



BLACKBUCKS INTERNSHIP REPORT

RENT4U WEBSITE HOSTING USING EC2 AND LOAD BALANCER

SUBMITTED BY

- 1.Shaik Sadulla (21B91A04M1)**
- 2.SVPKH Tejovarma (21B91A04K5)**

**UNDER THE GUIDANCE OF
MR. GURU SANTHOSH SEENIVASAN**

**Blackbuck Engineers Pvt. Ltd
Road No 36, Jubilee Hills, Hyderabad**

BLACKBUCK INTERNSHIP WORK

Team Members:

- 1.Shaik Sadulla(21B91A04M1)
- 2.SVPKH Tejovarma(21B91A04K5)

Title:

RENT4U WEBSITE HOSTING USING EC2 AND LOAD BALANCER

Abstract:

This abstract presents a comprehensive overview of website hosting using Amazon Elastic Compute Cloud (EC2) instances combined with a Load Balancer to achieve enhanced scalability, performance, and reliability. With the ever-increasing demand for online services, organisations must adopt a robust hosting architecture that can efficiently handle fluctuating traffic while maintaining high availability. Amazon EC2 offers scalable computing resources in the cloud, while the Load Balancer evenly distributes incoming traffic across multiple EC2 instances, ensuring optimised performance and fault tolerance.

TABLE OF CONTENTS

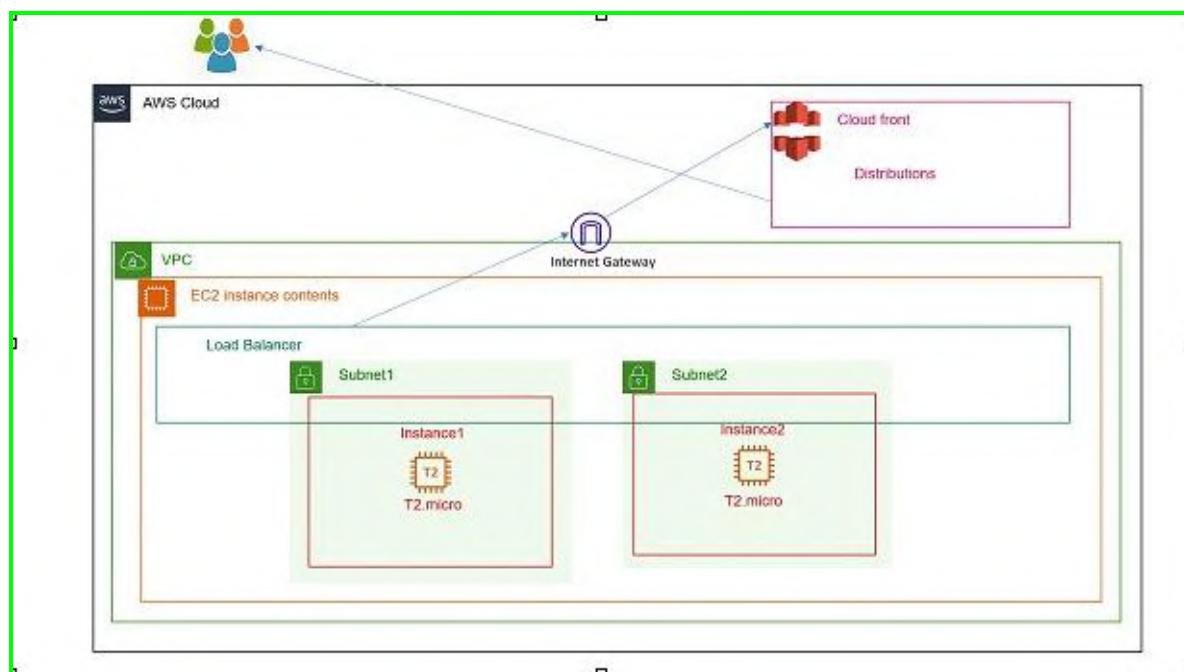
Services used:.....	3
Architecture:.....	3
Description:.....	3
Cloud computing:.....	4
Cloud Computing Services:.....	5
IaaS (Infrastructure-as-a-Service).....	5
PaaS (Platform-as-a-service).....	5
SaaS (Software-as-a-Service).....	6
Cloud Service Providers:.....	6
Amazon Web Services:.....	7
Why AWS?.....	8
List of AWS Services:.....	9
Amazon EC2:.....	10
Instance types:.....	11
Amazon RDS:.....	12
Multiple Availability Zone (AZ) Deployment.....	12
Read replicas.....	13
Performance metrics and monitoring.....	13
Amazon VPC:.....	13
Amazon S3:.....	15
Amazon IAM:.....	16
AWS Lambda:.....	18
AWS Cloud9:.....	19
Environments and computing resources.....	20
AWS Elastic BeanStalk:.....	21
AWS CodeCommit:.....	22
Amazon EBS (Elastic Block Store):.....	24
Features of Amazon EBS.....	25
Amazon Aurora:.....	27
AWS AutoScaling:.....	29
Auto scaling benefits.....	30
IMPLEMENTATION.....	31-46

