment app.

3. How to Create the Elastic Beanstalk App Step 1: Configure Your Environment

- The first step is to go to the AWS Management Console and select Elastic Beanstalk from the Services menu. Press the 'Create Application' button.
- Select the Web server environment and give a name to your app.



The screenshot shows the Amazon Elastic Beanstalk landing page. At the top, there's a navigation bar with 'Compute' selected. Below it, the title 'Amazon Elastic Beanstalk' and the subtitle 'End-to-end web application management.' are displayed. A brief description of the service follows, mentioning Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker support. To the right, there's a 'Get started' section with a 'Create application' button, and a 'Pricing' section stating there's no additional charge for Elastic Beanstalk.

Getting started with AWS Elastic Beanstalk

The screenshot shows the 'Configure environment' step of the AWS Elastic Beanstalk setup wizard. On the left, a sidebar lists steps: Step 1 (Configure environment), Step 2 (Configure service access), Step 3 (optional) Set up networking, database, and tags, Step 4 (optional) Configure instance traffic and scaling, Step 5 (optional) Configure updates, monitoring, and logging, and Step 6 (Review). The main area shows the 'Environment tier' section with 'Web server environment' selected. The 'Application information' section has 'sample-node-rds' entered in the 'Application name' field. The 'Environment information' section has 'Sample-node-rds-env' entered in the 'Environment name' field. A green box highlights the 'Application name' input field.

Choose Managed platform in “Platform type”, and Node.js in “Platform”, and leave the rest as it is.

Then choose Upload your code in the “Application code” section and upload the zip file.



Platform Info

Platform type

Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Node.js

Platform branch

Node.js 18 running on 64bit Amazon Linux 2

Platform version

5.8.1 (Recommended)

Application code Info

Sample application

Existing version
Application versions that you have uploaded.

Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

Version label
Unique name for this version of your application code.

Source code origin. Maximum size 2 GB

Local file

Upload application

Then set the version label to 1 and choose Single instance in the “Presets” section and click Next.

Note: Prefer High availability for the production environment.

Application code Info

Sample application

Existing version
Application versions that you have uploaded.

Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

Version label
Unique name for this version of your application code.

Source code origin. Maximum size 2 GB

Local file

Upload application

File must be less than 2GB max file size

Public S3 URL



Presets Info

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

Configuration presets

Single instance (free tier eligible)

Single instance (using spot instance)

High availability

High availability (using spot and on-demand instances)

Custom configuration

Cancel **Next**

Step 2: Configure service access

In this section, it is necessary to set up IAM roles. We must create two IAM roles, one for Elastic Beanstalk and one for EC2

For the service role, select Create and use new service role. It'll automatically create and provide the required permissions

In order to ssh into your EC2 instance via terminal, create a key-value pair and select it. Skip this step if you do not wish to log onto EC2.

Create an IAM role with the following permissions and add the role to the 'EC2 instance profile' and proceed next.

- AWS Elastic Beans talk Web Tier
- AWS Elastic Beans talk Worker Tier
- AWS Elastic Beans talk Multi container Docker

Step 1 Configure environment

Step 2 **Configure service access**

Step 3 - optional Set up networking, database, and tags

Step 4 - optional Configure instance traffic and scaling

Step 5 - optional Configure updates, monitoring, and logging

Step 6 Review

Configure service access Info

Service access

Iam roles assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

Service role

Create and use new service role

Use an existing service role

Service role name

Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to this role.

aws-elasticbeanstalk-service-role

[View permission details](#)

EC2 key pair

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

arunaaws

[View](#) [Delete](#)

EC2 Instance profile

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

sample_nodejs_eb

[View permission details](#)

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



Step 3: Set up networking, database, and tags

I'm going to skip this step because I'm using a Mongoose database, so I don't need to do this step.

Step 4: Configure instance traffic and scaling

It's not necessary to make any changes here unless you need them. If you're creating this sample app, leave the fields with their default values. An Amazon Linux machine will be created by Elastic Beanstalk by default.

The percentage of On-Demand instances as part of any additional capacity that your Auto Scaling group provisions beyond the On-Demand base instances.

0 %

Capacity rebalancing
Specifies whether to enable the capacity rebalancing feature for Spot Instances in your Auto Scaling Group. This option is only relevant when EnableSpot is true in the aws:ec2:instances namespace, and there is at least one Spot Instance in your Auto Scaling group.
 Turn on capacity rebalancing

Architecture
The processor architecture determines the instance types that are made available. You can't change this selection after you create the environment. [Learn more](#)

x86_64
This architecture uses x86 processors and is compatible with most third-party tools and libraries.

arm64 - new
This architecture uses AWS Graviton2 processors. You might have to recompile some third-party tools and libraries.

Instance types
Add instance types for your fleet. Change the order that the instances are in to set the preferred launch order. This only affects On-Demand instances. We recommend you include at least two instance types. [Learn more](#)

Choose x86 Instance types ▾

t3.micro X t3.small X

AMI ID
Elastic Beanstalk selects a default Amazon Machine Image (AMI) for your environment based on the Region, platform version, and processor architecture that you choose. [Learn more](#)

ami-Oaf78314da02722d5

Availability Zones
Number of Availability Zones (AZs) to use.
Any

Placement
Specify Availability Zones (AZs) to use.
Choose Availability Zones (AZs) ▾

Cancel Skip to review Previous Next

Step 5: Configure updates, monitoring, and logging

Choose Basic in "Health reporting" and uncheck Managed updates activation.



Configure updates, monitoring, and logging - optional Info

▼ Monitoring Info

Health reporting

Enhanced health reporting provides free real-time application and operating system monitoring of the instances and other resources in your environment. The **EnvironmentHealth** custom metric is provided free with enhanced health reporting. Additional charges apply for each custom metric. For more information, see [Amazon CloudWatch Pricing](#)

System



Health event streaming to CloudWatch Logs

Configure Elastic Beanstalk to stream environment health events to CloudWatch Logs. You can set the retention up to a maximum of ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming

Activated (standard CloudWatch charges apply.)

Retention

7

Lifecycle

Keep logs after terminating environment

▼ Managed platform updates Info

Activate managed platform updates to apply platform updates automatically during a weekly maintenance window that you choose. Your application stays available during the update process.

Managed updates



Weekly update window

Add your environment variables and click Next.

Instance log streaming to CloudWatch logs

Configure the instances in your environment to stream logs to CloudWatch logs. You can set the retention to up to 10 years and configure Elastic Beanstalk to delete the logs when you terminate your environment. [Learn more](#)

Log streaming
(standard CloudWatch charges apply)

Activated

Retention

7

Lifecycle

Keep logs after terminating env...

Environment properties

The following properties are passed in the application as environment properties. [Learn more](#)

Name	Value	Remove
NODE_ENV	production	Remove
PORT	8090	Remove

Add environment property

Cancel Continue Apply

In the end, examine all your configurations and proceed with the next step.



The screenshot shows the AWS Elastic Beanstalk environment dashboard for 'Sample-node-rds-env'. At the top, a message says 'Elastic Beanstalk is launching your environment. This will take a few minutes.' Below this, the 'Environment overview' section shows 'Health' (Grey) and 'Domain' (empty). To the right, the 'Platform' section shows 'Node.js 18 running on 64bit Amazon Linux 2/5.8.1' and 'Running version' (empty). The 'Events' tab is selected, displaying a table of deployment logs:

Time	Type	Details
May 6, 2023 21:56:54 (UTC+5:30)	INFO	Creating RDS database named: awseb-e-86yangpk5n3-stack-awsebredisdatabase-rgf7v7sawg. This may take a few minutes.
May 6, 2023 21:56:54 (UTC+5:30)	INFO	Created EP: 18.218.155.28
May 6, 2023 21:56:54 (UTC+5:30)	INFO	Created security group named: awseb-e-86yangpk5n3-stack-AWSEBRDSDBSecurityGroup-1DJKMHV2XKCS
May 6, 2023 21:56:59 (UTC+5:30)	INFO	Created security group named: awseb-e-86yangpk5n3-stack-AWSEBSecurityGroup-EKD4ZB8H3K
May 6, 2023 21:56:19 (UTC+5:30)	INFO	Using elasticbeanstalk-us-east-2-376292281996 as Amazon S3 storage bucket for environment data.
May 6, 2023 21:56:18 (UTC+5:30)	INFO	createEnvironment is starting.

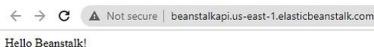
Now you can see why I spent hours on this process in the first place. Whenever I made a mistake, I had to wait about 10 to 15 minutes to check the result and redo all the steps above if anything went wrong. The Elasticated Bean will definitely test your patience, so be calm and relaxed.

When everything is finished, the health will turn green and a domain URL will be generated.

The screenshot shows the AWS Elastic Beanstalk environment dashboard for 'Sample-node-rds-env'. The 'Health' status is now 'Green' and the 'Domain' field contains 'sample-node-rds-env.eba-qhorezkm.us-east-2.elasticbeanstalk.com'. The 'Platform' section remains the same. The 'Events' tab is selected, showing a table of deployment logs:

Time	Type	Details
May 6, 2023 22:46:31 (UTC+5:30)	INFO	Deleted log fragments for this environment.
May 6, 2023 22:46:15 (UTC+5:30)	INFO	Deleted log fragments for this environment.

The following page will appear when you open the URL if you used my example repo.



That's all! We have successfully deployed our application on AWS Elastic Beanstalk. Our Pipeline enables us to make changes to our application continuously and return to old versions when necessary. Don't hesitate to explore the dashboard of your newly deployed application.

Add your domain to Route 53

Connect to Route 53, and you can either buy or register a domain name that has already been acquired from an external provider.



Network & Content Delivery

Amazon Route 53

A reliable way to route users to internet applications

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service.

Get started with Route 53

Get started by registering a domain, configuring DNS, or using another Route 53 feature.

Get started

Pricing (US)

View pricing

How it works

More resources

Documentation

API reference

FAQs

Forum - DNS and health checks

Forum - Domain name registration

Enter your domain name in the “Domain Name” field. The name of the custom domain you want to add (e.g., example.com) should be given here.

Route 53 > Hosted zones > Create hosted zone

Create hosted zone Info

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name Info

This is the name of the domain that you want to route traffic for.

Valid characters: a-z, 0-9, ! * # % & ' { } * *, - / ; < = > ? @ [\] ^ _ ' { } , ~

Description - optional Info

This value lets you distinguish hosted zones that have the same name.

The description can have up to 256 characters. 0/256

Type Info

The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

Public hosted zone
A public hosted zone determines how traffic is routed on the internet.

Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

Tags Info

Apply tags to hosted zones to help organize and identify them.

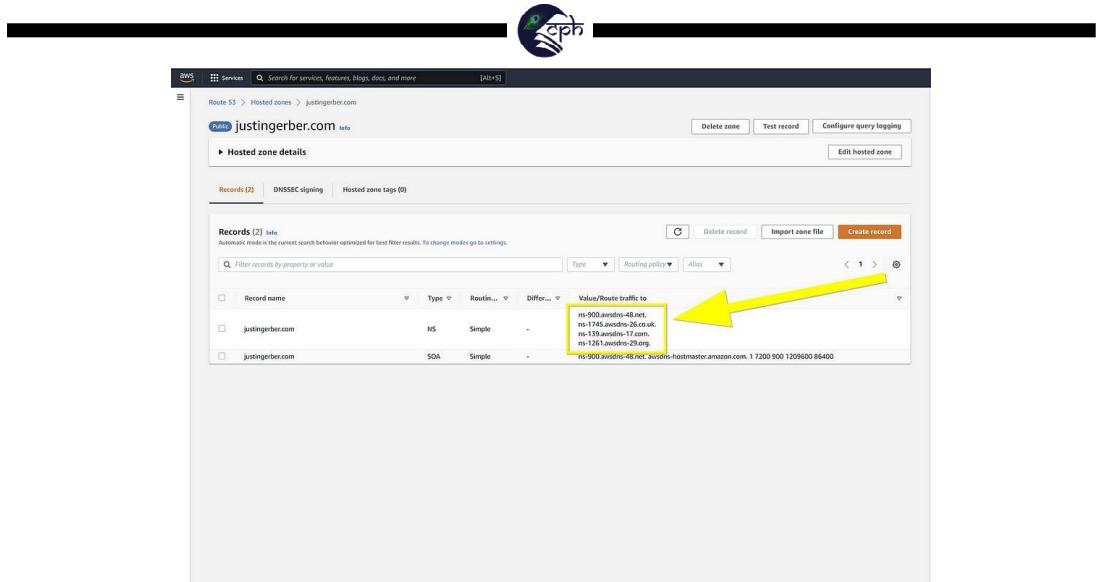
No tags associated with the resource.

Add tag

You can add up to 50 more tags.

Cancel **Create hosted zone**

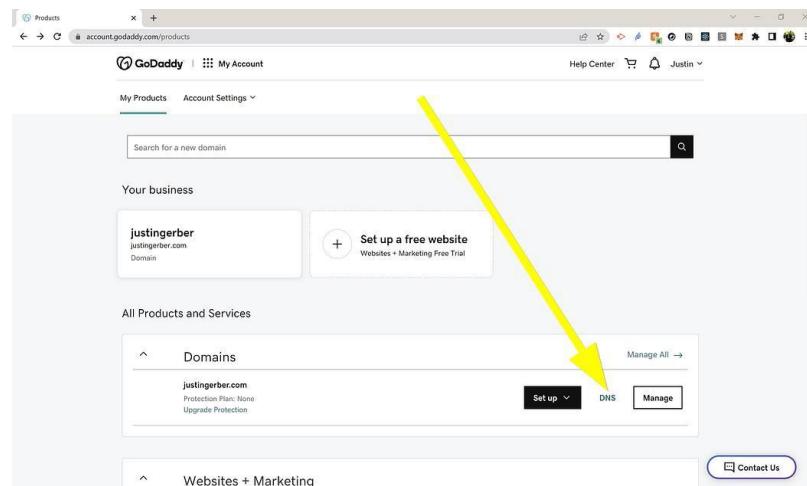
Once the hosted zone is established, you will be directed to the records management page. DNS records need to be added to point to the resources you want to associate with your custom domain. Records (for IPv4 addresses), AAAA records (for IPv6 addresses), CNAME records (for aliases), MX records (for mail servers), etc. are all common record types.



1. Add Name Server(NS) in your domain provider.

To point to the set of nameservers on AWS, custom NS records will be added to GoDaddy next.

Click on My Products on the top toolbar. The DNS button on the justingerber.com domain should be clicked next.



Depending on your requirements, you can choose between GoDaddy's default nameservers or custom nameservers. Here are the two options:

- **Default Nameservers:** If you're using GoDaddy's default nameservers, you will typically see an option to choose "GoDaddy Nameservers" or "Default Nameservers." Select this option if you want to use GoDaddy's own nameservers for your domain.

- **Custom Nameservers:** If you have your own nameservers (provided by a hosting provider, for example), you'll want to choose the option to enter custom nameservers. Enter the nameserver addresses provided by your hosting provider.

The screenshot shows the GoDaddy DNS Management interface for the domain `justingerber.com`. The page title is "DNS Management". Below it, the sub-page title is "My Domains / Domain Settings". The main content area is titled "DNS Records" with the sub-instruction: "DNS Records define how your domain behaves, like showing your website content and delivering your email." Below this, there is a table with columns: Type, Name, Data, TTL, Delete, and Edit. Two records are listed: one for type A with name @ and data Parked, and another for type NS with name @ and data ns11.domaincontrol.com. At the top of the table, there are buttons for Delete, Copy, Filter, Add (highlighted by a yellow arrow), and ... (more options). The bottom of the table has navigation arrows for previous and next pages.

Save your changes after entering the nameserver information. This might involve clicking a 'Save' button or a similar action. It's possible that your changes won't take effect immediately and may take some time to propagate across the internet.

4. Create a Certificate Manager for SSL/TLS

The AWS Certificate Manager is a service that allows for the provision, management, and deployment of public and private (SSL/TLS) certificates in AWS services.

The screenshot shows the "Request certificate" step of the AWS Certificate Manager wizard. The top navigation bar shows "AWS Certificate Manager > Certificates > Request certificate". The main section is titled "Request certificate". Under "Certificate type", there are two options: "Request a public certificate" (selected) and "Request a private certificate". A note below says: "Requesting a private certificate requires the creation of a private certificate authority (CA). To create a private CA, visit AWS Private Certificate Authority [link]". At the bottom right, there are "Cancel" and "Next" buttons.



Fill out the necessary information for the certification manager form. Make sure to use ‘*.doaminName.com’ when adding a name.

The screenshot shows the 'Request public certificate' page in the AWS Certificate Manager (ACM). The 'Domain names' section is filled with a placeholder domain. The 'Validation method' section has 'DNS validation - recommended' selected. The 'Key algorithm' section has 'RSA 2048' selected. The 'Tags' section is empty. At the bottom, there are 'Cancel', 'Previous', and 'Request' buttons, with 'Request' being highlighted in yellow.

After creating the certification, you will have a new screen that contains the necessary details related to your domain and certification. Click on Create a record in route 53, which creates an SSL certificate for your domain.

Domains (1)					
Domain	Status	Renewal status	Type	CNAME name	CNAME value
[REDACTED]	Success	-	CNAME	d4[REDACTED]	[REDACTED]d4[REDACTED]

Press the Create Records button. Route 53 will have a CNAME record added by this.

Return to the Certificate Manager service by navigating back.



Wait until the status is issued. This can take several minutes. **Please do not proceed until the status is Issued.**

The screenshot shows the AWS Certificate Manager interface. A modal window titled "Create DNS records in Amazon Route 53 (1/1)" is open. It displays a table with one row of data:

Domain	Validation status	Type	CNAME name	CNAME value	Is domain in Route 53?
weather.justin-gerber.com	Pending validation	CNAME	a-2597-90dab9c6cf-0f-4991-731a-197207weather.justin-gerber.com	a-2597-90dab9c6cf-0f-4991-731a-197207weather.justin-gerber.com.acm-validation.svc.acloudapi.net	Yes

A yellow arrow points from the bottom right towards the "Create records" button, which is highlighted with a yellow box.

Create a custom domain in route 53 and select an Elastic Beanstalk service.

5. Create a type A record in Route 53 in your custom domain add.

Create a record by clicking on the “Create record” button.

The screenshot shows the AWS Route 53 Hosted Zones interface. A modal window titled "Hosted zone details" is open. It displays a table with four rows of data:

Records (4)	DNSSEC signing	Hosted zone tags (0)
Record 1	Info	
Record 2	Info	
Record 3	Info	

A yellow arrow points from the bottom right towards the "Create record" button, which is highlighted with a yellow box.

Please provide the necessary information. I am naming “api” as the record name. Make sure that A is selected in the record type drop-down list.

- Make sure the alias switch is activated.



- Alias should be selected as the first drop-down list under Route traffic in the Elastic Beanstalk environment.
- Create a second list under Route traffic to Asia Pacific (Mumbai)[ap-south-1] for Route traffic to Asia Pacific (Mumbai)[ap-south-1]. you may select anyone according to your country.
- Ensure that the third drop-down list under Route traffic is set to your elastic Beanstalk environment.
- Press the Create records button.

The screenshot shows the 'Create record' wizard in the AWS Route 53 console. The 'Record name' field contains 'api'. The 'Record type' dropdown is set to 'A - Routes traffic to an IPv4 address and some AWS resources'. The 'Alias' checkbox is selected. Under 'Route traffic to', the 'Alias to Elastic Beanstalk environment' option is chosen, and 'Asia Pacific (Mumbai)' is listed as a target. The 'Routing policy' dropdown is set to 'Simple routing'. A yellow arrow points to the 'Record name' field.

Make sure that “A” and “CNAME” record is present on Route 53 for the weather application.

6. Adding SSL in your Elastic Beanstalk.

Under the Environment Name column, click the Weather-test-app-dev app that was added.

The screenshot shows the 'All environments' list in the AWS Elastic Beanstalk console. The 'Environment name' column lists 'Weather-test-app-dev'. A yellow arrow points to this row. The 'Health' column shows a green status. The 'Application name' column shows 'Weather'. The 'Last modified' column shows '2022-07-30 20:44:10 UTC-0700'. The 'URL' column shows 'weather-test-app-dev.us-east-1.elasticbeanstalk.com'. The 'Platform' column shows 'JET Core running on 64bit Amazon Linux 2'. A yellow arrow points to the 'Weather-test-app-dev' entry.

Afterwards, select the configuration option on the left menu.



Click the edit button under the Instance traffic and scaling. Continue scrolling until you come across Listeners.

The screenshot shows the AWS Lambda console. On the left, there's a sidebar with links: Application versions, Saved configurations, Environment [REDACTED]-env (which is expanded), Go to environment [Edit], Configuration (which is selected and highlighted in blue), Events, Health, and Logs. The main content area has a header "Instance traffic and scaling" with an "Info" link. Below it, a sub-header says "Customize the capacity and scaling for your environment's instances. Select specific platform-specific options." There's an "Edit" button. At the bottom of this section, there's a link "Instances".

Click on the configuration option on the left menu. Click the edit button located in the load balancer section. Click on the button that says Add listener.

The screenshot shows the "Listeners" configuration page. The title is "Listeners". A sub-instruction says: "You can specify listeners for your load balancer. Each listener routes incoming client traffic on a specified port using a specified protocol to your environment processes. By default, we've configured your load balancer with a standard web server on port 80." Below this is a table with columns: Listener Port (sorted by port), Listener Protocol, Instance port, Instance protocol, and SSL certificate. The table contains two rows:

	Listener Port ▲	Listener Protocol ▼	Instance port ▼	Instance protocol ▼	SSL certif
<input type="radio"/>	443	HTTPS	80	HTTP	arn:aws:ac
<input type="radio"/>	80	HTTP	80	HTTP	—

At the top right of the table are "Actions" and "Add listener" buttons.

Set the Listener port to 443.

Set the Listener protocol to HTTPS. Set the Instance port to 80.

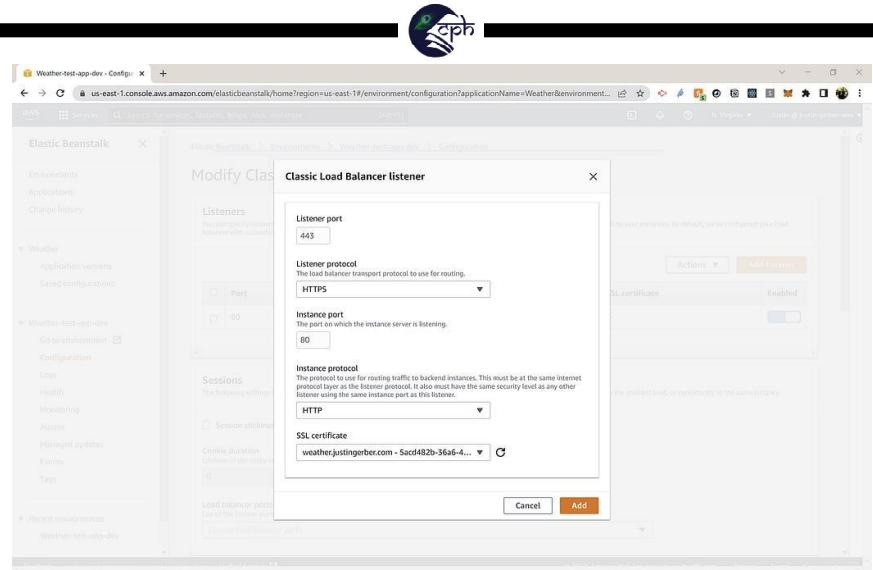
Set the Instance protocol to HTTP.

Set the SSL certificate which is created in step 4.

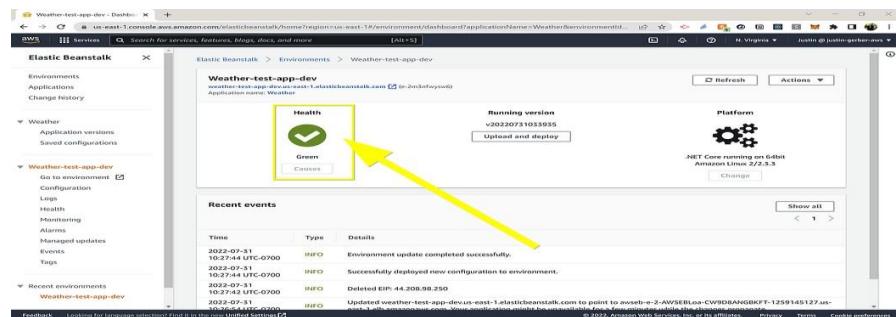
Click the Add button.

Important: Once this has been added, the changes have not been saved yet.

Scroll to the bottom of the page and click the Apply button.



Reflecting the environment on your custom domain will take some time.



That's all! We succeeded in adding a custom domain on AWS Elastic Beanstalk using the Route 53 service. Don't hesitate to explore the dashboard of your newly configured custom domain.

Using CloudWatch for Resource Monitoring, Create CloudWatch Alarms and Dashboards

Introduction:

What's Amazon CloudWatch?

Amazon CloudWatch is an AWS service for monitoring and managing resources in the cloud. It ensures the reliability, availability, and performance of AWS applications and infrastructure.

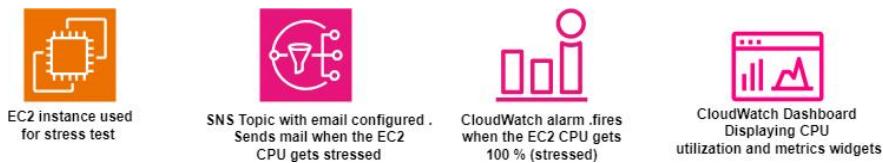
Key features of Amazon CloudWatch:

- Metrics and Alarms:** Collect and monitor metrics, set alarms for predefined thresholds.
- Dashboards:** Create customized dashboards for a centralized view of performance and health.



- **Logs:** Centralize logs, supporting aggregation, searching, and filtering for efficient log management.
- **Events:** Trigger automated actions in response to changes in AWS environment.
- **Insights:** Query log data interactively with CloudWatch Insights.
- **Synthetics:** Create canaries to monitor application availability and latency.
- **Container Insights:** Specialized monitoring for containerized applications.

Architecture Diagram:



Task Steps:

Step 1:

Sign in to AWS Management Console

On the AWS sign-in page ,enter your credentials to log in to your AWS account and click on the Sign in button.

Once Signed in to the AWS Management Console, Make the default AWS Region as US East (N. Virginia) us-east-1

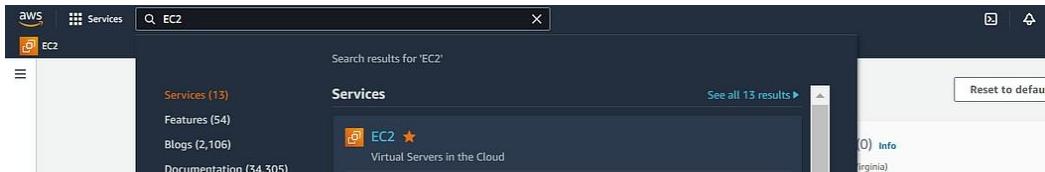
Step 2:

Launching an EC2 Instance

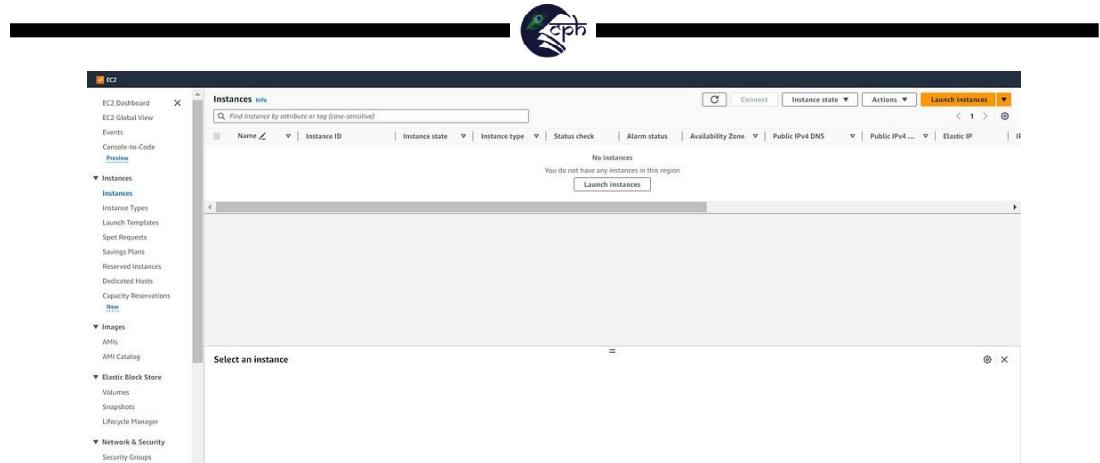
In this step, we are going to launch an EC2 Instance that will be used for checking various features in CloudWatch.

Make sure you are in the N.Virginia Region.

Navigate to EC2 by clicking on the Services menu in the top, then click on EC2 in the Compute section.



3. Navigate to Instances from the left side menu and click on Launch instances button.



4. Name : Enter MyEC2Server

The screenshot shows the 'Launch an instance' wizard. The first step, 'Name and tags', has a 'Name' field containing 'MyEC2Server'. There is also a link to 'Add additional tags'.

5. For Amazon Machine Image (AMI): Select Amazon Linux and the select Amazon Linux 2 AMI from the drop-down.

Note: if there are two AMI's present for Amazon Linux 2 AMI, choose any of them.

The screenshot shows the 'Application and OS Images (Amazon Machine Image)' search results. It features a search bar and a 'Quick Start' section with icons for various operating systems: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. Below this is a detailed view for the 'Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type' AMI. The description indicates it's a 64-bit (x86) HVM gp2 instance. The AMI ID is ami-00b8917ae86a424c9. The interface shows 'Free tier eligible' and a 'Verified provider' badge.

6. For Instance Type: Select t2.micro

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

Instance type

t2.micro Free tier eligible

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

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

7. For Key pair: Select Create a new key pair Button Key pair name: MyEC2Key
Key pair type: RSA
Private key file format: .pem

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

MyEC2Key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH

.ppk For use with PuTTY

⚠️ 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](#)

Cancel **Create key pair**

8. Select Create key pair Button.

▼ 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 - required

MyEC2Key

Create new key pair



9. In Network Settings Click on Edit button: Auto-assign public IP: Enable

Select Create new Security group

Security group name : Enter MyEC2Server_SG

Description : Enter Security Group to allow traffic to EC2

▼ Network settings [Info](#)

VPC - required [Info](#)
vpc-86c1a4fb (Default VPC) (default) [Edit](#)

Subnet [Info](#)
No preference [Edit](#) [Create new subnet](#)

Auto-assign public IP [Info](#)
Enable [Edit](#)

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Security group name - required
MyEC2Server_SG

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](#)
Security Group to allow traffic to EC2

To add SSH :

Choose Type: Select SSH Source: Select Anywhere

10. Keep Rest the things as Default and Click on Launch Instance Button.
11. Select View all Instances to View the Instance you created.
12. Launch Status: Your instance is now launching. Click on the instance ID and wait for complete initialization of the instance (until the status changes to running).

Instances (1) Info										
<input type="text"/> Find Instance by attribute or tag (case-sensitive)										
	Name Z	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
	MyEC2Server	i-035c0db7b676728e	Running Details Logs	t2.micro	-	No alarms +	us-east-1c	ec2-54-196-211-102.co...	54.196.211.102	-



Note: Select the instance and Copy the Instance-ID and save it for later, we need to search the metrics in CloudWatch based on this.

The screenshot shows the AWS CloudWatch Instances console. At the top, there's a search bar and several filter and action buttons. Below the header, a table lists one instance: 'MyEC2Server' with Instance ID 'i-035c0dbf7b676728e'. The instance is running in the 't2.micro' type, located in the 'us-east-1c' availability zone, with a public IPv4 of '54.196.211.102'. The 'Details' tab is selected, displaying various configuration details such as instance type, VPC ID, and IAM Role. Other tabs include Security, Networking, Storage, Status checks, Monitoring, and Tags.

Step 3 :

SSH into EC2 Instance and install necessary Software's Follow the instructions bellow to SSH to your EC2 instance

Once instance is launched, Select EC2 Instance Connect option and click on Connect button.(Keep everything else as default)

A new tab will open in the browser where you can execute the CLI Commands.

The screenshot shows an EC2 Instance Connect session. It features a terminal window with a decorative ASCII art banner at the top. The banner includes the text 'Amazon Linux 2', 'AL2 End of Life is 2025-06-30.', 'A newer version of Amazon Linux is available!', and 'Amazon Linux 2023, GA and supported until 2028-03-15.' followed by a link. Below the banner, the terminal prompt shows the user is logged in as 'ec2-user@ip-172-31-32-101 ~\$'. At the bottom of the session, there's a status bar with the instance ID 'i-035c0dbf7b676728e (MyEC2Server)' and its public and private IP addresses.

- Once you are logged into the EC2 instance, switch to root user. sudo su



3. Update : yum update -y
4. Stress Tool : Amazon Linux 2 AMI does not have the stress tool installed by default, we will need to install some packages sudo amazon-linux-extras install epel -y yum install stress -y
5. Stress tool will be used for simulating EC2 metrics. Once we create the CloudWatch Alarm, we shall come back to SSH and trigger CPUUtilization using it.

Step 4:

Create SNS Topic

In this step, we are going to create a SNS Topic. Make sure you are in the N.Virginia Region.

Navigate to Simple Notification Service by clicking on the Services menu available under the Application Integration section.

The screenshot shows the AWS Services search results for 'SNS'. The search bar at the top says 'Search results for "SNS"'. Below it, there are several service categories: EC2 Dashboard, EC2 Global View, Events, Console-to-Code, Preview, Services (11), Features (25), Documentation (15,310), and Knowledge Articles (224). The 'Simple Notification Service' is listed under Services, described as 'SNS managed message topics for Pub/Sub'. A 'Create topic' button is visible on the right side of the page.

3. Click on Topics in the left panel and then click on Create topic button.

The screenshot shows the 'Amazon Simple Notification Service' page. It features a summary of SNS benefits: 'Reliable delivery messages with durability', 'Automatically scale your workload', and 'No explicit costs'. On the right, there's a 'Create topic' form with fields for 'Topic name' (set to 'MyTopic') and 'Description'. Below the form is a 'Benefits and features' section with links to 'Reliability', 'Scalability', and 'Cost'. A 'Pricing' section notes that SNS has no explicit costs and provides a link to 'Learn more'.

4. Under Details:

Type: Select Standard

Name: Enter MyServerMonitor

Display name: Enter MyServerMonitor

The screenshot shows the 'Create topic' form in the AWS SNS console. The 'Details' tab is selected. Under 'Type', 'Standard' is chosen (radio button is checked). The 'Name' field contains 'MyServerMonitor'. The 'Display name - optional' field also contains 'MyServerMonitor'. There are informational sections for 'Topic type' (FIFO vs Standard) and 'Subscription protocols' (SQS, Lambda, HTTP, SMS, email, mobile application endpoints).



5. Leave other options as default and click on Create topic button. A SNS topic will be created.

The screenshot shows the AWS SNS 'Topics' page. A green header bar at the top indicates that the topic 'MyServerMonitor' was created successfully. Below the header, the topic name 'MyServerMonitor' is displayed. The 'Details' section shows the following information:

Name	MyServerMonitor	Display name	MyServerMonitor
ARN	arn:aws:sns:us-east-1:201559041471:MyServerMonitor	Topic owner	201559041471
Type	Standard		

Below the details, there are tabs for 'Subscriptions', 'Access policy', 'Delivery policy (HTTP/S)', 'Delivery status logging', 'Encryption', 'Tags', and 'Integrations'. The 'Subscriptions' tab is selected, showing a table with one row labeled '(0)'. The table has columns for 'ID', 'Endpoint', 'Status', and 'Protocol'. At the bottom of the 'Subscriptions' section is a prominent orange 'Create subscription' button.

Step 5:

Subscribe to an SNS Topic

Once SNS topic is created, click on SNS topic MyServerMonitor. Click on Create subscription button.

The screenshot shows the AWS SNS 'Topics' page for the 'MyServerMonitor' topic. The 'Details' section is identical to the previous screenshot. The 'Subscriptions' section shows a table with one row labeled '(0)'. The table has columns for 'ID', 'Endpoint', 'Status', and 'Protocol'. A message at the bottom of the table states 'No subscriptions found. You don't have any subscriptions to this topic.' Below the table is a 'Create subscription' button.

3. Under Details:

Protocol : Select Email

Endpoint : Enter your email address



Create subscription

Details

Topic ARN
Q arn:aws:sns:us-east-1:201559041471:MyServerMonitor

Protocol
The type of endpoint to subscribe

Endpoint
An email address that can receive notifications from Amazon SNS.
mrmerki@gmail.com

Info After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

Note: Make sure you give proper email address as this is where your notification will be delivered.

4. You will receive a subscription confirmation to your email address

AWS Notification - Subscription Confirmation

 MyServerMonitor <no-reply@sns.amazonaws.com>
to me

12:29 AM (0 minutes ago)

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:201559041471:MyServerMonitor

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

5. Click on Confirm subscription.


Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:us-east-1:201559041471:MyServerMonitor:1dbb5613-651b-4a96-917a-bb2d9547e0da

If it was not your intention to subscribe, [click here to unsubscribe](#).

6. Your email address is now subscribed to SNS Topic MyServerMonitor.



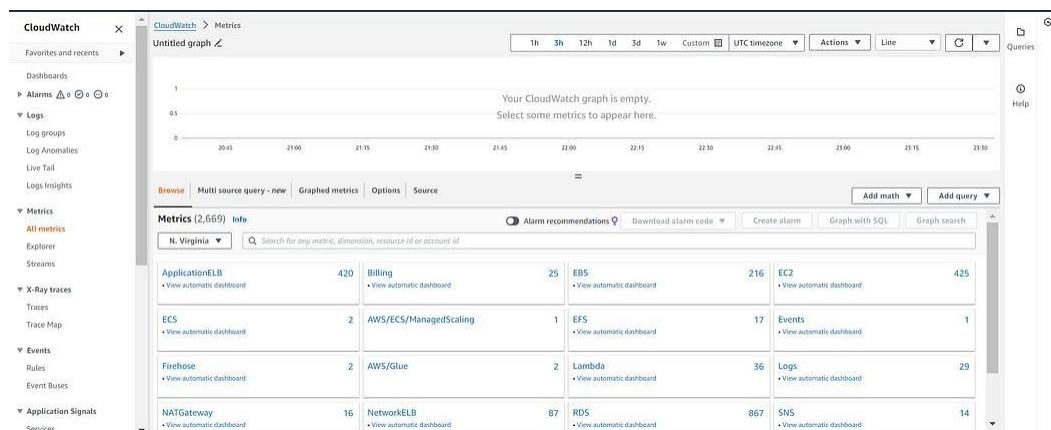
Step 6:

Using CloudWatch to Check EC2 CPU Utilization Metrics in CloudWatch Metrics

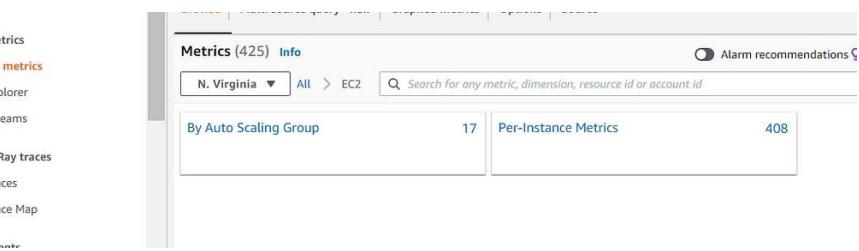
Navigate to CloudWatch by clicking on the Services menu available under the Management & Governance section.

The screenshot shows the AWS Services search interface. The search bar at the top contains 'CloudWatch'. Below the search bar, there is a list of services: 'Amazon SNS' (with 3 items), 'EC2' (selected), 'CloudWatch' (selected), and 'Documentation' (with 12,056 items). The 'CloudWatch' service card includes the text 'Monitor Resources and Applications'.

2. Click on All metrics under Metrics in the Left Panel.
3. You should be able to see EC2 under All Metrics. If EC2 is not visible, please wait for 5-10 minutes, CloudWatch usually takes around 5–10 minutes after the creation of EC2 to start fetching metric details.

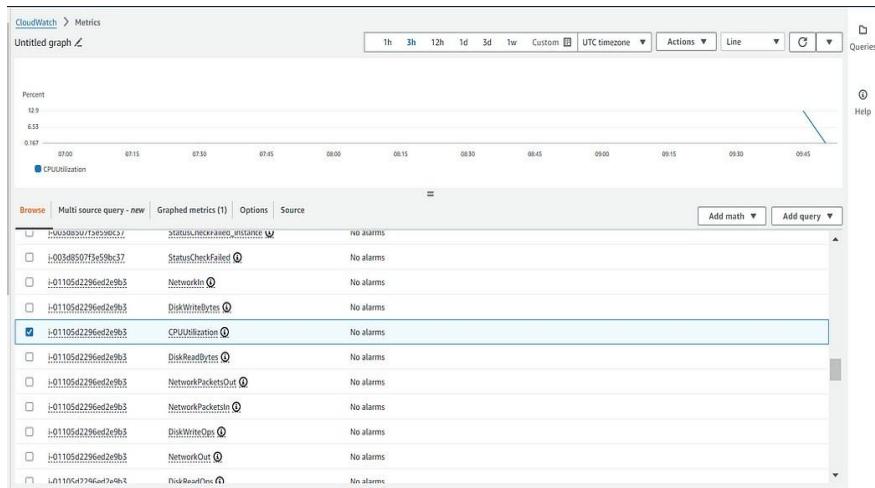


4. Click on EC2. Select Per-Instance Metrics.





5. Here you can see various metrics. Select the CPUUtilization metric to see the graph.



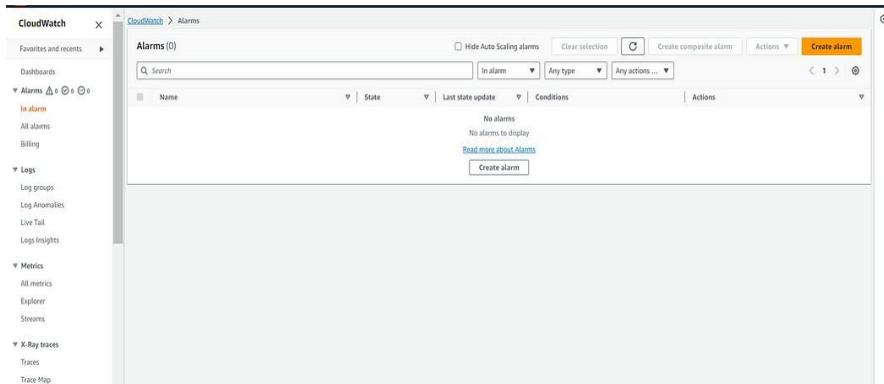
6. Now at the top of the screen, you can see the CPU Utilization graph (which is at zero since we have not stressed the CPU yet).

Step 7:

Create CloudWatch Alarm

CloudWatch alarms are used to watch a single Cloud Watch metric or the result of a math expression based on Cloud Watch metrics.

Click on In alarms under Alarms in the left panel of the Cloud Watch dashboard.



2. Click on Create alarm available on the top right corner.
3. In the Specify metric and conditions page:



Click on Select metric. It will open the Select Metrics page.

CloudWatch > Alarms > Create alarm

Step 1 Specify metric and conditions

Step 2 Configure actions

Step 3 Add name and description

Step 4 Preview and create

Specify metric and conditions

Metric

Graph Preview of the metric or metric expression and the alarm threshold.

Select metric

Cancel Next

Scroll down and Select EC2.

The screenshot shows the CloudWatch Metrics Graph interface. At the top, there's a navigation bar with 'Select metric' and 'Untitled graph'. Below it is a timeline selector with options for 1h, 3h, 12h, 1d, 3d, 1w, Custom, UTC timezone, Line, and a dropdown for 'Graph type'. The main area displays two data series: 'EC2' and 'AWS Glue'. Each series has four data points: CPUUtilization, EPS, Events, and Logs. The CPUUtilization values are 4.20, 374, 12, and 20 respectively. The EPS values are 100, 2, 10, and 22. The Events values are 47, 2, 36, and 22. The Logs values are 9, 16, 20, and 22. The interface includes a 'Browse' button, a 'Multi source query - new' button, and tabs for 'Graphed metrics', 'Options', and 'Source'. There are also buttons for 'Add math' and 'Add query'. A message at the top right says 'Your CloudWatch graph is empty. Select some metrics to appear here.' A 'Cancel' button and a 'Select a single metric to continue' link are at the bottom right.

Select Per-Instance Metrics

The screenshot shows the AWS CloudWatch Metrics console. At the top left, it says "Select metric" and "Untitled graph". Below that is a search bar with "Your CloudWatch graph is empty. Select some metrics to appear here." A timeline at the bottom shows hours from 07:01 to 09:01. The main area is empty, indicating no metrics have been selected.

Enter your EC2 Instance-ID in the search bar to get metrics for MyEC2Server
Choose the CPU Utilization metric.

Click on Select metric button.

The screenshot shows the Azure Metrics Explorer interface. At the top, there's a 'Select metric' dropdown and a 'Custom' button. Below that is a 'Untitled graph' section with a 'Line' chart type. The chart displays 'Percent' values over time from 00:15 to 10:00. A single data series, 'CPUUtilization', is shown with a sharp peak around 09:45. The Y-axis ranges from 0.00 to 12.0. The X-axis shows hours from 00:15 to 10:00. The bottom part of the screen shows a table of metrics for multiple instances, with columns for instance ID, metric name, and status. The 'CPUUtilization' row for instance i0110562796ed2bc8 has a checked checkbox, indicating it's selected.