Services used:

- IAM (Identity and Access Management)
- EC2 (Elastic Compute Cloud)
 - ELB (Elastic load balancing)
- CLOUDFRONT

Architecture:

RENT4U WEBSITE HOSTING USING EC2 AND LOAD BALANCER



Description:

Website hosting in EC2 architecture involves deploying web applications or websites on virtual servers called EC2 instances within Amazon Web Services (AWS) Elastic Compute Cloud (EC2) service. EC2 instances provide scalable computing resources and can be customised to suit the specific needs of the website, offering various instance types with varying CPU, memory, and storage capacities. Users can choose from standard Amazon Machine Images (AMIs) or create custom ones with the necessary software and configurations. Security groups act as virtual firewalls, controlling inbound and outbound

traffic to the instances, while Elastic IP addresses ensure a consistent public IP for the website. Data storage options like Amazon Elastic Block Store (EBS) and Amazon S3 allow users to store website files and data securely.

Optionally, Elastic Load Balancers (ELBs) distribute traffic among multiple instances for better performance and fault tolerance.

Cloud computing:

Cloud computing is on-demand access, via the internet, to computing resources—applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more—hosted at a remote data centre managed by a cloud services provider (or CSP). The CSP makes these resources available for a monthly subscription fee or bills them according to usage.

Compared to traditional on-premises IT, and depending on the cloud services you select, cloud computing helps do the following:

- **Lower IT costs:** Cloud lets you offload some or most of the costs and effort of purchasing, installing, configuring, and managing your own on-premises infrastructure.
- **Improve agility and time-to-value:** With cloud, your organisation can start using enterprise applications in minutes, instead of waiting weeks or months for IT to respond to a request, purchase and configure supporting hardware, and install software. Cloud also lets you empower certain users—specifically developers and data scientists.
- **Scale more easily and cost-effectively:** Cloud provides elasticity—instead of purchasing excess capacity that sits unused

During slow periods, you can scale capacity up and down in response to spikes and dips in traffic. You can also take advantage of your cloud provider's global network to spread your applications closer to users around the world.

The term ‘cloud computing’ also refers to the technology that makes cloud work. This includes some form of virtualized IT infrastructure—servers, operating system software, networking, and other infrastructure that's abstracted, using special software, so that it can be pooled and divided irrespective of physical hardware

boundaries. For example, a single hardware server can be divided into multiple virtual servers.

Cloud Computing Services:

- IaaS (Infrastructure-as-a-Service)
- PaaS (Platform-as-a-Service)
- SaaS (Software-as-a-service)

are the three most common models of cloud services, and it's not uncommon for an organisation to use all three.

IaaS (Infrastructure-as-a-Service)

IaaS provides on-demand access to fundamental computing resources—physical and virtual servers, networking, and storage—over the internet on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary on-premises or ‘owned’ infrastructure and for overbuying resources to accommodate periodic spikes in usage.

In contrast to SaaS and PaaS (and even newer PaaS computing models such as containers and serverless), IaaS provides the users with the lowest-level control of computing resources in the cloud.

IaaS was the most popular cloud computing model when it emerged in the early 2010s. While it remains the cloud model for many types of workloads, use of SaaS and PaaS is growing at a much faster rate.

PaaS (Platform-as-a-service)

PaaS provides software developers with on-demand platforms—hardware, complete software stack, infrastructure, and even development tools—for running, developing, and managing applications without the cost, complexity, and inflexibility of maintaining that platform on premises.

With PaaS, the cloud provider hosts everything—servers, networks,

storage, operating system software, middleware, databases—at their data centre. Developers simply pick from a menu to ‘spin up’ servers and environments they need to run, build, test, deploy, maintain, update, and scale applications.

Today, PaaS is often built around containers, a virtualized compute model one step removed from virtual servers. Containers virtualize the operating system, enabling developers to package the application with only the operating system services it needs to run on any platform, without modification and without need for middleware.

SaaS (Software-as-a-Service)

SaaS—also known as cloud-based software or cloud applications—is application software that’s hosted in the cloud, and that user’s access via a web browser, a dedicated desktop client, or an API that integrates with a desktop or mobile operating system. In most cases, SaaS users pay a monthly or annual subscription fee; some may offer ‘pay-as-you-go’ pricing based on your actual usage.

In addition to the cost savings, time-to-value, and scalability benefits of cloud, SaaS offers the following:

- **Automatic upgrades:** With SaaS, users take advantage of new features as soon as the provider adds them, without having to orchestrate an on-premises upgrade.
- **Protection from data loss:** Because SaaS stores application data in the cloud with the application, users don’t lose data if their device crashes or breaks.

SaaS is the primary delivery model for most commercial software today—there are hundreds of thousands of SaaS solutions available, from the most focused industry and departmental applications to powerful enterprise software databases and AI (artificial intelligence) software.