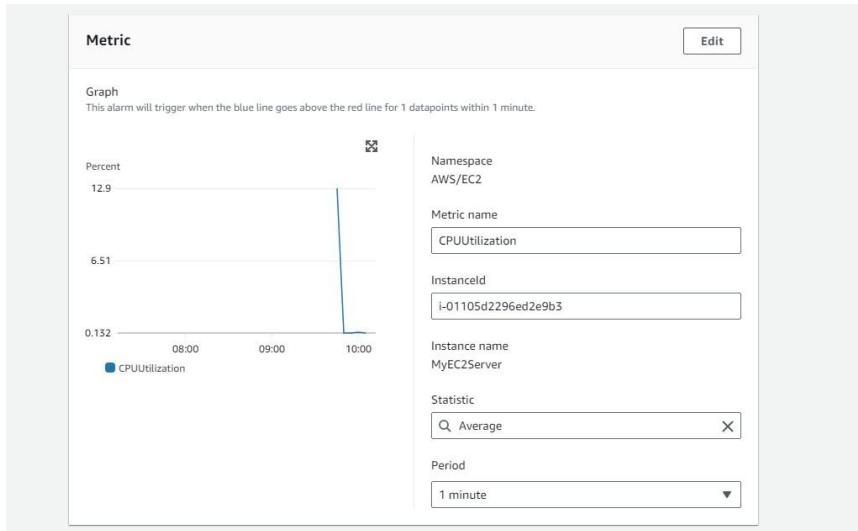
	Browse	Multi source query - new	Graphed metrics [I]	Options	Source	Add math ▾	Add query ▾
<input type="checkbox"/>	i0110562796ed2bc8	StatusCheckFailed_Instance	●	No alarms			
<input type="checkbox"/>	i0110562796ed2bc8	NetworkIn	●	No alarms			
<input type="checkbox"/>	i0110562796ed2bc8	DiskWriteBytes	●	No alarms			
<input checked="" type="checkbox"/>	i0110562796ed2bc8	CPUUtilization	●	No alarms			
<input type="checkbox"/>	i0110562796ed2bc8	DiskReadBytes	●	No alarms			
<input type="checkbox"/>	i0110562796ed2bc8	NetworkPacketsOut	●	No alarms			
<input type="checkbox"/>	i0110562796ed2bc8	NetworkAvailablePort	●	No alarms			



4. Now, configure the alarm with the following details:

Under Metrics

Period: Select 1 Minute



Under Conditions

Threshold type: Choose Static

Whenever CPUUtilization is...: Choose Greater than: Enter 30

Leave other values as default and click on Next button.

Conditions

Threshold type

Static
Use a value as a threshold

Anomaly detection
Use a band as a threshold

Whenever CPUUtilization is...

Define the alarm condition.

Greater
 $>$ threshold

Greater/Equal
 \geq threshold

Lower/Equal
 \leq threshold

Lower
 $<$ threshold

than...

Define the threshold value.

30

Must be a number

► Additional configuration

Cancel **Next**



5. In Configure actions page:

Under Notification

Alarm state trigger: Choose In Alarm

Select an SNS topic: Choose Select an existing SNS topic

Send a notification to... : Choose My Server Monitor SNS topic which was created earlier.

CloudWatch > Alarms > Create alarm
Step 1 Specify metric and conditions
Step 2 Configure actions
Step 3 Add name and description
Step 4 Preview and create

Configure actions

Notification

Alarm state trigger
Define the alarm state that will trigger this action.

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.

Send a notification to the following SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic
 Create new topic
 Use topic ARN to notify other accounts

Send a notification to...
MyServerMonitor

Email (endpoints)
mrnterki@gmail.com - View in SNS Console

Add notification

Leave other fields as default. Click on Next button.

6. In the Add a description page, (under Name and Description): Name: Enter the Name My Server CPU Utilization Alarm Click on Next button.

CloudWatch > Alarms > Create alarm
Step 1 Specify metric and conditions
Step 2 Configure actions
Step 3 Add name and description
Step 4 Preview and create

Add name and description

Name and description

Alarm name
MyServerCPUUtilizationAlarm

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

[Edit](#) [Preview](#)

This is an H1
double asterisks will produce strong character
This is [an example](https://example.com/) inline link.
Up to 1024 characters (0/1024)

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**

7. A preview of the Alarm will be shown. Scroll down and click on Create alarm button.

8. A new Cloud Watch Alarm is now created.



The screenshot shows the AWS CloudWatch Alarms interface. On the left, there's a sidebar with navigation links like 'CloudWatch', 'Dashboards', 'Alarms', 'Logs', and 'Metrics'. The main area is titled 'CloudWatch > Alarms' and contains a table with one row. The row has a checkbox, the name 'MyServerCPUUtilizationAlarm', a status indicator 'OK', the last update time '2024-01-04 10:21:00', the condition 'CPUUtilization > 30 for 1 datapoints within 1 minute', and an 'Actions' section with a note '0 actions enabled'.

Whenever the CPU Utilization goes above 30 for more than 1 minute, an SNS Notification will be triggered and you will receive an email

Step 8:

Testing Cloud Watch Alarm by Stressing CPU Utilization SSH back into the EC2 instance — MyEC2Server.

The stress tool has already been installed. Let's run a command to increase the CPU Utilization manually. sudo stress --cpu 10 -v --timeout 400s

3. This command shall monitor the process created by the stress tool(which we triggered manually). It will run for 6 minutes and 40 seconds. It will monitor CPU utilization, which should remain very near 100% for that amount of time.

```
Complete!
[root@ip-172-31-28-166 ec2-user]# sudo stress --cpu 10 -v --timeout 400s
stress: info: [3919] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd
stress: dbug: [3919] using backoff sleep of 30000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 10 [3920] forked
stress: dbug: [3919] using backoff sleep of 27000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 9 [3921] forked
stress: dbug: [3919] using backoff sleep of 24000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 8 [3922] forked
stress: dbug: [3919] using backoff sleep of 21000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 7 [3923] forked
stress: dbug: [3919] using backoff sleep of 18000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 6 [3924] forked
stress: dbug: [3919] using backoff sleep of 15000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 5 [3925] forked
stress: dbug: [3919] using backoff sleep of 12000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 4 [3926] forked
stress: dbug: [3919] using backoff sleep of 9000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 3 [3927] forked
stress: dbug: [3919] using backoff sleep of 6000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 2 [3928] forked
stress: dbug: [3919] using backoff sleep of 3000us
stress: dbug: [3919] setting timeout to 400s
stress: dbug: [3919] --> hogcpu worker 1 [3929] forked
```

4. Open another Terminal on your local machine and SSH back in EC2 instance — MyEC2Server.
5. Run this command to see the CPU utilization if you are a MAC or Linux User. For Windows User, you can navigate to Task manager.

top

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
1288	root	20	0	7584	96	0	R	10.3	0.0	0:03.06 stress
1290	root	20	0	7584	96	0	R	10.3	0.0	0:03.06 stress
1287	root	20	0	7584	96	0	R	10.0	0.0	0:03.05 stress
1289	root	20	0	7584	96	0	R	10.0	0.0	0:03.05 stress
1292	root	20	0	7584	96	0	R	10.0	0.0	0:03.06 stress
1293	root	20	0	7584	96	0	R	10.0	0.0	0:03.06 stress
1294	root	20	0	7584	96	0	R	10.0	0.0	0:03.06 stress
1296	root	20	0	7584	96	0	R	10.0	0.0	0:03.05 stress
1291	root	20	0	7584	96	0	R	9.6	0.0	0:03.05 stress
1295	root	20	0	7584	96	0	R	9.6	0.0	0:03.05 stress
1	root	20	0	123488	5408	3908	S	0.0	0.6	0:01.95 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 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
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:0H-ev
7	root	20	0	0	0	0	I	0.0	0.0	0:00.21 kworker/u30:0-e
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_tasks_rude_
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_tasks_trace_
11	root	20	0	0	0	0	S	0.0	0.0	0:00.14 ksoftirqd/0
12	root	20	0	0	0	0	I	0.0	0.0	0:00.15 rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.01 migration/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00 cpuhp/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kdevtmpfs
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 netns
21	root	20	0	0	0	0	S	0.0	0.0	0:00.01 kauditd
299	root	20	0	0	0	0	S	0.0	0.0	0:00.00 khungtaskd
300	root	20	0	0	0	0	S	0.0	0.0	0:00.00 com_reaper
301	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 writeback
303	root	20	0	0	0	0	S	0.0	0.0	0:00.10 kcompactd0
304	root	25	5	0	0	0	S	0.0	0.0	0:00.00 ksmd

6. You can now see that %Cpu(s) is 100. By running this stress command, we have manually increased the CPU utilization of the EC2 Instance.

7. After 400 Seconds, the %Cpu will reduce back to 0.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
1727	ec2-user	20	0	168948	4308	3764	R	0.3	0.4	0:00.38 top
1	root	20	0	123488	5408	3908	S	0.0	0.6	0:01.96 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 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
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kworker/0:0H-ev
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_tasks_rude_
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_tasks_trace
11	root	20	0	0	0	0	S	0.0	0.0	0:00.15 ksoftirqd/0
12	root	20	0	0	0	0	I	0.0	0.0	0:00.16 rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.02 migration/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00 cpuhp/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kdevtmpfs
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 netns
21	root	20	0	0	0	0	S	0.0	0.0	0:00.01 kauditd
299	root	20	0	0	0	0	S	0.0	0.0	0:00.00 khungtaskd
300	root	20	0	0	0	0	S	0.0	0.0	0:00.00 com_reaper
301	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 writeback
303	root	20	0	0	0	0	S	0.0	0.0	0:00.11 kcompactd0
304	root	25	5	0	0	0	S	0.0	0.0	0:00.00 ksmd
305	root	39	19	0	0	0	S	0.0	0.0	0:00.00 khugepaged
360	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kintegrityd
362	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 kblockd
363	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 blkcg_punt_bio
715	root	20	0	0	0	0	S	0.0	0.0	0:00.00 xen-balloon
721	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 rpm_dev_wq
730	root	0	-20	0	0	0	I	0.0	0.0	0:00.00 edac-poller
735	root	-51	0	0	0	0	S	0.0	0.0	0:00.00 watchdogd
767	root	20	0	0	0	0	I	0.0	0.0	0:00.00 kworker/u30:1-x



Step 9 :

Checking For an Email from the SNS Topic

Navigate to your mailbox and refresh it. You should see a new email notification for My Server CPU Utilization Alarm.

MyServerMonitor <no-reply@sns.amazonaws.com> to me ▾

11:26 AM (4 minutes ago)

You are receiving this email because your Amazon CloudWatch Alarm "MyServerCPUUtilizationAlarm" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [61.36666666666666 (04/01/24 10:20:00)] was greater than the threshold (30.0) (minimum 1 datapoint for OK->ALARM transition)" at "Thursday 04 January, 2024 10:26:00 UTC".

View this alarm in the AWS Management Console:
<https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink?is?region=us-east-1&alarmNameV2=alarm/MyServerCPUUtilizationAlarm>

Alarm Details:

- Name: MyServerCPUUtilizationAlarm
- Description:
- State Change: INSUFFICIENT_DATA->ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [61.36666666666666 (04/01/24 10:20:00)] was greater than the threshold (30.0) (minimum 1 datapoint for OK->ALARM transition).
- Timestamp: Thursday 04 January, 2024 10:26:00 UTC
- AWS Account: 316011138685
- Alarm Arn: arn:aws:cloudwatch:us-east-1:316011138685:alarm:MyServerCPUUtilizationAlarm

Threshold:

- The alarm is in the ALARM state when the metric is GreaterThanThreshold 30.0 for at least 1 of the last 1 period(s) of 60 seconds.

Monitored Metric:

- MetricNamespace: AWS/EC2
- MetricName: CPUUtilization
- Dimensions: [InstanceId = i-01105d2296ed2e9b3]
- Period: 60 seconds
- Statistic: Average
- Unit: not specified
- TreatMissingData: missing

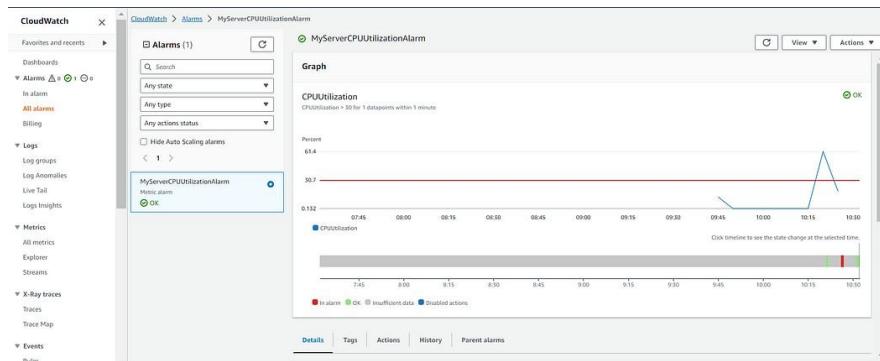
2. We can see that mail we received contains details about our Cloud Watch Alarm,(name of the alarm, when it was triggered, etc.).

Step 10:

Checking the Cloud Watch Alarm Graph

Navigate back to Cloud Watch page, Click on Alarms. Click on My Server CPU Utilization Alarm.

On the Graph, you can see places where CPU Utilization has gone above the 30% threshold.



4. We can trigger CPU Utilization multiple times to see the spike on the graph.
5. You have successfully triggered a Cloud Watch Alarm for CPU Utilization.

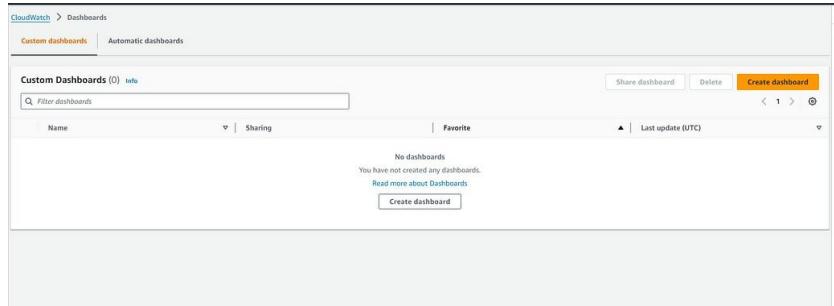


Step 11:

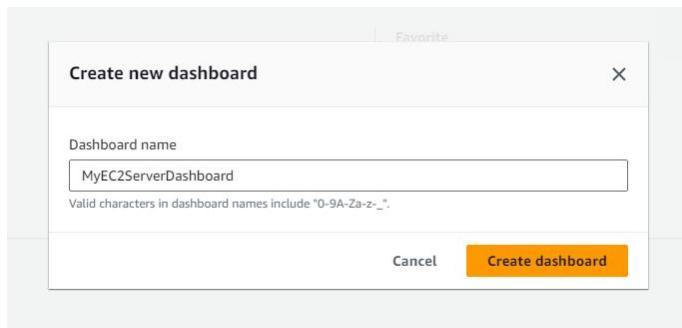
Create a Cloud Watch Dash board

We can create a simple Cloud watch dashboard to see the CPU Utilization and various other metric widgets.

Click on Dash board in the left panel of the Cloud Watch page. Click on Create dashboard button.

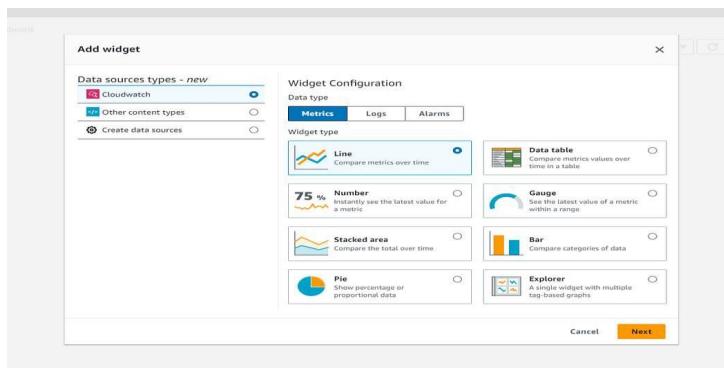


Dashboard name: Enter My EC2 Server Dashboard



Click on Create dashboard

Add widget: Select Line Graph.



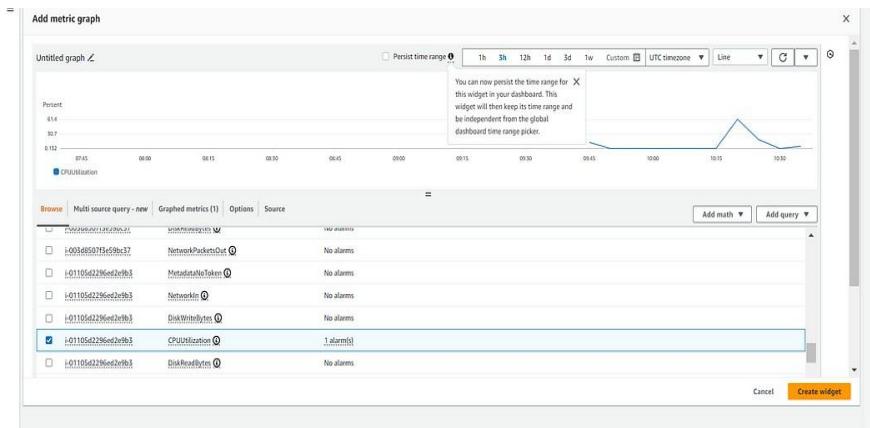


Click on Next button.

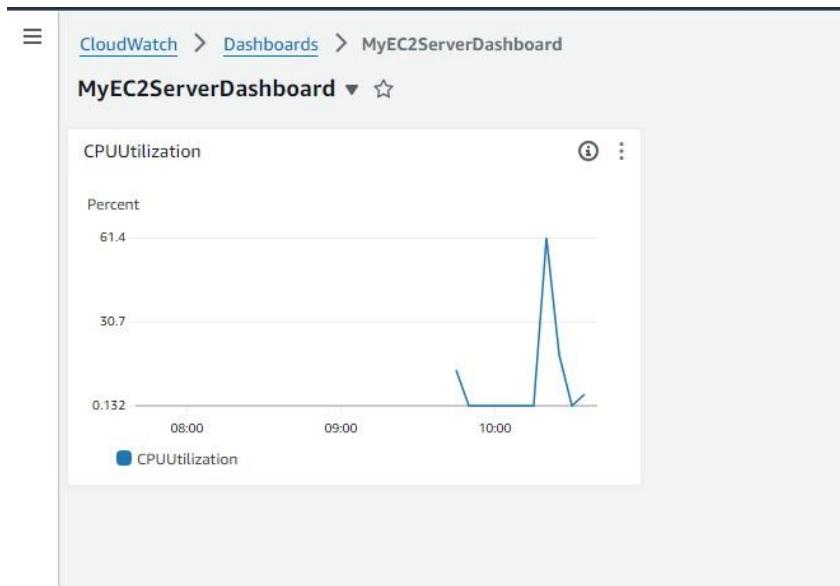
Select Metrics. Click on Next button.

On the next page, Choose EC2 under the Metrics tab. Choose Per- Instance Metrics.

In the search bar, enter your EC2 Instance ID. Select CPUUtilization.



Click on Create Widget button.

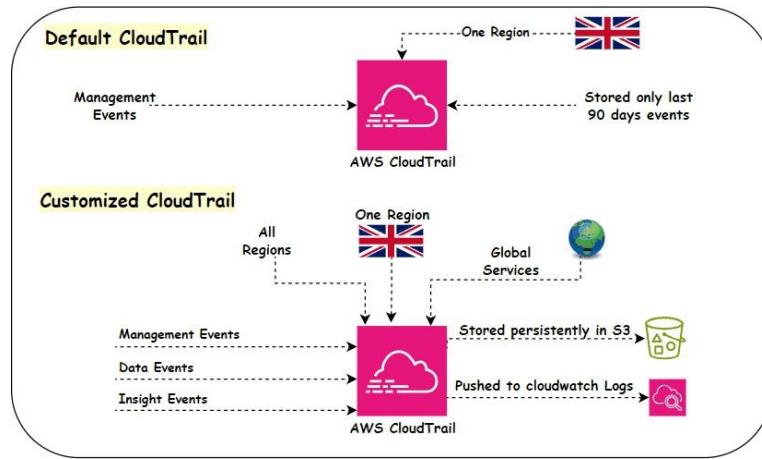


3. Depending on how many times you triggered the stress command, you will see different spikes in the timeline.
4. Now click on the Save button.



- You can also add multiple Widgets to the same Dashboard by clicking on Add widget button.

Exploring AWS CloudTrail: Auditing and Monitoring AWS API Activity



WS Cloud Trail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. It provides event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services.

What is Cloud Trail?

Cloud Trail continuously monitors and logs account activity across all AWS services, including actions taken by a user, role, or AWS service.

The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service.

Why Use Cloud Trail?

Here are some key reasons to use Cloud Trail:



- **Audit Compliance:** Cloud Trail logs provide detailed records of all API calls, which can be used to audit compliance with regulatory standards like HIPAA and PCI.
- **Security Analysis:** The API call logs can be analyzed to detect anomalous activity and unauthorized access to determine security issues.
- **Operational Issues:** The activity history can help troubleshoot operational issues by pinpointing when an issue began and what actions were taken.
- **Resource Changes:** You can identify what changes were made to AWS resources by viewing Cloud Trail events.

Cloud Trail Log Files

Cloud Trail log files contain the history of API calls made on your account. These log files are stored in Amazon S3 buckets that you specify. You can define S3 buckets per region or use the same bucket for all regions.

The log files capture API activity from all Regions and are delivered every 5 minutes. You can easily search and analyze the logs using Amazon Athena, Amazon Elasticsearch, and other tools.

Cloud Trail Events

Cloud Trail categorizes events into two types:

- **Management events:** Provides information about management operations that are performed on resources in your AWS account. These include operations like creating, modifying, and deleting resources.
- **Data events:** Provides information about resource operations performed on or in a resource. These include operations like Amazon S3 object-level API activity.

You can choose to log both management and data events or just management events. Data events allow more granular visibility into resource access.

Enabling Cloud Trail

Enabling Cloud Trail is simple and can be done in a few steps:

- Sign into the AWS Management Console and open the CloudTrail console.
- Get started by creating a new trail and specify a name.
- Choose whether to log management and/or data events.
- Specify an existing S3 bucket or create a new one where logs will be stored.
- Click Create to finish enabling Cloud Trail.
- Once enabled, Cloud Trail will begin recording events and delivering log files to the designated S3 bucket. You can customize trails further by adding tags, configuring log file validation, logging to Cloud Watch Logs, and more.



Use Cases

Here are some common use cases for CloudTrail:

- **User Activity Monitoring:** Review which users and accounts are performing actions across services.
- **Service Usage Optimization:** Analyze usage patterns to identify opportunities to reduce costs.
- **Security Forensics:** Investigate unusual activity when a security incident occurs by reviewing relevant events.
- **Regulatory Compliance:** Meet compliance requirements that mandate detailed activity logging and audit trails.

Cloud Trail provides a simple way to get visibility into account activity by recording API calls made across AWS. The event history and logs can be used for auditing, security analysis, troubleshooting, and more.

Businesses of all sizes can benefit from enabling Cloud Trail to gain insight into how their AWS resources are being accessed and modified. Tutorial

AWS Cloud Trail is a service that records AWS API calls for your account and delivers log files to you. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service.

In this tutorial, we will walk through how to enable Cloud Trail, view and analyze the log files, and leverage Cloud Trail logs for auditing and security.

Prerequisites

Before starting, you should have:

- An AWS account
- Basic understanding of AWS services
- An S3 bucket to store the Cloud Trail logs

Enabling Cloud Trail

Let's start by enabling Cloud Trail across all Regions:

- Go to the CloudTrail console in the AWS Management Console.
- Click “Trails” in the left sidebar and then “Create trail”.
- Enter a name for the trail such as “Cloud Trail-AllRegions”.
- Under Storage location, create or select an existing S3 bucket.
- For log file encryption, select AWS KMS to encrypt the logs.



- Click “Create” to enable the trail.
- Cloud Trail will now begin recording events and sending log files to the designated S3 bucket.

Viewing Cloud Trail Log Files

The log files can be viewed in the S3 bucket or analyzed using Athena, Elasticsearch, or other tools. Let’s take a look at the logs:

- Go to the S3 console and open the bucket storing the CloudTrail logs.
- Open one of the log files and inspect the JSON content.
- You will see API call details like source IP, user agent, resource affected, and parameters.

The logs provide a comprehensive audit trail of all API activity across services.

Using CloudTrail Insights

Cloud Trail Insights detects unusual activity by continuously analyzing event patterns. Let’s enable it:

- From the Cloud Trail console, go to “Trails” and select the trail.
- Under “Insights Events”, enable insights.
- In “Insights summary”, you can see detected anomalies.
- Click on events to see the anomalous activity details.
- Insights makes it easy to identify potential security issues.

In this tutorial, we enabled Cloud Trail across all Regions, viewed the generated log files, and enabled Cloud Trail Insights. The event history and anomaly detection allow for auditing, operational analysis, security monitoring, and more. Be sure to leverage Cloud Trail logs to gain visibility into your AWS account activity.

Common AWS CLI Commands for CloudTrail

Here are some common AWS CLI commands for working with AWS Cloud Trail:

Create Cloud Trail trail

```
aws cloudtrail create-trail --name my-trail --s3-bucket-name my-bucket
```

Describe Cloud Trail trail

```
aws cloudtrail describe-trails --trail-name-list my-trail
```



Start Cloud Trail logging

```
aws cloudtrail start-logging --name my-trail
```

Stop CloudTrail logging

```
aws cloudtrail stop-logging --name my-trail
```

List CloudTrail events

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName
```

Get CloudTrail log files

```
aws s3 cp s3://my-bucket/AWSLogs/my-account-id/CloudTrail . --recursive
```

Delete CloudTrail trail

```
aws cloudtrail delete-trail --name my-trail
```

Additional CloudTrail CLI commands

There are many additional CloudTrail CLI commands available:

- update-trail — Updates settings for a trail
- list-tags — Lists tags for a trail
- add-tags — Adds tags to a trail
- remove-tags — Removes tags from a trail
- list-public-keys — Lists public keys for log file validation
- get-trail-status — Returns status of Cloud Trail logging
- list-trails — Lists trails in the account

Final Words

AWS CloudTrail provides a simple yet powerful way to gain visibility into activity across your AWS account. By recording API calls made to various AWS services, CloudTrail delivers detailed audit logs that can be analyzed for security, compliance, and operational purposes. This tutorial guided you through enabling CloudTrail across all Regions, inspecting the generated log



files, and leveraging CloudTrail Insights to detect unusual activity. With CloudTrail activated, you now have comprehensive visibility into changes, user activity, and resource access within your AWS environment. Be sure to consult the CloudTrail logs regularly for auditing, monitoring AWS usage, troubleshooting issues, and investigating security incidents. We encourage you to explore the other capabilities of CloudTrail such as log file encryption, log validation, data event logging, and integrating logs with other AWS services. CloudTrail is a key component of the AWS shared responsibility model, enabling you to monitor the activity within your account and respond appropriately.

AWS Cloud Trail

Moving your complex resources and workloads to the cloud can make it challenging for your organization to analyze and understand everything in your AWS environment. AWS CloudTrail is a management service provided by AWS that enables governance, compliance, operational origin, and risk auditing of your AWS Account. AWS CloudTrail provides a comprehensive record history so you can easily see who made changes, where they made the changes, and when the changes were made. AWS Audit logs provide a wealth of information on every activity within your AWS environments.

With AWS Cloud Trail, you can search and track all account activities to monitor user changes, compliance, error rates, and risks.

The capabilities of CloudTrail are essential to simplifying your AWS environment troubleshooting and letting you identify areas that need improvements.

In this tutorial, we'll explore using AWS CloudTrail to monitor every activity and track user changes on our AWS Account.

Features of CloudTrail

- **Multi-Regional:** AWS CloudTrail allows the user to make trails from any part of the world, and you can enable this functionality from the actions tab.
- **Event History:** Event history is a tab on AWS CloudTrail that lets the user see what's happening in CloudTrail and all the services (S3, Lambda, Dynamo DB) integrated into CloudTrail.
- **File Encryption:** File encryption is done by AWS KMS, the key management system that allows you to encrypt the logs created from your environment to maintain the stability of your log files.
- **File Integrity:** File integrity checks for file validation and whether all the files are corrupt. If there's any form of corruption in any of the log files, it'll destroy the integrity of the file.

Getting Started

In your AWS Management Console, search and click on AWS Cloud Trail.

The screenshot shows the AWS CloudTrail landing page. It features a main heading "AWS CloudTrail" with the subtext "Continuously log your AWS account activity". Below this, there's a section titled "How it works" with two icons: "Capture" (cloud with a camera) and "Store" (cloud with a bucket). A callout text states: "Record activity in AWS services as AWS CloudTrail events" and "AWS CloudTrail delivers events to the AWS CloudTrail console, Amazon S3 buckets, and optionally Amazon CloudWatch Logs". To the right, there are boxes for "Create a trail with AWS CloudTrail", "Pricing", and "Getting started".

- Create a New Trail by clicking on Create Trail.

The screenshot shows the AWS CloudTrail Dashboard. On the left, under "Trails", there is a table with one row labeled "No trails" and a "Create trail" button. On the right, under "CloudTrail Insights", there is a message stating "CloudTrail Insights is not enabled" and a "Create a trail" button.

- Choose your Trail attributes. Enter your Trail name and storage location (select an existing S3 bucket or create a new S3 bucket). Enable your log file encryption with your file validation. This will ensure all aws resources are encrypted.

The screenshot shows the "Choose trail attributes" step in the AWS CloudTrail creation wizard. It includes fields for "General details" (a multi-region trail), "Trail name" (set to "workflow_trail"), "Storage location" (radio button selected for "Use existing S3 bucket" with "workflow-hosting" chosen), "Trail log bucket name" (prefix "workflow-hosting" entered), "Prefix - optional" (empty), "Log file SSE-KMS encryption" (checkbox checked, "Enabled"), "Customer managed AWS KMS key" (radio button selected for "New"), "AWS KMS alias" (empty), and "Additional settings" (checkbox checked, "Enabled").



- When you're done configuring your Trail attributes, click on **Next**. Next, choose your log events. In AWS CloudTrail, there are three types of events. Management events, Data events, and Insights events.
- Management events are free and can be viewed in the event history tab for 90 days. Data events are not free to the user and cannot be viewed in the event history tab. Insights events let you identify unusual activity, errors, or user behavior in your account.

Only Management events are free for your workloads. Data and Insights events will incur costs. In this tutorial, we'll be using **Management Events**.

Choose log events

Events Info
Record API activity for individual resources, or for all current and future resources in AWS account. [Additional charges apply](#)

Event type
Choose the type of events that you want to log.

Management events
Capture management operations performed on your AWS resources.

Data events
Log the resource operations performed on or within a resource.

Insights events
Identify unusual activity, errors, or user behavior in your account.

Management events Info
Management events show information about management operations performed on resources in your AWS account.

API activity
Choose the activities you want to log.

Read **Write**

Exclude AWS KMS events

Exclude Amazon RDS Data API events

Cancel **Previous** **Next**

- When you're done configuring log events, click on **Next**, you'll see the overview and general details of your configuration, and click on **Create Trail**.
- In your Trails dashboard, you'll see the Trail you just created.

CloudTrail > Trails

Name	Home region	Multi-region trail	Insights	Organization trail	S3 bucket	Log file prefix	CloudWatch Logs log group	Status
workflow_trail	US East (Ohio)	Yes	Disabled	No	workflow-hosting			Logging



- Integrate other AWS resources with your trail to see how it works and see different log events. For example, in my S3 bucket, I'll upload a new file into my S3 bucket. Once I'm done uploading the file, I'll automatically see the events in my CloudTrail.

The screenshot shows the 'Upload' interface in the Amazon S3 console. At the top, there's a breadcrumb navigation: 'Amazon S3 > Buckets > workflow-hosting > Upload'. Below this, a large central area has a dashed border and contains the text 'Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.' A file list titled 'Files and folders (1 Total, 60.0 KB)' shows one item: 'Screenshot 2023-02-25 at 23.13.12.png' (image/png, 60.0 KB). There are 'Remove', 'Add files', and 'Add folder' buttons above the list. A search bar labeled 'Find by name' is present. The 'Destination' section shows the target bucket as 's3://workflow-hosting'. Under 'Destination details', it says 'Bucket settings that impact new objects stored in the specified destination.' Below this, sections for 'Permissions' (with a note about granting public access) and 'Properties' (with a note about specifying storage class, encryption settings, tags, and more) are shown. At the bottom right are 'Cancel' and 'Upload' buttons.

- In your CloudTrail event history, you'll see all your events and logs from your S3 bucket.

The screenshot shows the 'Event history' interface in the CloudTrail console. At the top, there's a breadcrumb navigation: 'CloudTrail > Event history'. The main area has a title 'Event history (9) Info' with a note that it shows the last 90 days of management events. It includes 'Download events' and 'Create Athena table' buttons. A 'Lookup attributes' section has a dropdown for 'Resource type' and a search bar. Below is a table with columns: 'Event name', 'Event time', 'User name', 'Event source', 'Resource type', and 'Resource name'. The table lists four entries:

Event name	Event time	User name	Event source	Resource type	Resource name
UpdateTrail	February 26, 2023, 21:16:57 (UT...)	root	cloudtrail.amazonaws.com	AWS::CloudTrail::Trail, ...	arn:aws:cloudtrail:us-east-2:27035...
PutBucketPolicy	February 26, 2023, 21:16:57 (UT...)	root	s3.amazonaws.com	AWS::S3::Bucket	workflow-hosting
PutInsightSelectors	February 26, 2023, 20:45:04 (UT...)	root	cloudtrail.amazonaws.com	AWS::CloudTrail::Trail	arn:aws:cloudtrail:us-east-2:27035...

- You'll see your event records and referenced resources when you click on them.

Resources referenced (1) [Info](#)

Resource type	Resource name	AWS Config resource timeline
AWS::S3::Bucket	workflow-hosting Edit	Enable AWS Config resource recording Edit

Event record [Info](#)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Root",
    "principalId": "270355947833",
    "arn": "arn:aws:iam::270355947833:root",
    "accountId": "270355947833",
    "accessKeyId": "ASIAJ54TYME4UIJUKD9R",
    "userName": "thecrafterman",
    "sessionContext": {
      "sessionIssuer": {},
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-02-26T19:14:01Z",
        "mfaAuthenticated": "true"
      }
    }
  },
  "eventTime": "2023-02-26T20:16:57Z",
  "eventName": "PutBucketPolicy",
  "eventSource": "aws:s3:api",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "197.210.79.54",
  "userAgent": "[ANONYMOUS] vendor/Oracle Corporation cfg/retry-mode/standard",
  "requestParameters": {
    "bucketPolicy": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "ANGCloudTrailAclCheck20150319",
          "Effect": "Allow",
          "Principal": {
            "Service": "cloudtrail.amazonaws.com"
          },
          "Action": "s3:GetBucketAcl",
          "Resource": "arn:aws:s3:::workflow-hosting",
          "Condition": {
            "StringEquals": {
              "AWS:SourceArn": "arn:aws:cloudtrail:us-east-1:270355947833:trail/workflow_trail"
            }
          }
        }
      ]
    }
  }
}
```

- You can also filter your event history based on AWS access key, Event ID, Event Name, Event Source, Resource name, and user type.

[CloudTrail](#) > [Event history](#)

Event history (9) [Info](#)

Event history shows you the last 90 days of management events.

Lookup attributes

Read-only	▲	🔍 false
Lookup attributes		
AWS access key	▼	event time
Event ID	▼	February 26, 2023, 21:16:57 (UT...
Event name	▼	PutBucketPolicy
Event source	▼	aws:s3:api
Read-only	▼	February 26, 2023, 20:45:04 (UT...
Resource name	▼	workflow-hosting
Resource type	▼	AWS::S3::Bucket
User name	▼	thecrafterman



- You'll see the PUT event history in your Event Name, the S3 bucket we updated earlier.

The screenshot shows the AWS CloudTrail Event history interface. It displays three log entries:

Event name	Event time	User name	Event source	Resource type	Resource name
UpdateTrail	February 26, 2023, 21:16:57 (UT...)	root	cloudtrail.amazonaws.com	AWS::CloudTrail::Trail	arn:aws:cloudtrail:us-east-2:270355947833:tr... (redacted)
PutBucketPolicy	February 26, 2023, 21:16:57 (UT...)	root	s3.amazonaws.com	AWS::S3::Bucket	workflow-hosting

- In your AWS S3 storage bucket, you'll see your CloudTrail log events in the AWS logs folder.

The screenshot shows the AWS S3 bucket named "workflow-hosting". Inside the "AWSLogs" folder, there are two objects: "CloudTrail-Digest/" and "CloudTrail/".

- When you click on Cloud Trail, you can see the logs from each AWS Region.

The screenshot shows the AWS S3 bucket named "workflow-hosting". Inside the "CloudTrail" folder, there are 17 sub-folders representing different AWS Regions: ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-south-1, ap-southeast-1, ap-southeast-2, ca-central-1, eu-central-1, eu-north-1, eu-west-1, eu-west-2, eu-west-3, sa-east-1, us-east-1, us-east-2, us-west-1, and us-west-2.



Conclusion

You can see how fast it is to enable and configure AWS CloudTrail on your AWS resources and view log events in your Event History dashboard. CloudTrail is a service that has the primary function to record and track all AWS API requests made. These API calls can be programmatic requests initiated by a user using an SDK, from the AWS CLI, or within the AWS management console. With our Open-Source workflows, you can automatically send an API request with our ops cli to automatically enable logs and events into your AWS resources.

Route 53

we dive into Route 53, Amazon's highly scalable and reliable Domain Name System (DNS) web service. Route 53 offers a plethora of features to manage your domain names and direct internet traffic efficiently.

Let's explore the key concepts and functionalities of Route 53.



Introduction to Route 53 DNS service

Route 53 is a scalable and highly available Domain Name System (DNS) web service offered by Amazon Web Services (AWS). It is named after the TCP/IP port 53, where DNS requests are addressed. Route 53 effectively translates human-readable domain names (like example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other on the internet.

Key Features of Route 53:

- Scalability:** Route 53 is designed to handle large volumes of DNS queries without any degradation in performance. It can scale automatically to manage changes in traffic patterns and query loads.
- High Availability:** Route 53 is built on the same infrastructure that powers other AWS services. It is distributed across multiple geographically diverse data centers, ensuring high availability and reliability of DNS resolution.
- Global Coverage:** Route 53 has a global network of DNS servers strategically located around the world. This ensures fast and reliable DNS resolution for users accessing your applications from different geographic regions.
- Integration with AWS Services:** Route 53 seamlessly integrates with other AWS services such as Elastic Load Balancing (ELB), Amazon S3, Amazon EC2, and more. This allows you to easily map domain names to your AWS resources and manage traffic routing efficiently.



- **Advanced Routing Policies:** Route 53 supports various routing policies like simple routing, weighted routing, latency-based routing, geolocation routing, and failover routing. These policies enable you to implement sophisticated traffic management strategies based on your specific requirements.
- **Health Checks:** Route 53 provides health checks to monitor the health and availability of your resources. You can configure health checks for endpoints like web servers, load balancers, and more. Route 53 automatically routes traffic away from unhealthy endpoints, helping you maintain high availability and reliability.
- **DNS Failover:** Route 53 offers DNS failover functionality, which automatically redirects traffic from a failed or unhealthy resource to a healthy one. This helps minimize downtime and ensures continuous availability of your applications.

Use Cases for Route 53:

- **Hosting Websites:** Route 53 can be used to host your website's DNS records, including mapping domain names to web servers and configuring subdomains.
- **Load Balancing:** Route 53 works seamlessly with Elastic Load Balancers (ELB) to distribute incoming traffic across multiple EC2 instances or containers, ensuring optimal performance and fault tolerance.
- **Disaster Recovery:** Route 53's DNS failover feature can be used to implement disaster recovery strategies by automatically redirecting traffic to backup resources in case of primary resource failure.
- **Global Applications:** Route 53's global coverage and latency-based routing enable you to build and deploy applications that deliver low-latency experiences to users worldwide.
- **Hybrid Cloud Environments:** Route 53 can be integrated with on-premises infrastructure and hybrid cloud environments, allowing you to manage DNS for both cloud-based and traditional resources from a single interface.

Configuring DNS records and health checks:

Let's explore how to configure DNS records and health checks in Route 53.

- **Configuring DNS Records:**
- **Configuring DNS Records:**

DNS records in Route 53 define how domain names are mapped to resources such as EC2 instances, S3 buckets, load balancers, and other AWS services. Here are some common DNS record types and their purposes:

- **A Records (Address Record):** Maps a domain name to the IPv4 address of the server hosting the domain. This is commonly used for pointing domain names to web servers or other infrastructure.



- **AAAA Records (IPv6 Address Record):** Similar to A records but used for mapping domain names to IPv6 addresses.
- **CNAME Records (Canonical Name Record):** Points a domain or subdomain to another domain's canonical name. This is often used for creating aliases for domains.
- **Alias Records:** Route 53-specific records that function similarly to CNAME records but with some additional benefits, such as support for zone apex mapping and automatic updating of IP addresses.
- **MX Records (Mail Exchange Record):** Specifies mail exchange servers for the domain, allowing you to receive email for your domain.

Here's how you can configure DNS records in Route 53:

- **Using the AWS Management Console:** Navigate to the Route 53 console, select the hosted zone for your domain, and then create or edit DNS records using the interface provided.
- **Using the AWS CLI:** You can use the AWS CLI to manage Route 53 DNS records programmatically. Commands like '`aws route53 change-resource-record-sets`' enable you to add, update, or delete DNS records in your hosted zones.
- **Using AWS SDKs:** AWS SDKs for various programming languages provide APIs for interacting with Route 53 programmatically, allowing you to automate DNS management tasks in your applications.

Configuring Health Checks:

Health checks in Route 53 allow you to monitor the health and availability of your resources, such as web servers, load balancers, and endpoints. Route 53 periodically sends health check requests to your resources and evaluates their responses to determine their health status.

Here's how you can configure health checks in Route 53:

- **Define Health Check Settings:** Specify the endpoint or resource you want to monitor, along with the protocol (HTTP, HTTPS, TCP, or HTTPS), port, and other relevant settings.
- **Set Thresholds and Intervals:** Configure the frequency and thresholds for health checks, including the number of consecutive failed checks required to consider a resource unhealthy, and the interval between checks.
- **Configure Health Checkers:** Choose the regions from which Route 53 health checkers will send requests to your resources. Distributing health checkers across multiple regions helps ensure accurate monitoring and failover capabilities.
- **Associate Health Checks with DNS Records:** Associate health checks with the DNS records that route traffic to your resources. Route 53 automatically routes traffic away from unhealthy resources based on the results of health checks.



By configuring health checks in Route 53, you can ensure the high availability and reliability of your applications by automatically redirecting traffic away from unhealthy resources and minimizing downtime.

Implementing routing policies and latency-based routing:

It allows you to optimize traffic distribution and improve the performance of your applications. Let's delve into the details: Routing Policies in Route 53:

Route 53 supports several routing policies, each designed to meet specific requirements for traffic management and failover scenarios:

1. Simple Routing Policy:

- This is the most basic routing policy where you associate a single DNS record with a single resource. When a DNS query is received, Route 53 responds with the IP address associated with the DNS record.
- Useful for directing traffic to a single resource, such as a web server or a load balancer.

2. Weighted Routing Policy:

- With weighted routing, you can distribute traffic across multiple resources based on assigned weights.
- For example, you might allocate 70% of traffic to one resource and 30% to another to perform A/B testing or gradually shift traffic during deployments.

3. Latency-Based Routing Policy:

- Latency-based routing directs traffic to the resource with the lowest network latency based on the user's geographical location.
- Route 53 measures latency from multiple locations worldwide and directs traffic to the resource that provides the best performance for each user.
- Ideal for global applications where minimizing latency is crucial for user experience.

4. Failover Routing Policy:

- Failover routing is used for creating active-passive failover configurations. You designate one resource as primary and another as standby.
- Route 53 automatically redirects traffic to the standby resource if the primary resource becomes unavailable.
- Commonly used for disaster recovery scenarios.

5. Geolocation Routing Policy:

- Geolocation routing allows you to route traffic based on the geographic location of the user.
- You can define specific routing policies for different regions or countries, ensuring users are directed to the closest or most appropriate resources.



Latency-Based Routing:

Latency-based routing is particularly powerful for optimizing the performance of globally distributed applications. Here's how it works:

1. Route 53 Health Checks:

- Route 53 continually monitors the health and performance of your resources using health checks.
- This ensures that only healthy and responsive resources are considered when calculating latency.

2. Latency Measurements:

- Route 53 measures the latency between end users and your resources from multiple AWS regions.
- It uses this information to determine the optimal resource to which traffic should be directed based on the lowest latency.

2. Traffic Distribution:

- When Route 53 receives a DNS query, it evaluates the latency to each resource and directs the query to the resource with the lowest latency for that particular user.
- This ensures that users are automatically routed to the resource that offers the best performance from their location.

Implementation:

To implement latency-based routing in Route 53:

1. Create Resource Records:

Define the DNS records for your resources (e.g., EC2 instances, ELB endpoints) in your Route 53 hosted zone.

2. Enable Latency-Based Routing:

- In the Route 53 console, create a new record set and select “Latency” as the routing policy.
- Specify the regions where your resources are located and associate each region with the corresponding DNS records.

3. Health Checks and Monitoring:

- Ensure that health checks are configured for your resources to maintain high availability and reliability.
- Monitor latency and resource health through the Route 53 console or CloudWatch metrics to identify any performance issues.

Below are examples of code snippets demonstrating how to interact with AWS Route 53 using the AWS CLI and Python SDK (boto3). We'll cover creating DNS records, configuring health checks, and implementing latency-based routing.