Cloud Service Providers:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Oracle

- IBM cloud
- Salesforce

Amazon Web Services:

Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay as-you-go basis. Oftentimes, clients will use this in combination with autoscaling (a process that allows a client to use more computing in times of high application usage, and then scale down to reduce costs when there is less traffic). These cloud computing web services provide various services related to networking, computing, storage, middleware, IoT and other processing capacity, as well as software tools via AWS server farms. This frees clients from managing, scaling, and patching hardware, and operating systems.

One of the foundational services is Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, with extremely high availability, which can be interacted with over the internet via REST APIs, a CLI or the AWS console. AWS's virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard disk /SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

AWS services are delivered to customers via a network of AWS server farms located throughout the world. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), hardware, operating system, software, or networking features chosen by the subscriber required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either.

Amazon provides select portions of security for subscribers (e.g., physical security of the data centres) while other aspects of security are the responsibility of the subscriber (e.g., account management, vulnerability scanning, patching). AWS operates for many global geographical regions including seven in North America.

Amazon markets AWS to subscribers as a way of obtaining large-scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways. As of 2021 Q4, AWS has 33% market share for cloud infrastructure while the next two competitors Microsoft Azure and Google Cloud have 21%, and 10% respectively, according to Synergy Group.

Why AWS?

- **Easy to use:**

AWS is designed to allow application providers, ISVs, and vendors to host your applications quickly and securely – whether an existing application or a new SaaS based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

- **Flexible:**

AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.

- **Cost-effective:**

You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.

- **Reliable:**

With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion-dollar online business that has been honed for over a decade.

- **Scalable and High performance:**

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

- **Secure:**

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

List of AWS Services:

Amazon, the preeminent cloud vendor, broke new ground by establishing the first cloud computing service, Amazon EC2, in 2008. AWS offers more solutions and features than any other provider and has free tiers with access to the AWS Console, where users can centrally control their ministries.

Designed around ease-of-use for various skill sets, AWS is tailored for those unaccustomed to software development utilities. Web applications can be deployed in minutes with AWS facilities, without provisioning servers or writing additional code.

- Amazon EC2 (Elastic Compute Cloud)
- Amazon RDS (Relational Database Services)
- Amazon S3 (Simple Storage Service)
- Amazon Lambda
- Amazon Cognito

- Amazon Glacier
- Amazon SNS (Simple Notification Service)
- Amazon VPC (Virtual Private Cloud)
- Amazon Lightsail
- Amazon CloudWatch
- Amazon Cloud9
- Amazon Elastic Beanstalk
- Amazon CodeCommit
- Amazon IAM (Identity and Access Management)
- Amazon Inspector
- Amazon Kinesis
- Amazon DynamoDB
- Amazon Code Catalyst
- Amazon Kinesis
- AWS Athena
- AWS Amplify
- AWS Quicksight
- AWS Cloudformation

Amazon EC2:

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. In November

2010, Amazon switched its own retail website platform to EC2 and AWS.

Amazon announced a limited public beta test of EC2 on August 25, 2006, offering access on a first-come, first-served basis. Amazon added two new instance types (Large and Extra-Large) on October 16, 2007. On May 29, 2008, two more types were added, High-CPU Medium and High-CPU Extra Large. There were twelve types of instances available.

Amazon added three new features on March 27, 2008, static IP addresses, availability zones, and user selectable kernels. On August 20, 2008, Amazon added Elastic Block Store (EBS) This provides persistent storage, a feature that had been lacking since the service was introduced.

Instance types:

Initially, EC2 used Xen virtualization exclusively. However, on November 6, 2017, Amazon announced the new C5 family of instances that were based on a custom architecture around the KVM hypervisor, called Nitro. Each virtual machine, called an "instance", functions as a virtual private server. Amazon sizes instances based on "Elastic Compute Units". The performance of otherwise identical virtual machines may vary. On November 28, 2017, AWS announced a bare-metal instance type offering marking a remarkable departure from exclusively offering virtualized instance types.

As of January 2019, the following instance types were offered:

- General Purpose: A1, T3, T2, M5, M5a, M4, T3a
- Compute Optimised: C5, C5n, C4
- Memory Optimised: R5, R5a, R4, X1e, X1, High Memory, z1d
- Accelerated Computing: P3, P2, G3, F1
- Storage Optimised: H1, I3, D2

As of April 2018, the following payment methods by instance were offered:

- On-demand: pay by the hour without commitment.
- Reserved: rent instances with one-time payment receiving discounts on the hourly charge.

- Spot: bid-based service runs the jobs only if the spot price is below the bid specified by the bidder. The spot price is claimed to be supply-demand based, however a 2011 study concluded that the price was generally not set to clear the market but was dominated by an undisclosed reserve price.

Amazon RDS:

Amazon Relational Database Service (or **Amazon RDS**) is a distributed relational database service by Amazon Web Services (AWS). It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications. Administration processes like patching the database software, backing up databases and enabling point-in-time recovery are managed automatically. Scaling storage and compute resources can be performed by a single API call to the AWS control plane on-demand. AWS does not offer an SSH connection to the underlying virtual machine as part of the managed service.

Multiple Availability Zone (AZ) Deployment

In May 2010 Amazon announced Multi-Availability Zone deployment support. Amazon RDS Multi-Availability Zone (AZ) allows users to automatically provision and maintain a synchronous physical or logical "standby" replica, depending on database engine, in a different Availability Zone (independent infrastructure in a physically separate location). Multi-AZ database instances can be developed at creation time or modified to run as a multi-AZ deployment later. Multi-AZ deployments aim to provide enhanced availability and data durability for MySQL, MariaDB, Oracle, PostgreSQL and SQL Server instances and are targeted for production environments. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date standby, allowing database operations to resume without administrative intervention.

Multi-AZ RDS instances are optional and have a cost associated with them. When creating a RDS instance, the user is asked if they would like to use a multi-AZ RDS instance. In Multi AZ RDS deployments

backups are done in the standby instance so I/O activity is not suspended any time, but users may experience elevated latencies for a few minutes during backups.

Read replicas.

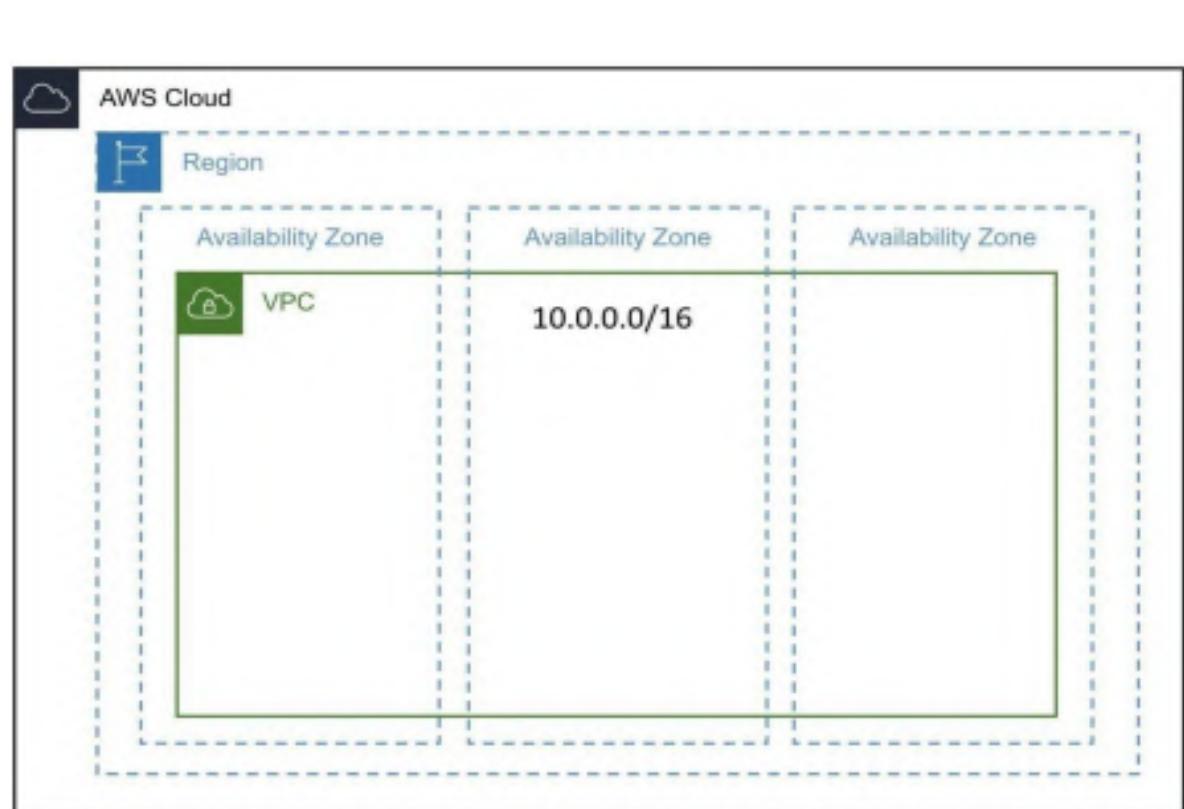
Read replicas allow different use cases such as scale in for read-heavy database workloads. There are up to five replicas available for MySQL, MariaDB, and PostgreSQL. Instances use the native, asynchronous replication functionality of their respective database engines. They have no backups configured by default and are accessible and can be used for read scaling. MySQL and MariaDB read replicas and can be made writeable again since October 2012; PostgreSQL read replicas do not support it. Replicas are done at database instance level and do not support replication at database or table level.

Performance metrics and monitoring

Performance metrics for Amazon RDS are available from the AWS Management Console or the Amazon CloudWatch API. In December 2015, Amazon announced an optional enhanced monitoring feature that provides an expanded set of metrics for the MySQL, MariaDB, and Aurora database engines.