Conclusion:

Route 53 is a powerful tool for managing DNS and routing traffic effectively within AWS and beyond. Understanding its features and configurations is essential for building scalable and reliable web applications.

CloudFront in AWS



CloudFront

1. **Content Delivery Network (CDN):** CloudFront is a CDN service provided by AWS. CDNs help deliver content (like web pages, images, videos) to users globally with low latency by caching content at edge locations.
2. **Edge Locations:** These are server clusters located in various parts of the world. CloudFront uses these edge locations to cache and deliver content to users with lower latency.
3. **Origin Server:** This is where your original, un-cached content is stored. It could be an Amazon S3 bucket, an EC2 instance, or even an on-premises server.
4. **Distribution:** A CloudFront distribution is the configuration specifying the origin server, the edge locations for caching, and other settings. You can create two types of distributions: Web and RTMP.
5. **Web Distribution:** Used for distributing websites, including static and dynamic content.
6. **RTMP Distribution:** Designed for streaming media files using Adobe Media Server and the Real-Time Messaging Protocol (RTMP).
7. **Cache Behavior:** Defines how CloudFront handles requests and responses between users and the origin.
8. **Security:** CloudFront provides several security features, including the ability to restrict access to content, use SSL/TLS for secure connections, and integrate with AWS WAF for additional security.
9. **Logging and Monitoring:** CloudFront provides logs and real-time monitoring through Amazon CloudWatch, helping you track user requests and system performance.
10. **Cost Management:** Pricing is based on data transfer out of CloudFront to end-users, requests, and data transfer between AWS regions. Utilizing features like caching and compression can help manage costs.

Uses of CloudFront

1. **Create a Distribution:** Set up a new CloudFront distribution in the AWS Management Console.



2. **Configure Origins:** Specify the origin server where CloudFront fetches the content.
3. **Configure Behavior:** Define cache behaviors, including how CloudFront handles various types of content.
4. **Set Security Measures:** Implement security features like SSL/TLS and access control.
5. **Configure DNS:** Map your domain to the CloudFront distribution using a domain name
6. **Testing and Optimization:** Test your distribution to ensure proper functionality and consider optimizing settings based on performance requirements.

CloudFront is a powerful tool for optimizing content delivery and enhancing the performance of your web applications globally.

AWS ACM

Introduction:

In the rapidly evolving landscape of web security, securing your website with SSL/TLS certificates has become paramount. Amazon Web Services (AWS) provides a robust solution for certificate management through AWS Certificate Manager (ACM). In this blog post, we'll delve into the key features of AWS ACM, its benefits, and how it simplifies the process of obtaining, managing, and deploying SSL/TLS certificates.

Additionally, we'll explore the concepts of Public and Private Certificate Authorities (CAs) and how they contribute to the security ecosystem.

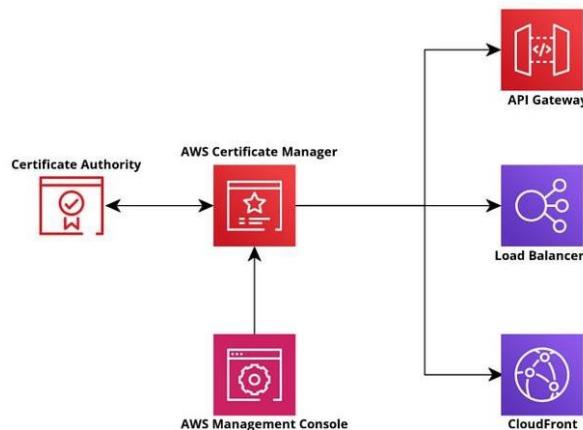
Understanding AWS ACM:

AWS Certificate Manager (ACM) is a fully managed service that makes it seamless to provision, manage, and deploy SSL/TLS certificates for your applications on AWS. ACM takes the complexity out of certificate management by automating the process of certificate renewal, validation, and deployment, allowing you to focus on building and scaling your applications.



Key Features of AWS ACM:

- **Automatic Certificate Renewal:** ACM automates the renewal process, ensuring that your certificates are always up-to-date and eliminating the risk of expiration-related disruptions. This feature is particularly beneficial for organizations managing a large number of certificates.
- **Integrated with AWS Services:** ACM seamlessly integrates with other AWS services, such as Elastic Load Balancer (ELB), CloudFront, and API Gateway. This integration simplifies the process of associating certificates with these services, reducing the time and effort required for deployment.



Global Coverage: ACM supports global deployments with certificates that can be used in multiple AWS regions. This is especially useful for businesses with a global presence, ensuring consistent security across different geographic locations.

Certificate Validation: ACM handles the validation of domain ownership automatically. This streamlines the process of obtaining certificates, saving users from the hassle of manual validation steps.

Public and Private Certificate Authorities (CAs):

Public Certificate Authorities (CAs): Public CAs, such as Let's Encrypt or Sectigo, are entities that issue SSL/TLS certificates to the public. These certificates are widely recognized by browsers, making them suitable for securing websites that need to establish trust with a broad audience.

Private Certificate Authorities (CAs): Private CAs, on the other hand, are used within a specific organization or network. They are ideal for internal communication where the trust is established within a closed environment. AWS ACM supports private CAs, allowing organizations to manage their internal certificates securely.



Benefits of Using AWS ACM:

- **Simplified Management:** ACM simplifies the traditionally complex process of certificate management. With a few clicks in the AWS Management Console or through API calls, you can request, renew, and deploy certificates effortlessly.
- **Enhanced Security:** SSL/TLS certificates play a crucial role in securing data in transit. ACM ensures that your certificates are always valid and up-to-date, reducing the risk of security breaches due to expired certificates.
- **Cost-Efficiency:** As a fully managed service, ACM eliminates the need for manual intervention in certificate management, saving time and reducing operational costs. Moreover, ACM is offered at no additional cost, making it a cost-effective solution.

Conclusion:

AWS ACM emerges as a powerful tool in the realm of certificate management, offering a seamless and secure experience for users. By automating the certificate lifecycle, integrating with various AWS services, and providing global coverage, ACM empowers businesses to prioritize application development while ensuring robust security.

Embrace the simplicity and efficiency of AWS ACM, whether you're utilizing public or private CAs, to fortify your web applications with the strength of SSL/TLS encryption.

Streamlining Mobile App Development with AWS Amplify Console



In today's fast-paced digital landscape, mobile app development can be a daunting task. However, with the right tools and strategies, it can become a seamless and efficient process. This is where AWS Amplify Console comes into play. Combining the power of AWS Amplify and AWS Mobile Hub, the Amplify Console offers a comprehensive solution for building, testing, and deploying mobile apps.

Benefits of using AWS Amplify Console for mobile app development

With AWS Amplify Console, developers can streamline the entire app development lifecycle. From code changes to continuous deployment and hosting, everything is managed in one place. This not only saves time and effort but also ensures a smooth and consistent user experience.

One of the key benefits of using AWS Amplify Console is its integration with AWS Amplify. Amplify is a set of tools and services that simplifies the process of building scalable and secure mobile applications. By leveraging Amplify Console, developers can easily connect their app to the cloud, set up authentication and authorization, and access other AWS services such as databases and storage.



Another advantage of using AWS Amplify Console is its scalability and flexibility. With its automatic branch deployments feature, developers can easily create new branches for different features or bug fixes and have them automatically deployed to separate environments. This allows for easy experimentation and iteration, ensuring that the app development process remains agile and efficient.

Furthermore, AWS Amplify Console provides a simple and intuitive user interface that makes it easy for developers to manage their app's deployment and hosting. With just a few clicks, developers can configure their app's settings, set up custom domains, and monitor the deployment process. This eliminates the need for complex manual configurations and reduces the risk of human error.

Key features of AWS Amplify Console

AWS Amplify Console is packed with powerful features that make it an essential tool for mobile app development. Here are some of its key features:

- **Continuous deployment:** AWS Amplify Console allows developers to set up automated deployments for their app. Whenever changes are pushed to the repository, Amplify Console automatically builds and deploys the updated app, ensuring a smooth deployment process.
- **Environment variables:** With Amplify Console, developers can easily manage environment variables for different stages of their app's development. This allows for easy configuration of variables such as API endpoints, database credentials, and third-party integrations.
- **Branch deployments:** Amplify Console enables developers to create separate branches for different features or bug fixes. Each branch can have its own environment and deployment settings, allowing for easy testing and experimentation.
- **Custom domains:** Developers can easily set up custom domains for their app with Amplify Console. This gives the app a professional and branded look, enhancing user trust and engagement.
- **Automatic SSL certificates:** Amplify Console automatically provisions and manages SSL certificates for custom domains, ensuring secure communication between the app and its users.

Setting up AWS Amplify Console for mobile app development

Getting started with AWS Amplify Console is quick and easy. Here's a step-by-step guide to setting it up for your mobile app development:

- **Create an AWS account:** If you don't already have one, sign up for an AWS account at aws.amazon.com. This will give you access to all the AWS services, including Amplify Console.
- **Install the Amplify CLI:** The Amplify CLI is a command-line tool that helps you create and manage your app's backend resources. Install it by running the following command in your terminal: `npm install -g @aws-amplify/cli`.



- **Initialize your app:** Navigate to your app's root directory and run the command amplify init. This will initialize your app with Amplify and create a new Amplify environment.
- **Connect your app to the cloud:** Once your app is initialized, you can start connecting it to the cloud. Use the Amplify CLI commands to add backend services such as authentication, storage, and databases.
- **Configure Amplify Console:** After setting up the backend, run the command amplify console to open the Amplify Console in your browser. Here, you can configure your app's deployment settings, custom domains, and environment variables.
- **Deploy your app:** Finally, use the Amplify CLI command amplify push to deploy your app to the Amplify Console. This will build your app, create the necessary resources, and deploy it to the specified environment.

Integrating AWS Amplify Console with your mobile app development workflow

AWS Amplify Console seamlessly integrates with popular development workflows, making it easy for developers to incorporate it into their existing processes. Here are a few ways you can integrate Amplify Console with your mobile app development workflow:

- **Version control integration:** Amplify Console supports integration with popular version control systems like GitHub, GitLab, and Bitbucket. This allows you to automatically build and deploy your app whenever changes are pushed to your repository.
- **Build hooks:** Amplify Console provides build hooks that can be used to trigger custom build scripts or external services. This enables you to incorporate additional build steps or automated testing into your app's deployment pipeline.
- **Webhooks:** Amplify Console can also send webhooks to external services, enabling you to trigger custom actions or notifications based on the app's deployment status. This can be useful for sending notifications to team members or integrating with other tools in your development workflow.
- **API integration:** Amplify Console provides a RESTful API that allows you to programmatically manage your app's deployments and settings. This enables you to automate certain tasks or integrate Amplify Console with other tools in your development workflow.

Streamlining the deployment process with AWS Amplify Console

One of the biggest challenges in mobile app development is the deployment process. Traditional deployment methods often involve manual configurations, complex build scripts, and potential human errors. However, with AWS Amplify Console, deploying your app becomes a breeze.

Amplify Console simplifies the deployment process by automating key tasks and providing an intuitive user interface. Here's how it streamlines the deployment process:

- **Continuous deployment:** With Amplify Console, every code change triggers an automated deployment. This means that as soon as you push changes to your repository, Amplify Console automatically builds and deploys the updated app. This eliminates the need for manual deployments and reduces the risk of human error.



- **Automatic branch deployments:** Amplify Console allows you to create separate branches for different features or bug fixes. Each branch can have its own environment and deployment settings. This enables you to test and iterate on new features without affecting the main production environment.
- **Preview deployments:** Amplify Console provides a preview URL for each deployment, allowing you to easily preview and test your app before making it live. This is particularly useful for testing new features or bug fixes in a controlled environment.
- **Rollback feature:** In case of any issues or bugs in a deployment, Amplify Console allows you to easily rollback to a previous version with just a few clicks. This ensures that you can quickly revert to a stable version of your app without any downtime.

Optimizing mobile app performance with AWS Amplify Console

Performance is a critical aspect of mobile app development. Users expect apps to be fast, responsive, and reliable. AWS Amplify Console provides several features and optimizations that can help you optimize your app's performance:

- **Content delivery network (CDN):** Amplify Console automatically deploys your app to a global CDN, ensuring that your app's static assets are served from the closest edge location. This reduces latency and improves the app's overall performance.
- **Automatic asset optimization:** Amplify Console automatically optimizes your app's static assets, including images, CSS, and JavaScript files. This reduces the file size of these assets, resulting in faster load times and better user experience.
- **GZIP compression:** Amplify Console automatically enables GZIP compression for your app's assets, reducing the size of transferred data and improving network performance.
- **Cache control:** Amplify Console allows you to configure cache control headers for your app's assets. This enables you to control how long assets are cached by the user's browser, reducing the number of requests made to the server and improving performance.

Monitoring and troubleshooting mobile app development with AWS Amplify Console

Monitoring and troubleshooting are essential aspects of mobile app development. AWS Amplify Console provides several tools and features that help you monitor and troubleshoot your app's development process:

1. **Deployment logs:** Amplify Console provides detailed deployment logs that allow you to track the progress of your app's deployment. These logs include information about build times, errors, and warnings, enabling you to quickly identify and fix any issues.
2. **Real-time metrics:** Amplify Console provides real-time metrics for your app's deployments, including build times, deployment durations, and success rates. These metrics help you monitor the performance of your app's deployment process and identify any bottlenecks or issues.



3. **Alerts and notifications:** Amplify Console allows you to set up alerts and notifications for your app's deployments. You can configure alerts based on criteria such as deployment failures, long build times, or high error rates. This enables you to proactively monitor your app's development process and take immediate action when necessary.
4. **Integration with AWS CloudWatch:** Amplify Console integrates seamlessly with AWS CloudWatch, allowing you to collect and analyze logs, metrics, and events from your app's deployments. This provides deeper insights into your app's performance and helps you troubleshoot any issues.

Case studies: Success stories of mobile app development with AWS Amplify Console

AWS Amplify Console has been used by numerous organizations to streamline their mobile app development process. Here are a couple of success stories:

1. **Company A:** Company A, a fast-growing startup, used AWS Amplify Console to build and deploy their mobile app. By leveraging Amplify Console's continuous deployment and automatic branch deployments features, they were able to rapidly iterate on new features and bug fixes. This allowed them to launch their app in record time and achieve a high level of user satisfaction.
2. **Company B:** Company B, a large enterprise, used AWS Amplify Console to simplify their complex mobile app development workflow. With Amplify Console's environment variables and integration with version control systems, they were able to automate their deployment process and reduce the risk of human error. This resulted in significant time and cost savings for the company.

These success stories highlight the effectiveness of AWS Amplify Console in streamlining mobile app development and enabling organizations to deliver high-quality apps in a timely manner.

Conclusion: Streamlining mobile app development with AWS Amplify Console

In conclusion, AWS Amplify Console is revolutionizing the way mobile apps are developed. Its powerful features and seamless integration with other AWS services make it a must-have tool for any app developer.

With Amplify Console, developers can streamline the entire app development lifecycle, from code changes to continuous deployment and hosting. Its scalability and flexibility enable easy adaptation and iteration of apps, making it ideal for both small startups and large enterprises.

Furthermore, Amplify Console's optimization and monitoring features help developers optimize their app's performance and troubleshoot any issues. With real-time metrics, detailed deployment logs, and integration with AWS CloudWatch, developers can ensure that their app is performing at its best.

So why not give AWS Amplify Console a try and experience the convenience and efficiency it brings to your mobile app development process? Streamline your workflow, deliver high-quality apps, and stay ahead in the fast-paced digital landscape.



AWS Lambda

Serverless Architecture:

The advancement of technology has generated new needs. The increasing demand, load and costs have accelerated the development of new methods. In addition, the development of cloud technology and innovations have brought new services and concepts into our lives. One of these concepts is serverless architecture.

While developing, our primary goal is to create a structure that will solve a problem. However, in doing so, we are also forced to consider other things. We have to think about the server configuration where the application will run, as well as authorization, load balancing, and many other aspects. Serverless architecture (another term used in place of “serverless” is “Functions as a Service”) is a design approach that enables you to build and run applications and services without the need to manage the infrastructure.

Serverless architecture is not a way of assuming that servers are no longer required or that applications will not run on servers. Instead, it is a pattern or approach that helps us think less about servers in the context of software development and management. This approach allows us to eliminate the need to worry about issues related to scaling, load balancing, server configurations, error management, deployment, and runtime. With serverless architecture, we are essentially outsourcing one of the most challenging aspects of running a software in production, which is managing operational tasks.

Every technology has its own drawbacks, and serverless is no exception. Here are the main situations in which it is generally not recommended to use serverless architecture:

- A “cold start” is a phenomenon that can occur when a serverless platform is required to initiate internal resources in order to handle a function request. This process can take some time and may result in slower performance for the initial request. To avoid this issue, it is possible to keep the function in an active state by sending periodic requests to it. This helps ensure that the necessary resources are already initialized and ready to handle incoming requests efficiently.
- Long-running workloads may be more expensive to run on serverless platforms compared to using a dedicated server, which can be more efficient in these cases. When deciding between these options, it is crucial to carefully consider the specific needs and requirements of the workload.
- Testing and debugging code in a serverless computing environment can be challenging due to the nature of these cloud systems and the lack of back-end visibility for developers.
- As a serverless application that relies on external vendors for back-end services, it is natural to have a certain level of reliance on those vendors. However, if you decide to switch vendors at any point, it can be challenging to reconfigure your serverless architecture to accommodate the new vendor’s features and workflows.

- Due to time limitations imposed by the vendor (for example, AWS allows up to 15 minutes), it is not possible to perform long-running tasks.

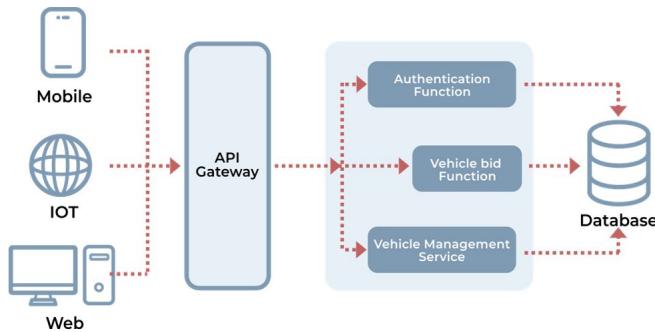


Figure 1 -Reference Architecture Serverless with Microservices

What benefits does serverless provide?

- Serverless computing runs on servers managed by cloud service providers, eliminating the need for users to manage the underlying infrastructure themselves. This allows organizations to focus on developing and deploying their applications without worrying about server management.
- Serverless architecture allows for applications to be scaled automatically. This means that as demand for the application increases, the necessary resources will be automatically allocated to meet that demand, without the need for manual intervention. This can provide a high degree of flexibility and scalability for organizations using serverless architectures for their applications.
- Serverless architectures enable the creation of development environments that are easier to set up, which can lead to faster delivery and more rapid deployment of applications.
- When using serverless services, you only pay for the specific instances or invocations you use rather than being charged for idle servers or virtual machines that you may not be utilizing.

Use Cases

- Serverless computing is well suited for tasks that are triggered by an event. If you have an event that needs to be run based on some trigger, serverless architecture can be an effective solution. An example of this is when a user signs up for a service on a website and a welcome email is automatically sent in response.
- Serverless computing allows for the creation of RESTful APIs that can be easily scaled as needed.



Serverless computing is a relatively new technology; it has advantages and disadvantages. However, it is not a suitable solution for every situation, and it is important to carefully consider all infrastructure requirements before deciding to use it as your execution model. If you currently host small functions on your own servers or virtual servers, it may be beneficial for you to consider the benefits of using a serverless computing solution.

There are a variety of platforms that offer a range of services for serverless architecture. One such platform is Amazon Web Services, which offers a number of serverless services. AWS provides AWS

Lambda, AWS Fargate for computing; Amazon EventBridge, AWS Step Functions, Amazon SQS, Amazon SNS, Amazon API Gateway, AWS AppSync for application integration; and Amazon S3, Amazon EFS, Amazon DynamoDB, Amazon RDS Proxy, Amazon Aurora Serverless, Amazon Redshift Serverless, Amazon Neptune serverless for data store.

I will now provide an explanation of one of the most widely utilized and practical services among these options, which is AWS Lambda.

AWS Lambda

AWS Lambda is an event-driven cloud service from Amazon Web Services (AWS) that enables users to execute their own code, known as “functions,” without the need to worry about the underlying infrastructure. These functions can be written in various programming languages and runtimes supported by AWS Lambda and be uploaded to the service for execution.

AWS Lambda automatically manages the scaling and allocation of resources for these functions, providing a convenient and efficient way to run code in the cloud.

AWS Lambda functions can be used to perform a wide range of computing tasks, such as serving web pages, processing streams of data, calling APIs, and integrating with other AWS services. These functions are designed to be flexible and can be used for a variety of purposes, making them a powerful tool for cloud computing.

What is the process behind AWS Lambda's functionality?

Lambda functions are run in their own isolated containers. When a new function is created, it is packaged into a container by Lambda and then run on a cluster of machines managed by AWS, which can serve multiple tenants. Before the functions begin execution, the required RAM and CPU capacity is allocated to each function's container. Once the functions have completed execution, the RAM allocated at the start is multiplied by the duration of the function's execution. Customers are charged based on the amount of allocated memory and the run time required for the function to complete.

AWS manages the entire infrastructure layer of AWS Lambda, so customers do not have visibility into how the system operates. However, this also means that customers do not need to worry about tasks, such as updating the underlying machines or managing network contention, as these responsibilities are handled by AWS.

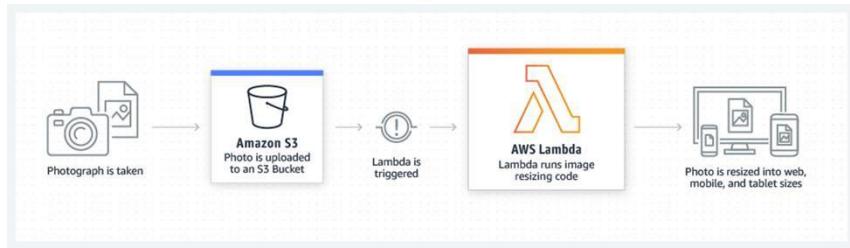


Figure 2 - Reference Architecture: Image File Processing

In this reference architecture, we can use AWS Lambda to create the thumbnails automatically. Lambda will be triggered by S3 Bucket events, and then Lambda will generate thumbnail.

What are the capabilities of Lambda?

AWS Lambda provides native support for Java, Go, PowerShell, Node.js, C#, Python, and Ruby, and offers a runtime API that enables the use of additional programming languages to write functions.

We can create web applications, mobile back-ends, and IoT back-ends by combining Lambda with other serverless components. AWS Lambda can be utilized to perform data transformation tasks, such as validation, filtering, sorting, or other processes, for every data change in a DynamoDB table, and load the transformed data to another data store.

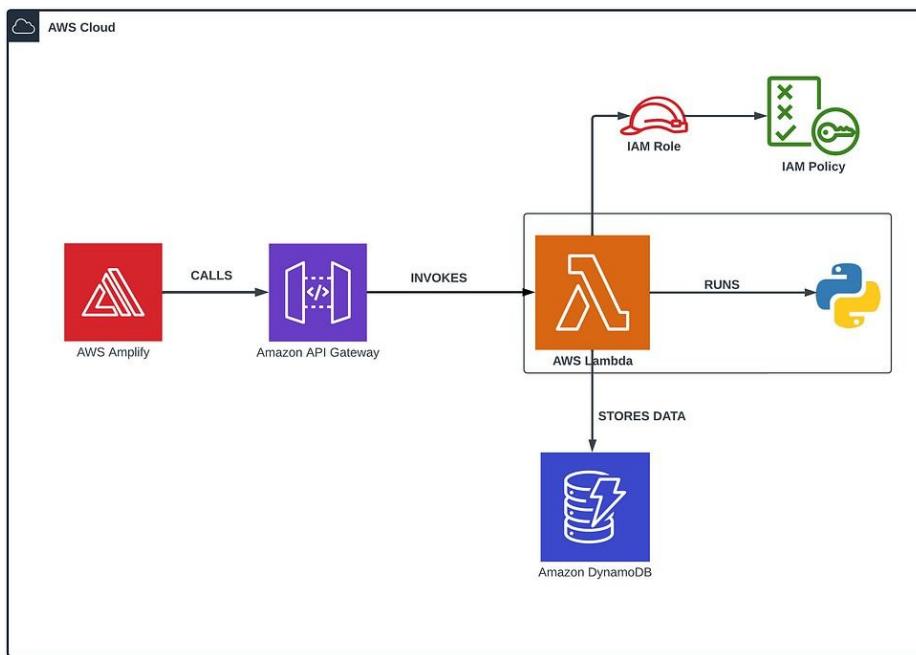
Scalable APIs, when building APIs using AWS Lambda, each execution of a Lambda function can serve a single HTTP request. The API's different components can be routed to different Lambda functions through Amazon API Gateway. AWS Lambda automatically scales individual functions based on demand, enabling different parts of the API to scale differently according to current usage levels. This enables cost-effective and flexible API set-ups.

AWS Lambda has few restrictions

- A Lambda function will end execution after 15 minutes. It is not possible to alter this limit. If your function typically takes more than 15 minutes to run, AWS Lambda may not be a suitable option for your task.
- The amount of memory that can be allocated to a Lambda function ranges from 128MB to 3,008MB in 64MB increments.
- The size of the zipped Lambda code package should not exceed 50MB, and the unzipped version should not be larger than 250MB.
- By default, the concurrent execution for all AWS Lambda functions within a single AWS account are limited to 1,000.



Serverless Web App Development Made Easy: A Complete Guide with AWS Amplify, DynamoDB, Lambda and API Gateway



Get ready to dive into the world of serverless web application development on AWS. In this series, we'll guide you through the process of creating a dynamic web app that calculates the area of a rectangle based on user-provided length and width values. We'll leverage the power of AWS Amplify for web hosting, AWS Lambda functions for real-time calculations, DynamoDB for storing and retrieving results, and API Gateway for seamless communication. By the end of this journey, you'll have the skills to build a responsive and scalable solution that showcases the true potential of serverless architecture. Let's embark on this development adventure together!

Prerequisites

- Have an AWS account. If you don't have one, sign up [here](#) and enjoy the benefits of the [Free-Tier Account](#)
- Access to the project files: [Amplify Web-app](#)

Creating the Front-end

- Use the index.html file from the project files. Or simply open a text editor and copy the following code into an index.html file. Note the part with "YOUR API URL" as we will be filling this part with the API URL later



```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Rectangle</title>
<!-- Styling for the client UI -->
<style>
h1 {
    color: #FFFFFF;
    font-family: system-ui;
    margin-left: 20px;
}
body {
    background-color: #222629;
}
label {
    color: #86C232;
    font-family: system-ui;
    font-size: 20px;
    margin-left: 20px;
    margin-top: 20px;
}
button {
    background-color: #86C232;
    border-color: #86C232;
    color: #FFFFFF;
    font-family: system-ui;
    font-size: 20px;
    font-weight: bold;
    margin-left: 30px;
    margin-top: 20px;
    width: 140px;
}
input {
    color: #222629;
    font-family: system-ui;
```

```
font-size: 20px;  
margin-left: 10px;  
margin-top: 20px;  
width: 100px;  
}  
</style>  
<script>  
// callAPI function that takes the length and width numbers as parameters  
var callAPI = (length,width)=>{  
    // instantiate a headers object var myHeaders = new Headers();  
    // add content type header to object myHeaders.append("Content-Type",  
“application/json”);  
    // using built in JSON utility package turn object to string and store in a  
variable  
    var raw = JSON.stringify({“length”:length,”width”:width});  
    // create a JSON object with parameters for API call and store in a  
variable  
    var requestOptions = { method: ‘POST’, headers: myHeaders, body: raw, redirect:  
‘follow’  
};  
    // make API call with parameters and use promises to get response  
fetch(“YOUR API URL”, requestOptions)  
    .then(response => response.text())  
    .then(result => alert(JSON.parse(result).body))  
    .catch(error => console.log(‘error’, error));  
}  
</script>  
</head>  
<body>  
    <h1>AREA OF A RECTANGLE!</h1>  
    <form>  
        <label>Length:</label>  
        <input type="text" id="length">  
        <label>Width:</label>  
        <input type="text" id="width">
```



```
<!-- set button onClick method to call function we defined passing input values as parameters
-->
<button type="button"
    onclick="callAPI(document.getElementById('length').value,document.getElementById('width').value)">CALCULATE</button>
</form>
</body>
</html>
```

2. The file should look like this when opened on a browser. It gives spaces to input the length and width of a rectangle and a ‘Calculate’ button.

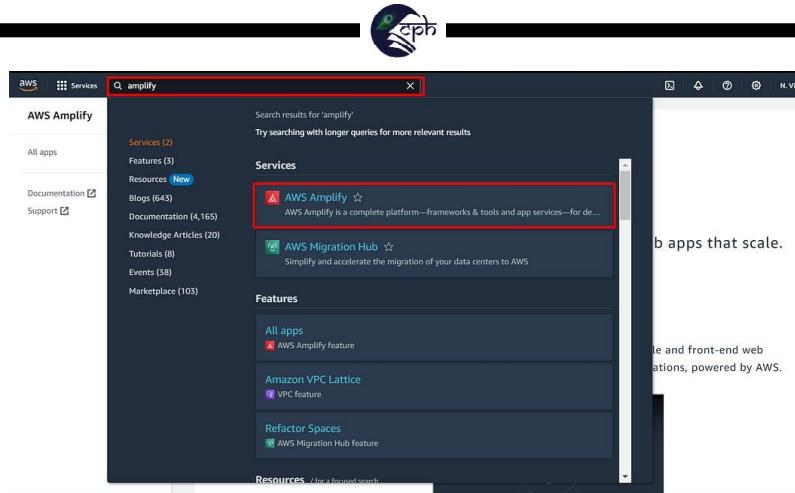
The screenshot shows a dark-themed web page with a light gray header bar. The header bar contains the text "AREA OF A RECTANGLE!". Below the header, there is a form with two input fields: "Length:" and "Width:". Both input fields have a placeholder value of "0.00". To the right of the input fields is a green button labeled "CALCULATE".

Hosting the App on AWS Amplify

- On your AWS console search box, search for ‘Amplify’ and click on the first option that appears.
- 2. Click on ‘GET STARTED’
- 3. Select ‘Get Started’ on the Amplify Hosting side
- 4. select the source for your app files. They can be in a remote repository or local. We will use ‘Deploy without Git provider’ since our files are local. We also need to use a compressed folder with our files. Click on ‘Continue’
- 5. Give the app a name, an environment name, choose the method as ‘Drag and drop’ and select the index.zip file (zip all the app files. In this case, it is only the index.html file). Click on ‘Save and deploy’
- 6. Once the deployment is complete, click on the Domain to access your app
- 7. The app opens on the browser. (You might need to refresh the deployment page on Amplify. Maybe it’s a bug or something %)

Creating a Lambda Function to do the Math

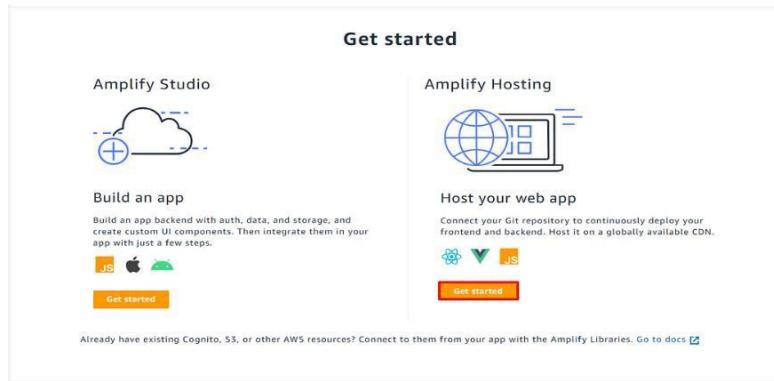
1. On the AWS console search bar, type ‘Lambda’ and select the Lambda service



- Click on 'Create function'



- Select 'Get Started' on the Amplify Hosting side



- select the source for your app files. They can be in a remote repository or local. We will use 'Deploy without Git provider' since our files are local. We also need to use a compressed folder with our files. Click on 'Continue'



Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

GitHub

Bitbucket

GitLab

AWS CodeCommit

Deploy without Git provider

Continue

5. Give the app a name, an environment name, choose the method as 'Drag and drop' and select the index.zip file (zip all the app files. In this case, it is only the index.html file). Click on 'Save and deploy'

Start a manual deployment

App name
Give this a name, or we will generate a default for you
rectangle

Environment name
Give this resource a meaningful environment name, like dev, test, or prod, or we will generate a default for you
dev

Method
 Drag and drop
 Amazon S3
 Any URL

index.zip

Save and deploy

6. Once the deployment is complete, click on the Domain to access your app

This tab lists all connected branches, select a branch to view build details.

Add environment

dev

Deployment successfully completed.
100%

Domain
<https://dev.d556nbx5khrz60.amplifyapp.com>

Last deployment
10/12/2023, 3:46:50 PM

Drag and drop your project's build output directory or zip file here to update your app, or, choose another method.

Choose files

7. The app opens on the browser. (You might need to refresh the deployment page on Amplify. Maybe it's a bug or something ?

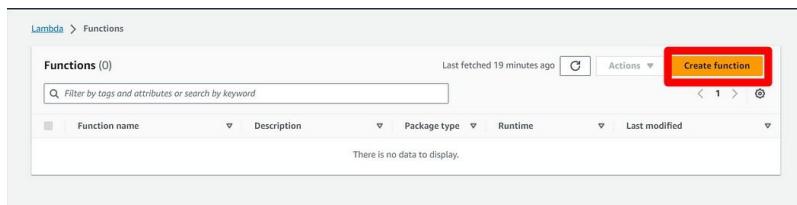


Creating a Lambda Function to do the Math

1. On the AWS console search bar, type 'Lambda' and select the Lambda service



2. Click on 'Create function'



3. Give the Function name, The Runtime(Latest Python), then scroll down and click on 'Create function'

Create function Info
AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Contain
Select a

Basic information

Function name Required. The name must be unique across all of your functions.

Runtime Info
Choose the language you want to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

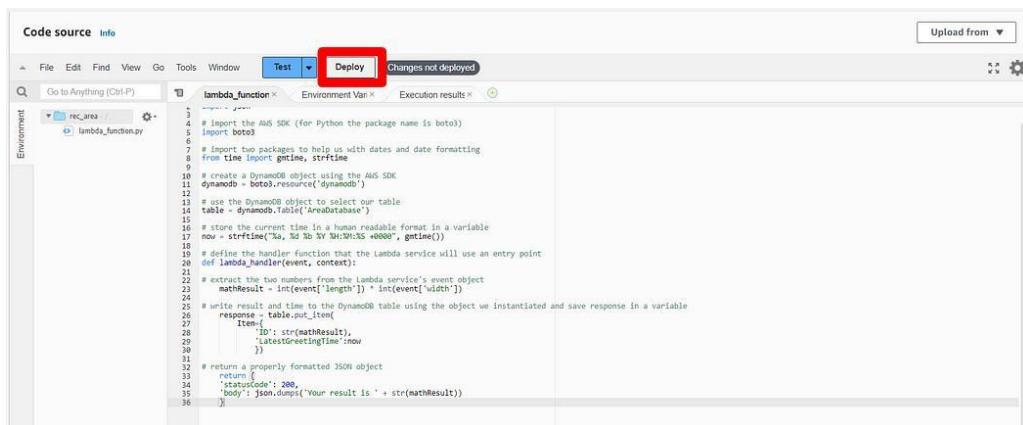
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.



4. Copy the following Lambda function onto your lambda_function.py file. Please note the DynamoDB name. We will be using this name later as we create the DB.

```
# import the JSON utility package import json
# import the AWS SDK (for Python the package name is boto3) import boto3
# import two packages to help us with dates and date formatting from time import gmtime,
# strftime
# create a DynamoDB object using the AWS SDK dynamodb = boto3.resource ('dynamodb')
# use the DynamoDB object to select our table table = dynamodb.Table ('AreaDatabase')
# store the current time in a human readable format in a variable now = strftime("%a, %d
# %b %Y %H:%M:%S +0000", gmtime())
# define the handler function that the Lambda service will use an entry point def
lambda_handler(event, context):
# extract the two numbers from the Lambda service's event object Area = int(event['length'])
* int(event['width'])
# write result and time to the DynamoDB table using the object we instantiated and save
response in a variable
response = table.put_item( Item={
'ID': str(Area), 'LatestGreetingTime':now
})
# return a properly formatted JSON object return {
'statusCode': 200,
'body': json.dumps('Your result is ' + str(Area))
}
```

5. Click on 'Deploy'





Create an API Gateway

On the AWS services search box, enter ‘API’ and select ‘API Gateway’ that appears

The screenshot shows the AWS Management Console search interface. A red box highlights the search bar at the top containing 'api'. Below it, the 'Services' section is expanded, showing various services like Amplify, CloudTrail, Amazon Location Service, and AWS Cloud Map. A large red box highlights the 'API Gateway' service card, which is described as 'Build, Deploy and Manage APIs'. Other services listed include Documentation, CloudWatch Metrics, and AWS AppSync.

2. In the list for ‘Choose an API type’, select ‘Build’ for ‘REST API’

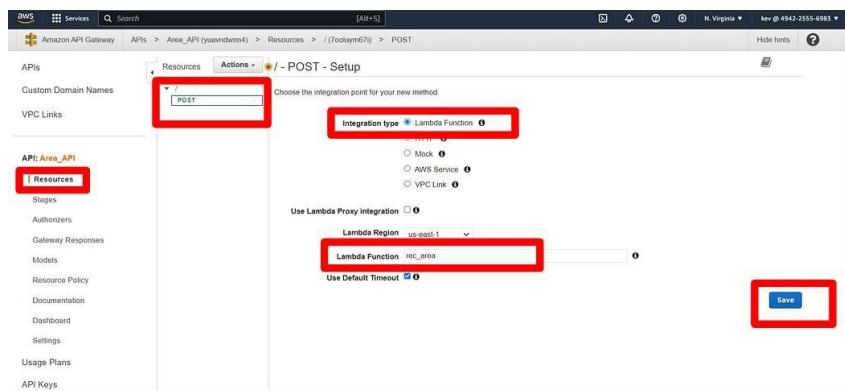
The screenshot shows the 'Choose an API type' dialog. It lists three options: 'HTTP API', 'WebSocket API', and 'REST API'. The 'REST API' section is expanded, showing its description: 'Develop a REST API where you gain complete control over the request and response along with API management capabilities.' It also lists 'Lambda, HTTP, AWS Services' as works with. At the bottom of this section, there are 'Import' and 'Build' buttons, with 'Build' being highlighted by a red box.

3. Choose the ‘REST’ protocol for the API, select ‘New API’ under ‘Create new API’ and give the API a name, then click on ‘Create API’

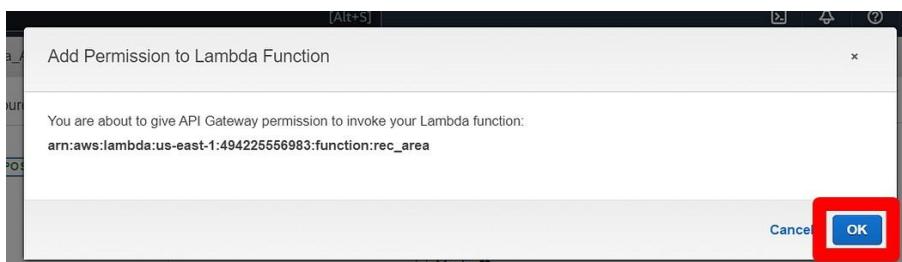
The screenshot shows the 'Create new API' dialog. It starts with 'Choose the protocol' step, where 'REST' is selected (highlighted by a red box). It then moves to 'Create new API' step, where 'New API' is selected (highlighted by a red box). Finally, it reaches the 'Settings' step, where the 'API name*' field is filled with 'Area_API' (highlighted by a red box). The 'Create API' button at the bottom right is also highlighted by a red box.



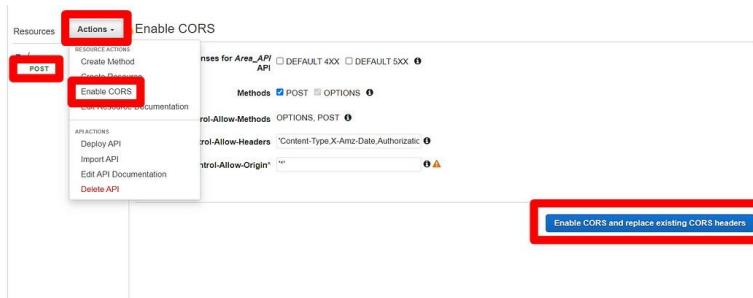
4. On the page that appears, select ‘Resources’ on the Left Panel, On the ‘Actions’ drop-down, select ‘Create method’. Select ‘POST’ on the drop down that appears then click on the —’. Select ‘Lambda Function’ as the Integration type and type the name of the lambda function in the ‘Lambda Function’ box. Click on ‘Save’



5. On the dialog box that appears to Add Permission to Lambda Function, click ‘OK’

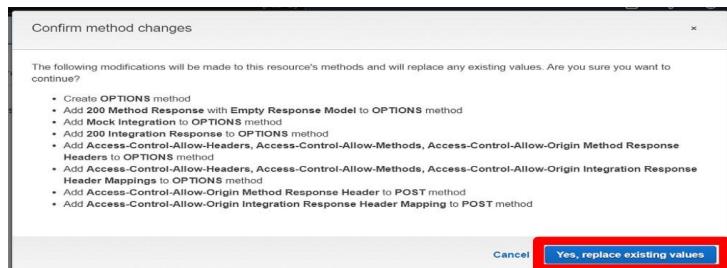


6. Select ‘POST’. On the ‘Actions’ drop down, click on ‘Enable CORS/ then click on ‘Enable CORS and replace existing CORS headers’ on the bottom right

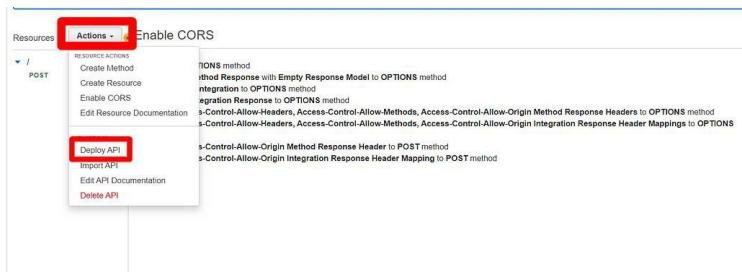




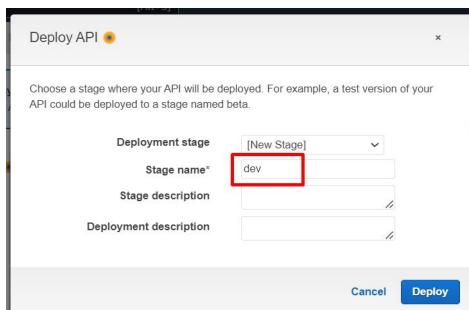
7. On the ‘Confirm method changes’ box that appears, click on ‘Yes, replace existing values’



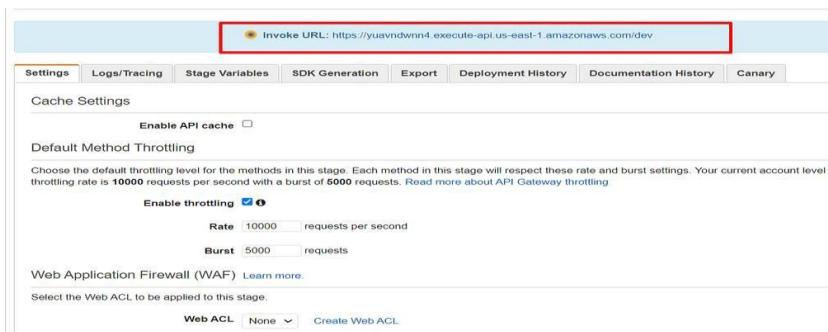
8. Once all the checks are complete, click on ‘Actions’, then, ‘Deploy API’



9. Give the ‘Stage name’, then click ‘Deploy’



10. The Invoke URL is what you replace “YOUR API URL” with on the index.html file. Insert the URL, regenerate the index.zip and reupload to Amplify





Invoke URL: <https://yuavndwnn4.execute-api.us-east-1.amazonaws.com/dev>

Setting up a Database on DynamoDB to store results On the services search box, search for ‘DynamoDB’ and select the DynamoDB service

The screenshot shows the AWS Services search interface. The search bar at the top contains the text 'dynamoDB'. Below the search bar, there is a sidebar with various links like 'Amazon API Gateway', 'Amazon Domain Name Service', 'Links', 'Area API', 'Resources', 'Pages', 'Authorizers', 'Gateway Responses', 'Models', 'Resource Policy', 'Documentation', 'Dashboard', and 'Settings'. The main area is titled 'Services (4)' and shows a list of services: 'DynamoDB' (highlighted with a red box), 'Athena', 'CloudFront', and 'AWS Cloud Map'. Each service entry includes a star icon and a brief description. At the bottom of the main area, there is a link 'See all 19 results ▶'.

2. Click on ‘Create table’

The screenshot shows the 'Amazon DynamoDB' landing page. The main heading is 'Amazon DynamoDB: A fast and flexible NoSQL database service for any scale'. Below the heading, there is a section titled 'How it works' with a link 'What is Amazon DynamoDB?'. To the right, there is a 'Get started' section with a 'Create table' button (highlighted with a red box) and a 'Pricing' section.

3. Give the table a name, for ‘Partition key’ input ‘ID’. Leave the rest as default, scroll to the bottom and click on ‘Create table’

The screenshot shows the 'Create table' form. The 'Table name' field is filled with 'Area_table' and highlighted with a red box. The 'Partition key' section shows a dropdown menu with 'ID' selected. The 'Sort key - optional' section is empty. At the bottom of the form, there is a note: '1 to 255 characters and case sensitive.' and a 'Create table' button.



4. Select the table name. Under the overview tab, expand ‘Additional info’, then take note of the ARN

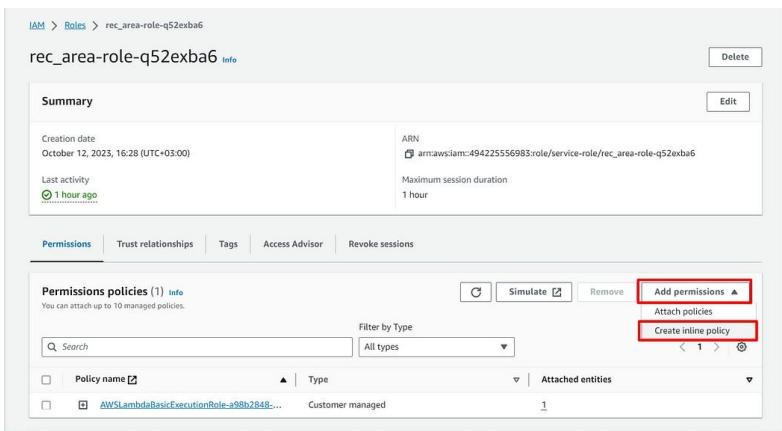
The screenshot shows the 'General information' and 'Additional info' sections of the DynamoDB table 'Area_table'. The 'Additional info' section includes fields for Table class (DynamoDB Standard), Indexes (0 globals, 0 locals), DynamoDB stream (Off), Time to Live (TTL) (Off), Replication Regions (0 Regions), Encryption (Owned by Amazon), Date created (October 12, 2023, 23:59:00 UTC+03:00), and Deletion protection (Off). The 'Amazon Resource Name (ARN)' field is highlighted with a red box and contains the value: arn:aws:dynamodb:us-east-1:494225556983:table/Area_table.

arn:aws:dynamodb:us-east-1:494225556983:table/Area_table

5. Let’s add permissions to our Lambda function to access DynamoDB. On the Lambda function window, select the ‘Configuration’ tab, then ‘Permissions’ on the left side panel and select the Role name.

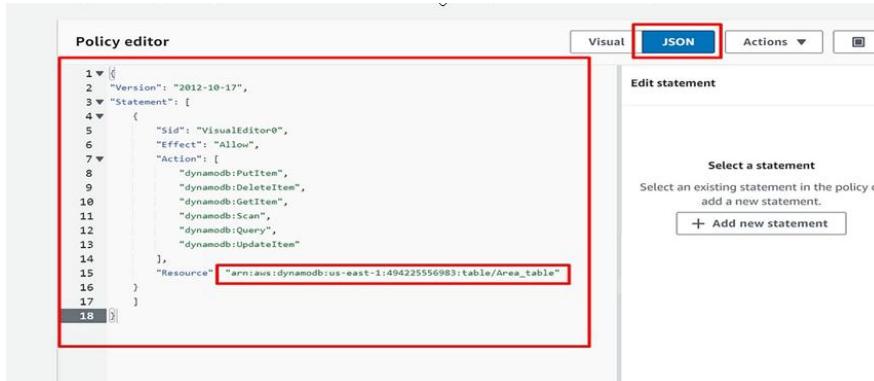
The screenshot shows the 'Function overview' and 'Configuration' tabs for the Lambda function 'rec_area'. The 'Configuration' tab is selected. In the sidebar, the 'Permissions' tab is highlighted with a red box. In the main area, under 'Execution role', the 'Role name' field is highlighted with a red box and contains the value: rec_area-role-q52exba6. Other visible fields include 'Description' (empty), 'Last modified' (8 hours ago), 'Function ARN' (arn:aws:lambda:us-east-1:494225556983:function:rec_area), and 'Function URL' (empty).

6. A new tab opens in IAM and we can add permissions to the role. Click on ‘Add permissions’, then ‘Create inline policy’



The screenshot shows the AWS IAM 'Roles' page with a selected role named 'rec_area-role-q52exba6'. The 'Permissions' tab is active, displaying a list of managed policies attached to the role. One policy, 'AWSLambdaBasicExecutionRole-a98b2848...', is highlighted with a red box. On the right side of the permissions list, there are buttons for 'Add permissions', 'Attach policies', and 'Create inline policy', also all highlighted with red boxes.

7. Select the JSON Tab and copy the following policy. Replace “YOUR- TABLE-ARN” with the ARN of your table that we copied in step 4, then click ‘Next’ at the bottom



The screenshot shows the AWS Policy Editor interface with the 'JSON' tab selected. A large red box highlights the JSON policy code. The policy is defined as follows:

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "VisualEditor0",
6        "Effect": "Allow",
7        "Action": [
8          "dynamodb:PutItem",
9          "dynamodb>DeleteItem",
10         "dynamodb:GetItem",
11         "dynamodb:Scan",
12         "dynamodb:Query",
13         "dynamodb:UpdateItem"
14       ],
15       "Resource": "arn:aws:dynamodb:us-east-1:494225556983:table/Area_table"
16     }
17   ]
18 }

```

```
{
"Version": "2012-10-17",
"Statement": [
{
  "Sid": "VisualEditor0",
  "Effect": "Allow",
  "Action": [
    "dynamodb:PutItem",
    "dynamodb>DeleteItem",
    "dynamodb:GetItem",
    "dynamodb:Scan",
    "dynamodb:Query",
    "dynamodb:UpdateItem"
  ],
  "Resource": "arn:aws:dynamodb:us-east-1:494225556983:table/Area_table"
}
]
```



```
"dynamodb:UpdateItem"
];
"Resource": "YOUR-TABLE-ARN"
}
]
```

8. On the ‘Review and create’ page, give the policy a name the click on ‘Create policy’ at the bottom of the page

52edba6 > Create policy

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name Enter a meaningful name to identify this policy.
AreaDynamoPolicy

Maximum 128 characters. Use alphanumeric and '+', '.', '_', '-' characters.

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Allow (1 of 384 services)

Service	Access level	Resource	Request condition
DynamoDB	Limited: Read, Write	TableName string like [area_table, region string like [us-east-1]	None

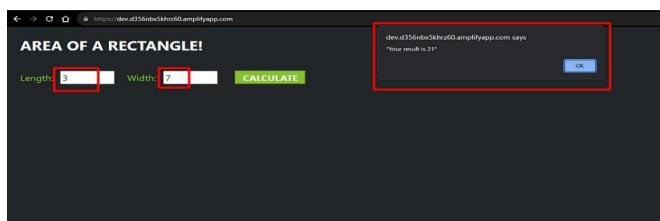
9. Now the Lambda function has permissions to write to the DB

Testing

Now that we are done, let's see what we have. Open the AWS Amplify domain. It should open our app.



2. Input values for the Length and Width and click on “Calculate”. The solution should pop up on the screen. (Returned in the browser through API Gateway)



3. Yaaaay!!!! And we are successful



Delete your Resources

Remember to delete your resources to avoid unnecessary charges: Delete the Amplify App

Delete the DynamoDB Table

Delete the Lambda function

Delete the API Gateway

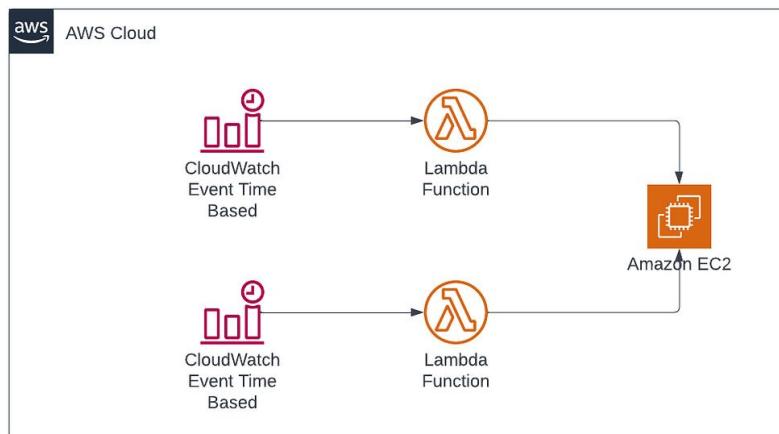
Conclusion

In this comprehensive guide, we've embarked on an exciting journey into the realm of serverless web application development on AWS. We've built a dynamic web app that calculates the area of a rectangle based on user-provided length and width values. Leveraging the power of AWS Amplify for web hosting, AWS Lambda functions for real-time calculations, DynamoDB for result storage, and API Gateway for seamless communication, we've demonstrated the incredible potential of serverless architecture.

Serverless EC2 Instance Scheduler for Company Working Hours

Scenario:

In some companies, there is no need to run their EC2 instances 24/7; they require instances to operate during specific time periods, such as company working hours, from 8:00 AM in the morning to 5:00 PM in the evening. To address this scenario, I will implement two Lambda functions responsible for starting and stopping instances. These Lambda functions will be triggered by two CloudWatch Events in the morning and evening. This solution is fully serverless.



Steps:

Step 1 :Creating the Instance :

- Navigate to the EC2 Console.



- Follow the Outlined steps below.

The screenshot displays three sequential steps for launching an Amazon Linux 2023 AMI instance on AWS. Each step shows the configuration of the instance type, network settings, and storage.

Step 1: Select Instance Type

The first step shows the "Instances" section of the AWS EC2 dashboard. A modal window titled "Select an instance" is open, showing the "Amazon Machine Image (AMI)" section. It lists the "Amazon Linux 2023 AMI" (ami-02ab7db191b50f6bb) as the selected AMI. The "Instance type" dropdown is set to "t2.micro". The "Launch instance" button is highlighted in orange at the bottom right of the modal.

Step 2: Configure Network Settings

The second step shows the "Network settings" configuration screen. It includes fields for "Subnet" (set to "No preference (Default subnet in any availability zone)"), "Auto-assign public IP" (set to "Enable"), and "Firewall (security group)" (set to "Create security group"). A note states: "We'll create a new security group called 'launch-wizard-5' with the following rules: Allow SSH traffic from Anywhere (0.0.0.0/0)". The "Launch instance" button is highlighted in orange.

Step 3: Configure Storage

The third step shows the "Configure storage" configuration screen. It specifies "1x 8 GB gp3 Root volume (Not encrypted)" and "Add new volume". A note states: "We'll create a new volume with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." The "Launch instance" button is highlighted in orange.



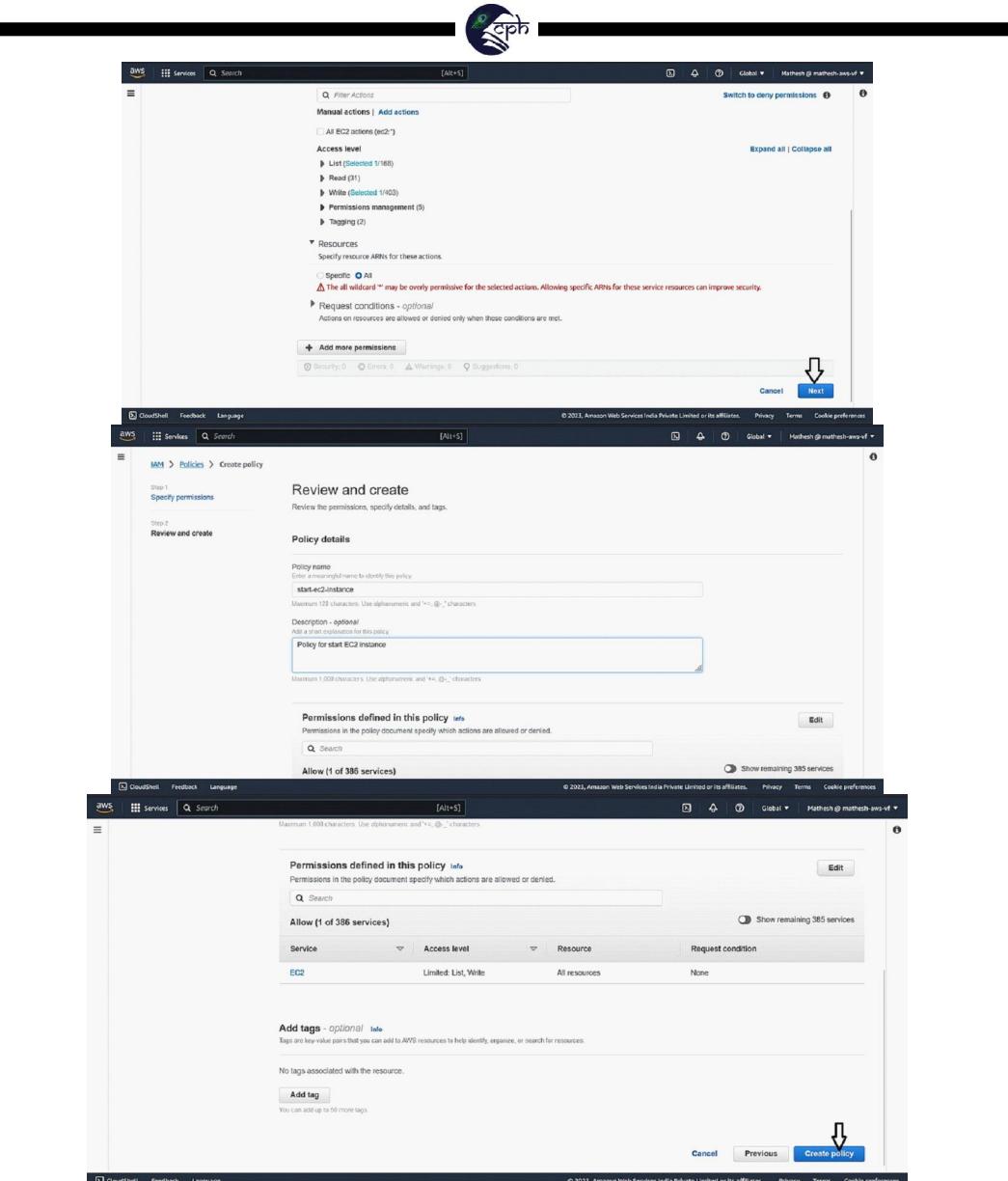
Step 2 :Creating the Policy:

- Navigate to the IAM Console.
- Click on “Policies” and then Click on “Create policy”

The screenshot shows the AWS IAM Policies page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. The main area displays a table of existing policies, each with a name, type (Customer managed), and usage count. At the top right of the table, there's a 'Create policy' button, which is highlighted with a blue arrow. The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and Language, along with a copyright notice for 2023.

- Select services as EC2.
- And Actions are DescribeInstances , Start Instances.

The screenshot shows the AWS IAM Policy editor for EC2. It's on 'Step 2: Review and create'. Under the 'Actions allowed' section, the 'describe' action is selected. The 'DescribeInstances' checkbox is checked and highlighted with a red box. Other actions listed include DescribeInstanceAttribute, DescribeInstanceConnectEndpoints, DescribeInstanceEventNotificationAttributes, DescribeInstanceEventWindows, DescribeInstanceState, DescribeInstanceTypeOfferings, and DescribeInstanceTypes. The bottom of the page includes standard AWS navigation links and a copyright notice for 2023.



5. Now we have created a policy for starting instances. We also need to create a policy for stopping the instances. This is because we are going to create two Lambda functions: one for starting and one for stopping the instances. Each function will have its own role, and we will attach these two policies to their respective roles.
6. Now we are going to repeat the same steps for Creating Stopping Policy also.
7. Everything is same , Except Actions because we are going to stop the instance.



8. The Actions are `DescribeInstances` , `Stop Instances` .

9. Keep your Policy name as “stop-ec2-instance”.

Step 3 :Creating the Lambda functions:

- Navigate to the lambda Console.
- Follow the Outlined steps below.

```
functions: []
```

```
function_name: lambda_function
description: ''
package_type: Zip
runtime: python3.8
last_modified: Never
```

Remove the default function and configuration values and replace with your own code and don't forget to replace Region name and instance ID

```
function_name: lambda_function
description: ''
package_type: Zip
runtime: python3.8
last_modified: Never
```

```
function_name: lambda_function
description: ''
package_type: Zip
runtime: python3.8
last_modified: Never
```

Successfully updated the function Start-EC2-demos.

```
function_name: Start-EC2-demos
description: ''
package_type: Zip
runtime: python3.8
last_modified: 2023-07-10T10:45:00+00:00
```

```
lambda_function.py
```

```
import json
import boto3
from botocore.exceptions import ClientError

# Set the AWS Region
region = "ap-south-1"

# Create an EC2 client
ec2_client = boto3.client("ec2", region_name=region)

# Specify the instance ID of the EC2 instance you want to start
instance_id = "i-01234567890abcdef12345678901234567890"

# Start the EC2 instance
try:
    response = ec2_client.start_instances(InstanceIds=[instance_id], DryRun=False)
    print(response["ResponseMetadata"]["HTTPStatusCode"])
    print(f"Success! EC2 instance {instance_id} is being started.")
except ClientError as e:
    print(f"An error occurred: {e}")
    print(f"HTTP Status Code: {e.response['HTTPStatusCode']}")
    print(f"Error Message: {e.response['Error']['Message']}")
```

The screenshot shows three sequential steps in the AWS Lambda console for creating a test event and configuring an execution role.

Step 1: Configure Test Event

A modal window titled "Configure test event" is open. It contains fields for "Event name" (set to "demo-start-instance"), "Event sharing settings" (set to "Private"), and "Template - optional" (set to "hello-world"). Below these is an "Event JSON" section with a "Format: JSON" button and a "Save" button highlighted with a red arrow. The background shows the Lambda function configuration page with the "Code" tab selected.

Step 2: Configuration Tab

The main Lambda function configuration page is shown. The "Configuration" tab is selected. A green success message at the top states: "The test event demo-start-instance was successfully saved." The "Code source" tab is also visible, showing the function code:

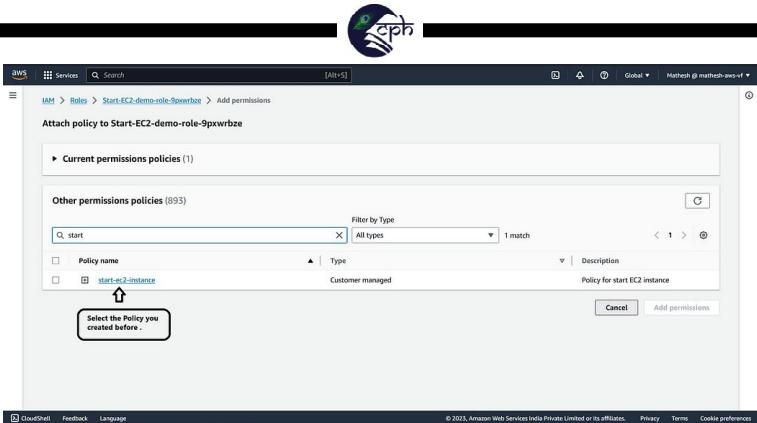
```

1 import boto3
2
3 def lambda_handler(event, context):
4     # Initialize the EC2 client
5     ec2_client = boto3.client('ec2', region_name='ap-south-1') # Replace your Region name
6
7     # Specify the instance ID of the EC2 instance you want to start
8     instance_id = 'i-05230280946f530a' # Replace your Instance ID
9
10    # Start the EC2 instance
11    try:
12        response = ec2_client.start_instances(InstanceIds=[instance_id], DryRun=False)
13        print(f"Started EC2 instance {instance_id}...")
14        print(response)
15        return {
16            'statusCode': 200,
17            'body': f"EC2 instance {instance_id} is being started."
18        }
19    except Exception as e:
20        print(f"Error starting EC2 instance: {e}")
21        return {
22            'statusCode': 500,
23            'body': f"Error starting EC2 instance: {str(e)}"
24        }

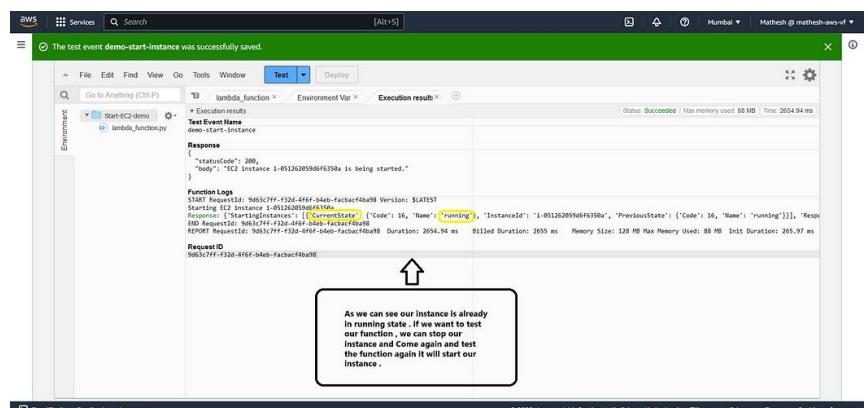
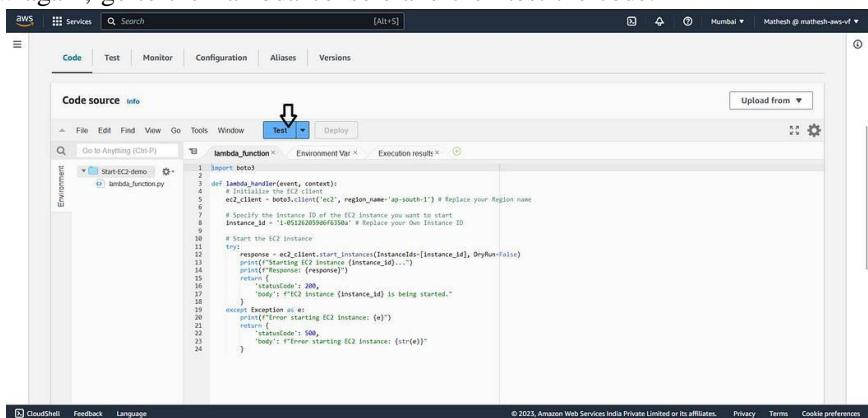
```

Step 3: Execution Role Configuration

The "Execution role" section of the configuration page is shown. The "Role name" field is set to "Start-EC2-demo-role-SouthAsia". A callout bubble points to this field with the text "Click". The "Resource summary" section shows an Amazon CloudWatch Logs resource with two actions and two resources. The table lists the resource "awslogs-loggroup-south-1:804937851364-log-group/awslambda/Start-EC2-demo" with actions "Allow: logs:CreateLogGroup" and "Allow: logs>CreateLogStream".



Now again, go to the Lambda console and then test the code.



1. Now we Created lambda function for Starting Instance.
2. We have to Repeat the same steps again to Create a Lambda function for Stopping Instance , keep your lambda function name as “Stop- EC2-demo”.



3. The only changes we have to make are to replace the default code with the ‘stop-ec2-instance.py’ code and attach the policy we created for stopping instances to the role of this Lambda function.
- As demonstrated above, when I test my Python code, it runs successfully and stops the instance.
 - Now, we are ready to proceed and create schedules for this functions.

Step 5 :Creating the Schedules Using Cloud Watch :

- Navigate to the Cloud Watch Console.
- Follow the Outlined Steps below.

The image contains three screenshots of the AWS CloudWatch and Amazon EventBridge consoles, illustrating the steps to create a scheduled rule:

- Screenshot 1: CloudWatch Overview**
Shows the CloudWatch Overview page with sections for Metrics, Logs, Alarms, and Events. A prominent "Get started with CloudWatch" section provides links to Create alarms, Create dashboards, and View logs.
- Screenshot 2: Amazon EventBridge - Rules**
Shows the Amazon EventBridge Rules page. It displays a table of existing rules and a "Create rule" button. A modal window titled "Important Message" provides instructions for creating event bus targets.
- Screenshot 3: Amazon EventBridge - Create rule**
Shows the "Define rule detail" step of the "Create rule" wizard. It includes fields for Name (set to "start-ec2-rule"), Description (optional), Rule type (set to "Schedule"), and Schedule (set to "Every 1 minute"). A note about the "EventBridge Scheduler" is visible at the bottom.

Note : Keep your rule name as “start-ec2-rule”, I mistakenly named it ‘role’ Please do not name it as ‘role’.

Schedule type

Cron-based schedule
A schedule using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

Rate-based schedule
A schedule that runs at a regular rate, such as every 10 minutes.

Cron expression Define the cron expression for the schedule

cron (00 Minutes 8 Hours ? Day of month JAN-DEC Month SUN-SAT Day of week 2023 Year)

Next 10 trigger dates

Sat, 09 Sep 2023 08:00:00 UTC+05:30
 Sun, 10 Sep 2023 08:00:00 UTC+05:30
 Mon, 11 Sep 2023 08:00:00 UTC+05:30
 Tue, 12 Sep 2023 08:00:00 UTC+05:30
 Wed, 13 Sep 2023 08:00:00 UTC+05:30
 Thu, 14 Sep 2023 08:00:00 UTC+05:30
 Fri, 15 Sep 2023 08:00:00 UTC+05:30
 Sat, 16 Sep 2023 08:00:00 UTC+05:30
 Sun, 17 Sep 2023 08:00:00 UTC+05:30

Flexible time window

If you choose a flexible time window, Scheduler invokes your schedule within the time window you specify. For example, if you choose 15 minutes, the schedule runs once between the schedule start time and end time.

Timeframe

Daylight saving time

Amazon EventBridge Scheduler automatically adjusts your schedule for daylight saving time. When time shifts forward in the Spring, if a cron expression falls on a non-existent date, your schedule invocation is skipped. When time shifts backwards in the Fall, your schedule runs only once and does not repeat its invocation. The following invocations occur normally at the specified date and time.

Timezone

The timezone for the schedule: **UTC-05:30 Asia/Kolkata**

Start date and time - optional

The start date and time of the schedule:
 YYYY/MM/DD **10/09/2023** Use 24-hour format timestamp (hh:mm)

End date and time - optional

The end date and time of the schedule:
 YYYY/MM/DD **10/09/2023** Use 24-hour format timestamp (hh:mm)

Select target

Target detail

Target API info

Select an API that will be invoked as a target for your schedule:

Templated targets All APIs

Amazon Lambda

Choose your Lambda function for start schedule

Lambda function

Start EC2 demo

Create new Lambda function

Configure verifications

Payload

Provide the JSON that you want to provide to your Lambda function as input. For example, - payload "key value": `{"key": "value"}`

JSON

Line 1, Col 1 Errors: 0 Warnings: 0

Skip to Review and create schedule

Previous

Next

Amazon EventBridge > Schedules > Create schedule

Step 1: Specify schedule details

Step 2: optional

Select targets

Step 3: optional

Settings

Step 4: Review and create schedule

Settings - optional

Schedule state

Enable schedule

You can choose to enable the schedule now. You will be able to enable the schedule after it has been created.

Enable

Action after schedule completion

Action after schedule completion

If you choose `DELETE`, EventBridge Scheduler will automatically delete the schedule after it has completed its last invocation and has no further target invocations planned.

NONE

Choose `NONE` and Scroll down and click "Next"

Retry policy and dead-letter queue (DLQ)

Retry policy

Max age of event: 24 hours

Dead-letter queue ARN

None

Encryption info

By default, Amazon EventBridge encrypts event metadata and message data that it stores under an AWS owned key (Encryption at rest). EventBridge Scheduler also encrypts data that passes between EventBridge Scheduler and other services using Transport Layer Security (TLS) whenever it is used.

Your data is encrypted by default with a key that AWS owns and manages for you. To choose a different key, customize your encryption settings.

Customize encryption settings (advanced)

Permissions info

EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.

Execution role

Create new role for this schedule

Use existing role

Role name

This role is the one we will be creating on your behalf. You can change the name.

Amazon_EventBridge_Scheduler_LAMBDA_7b3034c1f5

Go to IAM console

Cancel

Previous

Next

Action after schedule completion

NONE

Retry policy and dead-letter queue (DLQ)

Retry policy

Max age of event: 24 hours 0 minutes

Dead-letter queue ARN

None

Encryption

Customer master key (CMK) aws/lambda

Description

Default master key that protects my Amazon EventBridge Scheduler data when no other key is defined

Key ARN

Cancel

Previous

Create schedule



- We have now created a schedule for starting the instance every day at 8:00 AM.
 - Next, we need to create a schedule for stopping instances.
 - To create the schedule for stopping instances, follow the same steps as for starting instance scheduling with a few changes, Keep your rule name as “stop-ec2-rule”.
1. The changes include modifying the scheduled time and selecting the appropriate scheduling function.
 2. We need to change the schedule time to 17:00 because it will stop the Lambda function at 17:00 IST (5:00 PM).
 3. We have to Change the Function as Stop-EC2-demo

Lambda function

Stop-EC2-demo



Now, we have successfully created two schedules: one to start the instance every day at 8:00 AM and the other to stop the instance every day at 5:00 PM.

Deploy Your First Web App on AWS with AWS Amplify, Lambda, DynamoDB and API Gateway

This guide is designed for beginners or developers with some cloud experience who want to learn the fundamentals of building web applications on the AWS cloud platform. We'll walk you through deploying a basic contact management system, introducing you to key AWS services along the way.

In this project, as you can guess from the title, we will use AWS, which stands for Amazon Web Services; an excellent cloud platform with endless services for so many various use cases from training machine learning models to hosting websites and applications.

Cloud computing provides on-demand access to computing resources like servers, storage, and databases.

Serverless functions are a type of cloud computing service that allows you to run code without managing servers.

By the end of this tutorial, you'll be able to:

- Deploy a static website to AWS Amplify.
- Create a serverless function using AWS Lambda.
- Build a REST API with API Gateway.
- Store data in a NoSQL database using DynamoDB.
- Manage permissions with IAM policies. Integrate your frontend code with the backend services.



I recommend you follow the tutorial one time and then try it by yourself the second time. And before we begin, ensure you have an AWS account. Sign up for a free tier account if you haven't already.

Now let's get started!

Step 1: Deploy the frontend code on AWS Amplify

- In this step, we will learn how to deploy static resources for our web application using the AWS Amplify console.
- Basic web development knowledge will be helpful for this part. We will create our HTML file with the CSS (style) and Javascript code (functionality) embedded in it. I have left comments throughout to explain what each part does.

Here is the code snippet of the page:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <title>Contact Management System</title>
    <style>
      body {
        background-color: #202b3c;
        display: flex; /* Centering the form */
        justify-content: center; /* Centering the form */
        align-items: center; /* Centering the form */
        height: 100vh; /* Centering the form */
        margin: 0; /* Removing default margin */
      }
      form {
        display: flex;
        flex-direction: column; /* Aligning form elements vertically */
        align-items: center; /* Centering form elements horizontally */
        background-color: #fff; /* Adding a white background to the form */
        padding: 20px; /* Adding padding to the form */
        border-radius: 8px; /* Adding border radius to the form */
      }
      label, button {
        color: #FF9900;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 20px;
        margin: 10px 0; /* Adding margin between elements */
      }
      input {
        color: #232F3E;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 20px;
        margin: 10px 0; /* Adding margin between elements */
        width: 250px; /* Setting input width */
        padding: 5px; /* Adding padding to input */
      }
      button {
        background-color: #FF9900; /* Adding background color to button */
        color: #fff; /* Changing button text color */
        border: none; /* Removing button border */
      }
    </style>
  </head>
  <body>
    <form>
      <label>First Name</label>
      <input type="text" name="first-name" required="required" />
      <label>Last Name</label>
      <input type="text" name="last-name" required="required" />
      <label>Email Address</label>
      <input type="email" name="email" required="required" />
      <label>Phone Number</label>
      <input type="tel" name="phone" required="required" />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```



```
padding: 10px 20px; /* Adding padding to button */
cursor: pointer; /* Changing cursor to pointer on hover */
}

h1{
color: #202b3c;
font-family: Arial, Helvetica, sans-serif;
}

</style>
<script>

// Define the function to call the API with the provided first name, last
name, and phone number
let callAPI = (fname, lname, pnumber)=>{
    // Create a new Headers object and set the 'Content-Type' to
    'application/json'
    let myHeaders = new Headers();

    myHeaders.append("Content-Type", "application/json");

    // Create the JSON string from the input values
    let raw = JSON.stringify({ "firstname": fname, "lastname": lname,
    "phone_number": pnumber });

    // Define the request options including method, headers, body, and
    redirect behavior
    let requestOptions = {
        method: 'POST', // Method type
        headers: myHeaders, // Headers for the request
        body: raw, // The body of the request containing the JSON string
        redirect: 'follow' // Automatically follow redirects
    };