Amazon VPC:

Amazon Virtual Private Cloud (VPC) is a commercial cloud computing service that provides a virtual private cloud, by provisioning a logically isolated section of Amazon Web Services (AWS) Cloud. Enterprise customers are able to access the Amazon Elastic Compute Cloud (EC2) over an IPsec based virtual private network. Unlike traditional EC2 instances which are allocated internal and external IP numbers by Amazon, the customer can assign IP numbers of their choosing from one or more subnets.



Amazon Web Services launched Amazon Virtual Private Cloud on 26 August 2009, which allows the Amazon Elastic Compute Cloud service to be connected to legacy infrastructure over an IPsec VPN. In AWS, the basic VPC is free to use, with users being charged by usage for additional features. EC2 and RDS instances running in a VPC can also be purchased using Reserved Instances, however, will have a limitation on resources being guaranteed [citation needed].

IBM Cloud launched IBM Cloud VPC on 4 June 2019, provides an ability to manage virtual machine-based compute, storage, and networking resources. Pricing for IBM Cloud Virtual Private Cloud is applied separately for internet data transfer, virtual server instances, and block storage used within IBM Cloud VPC.

Google Cloud Platform resources can be provisioned, connected, and isolated in a virtual private cloud (VPC) across all GCP regions. With GCP, VPCs are global resources and subnets within that VPC are regional resources. This allows users to connect zones and regions without the use of additional networking complexity as all data travels, encrypted in transit and at rest, on Google's own global, private

network. Identity management policies and security rules allow for private access to Google's storage, big data, and analytics managed services. VPCs on Google Cloud Platform leverage the security of Google's data centres.

Amazon S3:

Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with high durability. The basic storage units of Amazon S3 are objects which are organised into buckets. Each object is identified by a unique, user assigned key. Buckets can be managed using the console provided by Amazon S3, programmatically with the AWS SDK, or the REST application programming interface.

Objects can be up to five terabytes in size. Requests are authorised using an access control list associated with each object bucket and support versioning which is disabled by default. Since buckets are typically the size of an entire file system mount in other systems, this access control scheme is very coarse-grained. In other words, unique access controls cannot be associated with individual files. [citation needed] Amazon S3 can be used to replace static web-hosting infrastructure with HTTP client-accessible objects, index document support and error document support.

The Amazon AWS authentication mechanism allows the creation of authenticated URLs, valid for a specified amount of time. Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.

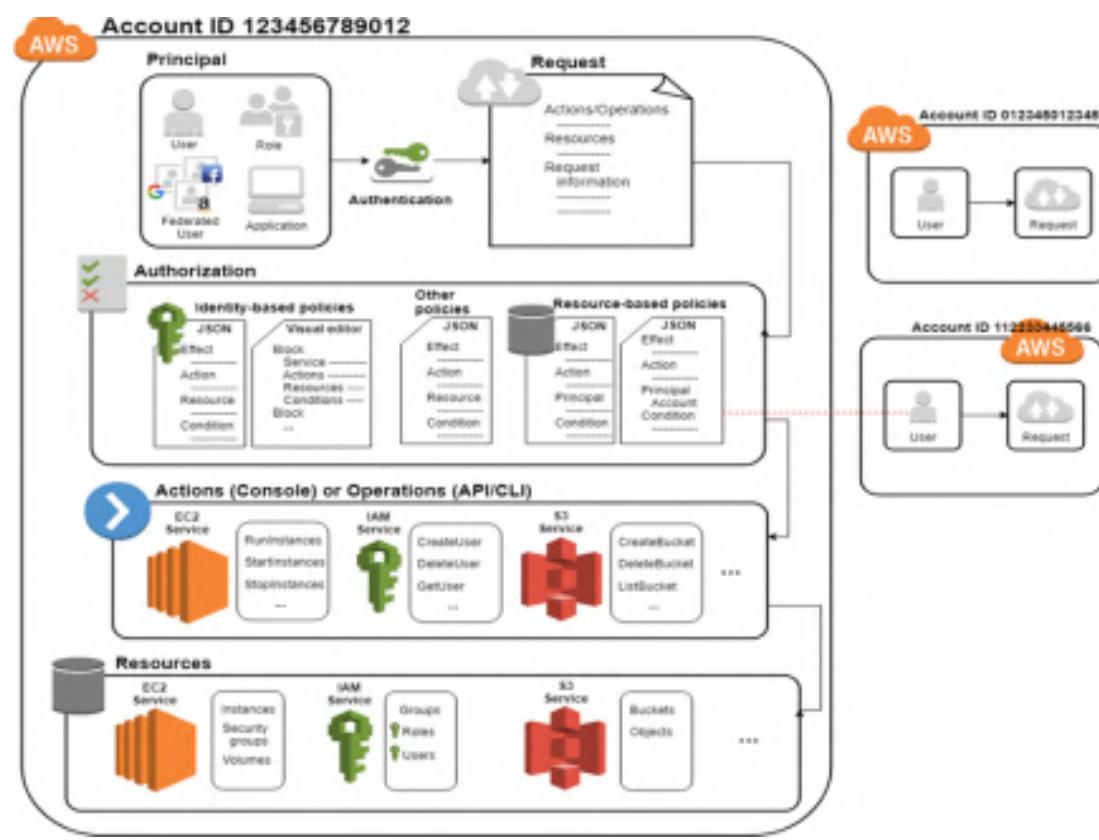
There are various User Mode File System (FUSE)-based file systems for Unix-like operating systems (for example, Linux) that can be used to mount an S3 bucket as a file system. The semantics of the Amazon

S3 file system is not that of a POSIX file system, so the file system may not behave entirely as expected.



Amazon IAM:

IAM provides the infrastructure necessary to control authentication and authorization for your AWS account. The IAM infrastructure is illustrated by the following diagram.



First, a human user or an application uses their sign-in credentials to authenticate with AWS. Authentication is provided by matching the sign-in credentials to a principal (an IAM user, federated user, IAM role, or application) trusted by the AWS account.

Next, a request is made to grant the principal access to resources. Access is granted in response to an authorization request. For example, when you first sign into the console and are on the console home page, you are not accessing a specific service. When you select a service, the request for authorization is sent to that service and it looks to see if your identity is on the list of authorised users, what policies are being enforced to control the level of access granted, and any other policies that might be in effect. Authorization requests can be made by principals within your AWS account or from another AWS account that you trust.

Once authorised, the principal can take action or perform operations on resources in your AWS account. For example, the principal could launch a new Amazon Elastic Compute Cloud instance, modify IAM group membership, or delete Amazon Simple Storage Service buckets.

The previous illustration we used specific terminology to describe how to obtain access to resources. These IAM terms are commonly used when working with AWS:

IAM Resources

The user, group, role, policy, and identity provider objects that are stored in IAM. As with other AWS services, you can add, edit, and remove resources from IAM.

The IAM resource objects that are used to identify and group. You can attach a policy to an IAM identity. These include users, groups, and roles.

IAM Entities

The IAM resource objects that AWS uses for authentication. These include IAM users and roles.

Principals

A person or application that uses the AWS account root user, an IAM user, or an IAM role to sign in and make requests to AWS. Principals include federated users and assumed roles.

Human users

Also known as human identities; the people, administrators, developers, operators, and consumers of your applications.

Workload

A collection of resources and code that delivers business value, such as an application or backend process. Can include applications, operational tools, and components.

AWS Lambda:

AWS Lambda is a compute service that lets you run code without provisioning or managing servers.

Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

You organise your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customise the operating system on provided runtimes.

Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you do need to manage your compute resources, AWS has other compute services to consider, such as:

- AWS App Runner builds and deploys containerized web applications automatically, load balances traffic with encryption, scales to meet your traffic needs, and allows for the configuration of how services are accessed and communicated with other AWS applications in a private Amazon VPC.
- AWS Fargate with Amazon ECS runs containers without having to provision, configure, or scale clusters of virtual machines.
- Amazon EC2 lets you customise the operating system, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.

You can use environment variables to adjust your function's behaviour without updating code. An environment variable is a pair of strings that is stored in a function's version-specific configuration. The Lambda runtime makes environment variables available to your code and sets additional environment variables that contain information about the function and invocation request.



AWS Cloud9:

AWS Cloud9 is an integrated development environment, or *IDE*.

The AWS Cloud9 IDE offers a rich code-editing experience with

support for several programming languages and runtime debuggers, and a built-in terminal. It contains a collection of tools that you use to code, build, run, test, and debug software, and helps you release software to the cloud.

You access the AWS Cloud9 IDE through a web browser. You can configure the IDE to your preferences. You can switch colour themes, bind shortcut keys, enable programming language specific syntax colouring and code formatting, and more.



Environments and computing resources

Behind the scenes, there are a couple of ways you can connect your environments to computing resources:

- You can instruct AWS Cloud9 to create an Amazon EC2 instance, and then connect the environment to that newly created EC2 instance. This type of setup is called an EC2 environment.
- You can instruct AWS Cloud9 to connect an environment to an existing cloud compute instance or to your own server. This type of setup is called an SSH environment.

EC2 environments and SSH environments have some similarities and

some differences. If you're new to AWS Cloud9, we recommend that you use an EC2 environment because AWS Cloud9 takes care of much of the configuration for you. As you learn more about AWS Cloud9, and want to understand these similarities and differences better, see EC2 environments compared with SSH environments in AWS Cloud9.