    // Use the fetch API to send the request to the specified URL
    fetch("https://uvibtoen42.execute-api.us-east-1.amazonaws.com/web-
app-stage", requestOptions) // Replace "API_KEY" with your actual API endpoint
        .then(response => response.text()) // Parse the response as text
        .then(result => alert(JSON.parse(result).message)) // Parse the
    result as JSON and alert the message
        .catch(error => console.log('error', error)); // Log any errors to
the console
}
```



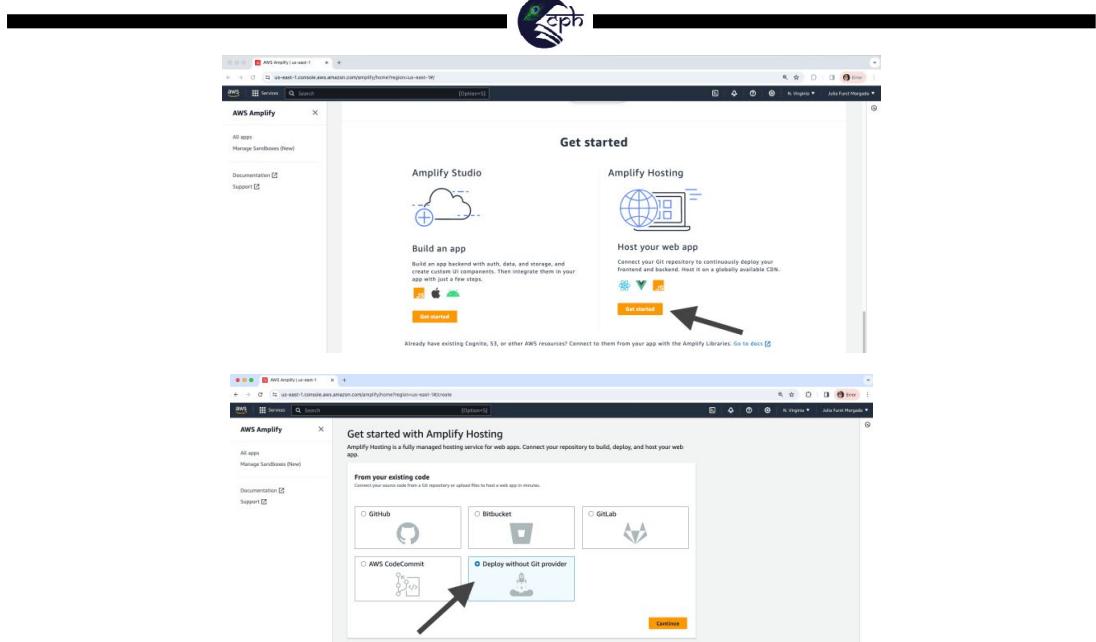
```
</script>
</head>
<body>
<form>
    <h1>Contact Management System</h1>
    <label>First Name :</label>
    <input type="text" id="fName">
    <label>Last Name :</label>
    <input type="text" id="lName">
    <label>Phone Number :</label>
    <input type="text" id="pNumber">
    <button type="button"
        onclick="callAPI(document.getElementById('fName').value,
        document.getElementById('lName').value,
        document.getElementById('pNumber').value)">Submit</button>
    <!-- Button to submit user input without reloading the page -->
    <!-- When clicked, it calls the callAPI function with values from the
    input fields -->
</form>
</body>
</html>
```

There are multiple ways to upload our code into Amplify console. For example, I like using Git and Github. To keep this article simple, I will show you how to do it directly by drag and drop method into Amplify. To do this — we have to compress our HTML file.

Now, make sure you're in the closest region to where you live, you can see the region name at the top right of the page, right next to the account name. Then let's go to the AWS Amplify console. It will look something like this:

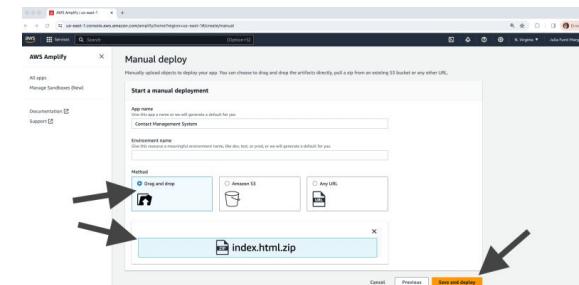


When we click "Get Started," it will take us to the following screen (we will go with Amplify Hosting on this screen):

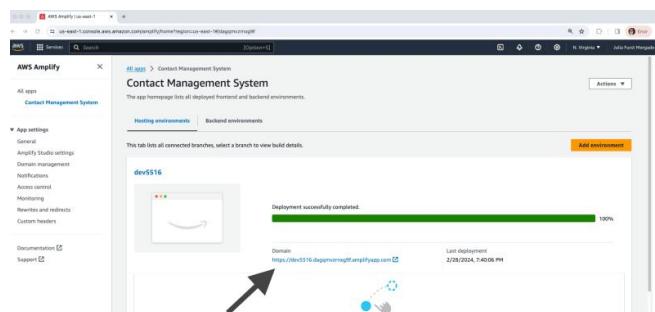


You will start a manual deployment. Give your app a name, I'll call it “Contact Management System”, and ignore the environment name.

Then, drop the compressed index file and click Save and Deploy.

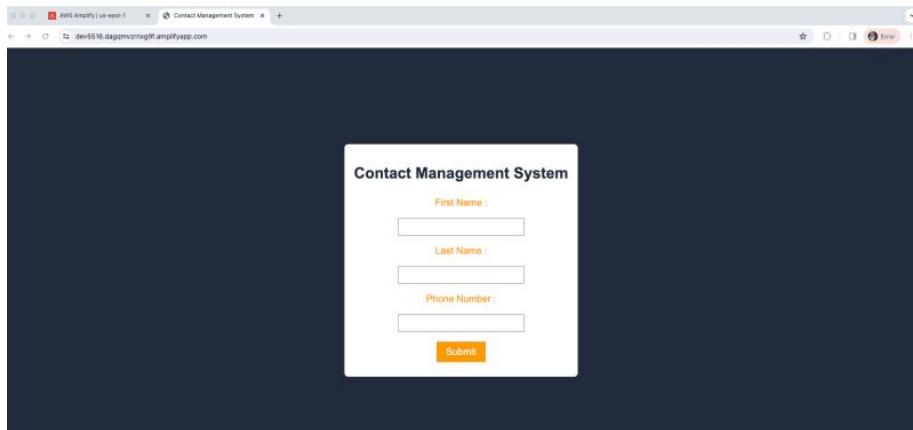


Amplify will deploy the code, and return a domain URL where we can access the website.



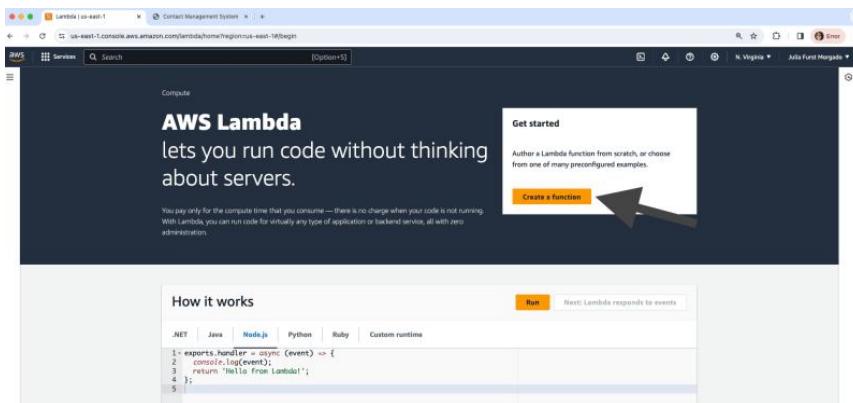


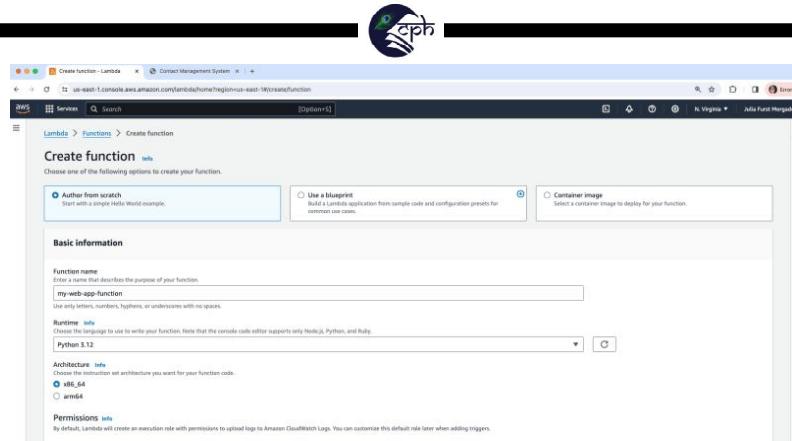
Click on the link and you should see this:



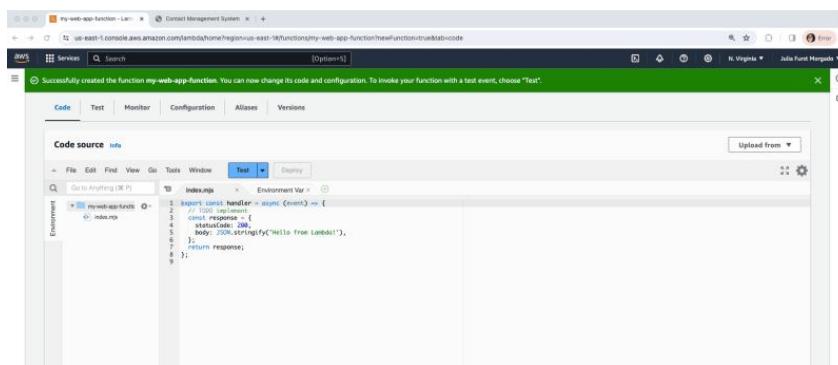
Step 2: Create an AWS Lambda Serverless function

- We will create a serverless function using the AWS Lambda service in this step. A Lambda function is a serverless function that executes code in response to events. You don't need to manage servers or worry about scaling, making it a cost-effective solution for simple tasks. To give you some idea, a great example of Serverless computing in real life is vending machines. They send the request to the cloud and process the job only somebody starts using the machine.
- Let's go to the Lambda service inside the AWS console. By the way, make sure you are creating the function in the same region in which you deployed the web application code in Amplify.
- Time to create a function. Give it a name, I'll call it "my-web-app- function", and for runtime programming language parameters: I've chosen Python 3.12, but feel free to choose a language and version that you are more comfortable and familiar with.

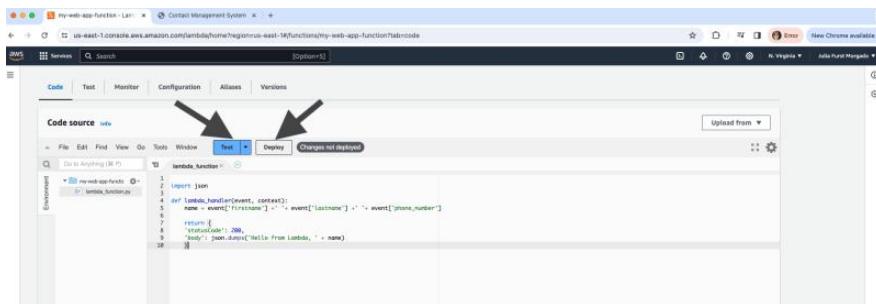




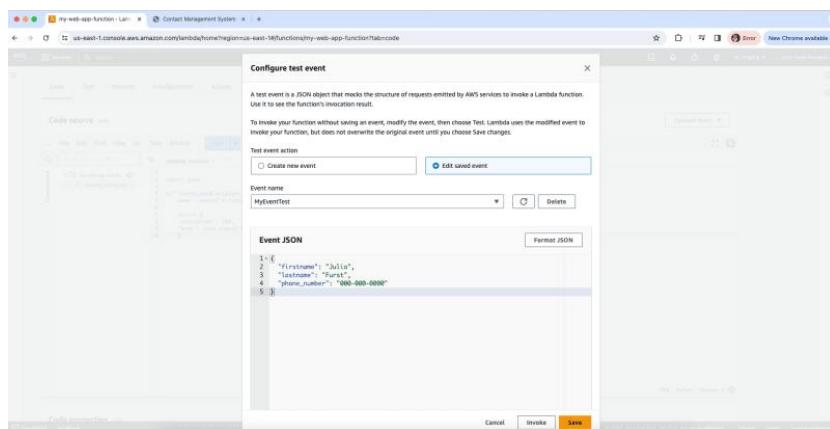
After our lambda function is created, scroll down and you will see the following screen:



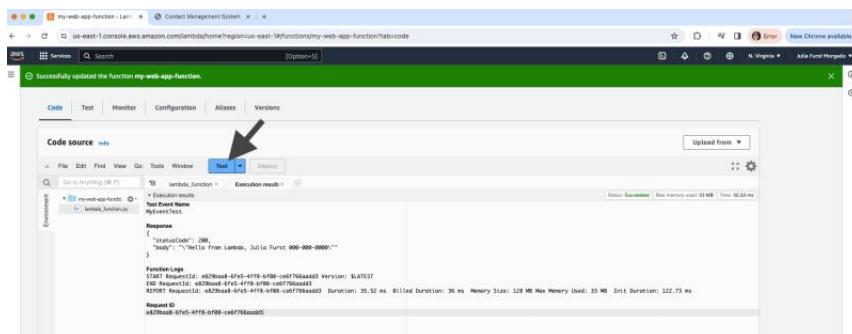
Now, let's edit the lambda function. Here is a function that extracts first and last names from the event JSON input. And then returns a context dictionary. The body key stores the JSON, which is a greeting string. After editing, click Deploy to save my-web-app-function, and then click Test to create an event.



To configure a test event, give the event a name like "MyEventTest", modify the Event JSON attributes and save it.



Now click on the big blue test button so we can test the Lambda function.



The execution result has the following elements:

- Test Event Name
- Response
- Function Logs
- Request ID

Step 3: Create Rest API with API Gateway

Now let's go ahead and deploy our Lambda function to the Web Application. We will use Amazon API Gateway to create a REST API that will let us make requests from the web browser. API Gateway acts as a bridge between your backend services (like Lambda functions) and your frontend application. It allows you to create APIs that expose functionality to your web app.

REST: Representational State Transfer.

API: Application Programming Interface.

Go to the Amazon API Gateway to create a new REST API.



The screenshot shows the AWS API Gateway - Create API page. It displays three API type options: **HTTP API**, **WebSocket API**, and **REST API**. Each section provides a brief description and compatibility information. Under the **REST API** section, there are two buttons: **Import** and **Build**. A large black arrow points to the **Build** button.

At the API creation page, we have to give it a name for example “Web App API”, and choose a protocol type and endpoint type for the REST API (select Edge-optimized).

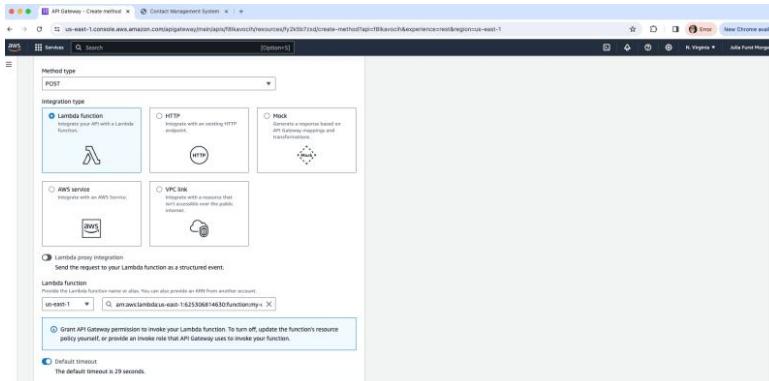
The screenshot shows the AWS API Gateway - Create REST API page. In the **API details** section, there are four options: **New API** (selected), **Clone existing API**, **Import API**, and **Example API**. The **API name** field is set to "Web app API". The **Description - optional** field is empty. Below these, the **API endpoint type** dropdown is set to "Edge-optimized". At the bottom right, there is a **Create API** button. A large black arrow points to this button.

Now we have to create a POST method so click on Create method.

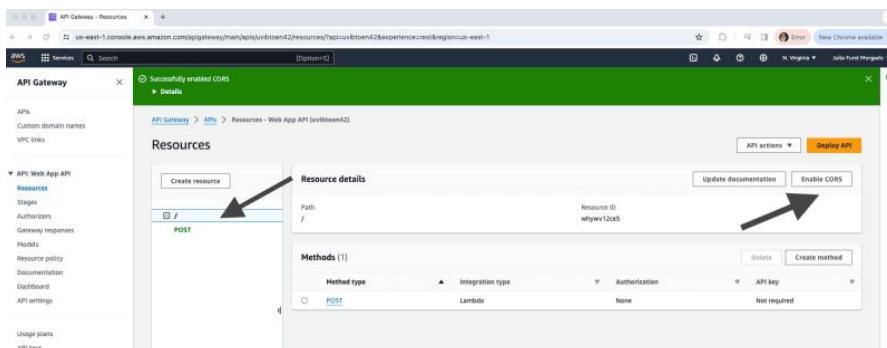
The screenshot shows the AWS API Gateway - Resources page. In the **Resource details** section, the path is set to "/". In the **Methods** section, there is a table with one row. The **Create method** button is located at the bottom right of this section. A large black arrow points to this button.



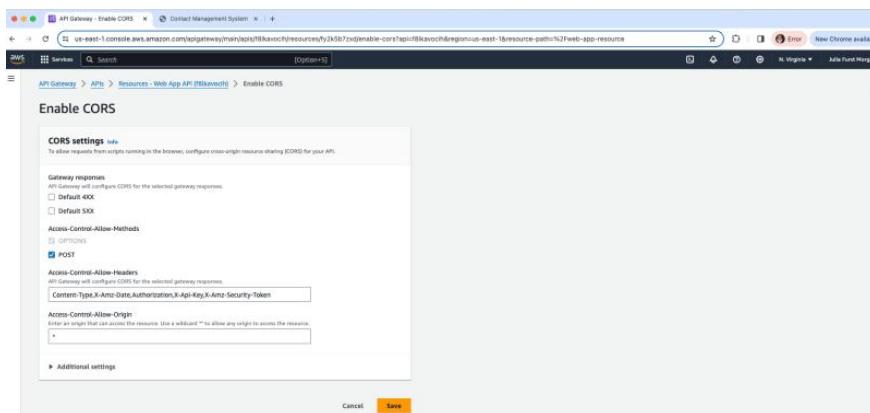
In the Create method page, select the method type as POST, the integration type should be Lambda function, ensure the Region is the same Region you've used to create the lambda function and select the Lambda function we just created. Finish by clicking on Create method at the bottom of the page.



Now we need to enable CORS, so select the / and then click enable CORS

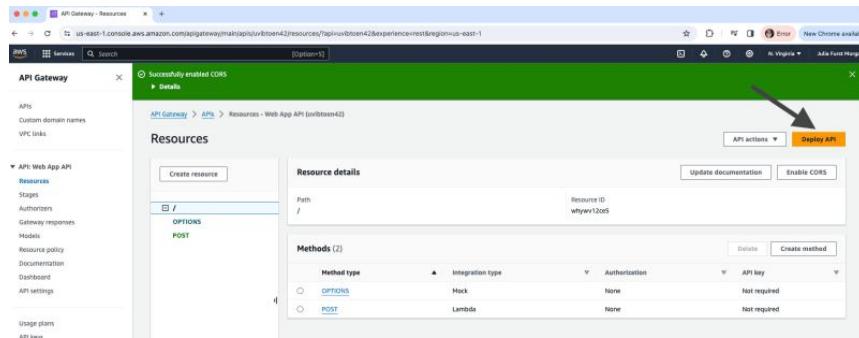


In the CORS settings, just tick the POST box and leave everything else as default, then click save.

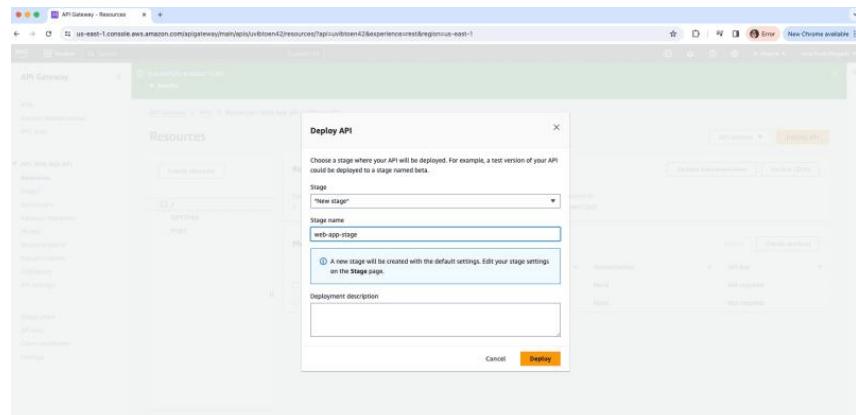




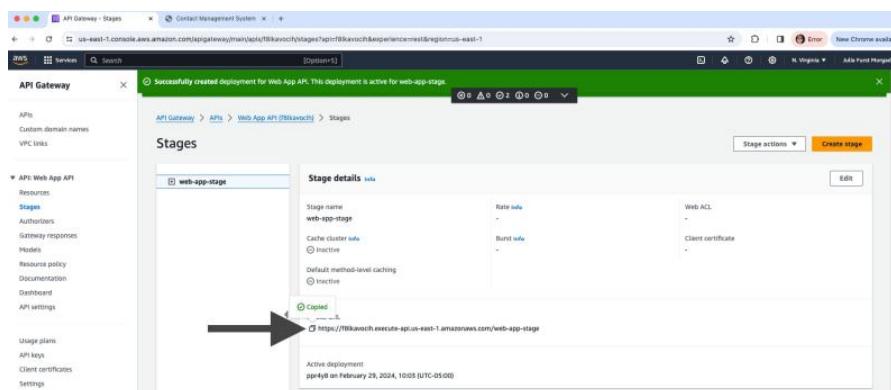
After enabling CORS headers, click on the orange Deploy API button.



A window will pop up, under stage select new stage and give the stage a name, for example “web-app-stage”, then click deploy.



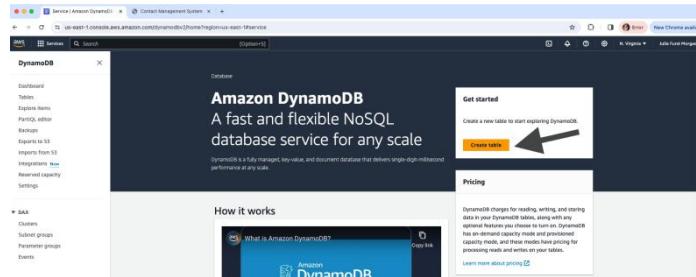
When you view the stage, there will be a URL named Invoke URL. Make sure to copy that URL; we will use it to invoke our lambda function in the final step of this project.



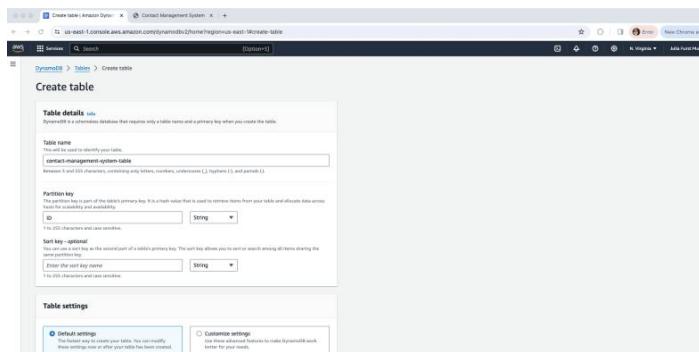


Step 4: Create a DynamoDB table

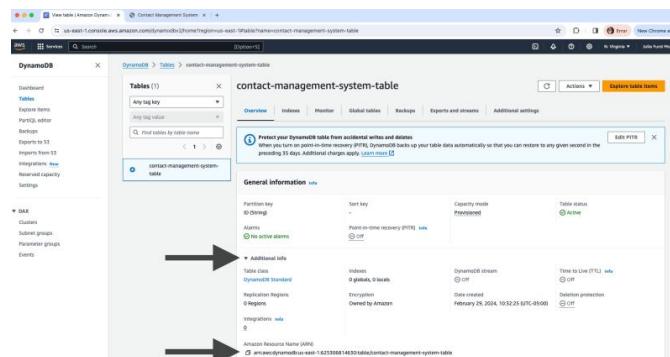
In this step, we will create a data table in Amazon DynamoDB, another AWS service. DynamoDB is a NoSQL database service that stores data in key-value pairs. It's highly scalable and flexible, making it suitable for various applications. Click on the orange create table button.



Now we have to fill out some information about our data table, like the name “contact-management-system-table”, and the partition key is ID. The rest leave as default. Click Create table.



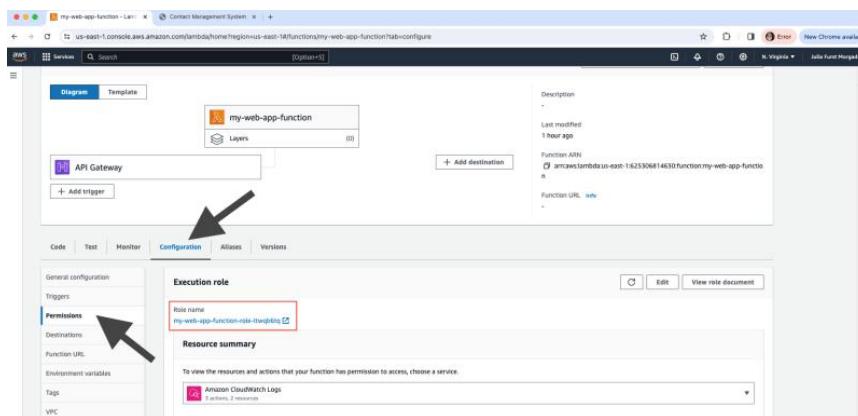
Once the table is successfully created, click on it and a new window with the details of the table will open up. Expand the Additional info and copy the Amazon Resource Name (ARN). We will use the ARN in the next step when creating IAM access policies.



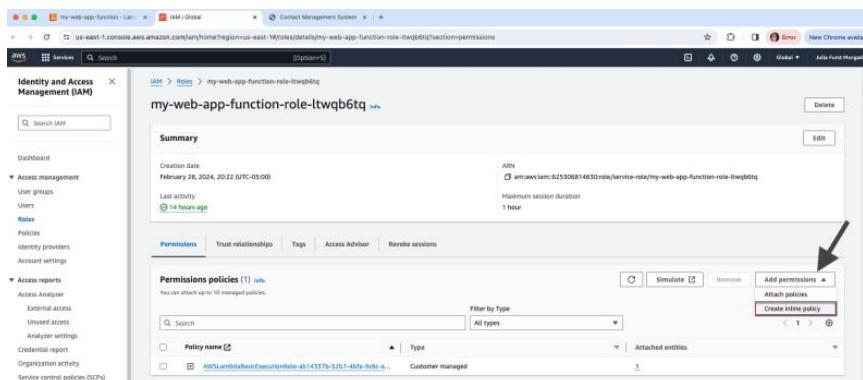


Step 5: Set up IAM Policies and Permissions

- AWS IAM is one of the most basic and important things to be set up, yet a lot of people neglect it. For improved security, it's always recommended a least-privilege access model, which means not giving a user more than needed access. For example, even for this simple web application project, we have already worked on multiple AWS services: Amplify, Lambda, DynamoDB, and API Gateway. It's essential to understand how they communicate with each other and what kind of information they share.
- Now back to our project, we have to define an IAM policy to give access to our lambda function to write/update the data in the DynamoDB table.
- So go back to the AWS Lambda console, and click on the lambda function we just created. Then go to the configuration tab, and on the left menu click on Permissions. Under Execution role, you will see a Role name.



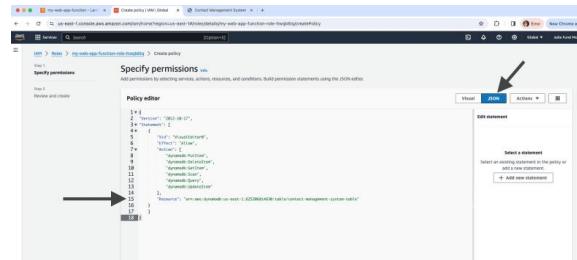
Click on the link, which will take us to the permissions configuration settings of this IAM role. Now click on Add permissions, then create an inline policy.



Then click on JSON, delete what's on the Policy editor and paste the following.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb>DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "YOUR-DB-TABLE-ARN"
        }
    ]
}
```

Make sure to substitute the “YOUR-DB-TABLE-ARN” with your real DynamoDB table ARN. Click Next, give the policy a name, like “lambda-dynamodb”, and then click Create policy. This policy will allow our Lambda function to read, edit, delete, and update items from the DynamoDB data table.



Now close this window, and back to the Lambda function, go to the Code tab and we will update the lambda function python code with the following.

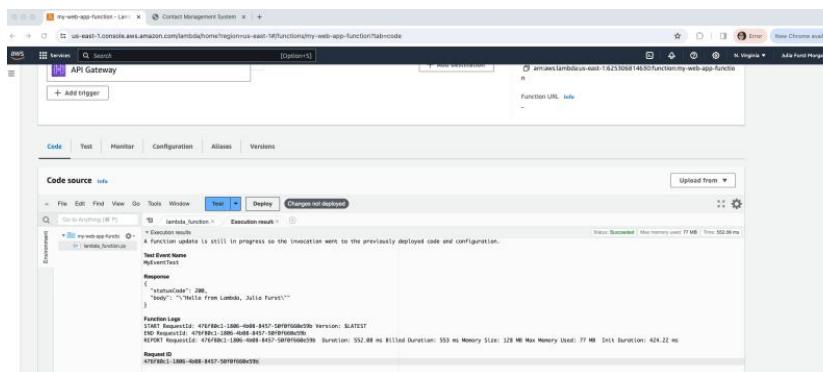
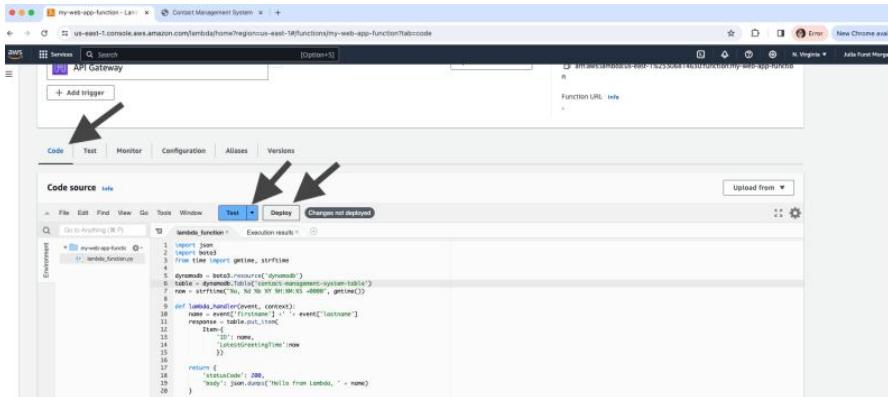
```
import json
import boto3
from time import gmtime, strftime

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('contact-management-system-table')
now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

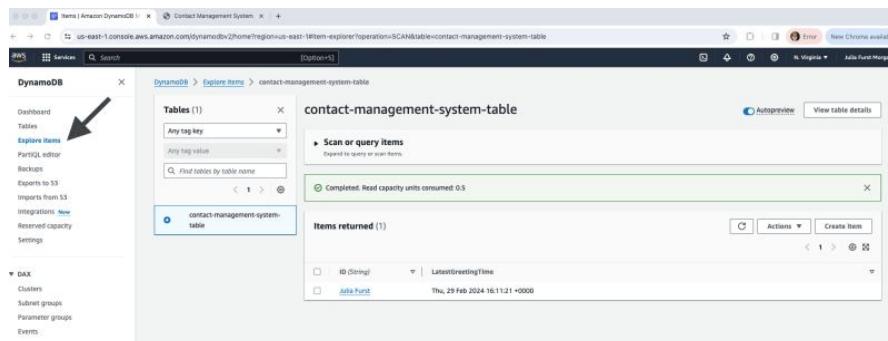
def lambda_handler(event, context):
    name = event['firstname'] + ' ' + event['lastname'] + ' ' + event['phone_number']
    response = table.put_item(
        Item={
            'ID': name,
            'latestGreetingTime': now
        })
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda, ' + name)
    }
    'statusCode': 200,
    'body': json.dumps('Hello from Lambda, ' + name)
}
```



The response is in REST API format. After making the changes, make sure to deploy the code. After the deployment is concluded, we can Test the program by clicking on the blue test button.



We can also check the results on the DynamoDB table. When we run the function it updates the data on our table. So go to AWS DynamoDB, click on explore items in the left nav bar, click on your table. Here is the object returned from the lambda function:



Step 6: Update frontend code with Rest API

- Congrats on making it this far!
- In this final step, we will see everything we just built in action. We will update the front-end to be able to invoke the REST API with the help of our lambda function and receive data.
- First, go back to your index.html on your code editor. See on line 68 you had “API_KEY”? Go ahead and swap that with the invoke URL you copied from the API Gateway service under your REST API details. Once you’ve done that, save and compress the file again, like we did in step 1, and upload it again to AWS using the console.

```

    ...
    <head>
        ...
        <script>
            // Define the function to call the API with the provided first name, last name, and phone number
            let callAPI = (fname, lname, pnumber) =>
                // Create a new Headers object and set the 'Content-Type' to 'application/json'
                let myHeaders = new Headers();
                myHeaders.append("Content-Type", "application/json");

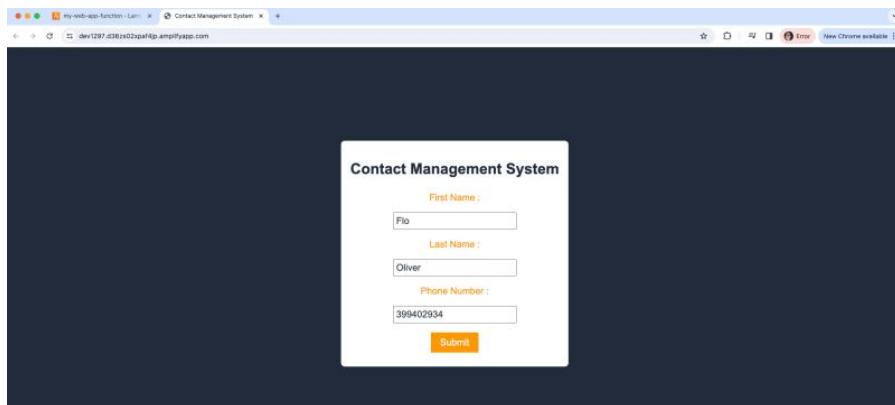
                // Create the JSON string from the input values
                let raw = JSON.stringify({ "firstname": fname, "lastname": lname, "phone_number": pnumber });

                // Define the request options including method, headers, body, and redirect behavior
                let requestOptions = {
                    method: 'POST', // Method type
                    headers: myHeaders, // Headers for the request
                    body: raw, // The body of the request containing the JSON string
                    redirect: 'follow' // Automatically follow redirects
                };

                // Use the fetch API to send the request to the specified URL
                fetch(`https://av1biten04.execute-api.us-east-1.amazonaws.com/web-app-stage`, requestOptions) // Replace "API_KEY" with your actual invoke URL
                    .then(response => response.text()) // Parse the response as text
                    .then(result => alert(JSON.parse(result).message)) // Parse the result as JSON and alert the message
                    .catch(error => console.log(error)); // Log any errors to the console
            }
        </script>
    </head>
</body>

```

Click on the new link you got and let's test it.



Our data tables receive the post request with the entered data. The lambda function invokes the API when the “Call API” button is clicked. Then using javascript, we send the data in JSON format to the API. You can find the steps under the callAPI function.



You can find the items returned to my data table below:

The screenshot shows the AWS DynamoDB console with the 'Explore Items' view for the 'contact-management-system-table'. The table has 4 items returned. The data is as follows:

ID	LatestGreetingTime
Paul Strnad 273840...	Fri, 01 Mar 2024 14:00:58 +00000
Mark Fehlha 578503...	Fri, 01 Mar 2024 14:00:58 +00000
Julia Furt 000-000-400...	Fri, 01 Mar 2024 13:44:57 +00000
Flo Oliver 399402934	Fri, 01 Mar 2024 13:44:57 +00000

Conclusion

You have created a simple web application using the AWS cloud platform. Cloud computing is snowballing and becoming more and more part of developing new software and technologies.

If you feel up for a challenge, next you could:

- Enhance the frontend design
- Add user authentication and authorization
- Set up monitoring and analytics dashboards
- Implement CI/CD pipelines to automate the build, test, and deployment processes of your web application using services like AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy.



How to Start and Stop an AWS EC2 Instance Automatically?

This text explains step by step how to automatically start and stop an EC2 instance in AWS using AWS Lambda function and Amazon Event Bridge.



We may not need the servers (EC2 instance) in AWS to run continuously. Running it on only when needed and shutting it down when the work is completed prevents waste of resources and saves our budget.

It can be managed manually at irregular intervals or for servers that are not tied to a specific schedule. However, on servers that need to be start and stop in a certain schedule, we can automate this process using the AWS Lambda function and Amazon Event Bridge. I describe step by step now.

Create AWS Lambda Function:

We go to the AWS Lambda service and click on the “Create Function” button. Then we give the function a name. We choose the language in which the function will be written, for example Python 3.8. In our example, the Lambda function is written in Python 3.8 and the AWS SDK using the Boto3 library. We create the basic properties of the function by clicking on the “Create” button.

Create function info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to display for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
myFunctionOnMedium

Use only letters, numbers, underscores, and hyphens with no spaces.

Runtime info
Choose the runtime language to use for your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.8

Architecture info
Choose the execution environment architecture you want for your function code.
 x86_64
 arm64

Permissions info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Advanced settings

Create function

AWS Create Function



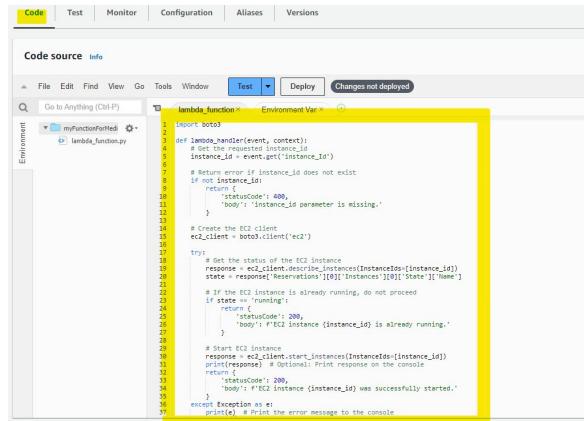
We go inside the created function. We delete the default code under the Code tab and paste the following code.

```
1 import boto3
2
3 def lambda_handler(event, context):
4     # Get the requested instance_id
5     instance_id = event.get('instance_Id')
6
7     # Return error if instance_id does not exist
8     if not instance_id:
9         return {
10             'statusCode': 400,
11             'body': 'instance_id parameter is missing.'
12         }
13
14     # Create the EC2 client
15     ec2_client = boto3.client('ec2')
16
17     try:
18         # Get the status of the EC2 instance
19         response = ec2_client.describe_instances(InstanceIds=[instance_id])
20         state = response['Reservations'][0]['Instances'][0]['State']['Name']
21
22         # If the EC2 instance is already running, do not proceed
23         if state == 'running':
24             return {
25                 'statusCode': 200,
26                 'body': f'EC2 instance {instance_id} is already running.'
27             }
28
29         # Start EC2 instance
30         response = ec2_client.start_instances(InstanceIds=[instance_id])
31         print(response) # Optional: Print response on the console
32         return {
33             'statusCode': 200,
34             'body': f'EC2 instance {instance_id} was successfully started.'
35         }
36     except Exception as e:
37         print(e) # Print the error message to the console
38         return {
39             'statusCode': 500,
40             'body': 'An error occurred while starting the EC2 instance.'
41         }

```

This script takes the instance_id as a parameter. If the instance_id is incorrect or missing, the warning ‘instance_id parameter is missing’ is returned. If the instance_id is correct, the process continues. An EC2 client is created and the status of the instance is checked. If the instance is already running, no action is taken and the warning “EC2 instance is already running” is returned. If the instance is not running, it is initialised.

After pasting the code, we load the code by pressing the “Deploy” button.



AWS Lambda Function Code



After installing the code, we go to the “General configuration” menu under the “Configuration” tab and click the “Edit” button to change the default settings.

The screenshot shows the AWS Lambda "General configuration" page. The left sidebar lists various configuration options: Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, RDS databases, File systems, and State machines. The main panel displays the "General configuration" section with the following details:

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	0 min 3 sec	SnapStart Info None

An "Edit" button is located in the top right corner of the main panel.

AWS Lambda Function Configuration

Under Basic Settings, a description can optionally be written. Then, we specify the amount of memory and storage required. In this example, the minimum values will be enough.

The **timeout** specifies the maximum running time of the function. This duration refers to the time from the start of the function to its completion. When the timeout duration is exceeded, the function execution is stopped and the result is returned.

This parameter determines how long the function has to complete its function.

Since the server can take a long time to initialise, we should set this time to **at least 5 minutes**. Other properties are by default left and saved.

The screenshot shows the "Edit basic settings" page for a function named "myFunctionForMedium". The top navigation bar includes "Lambda > Functions > myFunctionForMedium > Edit basic settings". The main panel is titled "Edit basic settings" and contains the "Basic settings" section. The configuration fields are as follows:

- Description - optional:** An empty text input field.
- Memory Info:** A text input field set to "128 MB", with a note: "Your function is allocated CPU proportional to the memory configured." Below it is a slider to "Set memory to between 128 MB and 10240 MB".
- Ephemeral storage Info:** A text input field set to "512 MB", with a note: "You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing". Below it is a slider to "Set ephemeral storage (/tmp) to between 512 MB and 10240 MB".
- SnapStart Info:** A dropdown menu currently set to "None". A note says: "Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations".
- Timeout:** A slider set to "5 sec". A note says: "Supported runtimes: Java 11, Java 17, Java 21".
- Execution role:** A note: "Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console". Two radio buttons are shown:
 - Use an existing role
 - Create a new role from AWS policy templates

AWS Lambda Function Basic Configuration



After Basic Settings, we go to the “Permission” menu under the “Configuration” tab and go to the role settings by clicking the role that will run the function.

The screenshot shows the AWS Lambda Configuration page. The top navigation bar has tabs: Code, Test, Monitor, Configuration (which is highlighted in yellow), Aliases, and Versions. On the left, a sidebar menu includes General configuration, Triggers, Permissions (which is also highlighted in yellow), Destinations, Function URL, Environment variables, Tags, VPC, and Monitoring and operations tools. The main content area is titled "Execution role". It shows a "Role name" field containing "myFunctionForMedium-role-79j5dfq4" with a copy icon. Below it is a "Resource summary" section with a heading "To view the resources and actions that your function has permission to access, choose a service." A list item "Amazon CloudWatch Logs" is shown, indicating "3 actions, 2 resources". At the bottom of this section are two buttons: "By action" and "By resource".

Attach Policy

The screenshot shows the AWS Lambda Permissions page. The top navigation bar has tabs: Permissions (highlighted in yellow), Trust relationships, Tags, Access Advisor, and Revoke sessions. Below the tabs is a section titled "Permissions policies (1) Info" with a note "You can attach up to 10 managed policies." It includes a search bar, a "Filter by Type" dropdown set to "All types", and a table with one row: "AWSLambdaBasicExecutionRole-55da99f1-1125-4dbe-94... Customer managed". To the right of the table are buttons for "Add permissions" (highlighted in yellow), "Attach policies" (highlighted in yellow), and "Create inline policy".

AWS Lambda Function Permission Configuration

The default policy is not authorised in ec2 instances. To provide this, we select the “Attach policies” option under the Add permissions button.

The screenshot shows the "Current permissions policies (1)" section. It lists one policy: "AWSLambdaBasicExecutionRole-55da99f1-1125-4dbe-94... Customer managed". Below this is a "Other permissions policies (1/939)" section. A search bar shows "ec2full". The table lists one policy: "AmazonEC2FullAccess" (selected, highlighted with a blue border). To the right of the table are buttons for "Cancel" and "Add permissions" (highlighted in yellow).

AWS Lambda Function Role Configuration

We find and select the **AmazonEC2FullAccess** policy and click the “Add permissions” button.

The screenshot shows the "Permissions policies (2) Info" section. It lists two policies: "AmazonEC2FullAccess" (selected, highlighted with a blue border) and "AWSLambdaBasicExecutionRole-55da99f1-1125-4dbe-94... Customer managed". Below the table are buttons for "Add permissions" (highlighted in yellow), "Simulate", and "Remove".

AWS Lambda Function Role Attach Policy



With the addition of the new policy, the authorizations of the role will change.

AWS Lambda Function Role

Now Test Time

We need to test our lambda function before scheduling it. For this we go back to the lambda function and click on the Test tab. Here we give a name to the event and write the instance id we want to start in the “Event JSON” code section.

You can find the Instance ID in the instance section under the EC2 dashboard.

AWS EC2 Instance



After typing the correct instance id to the instance_Id variable in the test section, save it and start the test process by clicking the test button.

The screenshot shows the AWS Lambda Test event interface. At the top, a green checkmark indicates "Executing function: succeeded (Logs)". Below this, there are tabs for "Test event" and "Info". A "Save" and "Test" button are at the top right. The "Test event" tab contains fields for "Test event action" (set to "Create new event"), "Event name" (set to "myTest"), and "Event sharing settings" (set to "Private"). There is also a "Template - optional" dropdown set to "hello-world".

Our test was successful. Now we make sure by checking the status on the ec2 dashboard.

The screenshot shows the AWS EC2 Instances dashboard with two instances listed. The first instance is redacted. The second instance has a yellow status bar indicating it is "Running". The third instance is redacted. The fourth instance has a yellow status bar indicating it is "Initializing". The columns include Name, Instance ID, Instance state, Instance type, and Status check.

Name	Instance ID	Instance state	Instance type	Status check
[REDACTED]	[REDACTED]	Running	t3a.medium	2/2 checks passed
[REDACTED]	[REDACTED]	Running	t3a.medium	0/0 checks passed

AWS EC2 Instance

In this way, we have seen that our Lambda function works successfully.

Create Schedule

We will trigger the Lambda Function that we have built by creating a Schedule on Amazon Event Bridge. For this, we go to Amazon EventBridge service and click on the “Create schedule” button.

The screenshot shows the Amazon EventBridge Scheduler creation interface. On the left, there is a sidebar with "Developer resources" (Learn, Sandbox, Quick starts), "Buses" (Event buses, Rules, Global endpoints, Archives, Replay), "Pipes" (Pipes), "Scheduler" (Schedules, Schedule groups), "Integration" (Partner event sources, API destinations), and "Schema registry". The main area is titled "Amazon EventBridge Scheduler" and describes it as a serverless scheduler. It highlights four features: "Templated targets" (Amazon EventBridge Scheduler supports templated targets that you can use to perform common API operations using Amazon SQS, Amazon SNS, Lambda, and EventBridge), "Universal targets" (Amazon EventBridge Scheduler provides a universal target parameter that you can use to create common triggers that target more than 270 AWS services and over 6,000 API operations on a schedule), "Flexible time windows" (Amazon EventBridge Scheduler supports flexible time windows, allowing you to disperse your scheduled tasks and improve the reliability of your triggers for low-cost targets that do not require precise scheduled invocation of targets), and "Retries" (Amazon EventBridge Scheduler supports at-least-once event delivery to targets, ensuring that at least one delivery succeeds with a response from the target). At the bottom, there is a "Schedules (0)" section with a "Create schedule" button.

Amazon EventBridge



In the page that appears, we give a name to “Schedule name” and select “Recurring schedule” and “Cron-based schedule” options.

Specify schedule detail

Schedule name and description

Schedule name
 myScheduleForMedium

Use only letters, numbers, dashes, dots or underscores. Max 64 characters.

Description - optional
 Enter description

Maximum of 512 characters.

Schedule group
Each schedule needs to be placed in a schedule group. By default, a schedule is placed in the 'Default' group. You can also [create your own schedule group](#). You can only add tags to a schedule group, not a schedule.

default

Schedule pattern

Occurrence | [Info](#)
You can define an one-time or recurrent schedule.

One-time schedule Recurring schedule

Amazon EventBridge Schedule

For example, if we want our server to start at 09.00 every day, we fill in the blanks as follows, answer the “Flexible time window” question as off and click next.

Schedule type

Choose the schedule type that best meets your needs.

Cron-based schedule

A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

Rate-based schedule

A schedule that runs at a regular rate, such as every 10 minutes.

Cron expression | [Info](#)

Define the cron expression for the schedule

cron (

)

Minutes

Hours

Day of month

Month

Day of the week

Year

Copy Clear

Amazon EventBridge Cron-based Schedule



On the following page, we select the AWS Lambda function, which is the target API.

The screenshot shows the 'Select target' interface with the 'Target detail' tab active. Under 'Target API', the 'Info' tab is selected. A list of 'Templated targets' is shown in a grid:

- CodeBuild StartBuild
- CodePipeline StartPipelineExecute...
- Amazon ECS RunTask
- Amazon EventBridge PutEvents
- Kinesis Data Firehose PutRecord
- Amazon Inspector V1 StartAssessmentRun
- Kinesis Data Streams PutRecord
- AWS Lambda Invoke** (highlighted)
- SageMaker StartPipelineExecute...
- AWS Step Functions StartExecution
- Amazon SNS Publish
- Amazon SQS SendMessage

Amazon EventBridge Schedule with AWS Lambda

Then we select the Lambda Function that we have created and write the instance id as a parameter in the Payload section, just as we wrote in the test section of the Lambda Function, and click the next button.

The screenshot shows the 'Invoke' configuration screen. The 'Lambda function' dropdown is set to 'myFunctionForMedium'. The 'Payload' section contains the following JSON code:

```
1 [ {  
2   "Instance_Id": "1" (redacted)  
3 } ]
```

At the bottom, there are buttons for 'Cancel', 'Skip to Review and create schedule', 'Previous', and 'Next' (which is highlighted).

Amazon EventBridge Schedule with AWS Lambda



On the following page, select NONE for the “Action after schedule completion” question and disable the “Retry policy”.

Schedule state

Enable schedule
You can choose not to enable the schedule now. You will be able to enable the schedule after it has been created.
 Enable

Action after schedule completion

Action after schedule completion [Info](#)
If you choose DELETE, EventBridge Scheduler will automatically delete the schedule after it has completed its last invocation and has no future target invocations planned.

NONE

Retry policy and dead-letter queue (DLQ)

Retry policy [Info](#)
By default, EventBridge Scheduler attempts to retry failed invocations for up to 24 hours. You can specify the maximum age of the event and the maximum number of times to retry.
 Retry

Dead-letter queue (DLQ)
Standard Amazon SQS queues that EventBridge Scheduler uses to store events that couldn't be delivered successfully to a target.

None
 Select an Amazon SQS queue in my AWS account as a DLQ
 Specify an Amazon SQS queue in other AWS accounts as a DLQ

Amazon EventBridge Schedule with AWS Lambda

Then select “Create new role for this schedule” in the “Execution role” question and click next.

Encryption [Info](#)
By default, EventBridge Scheduler encrypts event metadata and message data that it stores under an AWS owned key (encryption at rest). EventBridge Scheduler also encrypts data that passes between EventBridge Scheduler and other services using Transport layer Security (TLS) (encryption in transit).

Your data is encrypted by default with a key that AWS owns and manages for you. To choose a different key, customize your encryption settings.
 Customize encryption settings (advanced)

Permissions [Info](#)

Permissions
EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.

Execution role
 Create new role for this schedule Use existing role

Role name
This is the role name we will be creating on your behalf. You can change the name.
Amazon_EventBridge_Scheduler_LAMBDA_685397e602 [Go to IAM console](#)

Cancel Previous Next

Amazon event Bridge Schedule with AWS Lambda



In the appearing window, we check the details again and click on “Create schedule” button and complete the process.

Step 3: Settings

Edit

Schedule state and permissions

Schedule state Enabled	Execution role Amazon_EventBridge_Scheduler_LAMBDA_685397e602
Action after schedule completion NONE	

Retry policy and dead-letter queue (DLQ)

Retry policy Max age of event: -	Retry policy Maximum retries: 185
Dead-letter queue ARN None	

Encryption

Customer master key (CMK) aws/scheduler	Description Default master key that protects my Amazon EventBridge Scheduler data when no other key is defined
Key ARN -	

Cancel Previous Create schedule

Amazon event Bridge Schedule with AWS Lambda

We can now see our Schedule under Amazon event Bridge.

Schedules (5)						<input type="button" value="Create schedule"/>
<input type="text"/> Search loaded schedules			All states			
						All groups
<input type="checkbox"/>	Schedule name	Schedule group	Status	Target	Target type	
<input type="checkbox"/>	myScheduleForMedium	default	Enabled	myFunctionForMedium	LAMBDA_Invoke	

Amazon Even Bridge Schedule



Automatically Stop AWS EC2 Instance

We can also stop the server automatically, just like we just initialized the server. I won't go through the steps to stop the server automatically, because they are mostly the same as the starting process. The only difference is the code written inside the Lambda Function. I share this code below.

```
1 import boto3
2
3 def lambda_handler(event, context):
4     # Get instance_id from request
5     instance_id = event.get('instance_Id')  #
6
7     # Return error if instance_id does not exist
8     if not instance_id:
9         return {
10             'statusCode': 400,
11             'body': 'instance_id parameter is missing.'
12         }
13
14     # Create the EC2 client
15     ec2_client = boto3.client('ec2')
16
17     try:
18         # Get the status of the EC2 instance
19         response = ec2_client.describe_instances(InstanceIds=[instance_id])
20         state = response['Reservations'][0]['Instances'][0]['State']['Name']
21
22         # Do not proceed if the EC2 instance has already stopped
23         if state == 'stopped':
24             return {
25                 'statusCode': 200,
26                 'body': f'EC2 instance {instance_id} is already stopped.'
27             }
28
29         # Stop EC2 instance
30         response = ec2_client.stop_instances(InstanceIds=[instance_id])
31         print(response)  # Optional: Print response on the console
32         return {
33             'statusCode': 200,
34             'body': f'EC2 instance {instance_id} was successfully stopped.'
35         }
36     except Exception as e:
37         print(e)  # Print the error message to the console
38         return {
39             'statusCode': 500,
40             'body': 'An error occurred while stopping the EC2 instance.'
41         }
```

AWS SNS — Simple Notification Service

What if you want to send one message to many receivers? So, we have the possibility to do a direct integration.

So, we have our create service and needs to send email, talk to their XYZ service, talk to their shipping service, maybe talk to another SQS queue, so we could integrate all these things together, but it would be quite difficult.

The other approach is to do something called Pub / Sub, or publish- subscribe.

And so, our create service publishes data to our SNS topic, and our SNS topic has many subscribers and all these subscribers get that data in real time as a notification. So, it could be an email notification, text message, shipping service, SQS queue. You're basically able to send your message once to an SNS topic and have many services receive it.



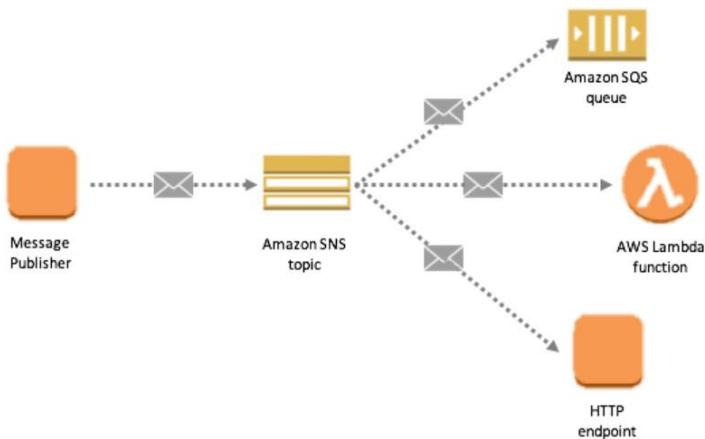
Basic Introduction

An Amazon SNS topic is a logical access point which acts as a communication channel. A topic lets you group multiple endpoints (such as AWS Lambda, Amazon SQS, HTTP/S, or an email address). To broadcast the messages of a message-producer system (for example, an e-commerce website) working with multiple other services that require its messages (for example, checkout and fulfillment systems), you can create a topic for your producer system. The first and most common Amazon SNS task is creating a topic.

Each subscriber to the topic will get all the messages now we have new feature to filter messages. Up to 10,000,000 subscriptions per topic & 100,000 topics limit.

Subscribers can be:

- SQS
- HTTP/HTTPS (with delivery retries — how many times).
- Lambda
- Emails
- SMS messages
- Mobile Notification



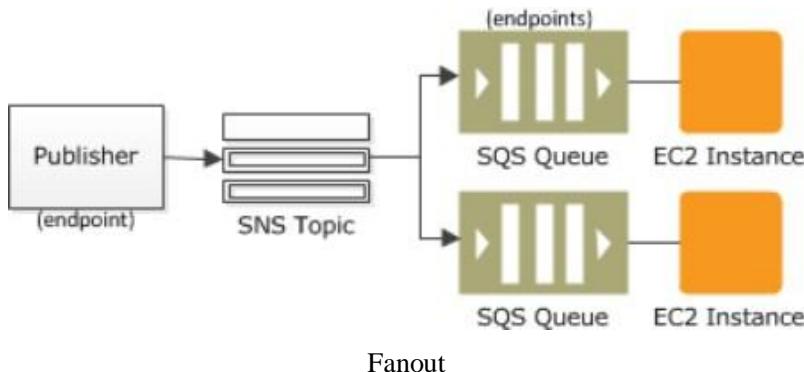
Our SNS integrates with a lot of Amazon Services in AWS.

- With Amazon S3, we use it on bucket events.
- For our ASG Auto-Scaling Notifications.
- In CloudWatch for Alarms.
- In CloudFormation for state changes.

SNS + SQS : Fan Out

The “fanout” scenario is when an Amazon SNS message is sent to a topic and then replicated and pushed to multiple Amazon SQS queues, HTTP endpoints, or email addresses. This allows for parallel asynchronous processing. For example, you could develop an application that sends an Amazon SNS message to a topic whenever an order is placed for a product. Then, the Amazon SQS queues that are subscribed to that topic would receive identical notifications for the new order. The Amazon EC2 server instance attached to one of the queues could handle the processing or fulfillment of the order while the other server instance could be attached to a data warehouse for analysis of all orders received.

It's fully decoupled & there is no data loss. It has ability to add receivers of data later.



Let's do Hands-On, we will go to SNS console & give the topic name, I'm giving “MyTestTopic” & click on Next Step.