AWS Elastic BeanStalk:

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

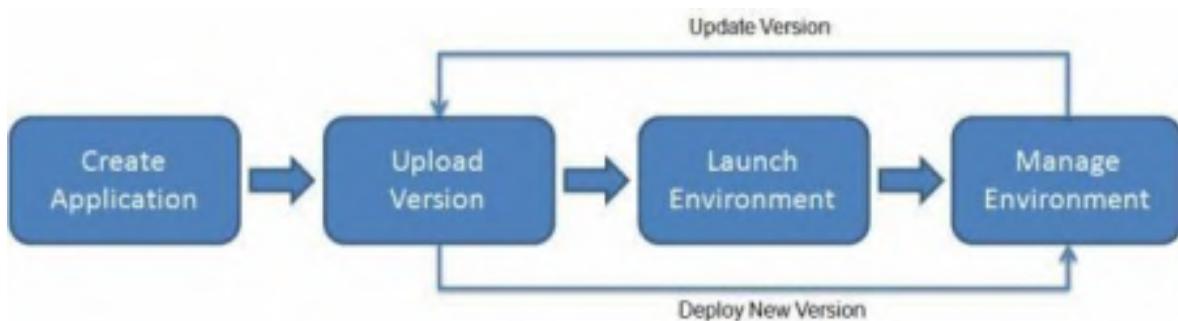
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or **eb**, a high-level CLI designed specifically for Elastic Beanstalk.

To learn more about how to deploy a sample web application using Elastic Beanstalk, see Getting Started with AWS: Deploying a Web App.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console).

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.



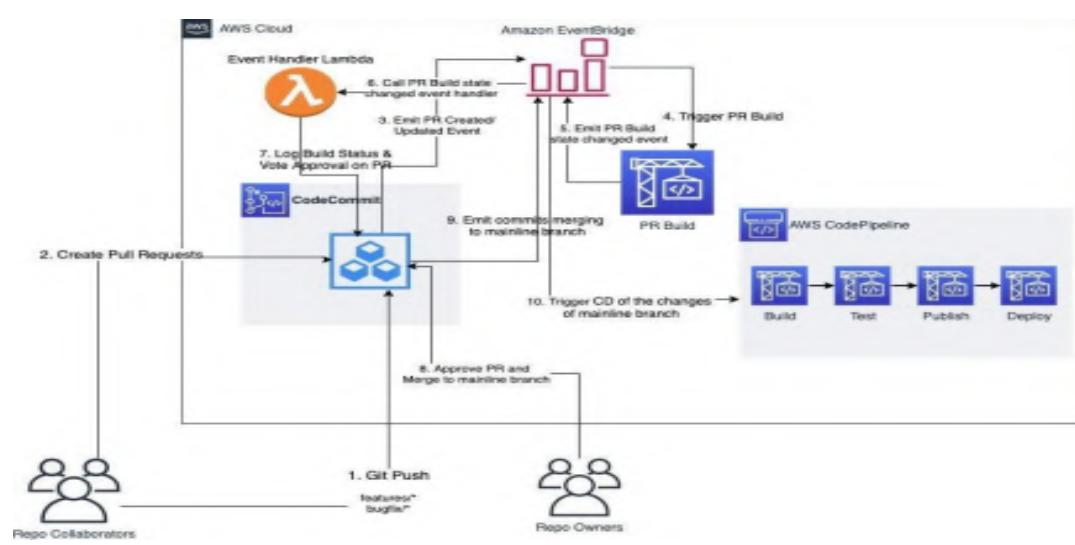
AWS CodeCommit:

CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

With CodeCommit, you can:

- **Benefit from a fully managed service hosted by AWS.** CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.

- **Store your code securely.** CodeCommit repositories are encrypted at rest as well as in transit.
- **Work collaboratively on code.** CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
- **Easily scale your version control projects.** CodeCommit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.
- **Store anything, anytime.** CodeCommit has no limit on the size of your repositories or on the file types you can store.
- **Integrate with other AWS and third-party services.** CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and service integrations with AWS CodeCommit.
- **Easily migrate files from other remote repositories.** You can migrate to CodeCommit from any Git-based repository.
- **Use the Git tools you already know.** CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.



Amazon CloudWatch:

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop underused instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilisation, application performance, and operational health.

Amazon EBS (Elastic Block Store):

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence. EBS volumes are particularly

well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

With Amazon EBS, you pay only for what you use. For more information about Amazon EBS pricing, see the Projecting Costs Section of the Amazon Elastic Block Store page.

Features of Amazon EBS

- You create an EBS volume in a specific Availability Zone, and then attach it to an instance in that same Availability Zone. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that Region. You can copy snapshots to other Regions and then restore them to new volumes there, making it easier to leverage multiple AWS Regions for geographical expansion, data centre migration, and disaster recovery.
- Amazon EBS provides the following volume types: General Purpose SSD, Provisioned IOPS SSD, Throughput Optimised HDD, and Cold HDD. For more information, see EBS volume types.

The following is a summary of performance and use cases for each volume type.

- General Purpose SSD volumes (gp2 and gp3) balance price and performance for a wide variety of transactional workloads. These volumes are ideal for use cases such as boot volumes, medium-size single instance databases, and development and test environments.