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

Next step

[Start with an overview](#)



We're not going to apply any custom changes so will keep as default & click on Create Topic.

Amazon SNS

Dashboard Topics Subscriptions

Mobile Push notifications Text messaging (SMS)

Name MyTestTopic Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).
Display name - optional To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info This is my first Topic Maximum 100 characters, including hyphens (-) and underscores (_).
Encryption - optional Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.
Access policy - optional This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. Info
Delivery retry policy (HTTP/S) - optional The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. Info
Delivery status logging - optional These settings configure the logging of message delivery status to CloudWatch Logs. Info
Tags - optional A tag is a metadata object that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. Learn more
Cancel Create topic

Now as we see in below picture, there is no Subscription in Topic so we will create subscription, click in Create Subscription.

Amazon SNS > Topics > MyTestTopic

MyTestTopic

Details

Name MyTestTopic Display name This is my first Topic

ARN arn:aws:sns:us-east-1:727206182954:MyTestTopic Topic owner 727206182954

Subscriptions Access policy Delivery retry policy (HTTP/S) Delivery status logging Encryption Tags

Subscriptions (0) Edit Delete Request confirmation Confirm subscription Create subscription

Now we will choose the protocol, there are many protocols available there like HTTP/ HTTPS, Email, Lambda, SQS, SMS etc. I'll choose Email & give the email id & Create subscription.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN arn:aws:sns:us-east-1:727206182954:MyTestTopic

Protocol The type of endpoint to subscribe Email

Endpoint An email address that can receive notifications from Amazon SNS. redacted@gmail.com

After your subscription is created, you must confirm it. Info

Subscription filter policy - optional This policy filters the messages that a subscriber receives. Info

Cancel Create subscription



Now you will see that status of subscription is in Pending Confirmation, Go to your email & click on received email from Amazon & Confirm it.

Subscription: 6cb97025-88fe-47d0-baf1-086be8613d06

Details	
ARN	arn:aws:sns:us-east-1:727206182954:MyTestTopic:6cb97025-88fe-47d0-baf1-086be8613d06
Endpoint	kumargaurav1247@gmail.com
Topic	MyTestTopic
Status	Pending confirmation
Protocol	EMAIL

Now you will see the status confirmed as I confirm the subscription in my email.

Subscription: 6cb97025-88fe-47d0-baf1-086be8613d06

Details	
ARN	arn:aws:sns:us-east-1:727206182954:MyTestTopic:6cb97025-88fe-47d0-baf1-086be8613d06
Status	Confirmed

So, this is my console, you can add more subscription. I have added 2 subscriptions. Let's publish some message. Click on Publish Message button on top right-hand side.

MyTestTopic

Details	
Name	MyTestTopic
ARN	arn:aws:sns:us-east-1:727206182954:MyTestTopic
Delivery status logging	This is my Test Topic
Topic owner	727206182954

Subscriptions | Access policy | Delivery entry policy (HTTP/2) | Delivery status logging | Encryptions | Tags

ID	Endpoint	Status	Protocol
6cb97025-88fe-47d0-baf1-086be8613d06	kumargaurav1247@gmail.com	Confirmed	EMAIL
4f9c7a5-18d2-4515-9d9d-588f953e50f3	arn:aws:sqs:us-east-1:727206182954:MyTestQueue	Confirmed	SQS

I'm publishing my message, gave the details.

Topic ARN
arn:aws:sns:us-east-1:727206182954:MyTestTopic

Subject - optional
This is my Message

Maximum 100 printable ASCII characters

Time to Live (TTL) - optional
This setting applies only to mobile application endpoints. The number of seconds that the push notification service has to deliver the message to the endpoint. [Info](#)

Message body

Message structure

Identical payload for all delivery protocols.
The same payload is sent to endpoints subscribed to the topic, regardless of their delivery protocol.

Custom payload for each delivery protocol.
Different payloads are sent to endpoints subscribed to the topic, based on their delivery protocol.

Message body to send to the endpoint
My Message

Now go to SQS Queue & click on View/Delete Message in Actions(Make sure you subscribe the Topic in SQS)



Actions — Subscribe to SNS Topic (Do this work before publishing message from SNS.

Create New Queue

Queue Actions

Filter by Prefix: En

Name
MyTestQueue

Send a Message
View/Delete Messages
Configure Queue
Add a Permission
Purge Queue
Delete Queue
Subscribe Queue to SNS Topic
Configure Trigger for Lambda Function

Now you can see your message in Queue as well as in your email.

View/Delete Messages in MyTestQueue

Message Details

Message Body

```
[{"Type": "Notification", "MessageID": "79262ec2-169-5973-b150-b4d201e3a3de", "TopicArn": "arn:aws:sns:us-east-1:79262ec2-169-5973-b150-b4d201e3a3de", "Subject": "This is my message", "Message": "79262ec2-169-5973-b150-b4d201e3a3de", "Timestamp": "2019-11-07 19:29:54"}]
```

Message ID: 79262ec2-169-5973-b150-b4d201e3a3de
Size: 953 bytes
MD5 of Body: 7f0f6fdec23be48112c92bd527568db
Sender Account ID: AD417UQ0QYAUVEKVKU
Sent: 2019-11-07 19:29:54
First Received: 2019-11-07 19:29:54
Receive Count: 1

Aws SQS- SIMPLE QUEUE SERVICE



What is SQS?

- SQS stands for SIMPLE QUEUE SERVICE.
- SQS was the first service available in AWS
- AMAZON SQS is a web service that gives you access to message queue that can be used to store message while waiting for a computer to process them.
- Amazon SQS is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component where a queue is a temporary repository for messages that are awaiting processing.

- with the help of SQS, you can send ,store and receive messages between software components at any volume without losing messages.
- using Amazon SQS, you can separate the components of an application so that they can run independently, easing message management between components.
- Any component of a distributed application can store the message in the queue.
- Messages can contain up to 256 kB of text in any format such as JSON, XML, etc.
- Amazon SQS can be described as commoditization of the messaging service. Well known examples of messaging service Technologies include IBM WEBSPHERE MQ and MICROSOFT MESSAGE

QUEUEING.

Unlike these Technologies,users do not need to maintain their own server. Amazon does it for them and sells the SQS service at a per-use rate.

Authentication:-

Amazon SQS provides authentication procedures to allow for secure handling of data. Amazon uses its AMAZON WEB SERVER (AWS) identification to do this, requiring users to have an Aws enabled account with amazon.com.Aws assigns a pair of related identifiers ,your Aws access keys,to an Aws enabled Account to perform identification.

Aws uses the access key ID provided in a service request to look up an account's secret Amazon .com then calculates a digital signature with the key .if they match then the user is considered authentic, if not then the authentication fails and the request is not processed.

MESSAGE delivery:-

Amazon SQS guarantees at least- once delivery. Messages are stored on multiple servers for redundancy and to ensure availability. If a message is delivered while a server is not available, it may not be removed from that server queue and may be resent.

The service supports both unlimited queues and message traffic.

you can get started with Amazon SQS for free .All customers can make 1 million Amazon SQS request for free each month. Some application might be able to operate within this FREE tier limit.Aws free tier includes 1 million requests with Amazon simple queue service.

How does SQS works?

SQS provides an Api endpoint to submit messages and another endpoint to read Messages from a queue. Each message can only be retrieved once, and you can have many clients submitting messages to and reading messages from a queue at the same time .

Conclusion:-

- SQS is a pull- based ,not push-based
- message are 256 kB in size

- message are kept in a queue from 1 minute to 14 days.
- the default retention period is 4 days .
- it guarantees that your message will be processed at least one. So, here's about the short information about AMAZON SQS.

Sending a message from AWS SQS and SNS via Lambda Function with API gateway



we will be sending a message from AWS SQS via Lambda Function with API gateway.

Scenario:

A company wants to create a system that can track customer orders and send notifications when orders have been shipped. They want to use AWS services to build the system. They have decided to use SQS,

Lambda, and Python for the project.

- 1) Create a Standard SQS Queue using Python.
- 2) Create a Lambda function in the console with a Python 3.7 or higher runtime
- 3) Modify the Lambda to send a message to the SQS queue. Your message should contain either the current time or a random number. You can use the built-in test function for testing.
- 4) Create an API gateway HTTP API type trigger.
- 5) Test the trigger to verify the message was sent.

Prerequisites:

- AWS CLI and Boto3 installed
- AWS account with I AM user access, NOT root user
- Basic AWS command line knowledge
- Basic Python programming language
- Basic knowledge of AWS Interactive Development Environment (IDE)



What is AWS SQS?

Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service provided by Amazon Web Services (AWS). It is a reliable, highly scalable, and flexible way to decouple and asynchronously process workloads or messages across distributed applications and services. The service allows messages to be sent between software components and systems in a reliable and fault-tolerant way.

What is AWS Lambda?

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run your code in response to events and automatically manage the underlying compute resources. With AWS Lambda, you can write and run code without having to provision, manage, or scale servers. It is highly scalable, cost-effective, and reliable, and integrates seamlessly with other AWS services to enable building of scalable and highly available distributed systems and microservices architectures.

What is AWS API Gateway?

Amazon API Gateway is a service provided by Amazon Web Services (AWS) that lets you create APIs for your backend services or applications without having to manage the infrastructure. With Amazon API Gateway, you can create APIs that act as front doors for applications to access data or functionality from your backend services, such as AWS Lambda, Amazon EC2 instances, and HTTP/HTTPS endpoints.

1. Create SQS Queue using Python

In order to create a Python script, here is the reference from Boto3 documentation ([click here to view](#)) It is the code requirements for creating an SQS queue.

We need to create a new file before we get started! Select File > New From Template > Python File > Save As... and give it a name. Make sure you KEEP the extension .py.

```
#!/usr/bin/env python3.7
#Create a Standard SQS Queue using Python boto3
import boto3
#Get the service resource
sq = boto3.resource('sqs')
#Creating sqs queue
queue = sq.create_queue(QueueName='Week15Project-sqs-queue')
#print the URL print(queue.url)
```



Alright, let's run the code in AWS Cloud9 IDE . Great! We see SQS queue url displayed. Let's copy and save this url, we will need it for later.

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are two tabs: 'README.md' and 'SQS.py'. The 'SQS.py' tab contains the following Python code:

```
1 #!/usr/bin/env python3.7
2 #
3 #Create a Standard SQS Queue using Python boto3
4 import boto3
5
6 #Set the service resource.
7 sqs = boto3.resource('sqs')
8
9 #Creating sqs queue.
10 queue = sqs.create_queue(QueueName="Week15Project-sqs-queue")
11
12 #Print the URL.
13 print(queue.url)
14
15
16
17
18
19
20
21
22
```

Below the code editor is a terminal window titled 'bash - ip-172-31-59-196.x LUITProjects/SQS.py - St x'. The terminal shows the command 'Run' was executed, and the output is:

```
https://sqs.us-east-1.amazonaws.com/251940354023/Week15Project-sqs-queue
```

At the bottom of the terminal, it says 'Process exited with code: 0'.

Ok, let's double check if our SQS queue is created in AWS console.

The screenshot shows the 'Amazon SQS > Queues' page. The table lists one queue:

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
Week15Project-sqs-queue	Standard	Apr 19, 2023, 17:00:31 EDT	0	0	Amazon SQS key (SSE-SQS)	-

Awesome, we see Week15ProjectSQS-queue is created!

2. Create Lambda Function

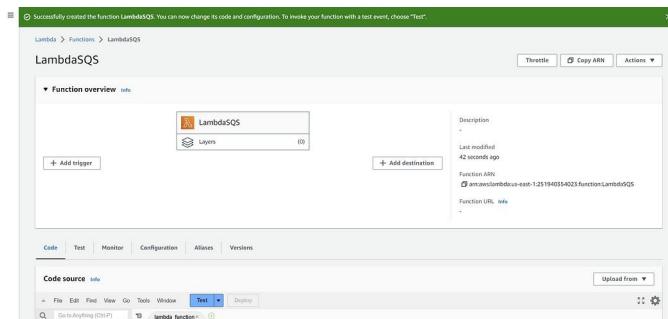
Creating our Lambda function will print a message to our newly created SQS queue when triggered.

Let's create a Lambda function in the AWS console > Create Function >

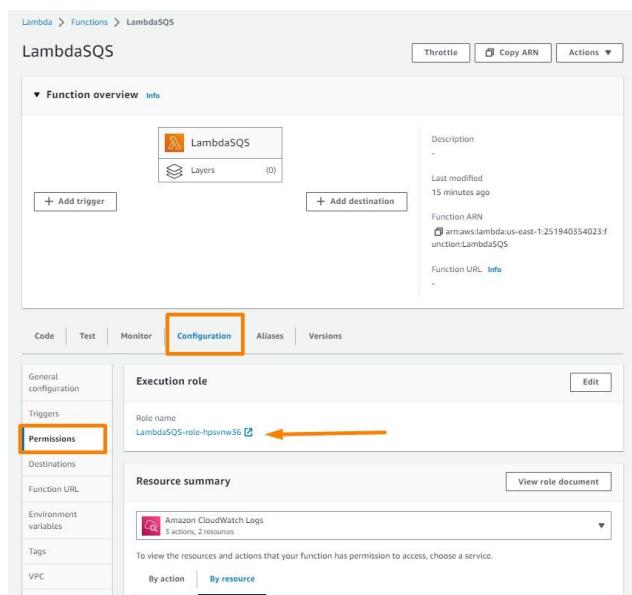
Author from scratch > Function Name: LambdaSQS > Runtime: Python 3.8 > Architecture x86_64 > Execution role: Create a new role with basic Lambda permissions > Create Function

The screenshot shows the AWS Lambda 'Create function' wizard. In the 'Basic information' step, the function name is 'LambdaSQS'. Under 'Execution role', the 'Create a new role with basic Lambda permissions' option is selected. In the 'Advanced settings' step, the 'Change default execution role' section is expanded, showing options to create a new role or reuse an existing one from AWS policy templates. A note states that role creation might take a few minutes. The 'Create function' button is at the bottom right.

We see our Lambda function has been created. Let's update the permissions for the role and attach to the Lambda function.



Under Configuration > Permissions > Add Permissions > Click on LambdaSQS-role-hpsvnw36





After click on LambdaSQS-role link, it will prompt you to IAM page, click >Add permissions > Attach policies.

IAM > Roles > LambdaSQS-role-hpsvnm36

LambdaSQS-role-hpsvnm36

Summary

Creation date: April 19, 2023, 17:40 (UTC-04:00)

Last activity: None

ARN: arn:aws:iam::251940354023:role/service-role/LambdaSQS-role-hpsvnm36

Maximum session duration: 1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (1) Info
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

Add permissions (highlighted with an orange arrow)

Create inline policy

Policy name: AWSLambdaBasicExecutionRole-72d85aa-9b75-4aea-9d43-da906c02b5ba

Type: Customer managed

Permissions boundary - (not set) Info
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others.

Set permissions boundary

Search SQS > Check AmazonSQSFullAccess > Add permission

IAM > Roles > LambdaSQS-role-hpsvnm36 > Add permissions

Attach policy to LambdaSQS-role-hpsvnm36

Current permissions policies (1)

Other permissions policies (Selected 1/0)

AmazonSQSFullAccess

Provides full access to Amazon SQS via the AWS Management Console.

AmazonSQSReadOnlyAccess

Provides read only access to Amazon SQS via the AWS Management Console.

AWSLambdaSQSQueueExecutionRole

Provides receive message, delete message, and read attribute access to SQS queues, and write permissions to CloudWatch logs.

Create policy (highlighted with an orange arrow)

AmazonSQSFullAccess

Provides full access to Amazon SQS via the AWS Management Console.

```
1: { "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "SQS:SendMessage", "Resource": "arn:aws:sqs:us-east-1:123456789012:my-queue", "Condition": {} }, { "Effect": "Allow", "Action": "SQS:DeleteMessage", "Resource": "arn:aws:sqs:us-east-1:123456789012:my-queue", "Condition": {} }, { "Effect": "Allow", "Action": "SQS:GetQueueAttributes", "Resource": "arn:aws:sqs:us-east-1:123456789012:my-queue", "Condition": {} }, { "Effect": "Allow", "Action": "SQS:ChangeMessageVisibility", "Resource": "arn:aws:sqs:us-east-1:123456789012:my-queue", "Condition": {} }, { "Effect": "Allow", "Action": "SQS:ReceiveMessage", "Resource": "arn:aws:sqs:us-east-1:123456789012:my-queue", "Condition": {} } ] }
```

AmazonSQSReadOnlyAccess

Provides read only access to Amazon SQS via the AWS Management Console.

AWSLambdaSQSQueueExecutionRole

Provides receive message, delete message, and read attribute access to SQS queues, and write permissions to CloudWatch logs.

Add permissions (highlighted with an orange arrow)

Great! Now our Lambda function have basic lambda function permission and Full SQS Access permission assigned to the attached role.



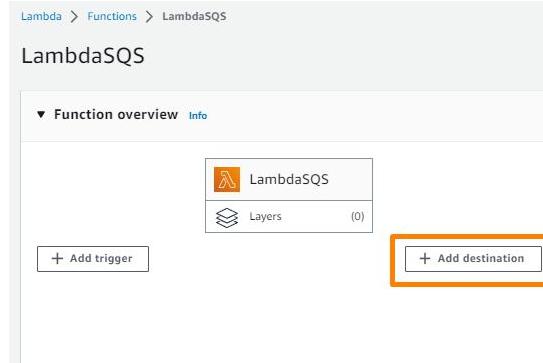
✓ Policy was successfully attached to role.

IAM > Roles > LambdaSQS-role-hpsvnw36

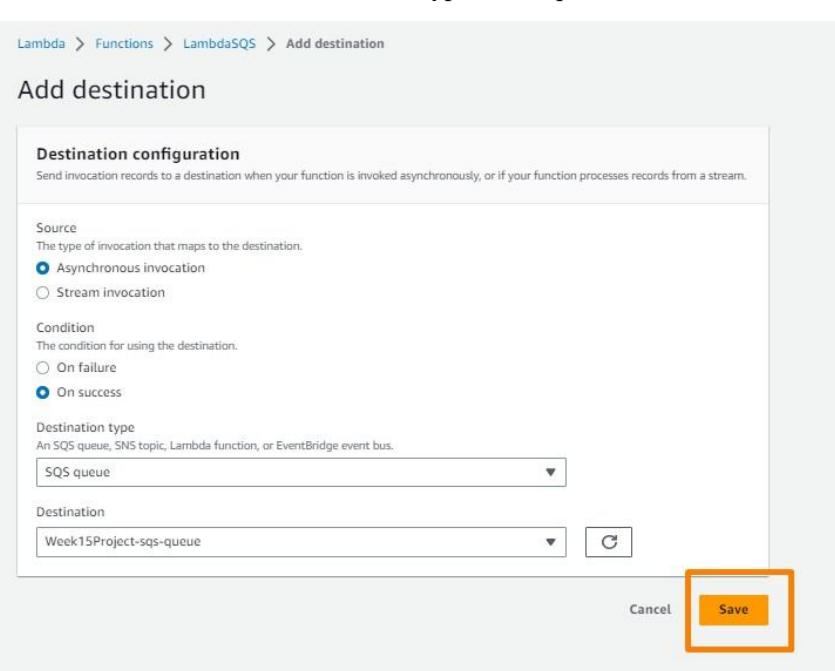
LambdaSQS-role-hpsvnw36

Summary

Next, we will add our SQS queue as a destination of the Lambda function. Go back to Lambda Function > Functions > LambdaSQS > Add destination



Add destination > Source: Asynchronous invocation (is a feature provided by AWS Lambda that allows you to invoke a Lambda function without waiting for the function to complete its execution) > Condition: On success > Destination type: SQS_queue > Destination: Week15Project-sqs-queue





Great we've completed the destination configuration.

Source	Stream	Condition	Destination
Asynchronous invocation	-	On success	arn:aws:sqs:us-east-1:251940354023:Week15Project-sqs-q

3. Edit the Lambda function to send a message to the SQS Queue

Below is the reference for Boto3 Documentation to find the code to send message to the SQS queue. We only need to use ‘QueueURL’ and ‘MessageBody’. to get the ‘QueueURL’

[send_message — Boto3 1.26.115 documentation \(amazonaws.com\)](#)

Go back in Lambda, use the Github gist Lambda SQS python script code and paste it in the lambda function code source, then click Deploy.

Lambda SQS Python Script Code ([github.com](#))

```
import json
import boto3
from datetime import datetime
import dateutil.tz

# Define the lambda handler
def lambda_handler(event, context):

    sqs = boto3.client('sns')

    # Get the current time
    est = dateutil.tz.gettz('US/Eastern')
    current_time = datetime.now(est)
    time = current_time.strftime("%I:%M:%S %p")

    # Send the message to the SQS queue
    response = sqs.send_message(
        QueueUrl = 'YOUR SQS URL',
        MessageBody = json.dumps(f'US Eastern current time: {time}')
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Message in a bottle sent to SQSqueue :D')
    }
```



Lambda > Functions > LambdaSQS

LambdaSQS

Function overview Info

Description

Last modified 1 hour ago

Function ARN arn:aws:lambda:us-east-1:251940354023:function:LambdaSQS

Function URL -

Code Test Monitor Configuration Aliases Versions

Code source Info

File Edit View Go Tools Window Test Deploy Changes not deployed

lambda_function.py

```
lambda_function.py
1 import json
2 import boto3
3 from datetime import datetime
4 import dateutil.tz
5
6 # Define the lambda handler
7 def lambda_handler(event, context):
8
9     sqs = boto3.client('sqs')
10
11     # Get the current time
12     est = dateutil.tz.gettz('US/Eastern')
13     current_time = datetime.now(est)
14     time = current_time.strftime("%Y-%m-%d %H:%M:%S")
15
16     # Send the message to the SQS queue
17     response = sqs.send_message(
18         QueueUrl = "https://sqs.us-east-1.amazonaws.com/251940354023/Week15Project-sqs-queue",
19         MessageBody = json.dumps("US Eastern current time: " + time)
20     )
21
22     return {
23         'statusCode': 200,
24         'body': json.dumps('Message sent to SQSqueue :D')
25     }
26
```

Successfully updated the function LambdaSQS.

We have successfully Deployed the code, lets test the code click Test.

Code source Info

File Edit Find View Go Tools Window Test Deploy Changes not deployed

lambda_function.py

```
lambda_function.py
1 import json
2 import boto3
3 from datetime import datetime
4 import dateutil.tz
5
6 # Define the lambda handler
7 def lambda_handler(event, context):
8
9     sqs = boto3.client('sqs')
10
11     # Get the current time
12     est = dateutil.tz.gettz('US/Eastern')
13     current_time = datetime.now(est)
14     time = current_time.strftime("%Y-%m-%d %H:%M:%S")
15
16     # Send the message to the SQS queue
17     response = sqs.send_message(
18         QueueUrl = "https://sqs.us-east-1.amazonaws.com/251940354023/Week15Project-sqs-queue",
19         MessageBody = json.dumps("US Eastern current time: " + time)
20     )
21
22     return {
23         'statusCode': 200,
24         'body': json.dumps('Message sent to SQSqueue :D')
25     }
26
```

Configure test event page will show up after. Select test event action: Create new event > Event Name: testSQSLambda > Event sharing settings: Private > Tempalte: apigateway-aws-proxy > Save

Configu11'e test event X

A \Qs.t eve:nt is a JSON object that mooks the stnJcturo oi rnque:sts Qmittoo by AWS seNices to invoke a ambda fum:fon_. Use it to ,;ee the function's illVOGltion result.

To [nvol<! your iuriction without saving an event, configurehe JSON event, then choose Test. Tes.t C'IICJlt action

I O Create new ewnt

E

Saved event



Eve:ntnarne

I testSQL.ambda

Maxanum of 25 characters consisting of letters, number,, doU, lryphens and underscores.

Eve:nt sharing settlings

O Private

This event Is. only .willab1eh 1he Lambda console and to the event creator. You can configure a to till 1 of 10. Learn more [!]:

Q Shareable:

This event Is. ;wajlab1e to 1AM us.er.. wllhln the s.ame aocount .mo h.we permission, to aocess and u,e >hareab e events. Learn more [!]:

Template: • *optimal*

I apigateway-aws-proxy

The screenshot shows the AWS Lambda Test Event configuration screen. On the left, there is a code editor window titled "EventJSON" containing a multi-line JSON event payload. The payload includes fields like "body", "resource", "path", "httpMethod", "isBase64Encoded", "queryStringParameters", "pathParameters", "stageVariables", "headers", and "stageHeaders". On the right, there is a "FormatJSON" button and a vertical scroll bar. At the bottom right of the editor area, there is a large orange "Save" button with a white border, which is highlighted with a red rectangle. Below the editor, there are "Cancel" and "Save" buttons.

```
1T "lil"
2  "body": "eyJ0ZXN0IjoiYm9keSj9",
3  "resource": "/{proxy+}",
4  "path": "/path/to/resource",
5  "httpMethod": "POST",
6  "isBase64Encoded": false,
7  "queryStringParameters": {
8    "foo": "bar"
9  },
10 "multivaluedQueryStringParameters": {
11   "foo": [
12     "bar"
13   ]
14 },
15 "pathParameters": {
16   "proxy": "/path/to/resource"
17 },
18 "stageVariables": {
19   "baz": null
20 },
21 "headers": {
22   "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
23   "Accept-Encoding": "gzip, deflate, sdch",
24   "Accept-Language": "en-US,en;q=0.8",
25   "Cache-Control": "max-age=0",
26   "CloudFront-Forwarded-Proto": "https",
27   "CloudFront-Is-Desktop-Viewer": "true",
28   "CloudFront-Is-Mobile-Viewer": "false",
29   "CloudFront-Is-smartTV-Viewer": "false",
30   "CloudFront-Is-tablet-Viewer": "false"
```

We can now click Test again and see if the Test Event was successful.



The test event testSQSLambda was successfully saved.

Lambda > Functions > LambdaSQS

LambdaSQS

Function overview [Info](#)

LambdaSQS

Amazon SQS

+ Add trigger

+ Add destination

Description

Last modified 8 minutes ago

Function ARN arn:aws:lambda:us-east-1:251940354023:function:LambdaSQS

Function URL [Info](#)

Code Test Monitor Configuration Aliases Versions

Code source [Info](#)

File Edit Find View Go Tools Window **Test** Deploy

lambda_function.py

```
1 import json
2 import boto3
3 from datetime import datetime
4 import dateutil.tz
5
6 # Define the lambda function
7 def lambda_handler(event, context):
8     sqs = boto3.client('sns')
9
10    # Get the current time
11    evn_time = dateutil.tz.gettz('UTC').localize(datetime.utcnow())
12    item_time = dateutil.tz.gettz('UTC').localize(datetime.now())
13    item_time = item_time.strftime("%Y-%m-%d %H:%M:%S")
14
15    # Create a message to queue
16    response = sqs.send_message(
17        QueueUrl = "https://sqs.us-east-1.amazonaws.com/251940354023/WorkShopQueue",
18        MessageBody = json.dumps(item_time)
19    )
20
21    return {
22        'statusCode': 200,
23        'body': json.dumps('Message sent to queue')
24    }
```

Awesome! We see StatusCode: 200 and the ‘body’ show our message along with the timestamp. This means our code is running correctly!

Code Test Monitor Configuration Aliases Versions

Code source [Info](#)

File Edit Find View Go Tools Window **Test** Deploy

lambda_function.py

Execution results

Test Event Name testSQSLambda

Response

```
{"statusCode": 200, "body": "\u201cMessage sent to queue\u201d"}
```

Function Logs

```
2019-08-12T21:56:00+00:00[1]: [Lambda] 2f966cb0-adba-4518-ad91-1731ef7e54bd Version: $LATEST
2019-08-12T21:56:00+00:00[1]: [Lambda] 2f966cb0-adba-4518-ad91-1731ef7e54bd
2019-08-12T21:56:00+00:00[1]: [Lambda] REPORT RequestId: 2f966cb0-adba-4518-ad91-1731ef7e54bd Duration: 1538.34 ms Billed Duration: 1530 ms Memory Size: 128 MB Max Memory Used: 66 MB Init Duration: 314.68 ms
2019-08-12T21:56:00+00:00[1]: [Lambda] RequestId: 2f966cb0-adba-4518-ad91-1731ef7e54bd
```

4. Create an API gateway HTTP API type trigger

In Lambda overview, click +Add trigger

The test event testSQSLambda was successfully saved.

Lambda > Functions > LambdaSQS

LambdaSQS

Function overview [Info](#)

+ Add trigger

Amazon SQS

+ Add destination

Description

Last modified 13 minutes ago

Function ARN arn:aws:lambda:us-east-1:251940354023:function:LambdaSQS

Function URL [Info](#)

Code Test Monitor Configuration Aliases Versions



Trigger configuration: > Select API Gateway > Intent: Create a new API
> API typ: HTTP API > Security: Open > Click Add

Lambda > Add trigger

Add trigger

Trigger configuration [Info](#)

API Gateway [app](#) [application-services](#) [aws](#) [serverless](#)

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

Intent
Use an existing API or have us create one for you.
 Create a new API
 Use existing API

API type
 HTTP API Build low-latency and cost-effective REST APIs with built-in features such as DDoS and OAuth2, and native CORS support.
 REST API Develop a REST API where you gain complete control over the request and response along with API management capabilities.

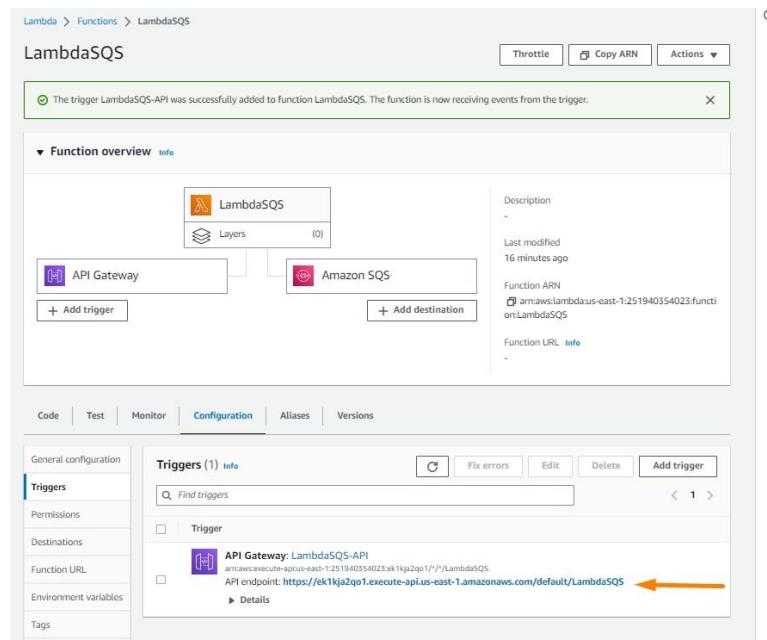
Security
Configure the security mechanism for your API endpoint.
 Open

Additional settings

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

In Configuration tab > Triggers > Click on the newly created API Gateway link.



We see “Message in a bottle sent to SQSqueue” is displayed in the URL!!

<https://ek1kja2qo1.execute-api.us-east-1.amazonaws.com/default/LambdaSQS>

“Message in a bottle sent to SQSqueue :)”



Let's go back to Amazon SQS Queue > Click on Send and receive message

Amazon SQS > Queues > Week15Project-sqs-queue

Week15Project-sqs-queue

Details Info

Name: Week15Project-sqs-queue Type: Standard ARN: arn:aws:sqs:us-east-1:251940354023:Week15Project-sqs-queue

Encryption: Amazon SQS key (SSE-SQS) URL: https://sqs.us-east-1.amazonaws.com/251940354023/Week15Project-sqs-queue

Dead-letter queue: -

Edit **Delete** **Purge** **Send and receive messages** **Start DLQ redrive**

Click on Poll for message to see if the Lambda function sent a message to the queue.

Receive messages Info

Messages available: 2 Polling duration: 30 Maximum message count: 10 Polling progress: 2 receives/second

Messages (2)

ID	Sent	Size	Receive count
ff06b0c8-bb64-4ecb-93a8-9da181dc56dc	Apr 19, 2023, 18:56:29 EDT	38 bytes	3
9be3ff4d-5e00-42ca-bb2d-4c639a57ad89	Apr 19, 2023, 18:48:10 EDT	38 bytes	3

Edit poll settings **Stop polling** **Poll for messages**

Click on the link to see what the latest message is:

Message: 5fc5d614-9a32-4de7-b859-e40c9c86f1ee

Details **Body** **Attributes**

"US Eastern current time: 09:54:00 PM"

Done

Current Time! Awesome we completed our objectives! We are able to send a message to the SQS queue by triggering our Lambda function with API gateway!



ADVANCED

- 1) Create a SNS topic using Python.
- 2) Use an email to subscribe to the SNS topic.
- 3) Create a Lambda function with a Python 3.7 or higher runtime
- 4) Modify the Lambda to trigger when new messages arrive in the SQS queue you created earlier.
- 5) The Lambda should publish the SQS message that triggered it to the SNS topic.
- 6) Validate the entire architecture works by triggering the API you created earlier. You should receive the notification from the SNS subscription.

In Amazon SNS console, create subscription tab.

The screenshot shows the 'Week15Project-sns-queue' details page in the Amazon SNS console. It displays the following information:

ID	Endpoint	Status	Protocol
2443a336-48ff-4b99-b61e-5d63c50dd...	arn:aws:lambda:us-east-1:251940354023:Week15Project-sns-queue	Confirmed	LAMBDA
82165638-56f2-4172-a162-356b6fad...	jlee7771@outlook.com	Confirmed	EMAIL

Topic ARN: Week15project-sns-quene> Protocol: Email > Endpoint: email address >Create Subscription

The screenshot shows the 'Create subscription' wizard step 1: Details. It includes the following fields:

- Topic ARN:** arn:aws:sns:us-east-1:251940354023:Week15Project-sns-queue
- Protocol:** Email
- Endpoint:** jlee7771@outlook.com

Below the form, there are optional sections for **Subscription filter policy** and **Redrive policy (dead-letter queue)**, both of which are currently collapsed.



Subscription successfully created

The screenshot shows the AWS SNS console with a green header bar indicating 'Subscription to Week15Project-sns-queue created successfully'. Below the header, the ARN of the subscription is listed as arn:aws:sns:us-east-1:251940354023:Week15Project-sns-queue:82165638-56f2-4172-a162-356b6facda7b. The main view displays the subscription details: ARN (arn:aws:sns:us-east-1:251940354023:Week15Project-sns-queue:82165638-56f2-4172-a162-356b6facda7b), Endpoint (jlee7711@outlook.com), Topic (Week15Project-sns-queue), Status (Confirmed), and Protocol (EMAIL). There are 'Edit' and 'Delete' buttons at the top right. At the bottom, there are links for 'Subscription filter policy' and 'Redrive policy (dead-letter queue)'.

Check your email: and confirm your subscription

AWS Notification - Subscription Confirmation

The email is from 'AWS Notifications <no-reply@sns.amazonaws.com>' at 4:00 PM. The recipient is 'jlee7711@outlook.com'. The subject is 'You have chosen to subscribe to the topic: arn:aws:sns:us-east-1:251940354023:Week15Project-sns-queue'. The body of the email contains a confirmation link: 'To confirm this subscription, click or visit the link below (If this was in error no action is necessary): [Confirm subscription](#)'. A note at the bottom states: 'Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)'.

Make sure to remember to delete the Lambda Function, SQS, API gateway to prevent any ongoing AWS charges

The email is from 'Simple Notification Service'. The subject is 'Subscription confirmed!'. The body of the email says: 'You have successfully subscribed. Your subscription's id is: arn:aws:sns:us-east-1:251940354023:Week15Project-sns-queue:82165638-56f2-4172-a162-356b6facda7b. If it was not your intention to subscribe, [click here to unsubscribe](#).'



Amazon Simple Email Service (SES)



What Is Amazon SES?

Welcome to the Amazon Simple Email Service (Amazon SES) Developer Guide. Amazon SES is an email platform that provides an easy, cost-effective way for you to send and receive email using your own email addresses and domains.

Amazon simple Email Service (Amazon SES) is a cloud-based email sending service designed to help digital marketers and application developers send marketing, notification, and transnational emails. ... you'll use our SMTP interface or one in all the AWS SDKs to integrate Amazon SES directly into your existing applications.

Why use Amazon SES?

Amazon SES and other AWS services. you'll send email from Amazon EC2 by using AN AWS SDK, by using the Amazon SES SMTP interface, or by making calls directly to the Amazon SES API. Use AWS Elastic beanstalk to create AN email-enabled application like a program that uses Amazon SES to send a report to customers.

Amazon SES and other AWS services:

Amazon SES integrates seamlessly with other AWS products. For example, you can:

- Add email-sending capabilities to any application. If your application runs in Amazon Elastic Compute Cloud (Amazon EC2), you can use Amazon SES to send 62,000 emails every month at no additional charge. You can send email from Amazon EC2 by using an AWS SDK, by using the Amazon SES SMTP interface or by making calls directly to the Amazon SES API.
- Use AWS Elastic Beanstalk to create an email-enabled application such as a program that uses Amazon SES to send a newsletter to customers.



- Set up Amazon Simple Notification Service (Amazon SNS) to notify you of your emails that bounced, produced a complaint, or were successfully delivered to the recipient's mail server. When you use Amazon SES to receive emails, your email content can be published to Amazon SNS topics.
- Use the AWS Management Console to set up Easy DKIM, which is a way to authenticate your emails. Although you can use Easy DKIM with any DNS provider, it is especially easy to set up when you manage your domain with [Route 53](#).
- Control user access to your email sending by using AWS Identity and Access Management (IAM).
- Store emails you receive in Amazon Simple Storage Service (Amazon S3).
- Take action on your received emails by triggering AWS Lambda functions.
- Use AWS Key Management Service (AWS KMS) to optionally encrypt the mail you receive in your Amazon S3 bucket.
- Use AWS CloudTrail to log Amazon SES API calls that you make using the console or the Amazon SES API.
- Publish your email sending events to Amazon CloudWatch or Amazon Kinesis Data Firehose. If you publish your email sending events to Kinesis Data Firehose, you can access them in Amazon Redshift, Amazon Elasticsearch Service, or Amazon S3.

Amazon SES Quick Start:

This procedure leads you through the steps to sign up for AWS, verify your email address, send your first email, think about how you'll handle bounces and complaints, and move out of the Amazon simple Email Service (Amazon SES) sandbox.

Use this procedure if you:

- Are just experimenting with Amazon SES.
- Want to send some test emails without doing any programming.
- Want to get set up in as few steps as possible.

Step 1: Sign up for AWS

- Before you can use Amazon SES, you need to sign up for AWS. When you sign up for AWS, your account is automatically signed up for all AWS services.



Step 2: Verify your email address

- Before you can send email from your email address through Amazon SES, you need to show Amazon SES that you own the email address by verifying it.

Step 3: Send your first email

- You can send an email simply by using the Amazon SES console. As a new user, your account is in a test environment called the sandbox, so you can only send email to and from email addresses that you have verified.

Step 4:

- Consider how you will handle bounces and complaints Before the next step, you need to think about how you will handle bounces and complaints. If you are sending to a small number of recipients, your process can be as simple as examining the bounce and complaint feedback that you receive by email, and then removing those recipients from your mailing list.

Step 5: Move out of the Amazon SES sandbox

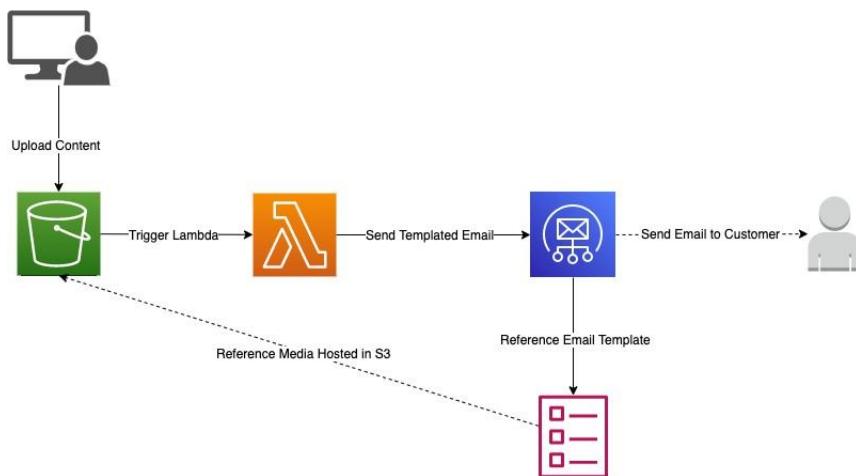
- To be able to send emails to unverified email addresses and to raise the number of emails you can send per day and how fast you can send them, your account needs to be moved out of the sandbox. This process involves opening an SES Sending Limits Increase case in Support Center.

Next steps:

- After you send a few test emails to yourself, use the Amazon SES mailbox simulator for further testing because emails to the mailbox simulator do not count towards your sending quota or your bounce and complaint rates. For more information on the mailbox simulator, see Testing Email Sending in Amazon SES .
- Monitor your sending activity, such as the number of emails that you have sent and the number that have bounced or received complaints. For more information, see Monitoring Your Amazon SES Sending Activity.
- Verify entire domains so that you can send email from any email address in your domain without verifying addresses individually. For more information, see Verifying Domains in Amazon SES.
- Increase the chance that your emails will be delivered to your recipients' inboxes instead of junk boxes by authenticating your emails. For more information, see Authenticating Your Email in Amazon SES .



Triggering S3 bucket to send mail via AWS SES via integrating Lambda

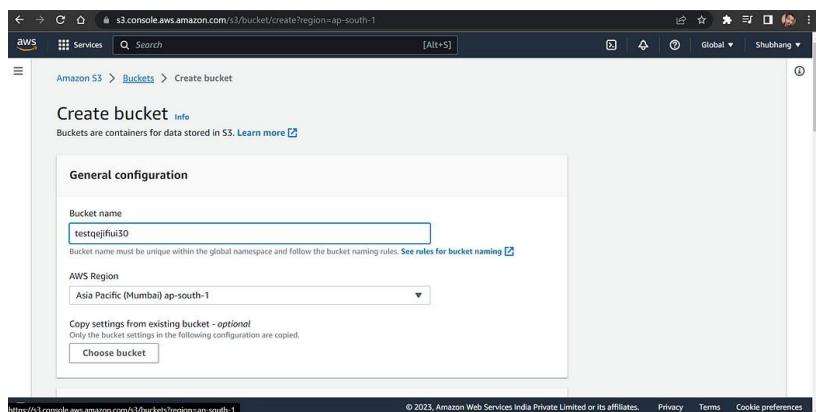


Introduction

Sending emails is a common requirement for many applications, whether it's sending notifications, newsletters, or transactional emails. In this blog post, we'll explore how to leverage the power of Amazon Simple Email Service (SES) and AWS Lambda to automate the process of sending emails when new objects are uploaded to an Amazon S3 bucket. This powerful combination allows you to build scalable and efficient email notification systems.

Setting up s3

here is how you can create a bucket in S3, here only change is the name and ACL and the rest is the same





creating bucket in S3

Object Ownership

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

⚠️ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

Bucket owner preferred
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Object writer
The object writer remains the object owner.

💡 If you want to enforce object ownership for new objects only, your bucket policy must specify that the 'bucket-owner-full-control' canned ACL is required for object uploads. [Learn more ↗](#)

enable ACL

Buckets (1) [info](#)

Buckets are containers for data stored in S3. [Learn more ↗](#)

Name	AWS Region	Access	Creation date
testopjifui30	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	July 6, 2023, 23:13:59 (UTC+05:30)

here our bucket is created simple.

Setting Lambda

below shows how you can create a lambda function handling our Python code

Create function

AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
s3_to_Lambda
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.10

Architecture



The screenshot shows the AWS Lambda console. The top navigation bar includes the AWS logo, Services, Search, and a Mumbai region dropdown. The main title is "Lambda > Functions > s3_to_Lambda". The function name is "s3_to_Lambda". Below the title are buttons for Throttle, Copy ARN, and Actions. A "Function overview" section is expanded, showing a thumbnail of the function icon, a "Layers" section (0 layers), and a "Description" field which is empty. The "Last modified" field shows "14 seconds ago". The "Function ARN" is listed as "arn:aws:lambda:ap-south-1:640714877922:function:s3_to_Lambda". A "Function URL" link is also present.

Triggering S3 and integrating Lambda with S3 object here is how step by step you can set up S3 object to trigger when something is uploaded in our bucket.

The screenshot shows the AWS S3 console. The top navigation bar includes the AWS logo, Services, Search, and a Global region dropdown. The main title is "Amazon S3 > Buckets > testqejifui30". The bucket name is "testqejifui30". Below the title are tabs for Objects, Properties (which is selected), Permissions, Metrics, Management, and Access Points. The "Bucket overview" section displays the AWS Region (Asia Pacific (Mumbai) ap-south-1), Amazon Resource Name (ARN) (arn:aws:s3:::testqejifui30), and Creation date (July 6, 2023, 23:13:59 (UTC+0:30)).

The screenshot shows the "Event notifications" tab for the "testqejifui30" bucket. It displays a table with columns: Name, Event types, Filters, Destination type, and Destination. There are no notifications listed under "Event notifications (0)". A "Create event notification" button is available. Below this, there is a section for "Amazon EventBridge" with a note about using it for event-driven applications. The bottom of the screen shows the browser address bar with the URL "s3.console.aws.amazon.com/s3/buckets/testqejifui30/property/notification/create?region=ap-south-1".



The screenshot shows the AWS S3 console interface for creating an event notification. In the 'General configuration' section, the event name is set to 'test_event'. Under 'Prefix - optional', there is a placeholder 'images/'. Under 'Suffix - optional', there is a placeholder '.jpg'. In the 'Destination' section, the 'Lambda function' option is selected, and the function name 's3_to_Lambda' is chosen from the dropdown. The 'Event types' section contains two main groups: 'Object creation' and 'Object removal'. Under 'Object creation', 'Put' (s3:ObjectCreated:Put) is checked. Under 'Object removal', 'Permanently deleted' (s3:ObjectRemoved:Delete) is checked.

here you have to connect the lambda with which you want your bucket to react or trigger when something is put in it.

The screenshot shows the AWS Lambda console for the function 's3_to_Lambda'. The 'Function overview' section shows the function name and its ARN. It indicates that the function was last modified 1 minute ago. The 'Triggers' section shows an S3 trigger for the bucket 'testqejiufl30'. The 'Layers' section shows '(0)' layers associated with the function.

refresh the lambda page and you'll see this, which shows our integration is done here with S3.



Giving permissions

our lambda needs permissions to perform activities, so here we will navigate to the permission section of our lambda and select these 2 permissions shown below as our work is with these two these permissions basically mean that our lambda is permitted to do anything with S3 and SES i.e full access

The screenshot shows the AWS IAM Permissions page. The left sidebar has 'Identity and Access Management (IAM)' selected. The main area shows 'Permissions policies (3)'. Three policies are listed:

Policy name	Type	Description
AWSLambdaBasicExecutionRole-9905da0b-530f-4d...	Customer managed	
AmazonS3FullAccess	AWS managed	Provides full access to all buck...
AmazonSESFullAccess	AWS managed	Provides full access to Amazor...

We'll set up our lambda to perform actions

```
import json  
import boto3
```

```
s3_client = boto3.client('s3')
```

```
def lambda_handler(event, context):
```

```
    # function used to get object, here we know our file name, for unknown  
file we can use prefix in S3 trigger event
```

```
a = s3_client.get_object(Bucket='testqejifiui30',Key='email_test')  
a = a['Body'].read()
```

```
object_data = a.decode('utf-8') # this is used to make this file in simple text
```

```
object_data = json.loads(object_data) # the above file comes in json format to extract the  
information
```

```
ses_client = boto3.client('ses', region_name='ap-south-1') # Replace with your desired region  
response = ses_client.send_email(
```

```
Source='shubhangorei@gmail.com', # Replace with the sender's email address
```

```
Destination={
```

```
    'ToAddresses': object_data['emails'] # Replace with the recipient's email address
```

```
},
```



```
Message={  
    'Subject': {  
        'Data': 'this is my subject', # Replace with the email subject  
    },  
    'Body': {  
        'Text': {  
            'Data': 'body is good', # Replace with the email body  
        }  
    }  
}  
)  
return {  
    'statusCode': 200,  
    'body': json.dumps('mail send')  
}
```

The Lambda function retrieves the object from S3, reads its content, converts it to a JSON format, and extracts the email addresses. Then, it uses the Boto3 library to interact with Amazon SES and sends the email to the recipients specified in the JSON file.

Make sure to replace the following placeholders in the code with your own values:

- Replace ‘shubhangorei@gmail.com’ with the sender’s mail used to create SES , below we’ll see how it is created.
- Replace ‘this is my subject’ with the email subject.
- Replace ‘body is good’ with the email body.
- Adjust the region name (‘ap-south-1’) and bucket name (‘testqejifiui30’) according to your specific configuration.

The screenshot shows the AWS Amazon SES Configuration: Verified identities page. The left sidebar has a navigation menu with options like Account dashboard, Reputation metrics, SMTP settings, Configuration (with sub-options like Verified identities, Configuration sets, Dedicated IPs, Email templates, Suppression list, and Cross-account notifications), and Virtual Deliverability Manager. The main content area is titled 'Verified identities' and contains a message about identity status update. Below this is a table titled 'Identities (2) Info' showing two entries:

Identity	Identity type	Identity status
shubhangorei2@gmail.com	Email address	Verified
shubhangorei@gmail.com	Email address	Verified



Additionally, ensure that the appropriate IAM role is assigned to the Lambda function to grant it the necessary permissions for accessing S3 and sending emails using Amazon SES.

Once you have customized the code and deployed the Lambda function, it will be triggered whenever a new object is uploaded to the specified S3 bucket, and it will send an email to the recipients specified in the JSON file.

Please note that the above code assumes that the email addresses are provided in the JSON file under the key ‘emails’. Adjust the code accordingly if your JSON file has a different structure.

The screenshot shows the AWS Lambda console interface. At the top, there's a green success message: "Successfully updated the function s3_to_Lambda." Below this, the left sidebar shows the environment and the function name "s3_to_Lambda". The main area displays the "lambda_function.py" code:

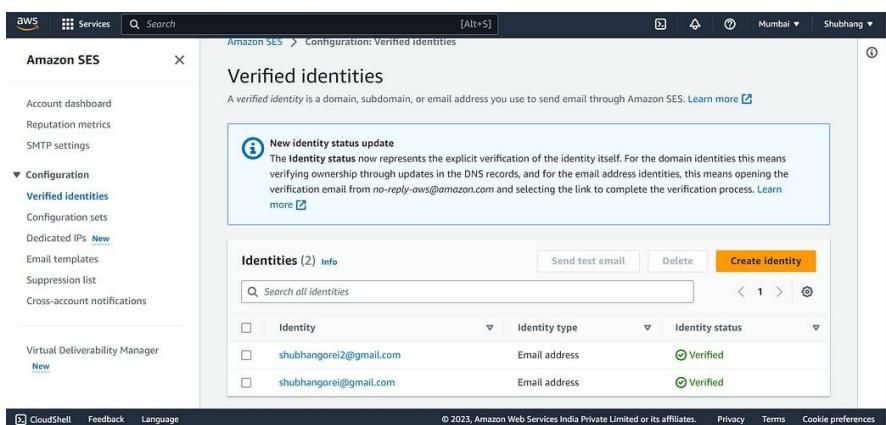
```
1 import json
2 import boto3
3
4 s3_client = boto3.client('s3')
5
6 def lambda_handler(event, context):
7     # function used to get object, here we know our file name, for unknown file we can use prefix in S3 trigger event
8
9     a = s3_client.get_object(Bucket="testqejifuiui30",Key="email_test")
10    a = a['Body'].read()
11
12    object_data = a.decode('utf-8') # this is used to make this file in simple text
13
14    object_data = json.loads(object_data) # the above file comes in json format to extract the information
15
16    ses_client = boto3.client('ses', region_name='ap-south-1') # Replace with your desired region
17
18    response = ses_client.send_email(
19        Source="shubhangorei@gmail.com", # Replace with the sender's email address
20        Destination={
21            'ToAddresses': object_data['emails'] # Replace with the recipient's email address
22        },
23        Message={
24            'Subject': {
25                'Data': 'this is my subject', # Replace with the email subject
26            },
27            'Body': {
28                'Text': {
29                    'Data': 'body is good', # Replace with the email body
30                }
31            }
32        }
33    )
34
```

Setting up SES

Hear my account is in the test phase, as all new users are in this phase if you want to go to the production phase where we can send mail to any unknown ID, you can request for it in the SES interface.

Here I have verified the mail which I'm going to upload and send mail otherwise an error will show up

It's important to note that when using Amazon SES in the sandbox mode, you can only send emails to verified email addresses. This means that the email addresses specified in the JSON file need to be verified in the Amazon SES console. If they are not verified, the emails will not be sent.



this is my file which I'm going to upload in S3, it's in JSON format

```
{
  "emails": ["vmrreddy913@gmail.com"]
}
```

Now upload this file in S3

this part of our code will do our stuff, you can customize this according to your needs, the code is all explanatory.

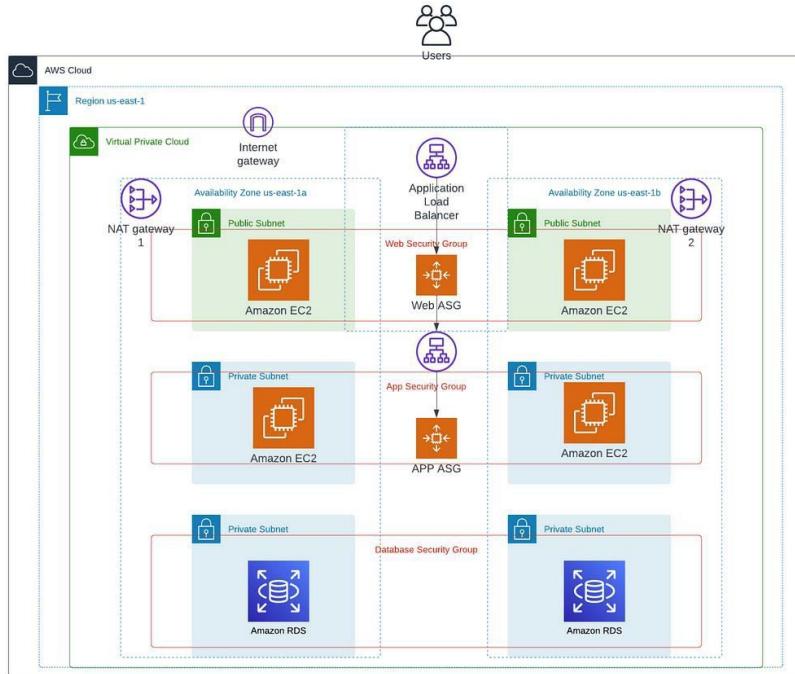
```
ses_client = boto3.client('ses', region_name='ap-south-1') # Replace with your desired region

response = ses_client.send_email(
    Source='vmrreddy913@gmail.com', # Replace with the sender's email address
    Destination={
        'ToAddresses': object_data['emails'] # Replace with the recipient's email address
    },
    Message={
        'Subject': {
            'Data': 'this is my subject', # Replace with the email subject
        },
        'Body': {
            'Text': {
                'Data': 'body is good', # Replace with the email body
            }
        }
    }
)
```

now upload the file in your created bucket



Creating a Highly Available 3-Tier Architecture for Web Applications in AWS



AWS provides a wide range of resources for developing and managing cloud applications, which can be customized to construct highly dependable and resilient cloud infrastructures. Suppose you are tasked with developing a three-tier architecture that is readily available for your organization's new web application. This tutorial is extensive but comprehensive. You may want to bookmark this guide for future reference on creating web, application, and data tiers.

What is a 3-Tier Architecture?

A three-tier architecture comprises three layers, namely the presentation tier, the application tier, and the data tier. The presentation tier serves as the front-end, hosting the user interface, such as the website that users or clients interact with. The application tier, commonly referred to as the back-end, processes the data. Finally, the data tier is responsible for data storage and management.

Benefits of a 3 Tier Architecture:

Scalability: Each tier can scale independently, allowing organizations to optimize their resources and minimize costs.

Reliability: Each tier can be replicated across multiple servers, improving application availability and reliability.

Performance: By dividing the application into separate layers, 3-tier architecture reduces network traffic and enhances application performance.



Security: Each tier can have its own security group, allowing different organizations to implement customized security measures for each layer.

Reduced development time: Different teams can work on different tiers simultaneously, resulting in faster deployments.

Flexibility: Each tier can be developed using different technologies and programming languages, enabling organizations to leverage the best tools for each layer.

Part 1:

Creating a VPC and Subnets

Using the architecture diagram as a reference, we will need to start by creating a new VPC with 2 public subnets and 4 private subnets.

Log into the AWS management console and click the Create VPC button.

The screenshot shows the AWS VPC dashboard. On the left sidebar, under 'Virtual private cloud', there is a 'Your VPCs' section with various options like Subnets, Route tables, Internet gateways, etc. In the center, there is a 'Create VPC' button highlighted with a yellow box. Below it, there's a 'Launch EC2 Instances' button. A note says 'Note: Your Instances will launch in the US East region.' To the right, there are sections for 'Service Health', 'Settings', 'Additional Information', and 'AWS Network Manager'. The 'Resources by Region' section shows counts for VPCs (3), NAT Gateways (1), Subnets (14), VPC Peering Connections (0), Route Tables (10), Network ACLs (3), Internet Gateways (4), Security Groups (22), Egress-only Internet Gateways (0), and Customer Gateways (0).

We are going to create a VPC with multiple public and private subnets, availability zones, and more, so let's choose “VPC and more.”

Name your VPC. I am using the auto-assigned IPV4-CIDR block of “10.0.0.0/16.” Choose these settings:

no IPv6

default Tenancy

2 Availability Zones

2 public subnets

4 private subnets



A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources
 VPC only VPC and more

Name tag auto-generation [Info](#)
Enter a value for the Name tag. This value will be used to auto-generate Name tags for other resources in the VPC.
 Auto-generate project

IPv4 CIDR block [Info](#)
Show the starting IP and the size of your VPC using CIDR notation.
10.0.0.0/16 65,536 IP's

Tenancy [Info](#)
Default

Number of Availability Zones (AZs) [Info](#)
Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.
 1 2 3

► Customize AZs

Number of public subnets [Info](#)
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.
 0 2 4

Number of private subnets [Info](#)
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.
 0 1 4

► Customize subnets CIDR blocks

NAT gateways (\$) [Info](#)
Choose the number of Availability Zones (AZs) in which to create NAT gateways. There is no charge for each NAT gateway.
 None In 1 AZ 1 per AZ

Preview

VPC [Show details](#)
Your AWS virtual network
project-vpc

Subnets (6)
Subnets within this VPC

us-east-1a
project-subnet-public1-us-east-1a
project-subnet-private1-us-east-1a
project-subnet-private3-us-east-1a

us-east-1b
project-subnet-public2-us-east-1b
project-subnet-private2-us-east-1b
project-subnet-private4-us-east-1b

Route tables (5)
Route network traffic to resources

project-rtb-public
project-rtb-private1-us-east-1a
project-rtb-private2-us-east-1b
project-rtb-private3-us-east-1a
project-rtb-private4-us-east-1b

Next, for Nat gateway choose “in 1 AZ,” none for VPC endpoints, and leave the Enable DNS hostnames and Enable DNS resolutions boxes checked.

Before we create the VPC expand the customize AZ’s and customize subnets CIDR blocks tabs.

Number of Availability Zones (AZs) [Info](#)
Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

1	2	3
---	---	---

▼ **Customize AZs**

First availability zone
us-east-1a

Second availability zone
us-east-1b

Number of public subnets [Info](#)
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

0	2
---	---

Number of private subnets [Info](#)
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.

0	2	4
---	---	---

▼ **Customize subnets CIDR blocks**



Public subnet CIDR block in us-east-1a

10.0.0.0/20

4,096 IPs

Public subnet CIDR block in us-east-1b

10.0.16.0/20

4,096 IPs

Private subnet CIDR block in us-east-1a

10.0.128.0/20

4,096 IPs

Private subnet CIDR block in us-east-1b

10.0.144.0/20

4,096 IPs

Private subnet CIDR block in us-east-1a

10.0.160.0/20

4,096 IPs

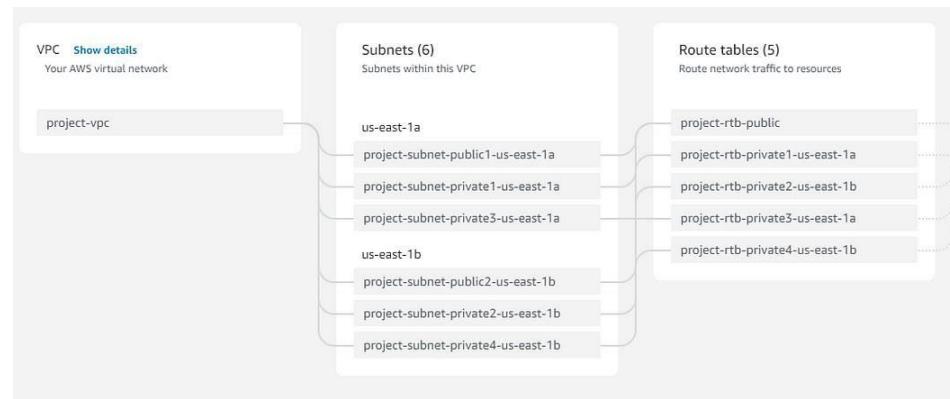
Private subnet CIDR block in us-east-1b

10.0.176.0/20

4,096 IPs

Click the “Create VPC” button.

The diagram below highlights the route that your new VPC will take.



You will then be shown a workflow chart that shows your resources being created.



Create VPC workflow

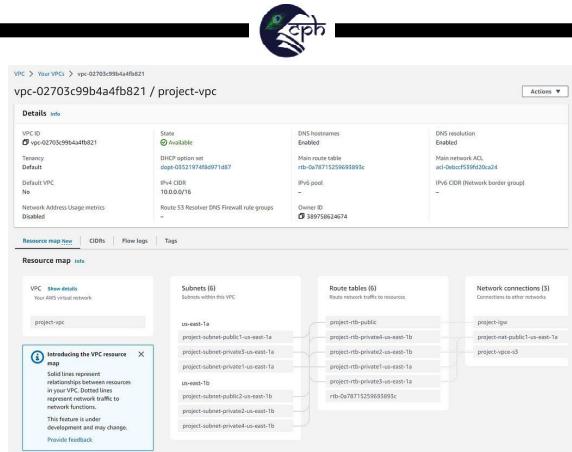
Create subnet

24%

▼ Details

- ✓ Create VPC: [vpc-02703c99b4a4fb821](#)
- ✓ Enable DNS hostnames
- ✓ Enable DNS resolution
- ✓ Verifying VPC creation: [vpc-02703c99b4a4fb821](#)
- ✓ Create S3 endpoint: [vpce-0dc40d8a903233b24](#)
- ✓ Create subnet: [subnet-0a3ca94b6aa57e2a4](#)
- ✓ Create subnet: [subnet-0f76c206ef6293ac1](#)
- ✓ Create subnet: [subnet-064828e076c2a7bc5](#)
- ⊕ Create subnet
- ⊕ Create subnet
- ⊕ Create subnet
- ⊕ Create internet gateway
- ⊕ Attach internet gateway to the VPC
- ⊕ Create route table
- ⊕ Create route
- ⊕ Associate route table
- ⊕ Associate route table
- ⊕ Allocate elastic IP
- ⊕ Create NAT gateway
- ⊕ Wait for NAT Gateways to activate
- ⊕ Create route table
- ⊕ Create route
- ⊕ Associate route table
- ⊕ Create route table
- ⊕ Create route
- ⊕ Associate route table
- ⊕ Create route table
- ⊕ Create route
- ⊕ Associate route table
- ⊕ Create route table
- ⊕ Create route
- ⊕ Associate route table
- ⊕ Verify route table creation
- ⊕ Associate S3 endpoint with private subnet route tables: [vpce-0dc40d8a903233b24](#)

View your new VPC after everything is created.



Next click on the Subnets tab in the VPC console. Select one of the new subnets that was created, then under the “Actions” tab, expand the down arrow and select “Edit subnet settings.”

Name	Subnet ID	State	VPC	IPv4 CIDR
<input checked="" type="checkbox"/> project-subnet-private3-us-east-1a	subnet-05b2eab8c180ef648	Available	vpc-02703c99b4a4fb821 proj...	10.0.160.0/20
<input type="checkbox"/> project-subnet-private1-us-east-1a	subnet-06482be076c2a7bc5	Available	vpc-02703c99b4a4fb821 proj...	10.0.128.0/20
<input type="checkbox"/> project-subnet-private2-us-east-1b	subnet-0609c808b75465097	Available	vpc-02703c99b4a4fb821 proj...	10.0.144.0/20
<input type="checkbox"/> project-private1-us-east-1a	subnet-02e850937e163a2b	Available	vpc-030489427fb8c553b my...	10.0.128.0/20
<input type="checkbox"/> my-subnet	subnet-09cafde376d54d717	Available	vpc-01f17772c79b30475	172.31.0.0/16
<input type="checkbox"/> subnet-public1-us-east-1a	subnet-01e95035718b30c12	Available	vpc-030489427fb8c553b my...	10.0.0.0/20
<input type="checkbox"/> project-subnet-public2-us-east-1b	subnet-07f6206fe6293ac1	Available	vpc-02703c99b4a4fb821 proj...	10.0.160.0/20
<input type="checkbox"/> project-subnet-private4-us-east-1b	subnet-05822727937de8e9	Available	vpc-02703c99b4a4fb821 proj...	10.0.176.0/20
<input type="checkbox"/> project-subnet-public1-us-east-1a	subnet-03c9466a57e2d4	Available	vpc-02703c99b4a4fb821 proj...	10.0.0.0/20



Check “Enable auto-assign IPv4 address” and “Save.” We need to do this for all 6 new subnets that were created.

Update Web Tier Public Route Table:

We need to navigate to the route table tag under the VPC dashboard to make sure that the Route table that was automatically created is associated with the correct subnets. Below I highlighted in green the correct public subnets that already associated. If you have none or you only created a stand-alone VPC, you will have to click on “edit subnet associations” and select the ones needed.

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0b4963f5d345fa11	Active	No
10.0.0.16	local	Active	No

Part 2: Creating a Web Server Tier

Next, we will create our first tier that represents our front-end user interface (web interface). We will create an auto scaling group of EC2 instances that will host a custom webpage for us. Start by heading to

Next, we are going to launch an EC2 instance.

Name your instance and select an AMI. Please use Amazon linux2; I used the updated version and was not able to ssh into it

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-04823729c75214919 (64-bit (x86)) / ami-000a54989336780b5 (64-bit (Arm))
Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.
Platform: amazon Root device type: ebs Virtualization: hvm ENA enabled: Yes
Select
64-bit (x86)
64-bit (Arm)



Name and tags [Info](#)

Name Add additional tags

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

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

AMI from catalog Recents My AMIs Quick Start

Amazon Machine Image (AMI)
amzn2-ami-kernel-5.10-hvm-2.0.20230628.0-x86_64-gp2
ami-04823729c75214919

Verified provider Free tier eligible

Browse more AMIs Including AMIs from AWS, Marketplace and the Community

Catalog Published Architecture Virtualization Root device type ENA Enabled

Quickstart AMIs 2023-06-28T20:23:57.00Z x86_64 hvm ebs Yes

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t3.micro is unavailable) instance usage on free tier AMIs per month, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel Launch instance Review commands

Select the key pair that you will use, and make sure to select your new VPC and the correct subnet. Auto-assign IP should be enabled.

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 - *required* Create new key pair

Network settings [Info](#)

VPC - *required* [Info](#)

vpc-05df2f0360171ad65 (3-tier-architecture-vpc)
10.0.0.0/16

Subnet Info

subnet-0b0b6abce9ff076aa 3-tier-architecture-subnet-public1-us-east-1a
VPC: vpc-05df2f0360171ad65 Owner: 389758624674 Availability Zone: us-east-1a
IP addresses available: 4090 CIDR: 10.0.0.0/20

Create new subnet

Auto-assign public IP [Info](#)

Enable



Create a new security group. For inbound security group rules, add rules for ssh, HTTP, and HTTPS from anywhere. This is not standard or safe practice, but for this demonstration it is fine.

Firewall (security groups) [Info](#)

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

[Create security group](#)

[Select existing security group](#)

Security group name - required

Company-Web-Tier-SG

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](#)

Company-Web-Tier-SG

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

[Remove](#)

Type [Info](#)

ssh

Protocol [Info](#)

TCP

Port range [Info](#)

22

Source type [Info](#)

Anywhere

Source [Info](#)

Add CIDR, prefix list or security

Description - optional [Info](#)

e.g. SSH for admin desktop

0.0.0.0/0 [X](#) ::/0 [X](#)

▼ Security group rule 2 (TCP, 80, Multiple sources)

[Remove](#)

Type [Info](#)

HTTP

Protocol [Info](#)

TCP

Port range [Info](#)

80

Source type [Info](#)

Anywhere

Source [Info](#)

Add CIDR, prefix list or security

Description - optional [Info](#)

e.g. SSH for admin desktop

0.0.0.0/0 [X](#) ::/0 [X](#)

▼ Security group rule 3 (TCP, 443, Multiple sources)

[Remove](#)

Type [Info](#)

HTTPS

Protocol [Info](#)

TCP

Port range [Info](#)

443

Source type [Info](#)

Anywhere

Source [Info](#)

Add CIDR, prefix list or security

Description - optional [Info](#)

e.g. SSH for admin desktop

Leave the configuration storage settings alone. In the Advanced details, head all the way to the bottom. We are going to use a script to launch an Apache web server when the instance starts.



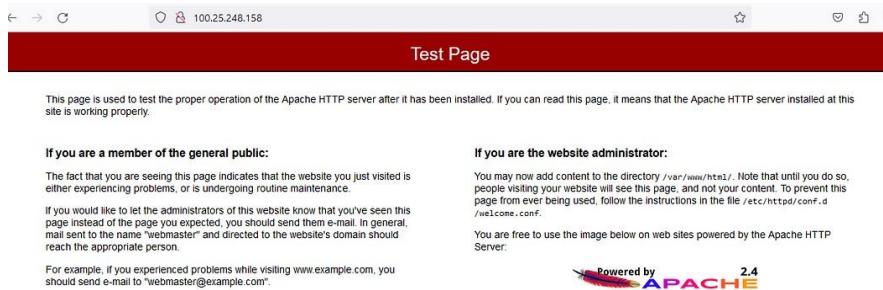
```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

cd/var/www/html

sudo echo "<h1>My Company Website</h1>">index.html
```

Launch your new instance!

Once your instance is up and running, copy the public IP address and paste it into a web server.



I am having an issue getting mine to say “My Company Website”. I will solve this at a later date. All that really matters is that the instance has Apache installed, and we can reach it. Next, I will SSH into the instance to make sure that works as well.

For this project to work we need to create an auto scaling group that we attach to our EC2 instance. This will increase our reliability and availability. Next, we are going to create a launch template. Before we do this, we will need to define a launch template; this template will

outline what resources are going to be allocated when an auto scaling group launches on-demand instances. Under the EC2 dashboard, select Launch Templates, and click the “Create launch template” button.



New EC2 Experience Tell us what you think

Compute

EC2 launch templates

Streamline, simplify and standardize instance launches

Use launch templates to automate instance launches, simplify permission policies, and enforce best practices across your organization. See launch parameters in a template that can be used for on-demand launches and with managed services, including EC2 Auto Scaling and EC2 Fleet. Easily update your launch parameters by creating a new launch template version.

Benefits and features

Streamline provisioning Minimize steps to provision instances. With EC2 Auto Scaling, updates to a launch template can be automatically passed to an Auto Scaling group. Learn more	Simplify permissions Create shorter, easier to manage IAM policies. Learn more
Governance Ensure best practices are used across your organization. Learn more	

New launch template

Create launch template

Documentation

Documentation [View](#)

API reference [View](#)

Name your template, and check the box to “provide guidance.”

Launch template name and description

Launch template name - required

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

Max 255 chars

Auto Scaling guidance [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► [Template tags](#)

► [Source template](#)

Use our recently launched AMI t2.micro instance type and select your key pair.



Recents

My AMIs

Quick Start

 Currently in use[Browse more AMIs](#)

Including AMIs from
AWS, Marketplace and
the Community

Amazon Machine Image (AMI)

amzn2-ami-kernel-5.10-hvm-2.0.20230628.0-x86_64-gp2

ami-04823729c75214919

2023-06-28T20:23:37.000Z architecture: 64-bit (x86) Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20230628.0 x86_64 HVM gp2

Architecture

x86_64

AMI ID

ami-04823729c75214919

Verified provider

▼ Instance type [Info](#)

[Advanced](#)

Instance type

t2.micro

Free tier eligible

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

All generations

[Compare instance types](#)

▼ 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

LUITThursday

[Create new key pair](#)

For the firewall, use “Select existing security group,” and make sure the security group (SG) that we created for the web tier is selected. Under Advanced network configuration, enable Auto-assign public IP.



▼ Network settings [Info](#)

Subnet [Info](#)

Don't include in launch template

Create new subnet [\[x\]](#)

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

Common security groups [Info](#)

Select security groups

Compare security group rules

Company-Web-Tier-SG sg-0b7fe3df4ef842ad4 X
VPC: vpc-05df2f0360171ad65

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Advanced network configuration

Network interface 1

Device index [Info](#)

0

Network interface [Info](#)

New interface

Description [Info](#)

Remove

Existing network interfaces are not recommended when creating a template for auto-scaling.

Subnet [Info](#)

Don't include in launch template

Not applicable for EC2 Auto Scaling

Security groups [Info](#)

Select security groups ▾

Auto-assign public IP [Info](#)

Enable



Primary IP [Info](#)

Not applicable for EC2 Auto Scaling

Secondary IP [Info](#)

Don't include in launch tem... ▾

IPv6 IPs [Info](#)

Don't include in launch tem... ▾

IPv4 Prefixes [Info](#)

Don't include in launch tem... ▾

IPv6 Prefixes [Info](#)

Don't include in launch tem... ▾

Delete on termination [Info](#)

Don't include in launch tem... ▾

Elastic Fabric Adapter [Info](#)

Enable

EFA is only compatible with certain instance types.

Network card index [Info](#)

Don't include in launch tem... ▾

The selected instance type does not support multiple network cards.

Add network interface

We are going to leave the storage options alone for now. Click on the Advanced details tab, scroll down, and enter the same script as we did earlier for our EC2 instance.



Click the “Create launch template” button.

The screenshot shows the AWS Lambda function configuration page. On the left, there are several dropdown menus for metadata settings like "Metadata version" (set to "Don't include in launch template"), "Metadata response hop limit" (set to "Don't include in launch template"), and "Allow tags in metadata" (set to "Don't include in launch template"). Below these is a "User data - optional" section with a file upload field containing a shell script:

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

cd/v/www/html

sudo echo "<h1>My Company Website</h1>"> index.html
```

On the right, there is a large text area with a note about instance types and usage, followed by a "Cancel" and a prominent orange "Create launch template" button.

Navigate to the autoscaling tab at the bottom of the EC2 dashboard. Click “Create auto scaling group.” The launch template that we just finished creating is the template that our auto scaling group will use to launch new EC2 instances when scaling up.

Name your auto scaling group (ASG), choose the launch template that you created, then click the Next button.

The screenshot shows the "Create Auto Scaling group" wizard. The left sidebar lists steps: Step 1 (selected), Step 2, Step 3 - optional, Step 4 - optional, Step 5 - optional, Step 6 - optional, and Step 7. The main area is titled "Choose launch template or configuration" with a sub-instruction: "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." It includes fields for "Name" (set to "My Company Web Server Auto Scaling Group") and "Launch template". The "Launch template" dropdown is set to "Company-Web-Tier-Server". The "Additional details" section shows storage (empty) and date created ("Sun Jul 23 2023 06:31:37 GMT-0400 (Eastern Daylight Time)"). At the bottom are "Cancel" and "Next" buttons.



Under Network, make sure to select the VPC that you created earlier, then also under availability zones and subnets select the public subnets that were created; yours may differ.

Click the Next button.

Choose instance launch options Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

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-05df2f0360171ad65 (3-tier-architecture-vpc)
10.0.0.0/16

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

us-east-1a | subnet-0b0b6abce9ff076aa (3-tier-architecture-subnet-public1-us-east-1a)
10.0.0.0/20

us-east-1b | subnet-027f35c3bd7718d12 (3-tier-architecture-subnet-public2-us-east-1b)
10.0.16.0/20

Create a subnet

Instance type requirements Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
Company-Web-Tier-Server	Default	A Production web server lt-0b4a7740f60e1fec6

Instance type
t2.micro

Now we are given the option to allocate a load balancer for our ASG. A load balancer will distribute the load from incoming traffic across multiple servers. This helps with availability and performance.

Select “Attach to a new load balancer” and “Application load balancer,” name your load balancer, then select “Internet facing” as this is for our web tier.



Configure advanced options - *optional* Info

Integrate your Auto Scaling group with other services to distribute network traffic across multiple servers using a load balancer or to establish service-to-service communications using VPC Lattice. 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.

Attach to a new load balancer

Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type

Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the Load Balancing console [\[?\]](#)

Application Load Balancer

HTTP, HTTPS

Network Load Balancer

TCP, UDP, TLS

Load balancer name

Name cannot be changed after the load balancer is created.

My Company Web Server Auto Scaling Group-1

Load balancer scheme

Scheme cannot be changed after the load balancer is created.

Internal

Internet-facing

Network mapping

Your new load balancer will be created using the same VPC and Availability Zone selections as your Auto Scaling group. You can select different subnets and add subnets from additional Availability Zones.

VPC

vpc-05df0360171ad65 [\[?\]](#)

3-tier-architecture-vpc

Availability Zones and subnets

You must select a single subnet for each Availability Zone enabled. Only public subnets are available for selection to support DNS resolution.

us-east-1a

subnet-0b0b6abce9ff076aa

▼

us-east-1b

subnet-027f35c3bd7718d12

▼

Your VPC and the two public subnets should already be selected.

Under the “Listeners and routing” section, select “Create a target group,” which should be on port 80 for HTTP traffic.



Listeners and routing
If you require secure listeners, or multiple listeners, you can configure them from the [Load Balancing console](#) after your load balancer is created.

Protocol	Port	Default routing (forward to)
HTTP	80	<input type="button" value="Create a target group"/> ▼ New target group name An instance target group with default settings will be created. <input type="button" value="My Company Web Server Auto Scaling Group-1"/>

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

50 remaining

Leave No VPC Lattice service checked.

Click to turn on Elastic Load Balancing health checks.

VPC Lattice integration options [Info](#)
To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

No VPC Lattice service
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

[Create new VPC Lattice service](#)

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 [X](#)

Turn on VPC Lattice health checks
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

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.

seconds



Check “enable group metrics collection within CloudWatch.”

Additional settings

Monitoring [Info](#)

Enable group metrics collection within CloudWatch

Default instance warmup [Info](#)
The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.

Enable default instance warmup

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

Next, we configure the group size and scaling policy for our ASG. For reliability and performance, enter 2 for desired capacity and minimum capacity. For maximum capacity, enter 3.

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

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - *optional* [Info](#)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

Minimum capacity

Maximum capacity

Scaling policies - *optional*

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

Target tracking scaling policy

Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

None

Scaling policy name

Metric type

Target value

Instances need

seconds warm up before including in metric

Disable scale in to create only a scale-out policy



In scaling policies, select “Target tracking scaling policy.” The metric type should be set to “average CPU utilization” with a target value of 50. Click the Next button.

On the next screen we could add notifications through SNS topics, but I skipped this for now. Click the Next button.

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.

Add notification

Cancel **Skip to review** **Previous** **Next**

The next screen allows you to add tags which can help to search, filter, and track your ASG’s across AWS. For now, just click the Next button.

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 (0)

Add tag

50 remaining

Cancel **Previous** **Next**

Review your setting on the next page, and at the bottom click the “Create auto scaling group” button.

You should see a lovely green banner declaring your success here. After the ASG is finished updating capacity, navigate to your EC2 dashboard to confirm that your new instance has been created.

Note: In my previous examples my names were not accepted for the auto scaling group, make sure you follow the naming conventions.



Auto Scaling groups (1) info

Search your Auto Scaling groups

Name: WebTierASG

Launch template/configuration: Company-Web-Tier-Server | Version Def.: 0

Instances: 2

Status: Updating capacity...

Desired capacity: 2

Min: 2

Max: 3

Availability Zones: us-east-1a, us-east-1b

As you can see, the ASG is doing its job.

	ID	State	Type	Initial Status	Alarms	Zone	IP	Size
WebTierASG	i-0e686e41f62e28892	Running	t2.micro	Initializing	No alarms	+ us-east-1b	ec2-3-82-163-8.compute...	3.82.163.8
Kubernetes_Wk18_project	i-0d8f41f821e57514e	Running	t2.medium	2/2 checks passed	No alarms	+ us-east-1a	ec2-44-200-18-223.co...	44.200.18.223
Company Web-Tier	i-0da5a290000386f78	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	ec2-100-220-248-158.co...	100.220.248.158
-	i-031e25e3f1141ae6	Running	t2.micro	Initializing	No alarms	+ us-east-1a	ec2-3-239-189-109.co...	3.239.189.109

Before we move on it is a good idea to take a moment and connect to the instances that were created; as you can see: success!

Test Page

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

If you are the website administrator:

You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache HTTP Server:

Powered by APACHE 2.4

Part 3: Creating an Application Tier

Next up we are going to create the back-end of our 3-tier architecture. We could create an EC2 instance first, but for this portion I will start by heading to the Launch Templates tab under the EC2 dashboard.

EC2 launch templates

Streamline, simplify and standardize instance launches

Use launch templates to automate instance launches, simplify permission policies, and enforce best practices across your organization. Save launch parameters in a template that can be used for on-demand launches and with managed services, including EC2 Auto Scaling and EC2 Fleet. Easily update your launch parameters by creating a new launch template version.

New launch template

Create launch template

Name your new template, and select the Guidance tab again.



Launch template name and description

Launch template name - required
Company-Application-Tier
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description
Application-Tier
Max 255 chars

Auto Scaling guidance [Info](#)
Select this if you intend to use this template with EC2 Auto Scaling
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags
► Source template

Launch template contents
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

▼ Application and OS Images (Amazon Machine Image) - required [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

AMI from catalog Recents My AMIs Quick Start

Amazon Machine Image (AMI) amzn2-ami-kernel-5.10-hvm-2.0.20230628.0-x86_64-gp2 ami-04823729c75214919	Verified provider Free tier eligible				
Catalog Quickstart AMIs	Published 2023-06-28T20:23:37.00Z	Architecture x86_64	Virtualization hvm	Root device type ebs	ENI Enabled Yes

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Select browse more AMI's, then Amazon Linux 2 for your AMI. Select t2.micro for instance type, and also select you keypair.



▼ Instance type [Info](#)

Advanced

Instance type

t2.micro Free tier eligible

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

All generations [Compare instance types](#)

▼ 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

LUITThursday [Create new key pair](#)

Under the network setting, we want to limit access to the application tier for security purposes. You don't want any public access to the the application layer or the data tier after it. We will create a new security group. Select our VPC; I realize now that for this part I could have chosen a better name!

Name your new SG and select the VPC that we created when we started.

▼ 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 group name - required

Application-Tier-SG

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](#)

Allows ssh access to application tier

VPC - required [Info](#)

vpc-05df2f0360171ad65 (3-tier-architecture-vpc) [10.0.0.0/16](#)



We will create 3 security group rules starting with ssh; use My IP as the source. For Security Group 2, use Custom TCP; the source here is the security group from our web tier (tier-1). For the third group, select All ICMP-IPv4, and set the Source type as anywhere. This will allow us to ping our application tier from the public internet to test if traffic is routed properly.

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 0, sg-0b7fe3df4ef842ad4) Remove

Type Info Custom TCP	Protocol Info TCP	Port range Info 0
Source type Info Custom	Source Info <input type="text"/> Add CIDR, prefix list or security	Description - optional Info e.g. SSH for admin desktop sg-0b7fe3df4ef842ad4 X

▼ Security group rule 2 (TCP, 22, 0.0.0.0/0) Remove

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

▼ Security group rule 3 (ICMP, All, 0.0.0.0/0) Remove

Type Info All ICMP - IPv4	Protocol Info ICMP	Port range Info All
Source type Info Anywhere	Source Info <input type="text"/> Add CIDR, prefix list or security	Description - optional Info e.g. SSH for admin desktop 0.0.0.0/0 X

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. X

Add security group rule

► Advanced network configuration

This is my updated screenshot

We will again leave the storage volumes alone, and head to the bottom of Advanced details to enter our script. And then click next.



Metadata response hop limit [Info](#)

Don't include in launch template

Allow tags in metadata [Info](#)

Don't include in launch template

User data - optional [Info](#)

Upload a file with your user data or enter it in the field.

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

cd/var/www/html

sudo echo "<h1>My Company Website</h1>">index.html
```

Cancel
Create launch template

EC2 > Launch templates > Create launch template

 **Success**
Successfully created Company-Application-Tier (t2-06cf1f6e1fd240f48)

▶ Actions log

Next steps

Launch an instance

With On-Demand instances, you pay for compute capacity by the second (for Linux, with a minimum of 60 seconds) or by the hour (for all other operating systems) with no long-term commitments or upfront payments. Launch an On-Demand instance from your launch template.

[Launch instance from this template](#)

Create an Auto Scaling group from your template

Amazon EC2 Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

[Create Auto Scaling group](#)

Create Spot Fleet

A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance of each instance type in each Availability Zone is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Spot instances are well-suited for data-analysis, batch jobs, background processing, and optional tasks.

[Create Spot Fleet](#)

[View launch templates](#)

After I had created the launch template, I realized I made a couple of mistakes. Instead of modifying the template I decided to start from scratch, but again upon further reflection, in the future I would just modify the template as this would save time, and we all know time is money. The whole reason for these exercises is to learn, right!

I went back to the security group and updated my inbound rules.

	sg-0aacf4be2fe789806	Application-Tier-5G	vpc-05df2f0360171ad65	Allows ssh access to application tier	389758624674	3 Permission entries	1 Permission entries
	sg-05ddcc43581a6be1	launch-wizard-1	vpc-01f17772c79b30475	launch-wizard-1 creates	389758624674	3 Permission entries	1 Permission entries
=							
You can now check network connectivity with Reachability Analyzer							Run Reachability Analyzer
							X

Inbound rules (3)		Edit inbound rules
	Name	Manage tags
<input type="checkbox"/>	sg-0453b7c5d92008...	Edit inbound rule
<input type="checkbox"/>	sg-0453b7c5d92008...	Edit inbound rule
<input type="checkbox"/>	sg-0453b7c5d92008...	Edit inbound rule

[Filter security group rules](#)

Once this was fixed, I went back and recreated the application layer template using the ApplicationTierSG1 that I just altered. I just double checked the security group rules and used the same settings for everything else above.



Application Tier Auto Scaling Group:

Ok, now we are ready to create our auto scaling group for the application layer. Under the EC2 dashboard go to create an auto scaling group.

The screenshot shows the 'Auto Scaling groups (1) Info' section of the EC2 dashboard. It lists a single Auto Scaling group named 'WebTierASG' with the following details:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
WebTierASG	Company-Web-Tier-Server Version Def: 2	-	-	2	2	3	us-east-1a, us-east-1b

Name your new ASG and select the proper launch template, then click the Next button.

Choose the correct VPC and 2 private subnets, then click the Next button.

Choose launch template or configuration Info

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.

Name

Auto Scaling group name
Enter a name to identify the group.

Company-Application-Tier ASG

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info

[Switch to launch configuration](#)

Launch template

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Company-Application-Tier



[Create a launch template](#)

Version

Default (1)



[Create a launch template version](#)

Description

Application-Tier

Launch template

Company-Application-Tier
lt-08cff1ee1d8240f48

Instance type

t2.micro

AMI ID

ami-04823729c75214919

Security groups

- Request Spot Instances

No

Key pair name

LUITTThursday

Security group IDs

sg-0aacf4be2ef8798d6

Additional details

Storage (volumes)

-

Date created

Sun Jul 23 2023 07:31:15 GMT-0400
(Eastern Daylight Time)

Cancel

Next

We are again given the option to attach a load balancer, and we want to do this. Select an application load balancer, name it, and set it as an internal load balancer. Double check that the VPC and subnets are correct. Mine are.



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.

Attach to a new load balancer

Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type

Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the Load Balancing console.

Application Load Balancer

HTTP, HTTPS

Network Load Balancer

TCP, UDP, TLS

Load balancer name

Name cannot be changed after the load balancer is created.

CompanyAppTierASG

Load balancer scheme

Scheme cannot be changed after the load balancer is created.

Internal

Internet-facing

Network mapping

Your new load balancer will be created using the same VPC and Availability Zone selections as your Auto Scaling group. You can select different subnets and add subnets from additional Availability Zones.

VPC

vpc-05df2f0360171ad65

3-tier-architecture-vpc

Availability Zones and subnets

You must select a single subnet for each Availability Zone enabled. Only public subnets are available for selection to support DNS resolution.

us-east-1a

subnet-0be40b047d76f319a



us-east-1b

subnet-0e4a44d5355b23088



Under “Listeners and routing” create a new target group, and use port 80 once again.

Listeners and routing

If you require secure listeners, or multiple listeners, you can configure them from the Load Balancing console after your load balancer is created.

Protocol

Port

Default routing (forward to)

HTTP

80

Create a target group



New target group name

An instance target group with default settings will be created.

CompanyAppTierASG

Tags - optional

Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add tag

50 remaining

Below I have again chosen to turn on health checks and enable group metrics within CloudWatch.



VPC Lattice integration options Info

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

No VPC Lattice service

VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service

Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

[Create new VPC Lattice service](#)

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](#)

Turn on VPC Lattice health checks

VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

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

Additional settings

Monitoring Info

Enable group metrics collection within CloudWatch

Default instance warmup Info

The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.

Enable default instance warmup

[Cancel](#)

[Skip to review](#)

[Previous](#)

[Next](#)

On the next screen set your desired capacity, minimum capacity, and maximum capacity again.



Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity
2

Minimum capacity
2

Maximum capacity
3

Scaling policies - optional

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

Target tracking scaling policy
Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

None

Scaling policy name
Target-Tracking Policy

Metric type
Average CPU utilization

Target value
50

Instances need
300 seconds warm up before including in metric

Disable scale in to create only a scale-out policy

Instance scale-in protection - optional

Instance scale-in protection
If protect from scale in is enabled, newly launched instances will be protected from scale in by default.

Enable instance scale-in protection

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

Then I have chosen to select target tracking with a CPU utilization of 50%.

Click the Next button, add notifications if you want or need and then tags. Review your new ASG settings and create it.

As you can see below, my new application layer ASG is updating the capacity.

EC2 > Auto Scaling groups								
Auto Scaling groups [2] info								
<input type="text"/> Search your Auto Scaling groups								
Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones	
Company-Application-Tier ASG	Company-Application-Tier Version Def: 0	2	Updating capacity...	2	2	3	us-east-1a, us-east-1b	Edit
WebTierASG	Company-Web-Tier-Server Version Def: 2	2	-	2	2	3	us-east-1a, us-east-1b	Edit

Once the new EC2 instances are created and running, we will try to ssh into them. If we set it up correctly, we should not be able to.

When I tried to SSH into the application tier EC2 instance running, the connection timed out, this is exactly what we want here.



```
aaronbachman@Aarons-MacBook-Pro ~ % cd Desktop/Keys  
aaronbachman@Aarons-MacBook-Pro Keys % ssh -i "LUITThursday.pem" ec2-user@ec2-54-  
-152-34-134.compute-1.amazonaws.com  
ssh: connect to host ec2-54-152-34-134.compute-1.amazonaws.com port 22: Operation  
in timed out  
aaronbachman@Aarons-MacBook-Pro Keys %
```

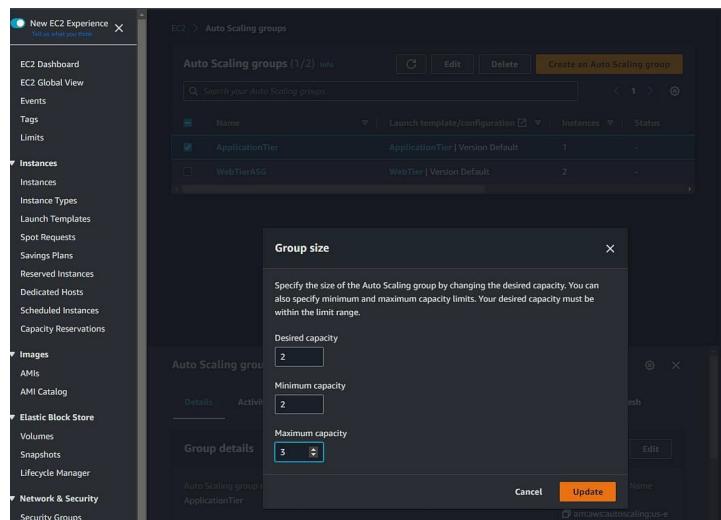
I also tried to connect using EC2 connect. This failed as well.

We still need to check if our tier-1 servers interact with our tier-2 servers. To test this, you will need to log into your tier-1 EC2 instances via SSH and run a ping command to a private IP address of our tier-2 servers. Below you can see a successful ping.

```
aaronbachman@Aarons-MacBook-Pro Keys % ssh -i "LUITThursday.pem" ec2-user@ec2-100-  
-25-248-158.compute-1.amazonaws.com  
The authenticity of host 'ec2-100-25-248-158.compute-1.amazonaws.com (100.25.248.  
.158)' can't be established.  
ED25519 key fingerprint is SHA256:K/UyzsufPjR08+PoY+srAi71kZWCMURCgcaGu67vfDs.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-100-25-248-158.compute-1.amazonaws.com' (ED25519  
) to the list of known hosts.  
  
_ _| _ _|_ )  
_ | ( _ _ / Amazon Linux 2 AMI  
___| \_ _|_ _|  
  
https://aws.amazon.com/amazon-linux-2/  
[ec2-user@ip-10-0-1-29 ~]$
```

After waiting for my new ASG to update its capacity, it only started 1 instance. I must have clicked back and reset my capacity selections to the default 1 instance. Below I updated the capacity that I wanted.

I apologize for the changes in settings here. I completed this project over a long period of time and multiple computers.





Update the application (Tier-2) Route table:

Head back to the VPC dashboard, select Route tables, and select one of the route tables that was automatically created when we created our VPC. I only have 1 subnet associated with this table so click on Edit subnet associations.

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
WebTier-subnet-private1-us-east-1a	subnet-0f0ae8db5d8db7ba8	10.0.128.0/20	-

Add another subnet that is private.

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
WebTier-subnet-private1-us-east-1a	subnet-0f0ae8db5d8db7ba8	10.0.128.0/20	-	rtb-093ef2637e8079c0c
WebTier-subnet-public1-us-east-1a	subnet-060b020e80971a83	10.0.0.0/20	-	rtb-0eddf6f2419df3f
WebTier-subnet-private3-us-east-1a	subnet-0fa241be1b1995d5	10.0.16.0/20	-	rtb-0765bab3fa53a
WebTier-subnet-private4-us-east-1b	subnet-03b00837d7aea1f	10.0.176.0/20	-	rtb-090ce3f34bf691c
WebTier-subnet-private2-us-east-1b	subnet-067bed699737ec224	10.0.144.0/20	-	rtb-0724d71150f401
WebTier-subnet-public2-us-east-1b	subnet-03f44438c9bbab395	10.0.16.0/20	-	rtb-0eddf6f2419df3f

Part 4: Created a Database Tier

Almost there! We have created 2 out of the 3 tiers and tested both successfully.

We are now going to build our database; AWS offers several types of databases but for this exercise we are going to use a MySQL RDS database.

Create a DB Subnet Group:

We will begin by creating a subnet groups. Navigate to the RDS console and on the left side menu, click “Subnet Groups” and then the orange

“Create DB subnet group.”



Amazon RDS

- Dashboard
- Databases
- Query Editor
- Performance insights
- Snapshots
- Exports in Amazon S3
- Automated backups
- Reserved instances
- Proxies

Subnet groups

- Parameter groups
- Option groups
- Custom engine versions

Events

RDS > Subnet groups > Create DB subnet group

Create DB subnet group

To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.

Subnet group details

Name
You won't be able to modify the name after your subnet group has been created.
DatabaseTierSubnetGroup

Description
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Private subnet group for database tier

VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
WebTier-vpc (vpc-0413d0b7f3ddb955a)

For the next part we need to know the availability zones for the last two subnets that were automatically created. Head back to the VPC console. Under Subnets, find the last 2 subnets that you have; make sure *not* to select the private subnets that you already used in tier 2.

<input checked="" type="checkbox"/> WebTier-subnet-private3-us-east-1a	subnet-0f5a2d1be1b1995d5	<input checked="" type="radio"/> Available	vpc-0413d0b7f3ddb955a We...
<input checked="" type="checkbox"/> WebTier-subnet-private4-us-east-1b	subnet-0c3b00837dd7aea1f	<input checked="" type="radio"/> Available	vpc-0413d0b7f3ddb955a We...

Back at the RDS console, select the availability zones that you are going to use.

Add subnets

Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone

us-east-1a X us-east-1b X

Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets

subnet-0f5a2d1be1b1995d5 (10.0.160.0/20) X
subnet-0c3b00837dd7aea1f (10.0.176.0/20) X

Subnets selected (2)

Availability zone	Subnet ID	CIDR block
us-east-1a	subnet-0f5a2d1be1b1995d5	10.0.160.0/20
us-east-1b	subnet-0c3b00837dd7aea1f	10.0.176.0/20

Cancel Create

Next up we need to select the proper subnet; the drop down menu only lists the subnet ID. Below is another screenshot of my subnets; the second column is the ID.



<input checked="" type="checkbox"/>	WebTier-subnet-private3-us-east-1a	subnet-0f5a2d1be1b1995d5
<input checked="" type="checkbox"/>	WebTier-subnet-private4-us-east-1b	subnet-0c3b00837dd7aea1f

Back in the RDS console click Create database.

Select the MySQL DB.

Successfully created DatabaseTierSubnetGroup. [View subnet group](#)

We listened to your feedback! Now, create a database with a single click using our pre-built configurations! Or choose your own configurations. [Share your feedback](#)

RDS > Create database

Create database

Choose a database creation method [Info](#)

Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

Aurora (MySQL Compatible)

Aurora (PostgreSQL Compatible)

MySQL

MariaDB

PostgreSQL

Oracle

Microsoft SQL Server

Next you can choose a multi-AZ deployment with 3 database instances. One is a primary instance with two read-only standby instances. This makes for a very reliable system, but we do not need this at this time.

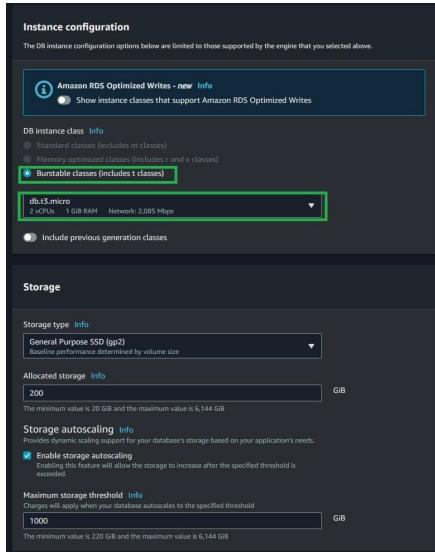
There are also availability and durability options; however, with the free-tier, none are available. We do not need them either.

Under Settings, name your DB and create a master username and password. This username and password these should be different than your AWS account user login, as it is specific to the database you are creating.

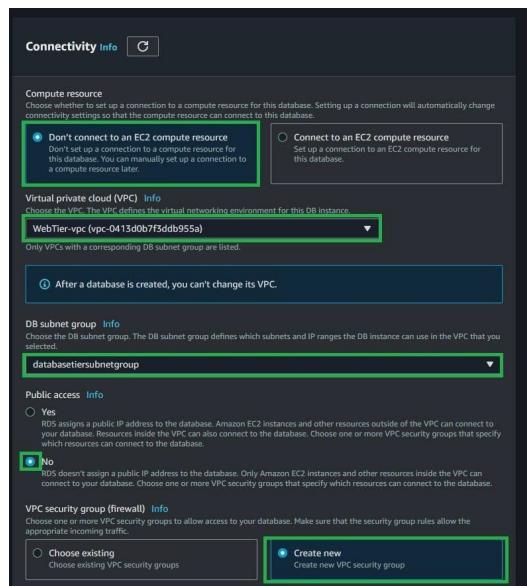


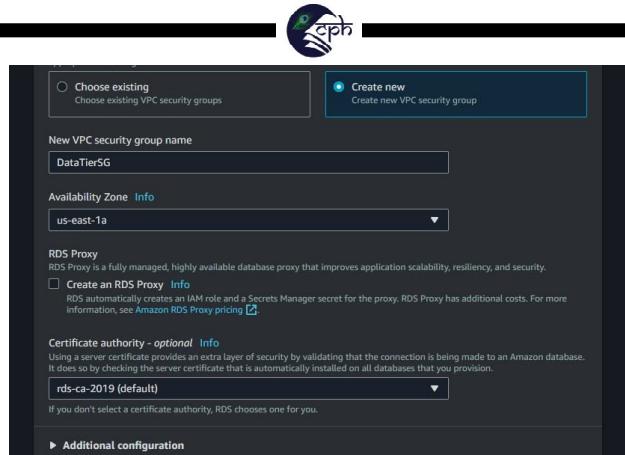
You will need your username and password. Make sure to store them in a secure place!

Under Instance configuration, the burstable classes option is pre-selected because it's the only one available for the free tier. I left my instance type as adb.t2.micro. You can add storage as needed; I left mine on default settings.

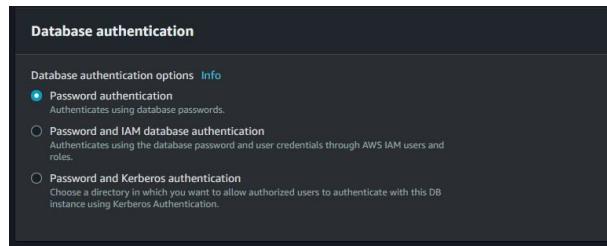


We are going to set up our network manually so choose not to connect to an EC2 resource. Select the proper VPC; the subnet group that you created earlier should be listed as default. Select Create new VPC security group (firewall).

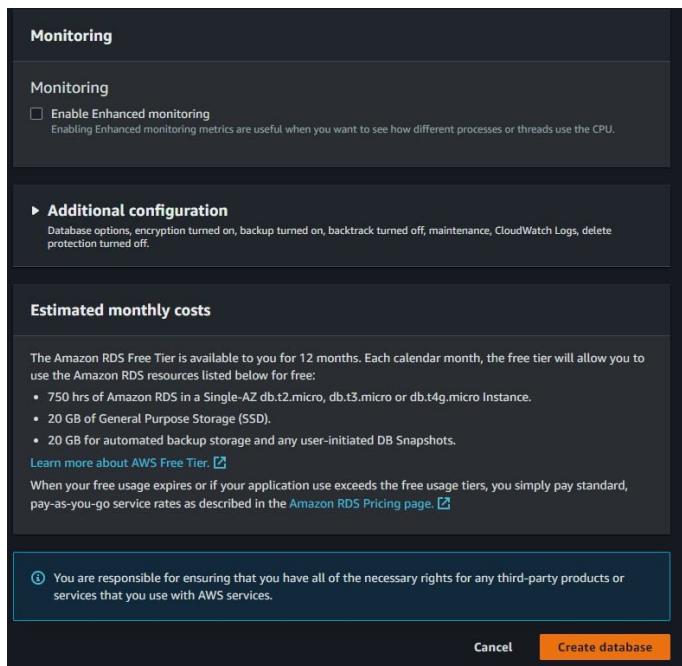




In Database authentication I left the default checked.



Click the “Create database” button.

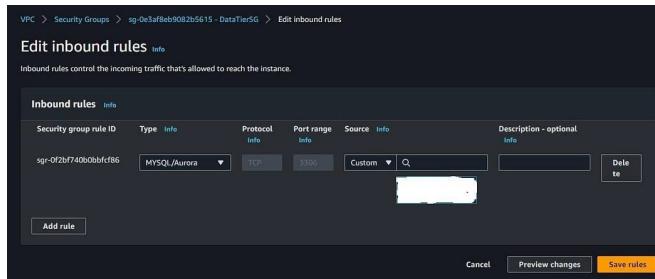




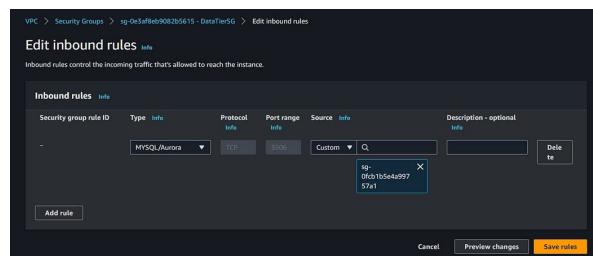
Update the Database Tier Security Group:

Navigate over to the VPC console, select Security groups on the left side menu, and then find the database tier security group you just created.

Select the security group that you just created. You need to edit inbound rules; by default the database SG has an inbound rule to allow MySQL/Aurora traffic on port 3306 from your IP address. Delete this rule.



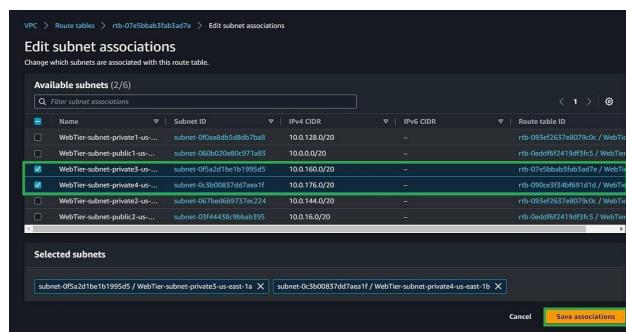
Create a new rule for MySQL/Aurora on port 3306: for the Source, select Custom to add your security group for your application layer (tier- 2 SG).



Update Tier 3 Private Route Tables:

It's the home stretch, people—I hope you're still here!

In the last step for our database tier, we need to make sure that the route table we associate with our databases private subnets has both subnets listed in subnet associations. If not, add the other subnet, and save.





Our three-tier architecture is done! We have already tested our web and application layers, but we are going to go a step further here.

Part 5: Testing

We can't directly SSH to the database, but we can use an SSH forwarding agent to achieve this.

You need to add your access key pair file to your keychain. To do this, first make sure you are in your local host (use the command *exit* to get out of any EC2 instance you're connected to). Then use the following command:

```
ssh-add -K <keypair.pem>
```

```
Identity added: LUIT_Project1.pem (LUIT_Project1.pem)
aaronbachman@Aarons-MacBook-Pro Keys %
```

Now that your key pair file is added to your keychain, the SSH agent will scan through all of the keys associated with the keychain and find your matching key.

Now reconnect to the web tier EC2; however, this time use -A to specify you want to use the SSH agent.

```
ssh -A ec2-user@<public-ip-address>
```

```
'      #_
~\_ #####_          Amazon Linux 2023
~~ \#####\_
~~ \###|
~~  \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~  \~' '->
~~  /
~~ .-.
~~ /_/
~/m/'
```

Once you are logged back into your tier-1 EC2, use the following command to check if the SSH agent forwarded the private key.

```
ssh-add -l
```

```
[ec2-user@ip-10-0-22-14 ~]$ ssh-add -l
                                     LUIT_Project1.pem (RSA)
```

Our key pair has been forwarded to our public instance. Go copy your tier-2 application layer private IP address and copy it into the next command.

```
ssh -A ec2-user@<private-ip-address>
```

```
'      #_
~\_ #####_          Amazon Linux 2023
~~ \#####\_
~~ \###|
~~  \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~  \~' '->
~~  /
~~ .-.
~~ /_/
~/m/'
```



We have now SSH'ed from your public tier 1 web instance into your private tier 2 application instance!

Testing Connectivity to the Database Tier

There are a few ways you can connect to your RDS database from your application tier. One way is to install MySQL on your private tier 2 instance to access your database. We are going to utilize this method.

While logged into your application tier instance, use this command: sudo dnf install mariadb105-server

```
Installed:
  mariadb-connector-c-3.1.13-1.amzn2023.0.3.x86_64
  mariadb-connector-c-config-3.1.13-1.amzn2023.0.3.noarch
  mariadb105-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-backup-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-common-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-cracklib-password-check-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-errmsg-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-gssapi-server-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-server-3:10.5.16-1.amzn2023.0.7.x86_64
  mariadb105-server-utils-3:10.5.16-1.amzn2023.0.7.x86_64
  mysql-selinux-1.0.4-2.amzn2023.0.3.noarch
  perl-B-1.80-477.amzn2023.0.3.x86_64
  perl-DBD-MariaDB-1.22-1.amzn2023.0.4.x86_64
  perl-DBI-1.643-7.amzn2023.0.3.x86_64
  perl-Data-Dumper-2.174-460.amzn2023.0.2.x86_64
  perl-File-Copy-2.34-477.amzn2023.0.3.noarch
  perl-FileHandle-2.03-477.amzn2023.0.3.noarch
  perl-Math-BigInt-1:1.9998.18-458.amzn2023.0.2.noarch
  perl-Math-Complex-1.59-477.amzn2023.0.3.noarch
  perl-Sys-Hostname-1.23-477.amzn2023.0.3.x86_64
  perl-base-2.27-477.amzn2023.0.3.noarch

Complete!
```

This command installs the MariDB package, which is used to read MySQL. Once installed, you should be able to use the following command to log into your RDS MySQL database. You will need your RDS endpoint, user name, and password. To find your RDS database endpoint, navigate to the database you created and find the endpoint under Connectivity & Security.

```
mysql -h <rds-database-endpoint> -P 3306 -u <username> -p
```

```
[ec2-user@ip-10-0-138-80 ~]$ mysql -h datatierdb.cmytqkhcetki.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 245
Server version: 8.0.32 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

We have now successfully connected to our MySQL database from the application tier. We have connectivity with all of our tiers!



How to Deploy a Static Website with AWS Using S3, Route 53, CloudFront, and ACM



Launching your own website is exciting, but figuring out how to get it online can be a bit overwhelming, especially if you're new to the process.

If you've created a website using for example HTML, CSS, and JavaScript and want to get it up and running on the internet, **AWS (Amazon Web Services)** can help.

In this beginner-friendly guide, we'll walk through the steps together. We'll use AWS services like **S3** (for storage), **Route 53** (for managing your domain name), **CloudFront** (for delivering your content quickly), and **ACM** (for keeping your site secure). Additionally, **we'll explore a straightforward process to purchase a domain for your website**, making it incredibly easy to have your very own web address.

By the end, you'll have a clear understanding of how to host your website on AWS and make it accessible to anyone on the web!

Is this process free? If not, how much does it cost?

Before we delve into the process, let's understand the potential costs involved in using these services:

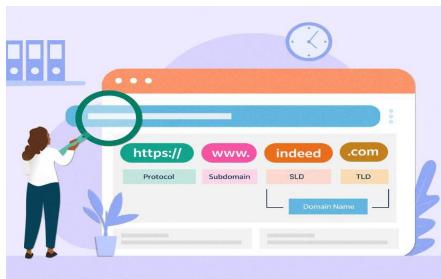
- **Buy a Domain:** There's a multitude of places to purchase domains, and the pricing varies based on several factors. Different extensions (like .com, .org, .net) and the duration you opt for (you usually buy a domain for a specific period) influence the costs. Generally, buying a domain with a common extension might range from 5-10\$/year
- **S3:** If you're within the AWS Free Tier, hosting your website files on S3 should incur no cost. Outside of the Free Tier, expenses are typically minimal, often just a few cents.
- **Route 53:** Maintaining a hosted zone adds around 50 cents monthly, while Route 53 queries cost approximately 40 cents per million queries.
- **AWS Certificate Manager:** Securing your site with a TLS/SSL certificate through ACM is free of charge.
- **CloudFront:** Operating within the Free Tier carries no cost. Beyond that, expenses will depend on website traffic, but for most small-scale usage, it often amounts to just a few cents. Refer to the detailed pricing page for more specifics.

If you plan to delete everything after the tutorial, consider setting up an AWS Budget to monitor and limit your spending. This ensures you're alerted before exceeding your set limit, preventing any unexpected bills.



Taking all this into account, let's dive into deploying our static website using AWS!

Buy a Domain



Before diving into the intricacies of using AWS services to host your website, let's first explore the fundamental step of acquiring a domain. Your domain is like your home's address on the internet; it's how people will find and remember your website. This step is crucial as it sets the foundation for your online presence, giving your website its unique identity in the vast landscape of the web.

There are various platforms where you can purchase a domain, such as:

- **Hostinger**
- **Namecheap**
- **GoDaddy**

Each of them offer simple processes to acquire your unique web address. For our example, we'll navigate through Hostinger, but the process remains fairly consistent across these platforms. Once on their website,

- you'll notice how straightforward it is to secure a domain.
- Simply input your desired name and check its availability.

If it's free, you'll then select your preferred domain extension(options like .com, .es, .dev, among others)

Choose the duration, whether it's for a year, two, or more.

AWS-S3



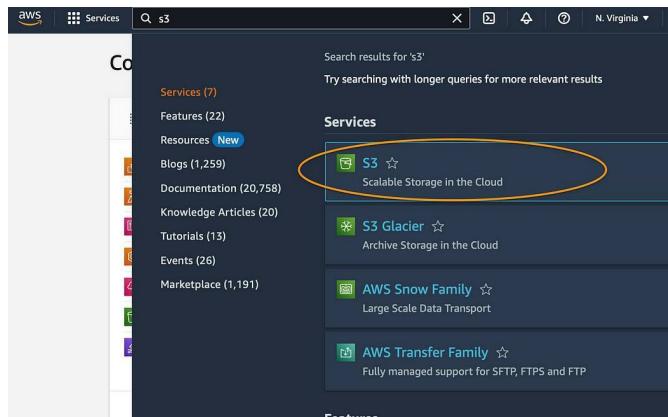
1. Create an AWS account

To create an account you just have to go to <https://aws.amazon.com/>



2. Search for S3

Navigate to the “Services” section and in the search bar type “S3” and press enter. Amazon S3 will appear as the first option in the list of services.



3. Create a S3 Bucket

AWS S3 is perfect for storing files affordably. When your website consists of client-side code only, you can set up S3 for hosting a static website easily.

- Click on the “Create bucket” button.
- **Bucket Name:** THIS NAME SHOULD MATCH THE DOMAIN

NAME EXACTLY. So if you bought a domain called for example “anaquirosa.com” the bucket name should be “anaquirosa.com”

- **AWS Region:** Decide on the region where you want your bucket to reside. From the dropdown menu, select your desired region. It’s important to choose a region that aligns with your specific requirements, such as data sovereignty or proximity to your target audience.
- Once you have chosen the bucket name and region, proceed with the creation of your bucket. This will establish a dedicated storage space for your objects within the selected region, allowing you to store and manage your data effectively using Amazon S3.



General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type [Info](#)

General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory - New

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

anaquirosa.com

⚠ Bucket with the same name already exists

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) ↗

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

In my case, the bucket name already exists because I created it before, but you shouldn't have any issues.

As you scroll down you have to **uncheck** the option for **Block all public access**. Typically, it's not advised to do so, as indicated by the warning prompt you'll receive upon disabling it. However, since you're crafting a website that you want to be accessible worldwide, turning this off is suitable for your purpose



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

acknowledge that the current settings might result in this bucket and the objects within becoming public.

- Proceed by utilising the **default settings** for the remaining bucket configurations.
- Then proceed to click on “**Create bucket**”

By following these steps, you'll create an S3 bucket that is accessible to the public, allowing you to host and share your website's content effectively.

4. Upload the content to the new Bucket

To upload your website content to the newly created bucket, follow these steps:

- Click on the name of your bucket in the S3 management console.
- Look for the “Upload” button and click on it. This will open the upload interface.
- Choose whether you need to upload a folder or individual files. If you have folders containing your website content, click on “Add folder.” Otherwise, if you only have specific files, click on “Add files.”
- Browse and select all the necessary files and folders from your local machine that make up your website, including files like index.html, CSS files, and images.
- After selecting all the relevant files, click on the “Upload” button located at the bottom right corner.

5. Enable Web Hosting

- Access the Properties menu of your bucket.



- Scroll down to the bottom and click on the “Edit” button. This will allow you to modify the properties of your bucket.
- Look for the option to enable static website hosting and click on it.
- Enter the name of the index file for your project, such as “index.html.” This file will be loaded as the default page for your website.
- If you have an error file, you can also specify its name under the “Error document” option. This file will be displayed if any errors occur while accessing your website.

Edit static website hosting [Info](#)

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting
 Disable
 Enable

Hosting type
 Host a static website
Use the bucket endpoint as the web address. [Learn more](#)
 Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#)

ⓘ For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document
Specify the home or default page of the website.

Error document - optional
This is returned when an error occurs.

Redirection rules - optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. [Learn more](#)

1	
---	--

After entering the necessary information, scroll down to the end of the page and click “**Save changes**” located at the bottom right corner.

6. Create bucket policies

- Navigate to the Permissions section in your bucket menu.
- Click on the “Edit” button located under the Bucket Policy section.
- Copy and paste the following JSON into the policy editor, ensuring to replace “your-bucket-name” with the actual name of your S3 bucket. You can find the Bucket ARN above the editor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::your-bucket-name/*"
    }
  ]
}
```

- **Remember not to remove the “/*” after the bucket name** in the JSON policy. This ensures that all objects within the bucket are selected to be made public.
- Finally, click on the “**Save changes**” option at the end to apply the bucket policy.

It is important to create bucket policies for granting public access to files otherwise, accessing the content of your website would not be possible. Instead, you would encounter a “403 Forbidden” message.

7. Test that your website works

- Navigate to the Properties tab of the bucket.
- Scroll down to the bottom page to Static website hosting and click on the link.

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

Enabled

Hosting type

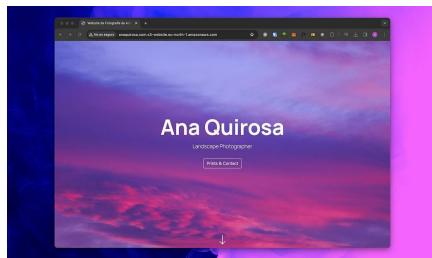
Bucket hosting

Bucket website endpoint

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://anaquirosa.com.s3-website.eu-north-1.amazonaws.com>

- If everything went well, you should observe your web showcased in the browser



Congratulations on successfully hosting your web in S3 with public access! However, to elevate its professionalism, it would be even more impressive if it were linked to a custom domain. **We've previously purchased a domain, and now we'll utilise it to enhance the final result.**



Add your Domain Name

1. Create a Hosted Zone

A public hosted zone, the one you'll be dealing with, manages internet traffic. On the other hand, a private hosted zone governs traffic within an AWS Virtual Private Cloud (VPC).

- Navigate to the “Services” section and in the search bar type “Route 53” and press enter. Route 53 will appear as the first option in the list of services.

The screenshot shows the AWS search interface with the query 'route 53'. The results list includes:

- Route 53** (Scalable DNS and Domain Name Registration)
- Route 53 Resolver** (Resolve DNS queries in your Amazon VPC and on-premises network.)
- Route 53 Application Recovery Controller**

- Click on “Hosted zones” and then on “Create hosted zone”.
- Domain name: Enter the domain you purchased from the third-party provider, select “Public hosted zone”, then click “Create hosted zone”.

The screenshot shows the 'Create hosted zone' wizard. Step 1: Hosted zone configuration. The 'Domain name' field is highlighted with an orange border and contains 'anaquirosa.com'. Other fields include 'Description - optional' (empty), 'Type' (radio button selected for 'Public hosted zone'), and 'Tags' (empty). Buttons at the bottom include 'Cancel' and 'Create hosted zone'.

After creating your public hosted zone, **you will find 4 listed name servers**.

The screenshot shows the AWS Route 53 Hosted Zone Details page for the domain `anaquirosa.com`. Under the 'Records (4)' tab, there are two entries:

- The first entry is an `A` record for `anaquirosa.com` with the value `4 servers`, which is highlighted with an orange box and an arrow pointing to it.
- The second entry is an `NS` record for `anaquirosa.com` with the value `Simple`.

Take a note of these for later!

2. DNS for your current domain provider

Go to the **DNS settings** for your current domain provider (in my case it is Hostinger) Find your name server settings, and replace them with the name servers from Route 53!

If you are doing it with Hostinger follow these steps:

Click on Manage

The screenshot shows the Hostinger Pro Panel interface. In the top left, it says "Hi, Ana" with a hand icon. On the right, there are icons for "Pro Panel BETA", search, and account settings. Below the header, there's a "Domain" section for `anaquirosa.com`, which expires on 2025-12-07. To the right of the domain info is a purple "Manage" button, which is highlighted with an orange border.

- Go to “DNS / Nameservers” and then to “Change Nameservers”
- Copy the links you found in the previous step in ‘Route 53’ and paste them one by one and then click on “Save”.

3. Create a Record to point to the S3

Having set up a public hosted zone, the next step is creating a record dictating how traffic should be directed when visitors access your domain name. For that follow these steps:

Enter your “Hosted zone” and click on “Create record”

The screenshot shows the AWS Route 53 Hosted Zone Details page for the domain `anaquirosa.com`. At the top, there are buttons for 'Delete zone', 'Test record', and 'Configure query logging'. Below that, a section titled 'Hosted zone details' has a 'Edit hosted zone' button. Underneath, there are tabs for 'Records (4)', 'DNSSEC signing', and 'Hosted zone tags (0)'. The 'Records (4)' tab is selected. It contains a table with columns for 'Name', 'Type', 'Value', and 'TTL'. A search bar at the bottom allows filtering by property or value, type, routing policy, and alias. The 'Create record' button is highlighted with a red box.

- **Record name:** Leave it blank.
- **Record type:** A — Routes traffic to an IPv4 address and some AWS resources.
- **Alias:** Switch this **on**. Enabling an alias allows you to direct traffic to various AWS resources such as S3, CloudFront, Elastic Beanstalk, and more.

About Route traffic:

1º Alias to S3 website endpoint. 2º Choose your region.

3º This menu should auto-fill with your S3 website.

If nothing appears here, it might be because your bucket is not named the same as your domain. In that case, you'll have to recreate the bucket with your domain's exact name.

And finally:

Routing policy: Simple routing. **Evaluate target health:** YES. Click on “**Create records**”.

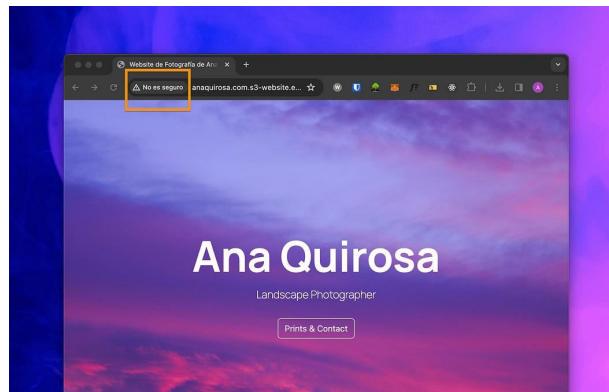
Your changes may take up to 60 seconds to become active. Once the **Status** switches from **PENDING** to **INSYNC**, you're all set to test out your modifications.

Let's run a test! If everything went good, entering your domain name into a browser (like `anaquirosa.com`) should lead Route 53 to direct you to the S3 website. **This means you should see your website!!**

Establishing a Secure Connection



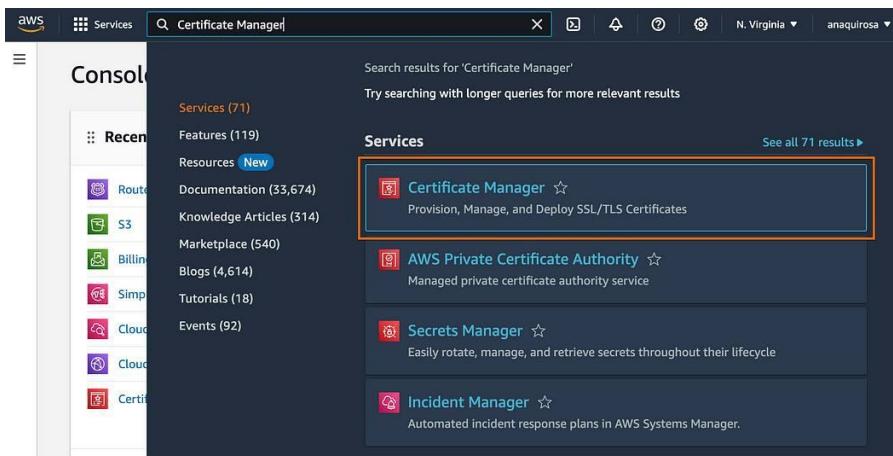
Now the final step involves establishing a secure connection (HTTPS) with a TLS/SSL certificate. This will help eliminate the “Not secure” message from your browser. This is very important because a secure connection will reassure visitors that they haven’t landed on a suspicious or unreliable website.



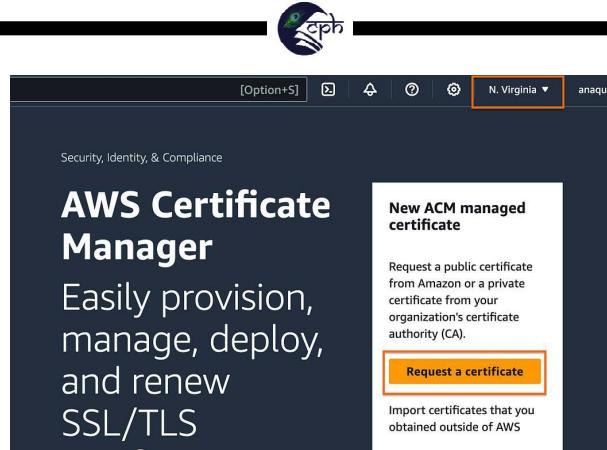
“Not secure” alert

For that we are going to use AWS Certificate Manager. Generate a public TLS/SSL Certificate

- Navigate to the “Services” section and in the search bar type “Certificate Manager” and press enter.

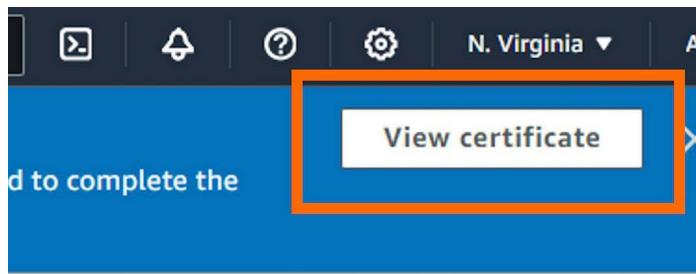


- **Ensure to change your region to u-east-1 (N.Virginia) for this section.** Creating a certificate in any other region will render it unusable with CloudFront, where you will ultimately need it.
- Click on “Request a certificate”



- Select “Request a public certificate” and click on NEXT.
- Fully qualified domain name: Enter your domain
- Leave the rest as default and then click “Request”.

The request went successful but it will remain in “pending validation” status until you validate it through DNS. For that click on “View certificate”



Before the certificate can be issued, Amazon requires confirmation of your domain ownership and your ability to modify DNS settings (within Route 53).

For that we need to :

- Click on “Create records in Route 53”
- Select your domain and click on “Create records”

If the record creation was completed without any issues, you'll be notified with a success message

Now you have to:

- Navigate to “Route 53” again and go to your “hosted zone”
- You'll notice a fresh CNAME record generated by Certificate Manager among the listings.

The screenshot shows the AWS Route 53 Hosted Zone Details page for the domain `anaquirosa.com`. The top navigation bar includes `Route 53 > Hosted zones > anaquirosa.com`. Below the navigation, there are tabs for `Public`, `anaquirosa.com`, and `Info`. The main section is titled `Hosted zone details` and has three tabs: `Records (4)`, `DNSSEC signing`, and `Hosted zone tags (0)`. The `Records (4)` tab is selected. It displays four records:

		Type	Routing	Differences
<input type="checkbox"/>	anaquirosa...	A	Simple	-
<input type="checkbox"/>	anaquirosa...	NS	Simple	-
<input type="checkbox"/>	anaquirosa...	SOA	Simple	-
<input type="checkbox"/>	_cc914d9...	CNAME	Simple	-

Congrats! You have a TLS/SSL certificate

Create a CloudFront Distribution

Your website's files are sitting in S3, but here's the catch: **certificates don't work directly with S3 buckets**. What you'll need is a **CloudFront distribution to link to that S3 bucket**. Then, you apply the certificate to this CloudFront setup.

CloudFront is Amazon's content delivery network (CDN) that speeds up content delivery worldwide by storing it closer to users. It's fantastic for videos and images, making them load faster. If your website is basic or has small files, you might not notice a huge difference in speed. But using CloudFront is essential to apply the TLS/SSL certificate you made earlier.

- Navigate to **CloudFront**.

The screenshot shows the AWS search interface with the query `cloudfront`. The search results are displayed under the `Services` and `Resources` sections. The `CloudFront` service is highlighted with an orange box. The results include:

- Services (1)**: CloudFront
- Resources (New)**: Documentation (6,947), Knowledge Articles (73), Marketplace (196), Blogs (281), Events (2), Tutorials (1)
- Documentation (6,947)**: Try searching with longer queries for more relevant results
- CloudFront**: Global Content Delivery Network
- Resources / for a focused search**: Introducing resource search

- Click on “Create distribution”.
- Origin domain: If you click on the filter it should appear in Amazon S3 your website files.



Create distribution

Origin

Origin domain

Choose an AWS origin, or enter your origin's domain name.

Choose origin domain

Amazon S3

anaquirosa.com.s3.amazonaws.com

Elastic Load Balancer

- Click on “**Use website endpoint**” and AWS will update the endpoint for you.
- Scroll down until the section “**Default cache behaviour**”
- Viewer protocol policy:** Redirect HTTP to HTTPS

Default cache behavior

Path pattern | [Info](#)

Default (*)

Compress objects automatically | [Info](#)

No

Yes

Viewer

Viewer protocol policy

- HTTP and HTTPS
- Redirect HTTP to HTTPS
- HTTPS only

Allowed HTTP methods

- GET, HEAD
- GET, HEAD, OPTIONS
- GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Restrict viewer access

If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.

No

Yes

Cache key and origin requests

We recommend using a cache policy and origin request policy to control the cache key and origin requests.

- Cache policy and origin request policy (recommended)
- Legacy cache settings



Scroll down to “Web Application Firewall (WAF)”

- Select “Do not enable security protections” Scroll down to “**Settings**”
- **Alternate domain name (CNAME): Add item** and **enter your domain name** (in my case “anaquirosa.com”)
- **Custom SSL certificate:** Choose the certificate you established earlier doing click on it. Remember, if you created it in a region other than us-east-1 (N. Virginia), it won’t appear in this list.
- **Default root object:** Type “index.html” (your default homepage) and then click “**Create distribution**”.

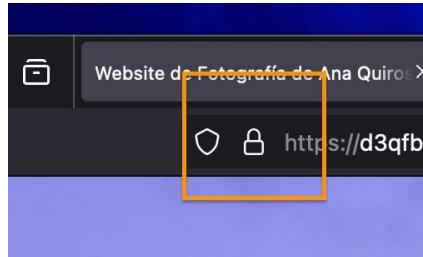
It can take a few minutes to for the CloudFront distribution to finish deploying, even if it initially shows “Successfully created” at the top of the page. **You will see it’s finished when the “Last modified” value displays a date and time**

ID	Type	Domain Name	Alternate Domains	Origins	Status	Last modified
d3qfbui1...	Production	anaquirosa.com	anaquirosa.com	Enabled	Enabled	December 7, 2023 at 10:57:53 AM UTC

To confirm that everything is functioning correctly with CloudFront and the TLS/SSL certificate:

- Copy the Distribution domain name.
- Open a new browser tab and paste that address into the navigation bar.

If everything went good, you should now notice the padlock icon in your browser (or something similar, it depends on the browser) signaling that you’re securely connected via the certificate configured in Certificate Manager.

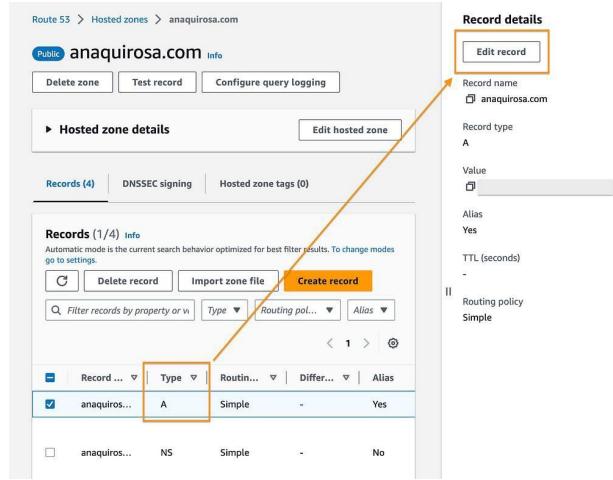


Update Route 53 to direct to the CloudFront Distribution.

Right now, Route 53 sends traffic to the S3 bucket. We want it to send traffic to the CloudFront distribution, which then leads to S3. For that:

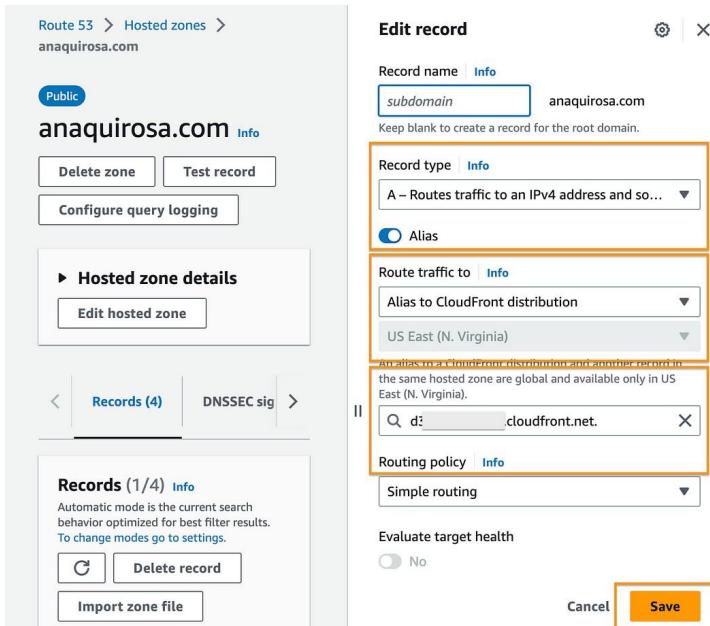
• Navigate to “Route 53” > “Hosted zones”

• Select the A record and then click “Edit record”



The screenshot shows the AWS Route 53 Hosted Zones interface for the domain anaquirosa.com. On the left, under ‘Hosted zone details’, there is a table with one row selected. The first column has a checkbox, the second column is labeled ‘Type’ with ‘A’ selected, and the third column is labeled ‘Alias’ with ‘Yes’ checked. An orange box highlights the ‘Type’ dropdown. On the right, a modal window titled ‘Record details’ is open, showing the record name as ‘anaquirosa.com’ and the record type as ‘A’. The ‘Value’ field is empty. Below it, the ‘Alias’ field is set to ‘Yes’. The ‘TTL (seconds)’ field is set to ‘-’. Under ‘Routing policy’, it says ‘Simple’. An orange arrow points from the ‘Type’ dropdown in the main table to the ‘Record type’ field in the modal.

- **Route traffic to:** Alias to Cloudfront distribution.
- **Choose Region:** This option is selected for you and grayed out.
- **Choose your distribution** (it should automatically populate in the third dropdown)
- Click “Save”.



The screenshot shows the ‘Edit record’ dialog for the ‘subdomain’ record of the ‘anaquirosa.com’ hosted zone. The ‘Record type’ is set to ‘A – Routes traffic to an IPv4 address and so...’ and the ‘Alias’ option is selected. In the ‘Route traffic to’ section, ‘Alias to CloudFront distribution’ is chosen, and ‘US East (N. Virginia)’ is selected. A note below states: ‘An alias to a CloudFront distribution can’t have other records in the same hosted zone are global and available only in US East (N. Virginia).’ A search bar contains ‘d: cloudfront.net.’. The ‘Routing policy’ is set to ‘Simple routing’. At the bottom, the ‘Evaluate target health’ switch is off, and there are ‘Cancel’ and ‘Save’ buttons. An orange box highlights the ‘Record type’ dropdown and the ‘Route traffic to’ section.



You did it!!

If everything worked, you should be able to navigate to your domain name and have it load your website on a secure connection!