- Provisioned IOPS SSD volumes (io1 and io2) are designed to meet the needs of I/O-intensive workloads that are sensitive to storage performance and consistency. They provide a consistent IOPS rate that you specify when you create the volume. This enables you to predictably scale to tens of thousands of IOPS per instance. Additionally, io2 volumes provide the highest levels of volume durability.

- Throughput Optimised HDD volumes (st1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.
 - Cold HDD volumes (sc1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential, cold-data workloads. If you require infrequent access to your data and are looking to save costs, these volumes provide inexpensive block storage.
- You can create your EBS volumes as encrypted volumes, in order to meet a wide range of data-at-rest encryption requirements for regulated/audited data and applications. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. Encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage. For more information, see [Amazon EBS encryption](#).
 - Performance metrics, such as bandwidth, throughput, latency, and average queue length, are available through the AWS Management Console. These metrics, provided by Amazon CloudWatch, allow you to monitor the performance of your volumes to make sure that you are providing enough performance for your applications without paying for resources you don't need.



Fig. High-Performance Block Storage

Amazon Aurora:

Amazon Aurora (Aurora) is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. You already know how MySQL and PostgreSQL combine the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. The code, tools, and applications you use today with your existing MySQL and PostgreSQL databases can be used with Aurora. With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

Aurora includes a high-performance storage subsystem. Its MySQL- and PostgreSQL compatible database engines are customised to take advantage of that fast distributed storage. The underlying storage grows automatically as needed. An Aurora cluster volume can grow to a maximum size of 128 tebibytes (TiB). Aurora also automates and standardised database clustering and replication, which are typically among the most challenging aspects of database configuration and administration.

Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the

cloud. If you are not already familiar with Amazon RDS, see the Amazon Relational Database Service User Guide.

The following points illustrate how Amazon Aurora relates to the standard MySQL and PostgreSQL engines available in Amazon RDS:

- You choose Aurora MySQL or Aurora PostgreSQL as the DB engine option when setting up new database servers through Amazon RDS.
- Aurora takes advantage of the familiar Amazon Relational Database Service (Amazon RDS) features for management and administration. Aurora uses the Amazon RDS AWS Management Console interface, AWS CLI commands, and API operations to handle routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.
- Aurora management operations typically involve entire clusters of database servers that are synchronised through replication, instead of individual database instances. The automatic clustering, replication, and storage allocation make it simple and cost effective to set up, operate, and scale your largest MySQL and PostgreSQL deployments.
- You can bring data from Amazon RDS for MySQL and Amazon RDS for PostgreSQL into Aurora by creating and restoring snapshots, or by setting up one-way replication. You can use push-button migration tools to convert your existing RDS for MySQL and RDS for PostgreSQL applications to Aurora.

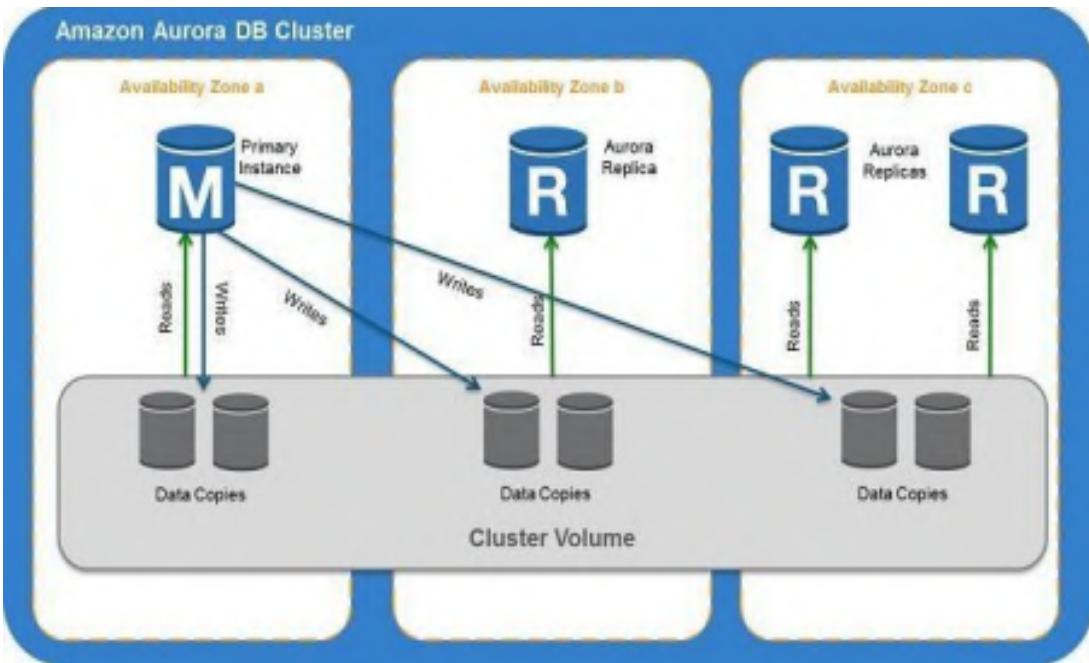


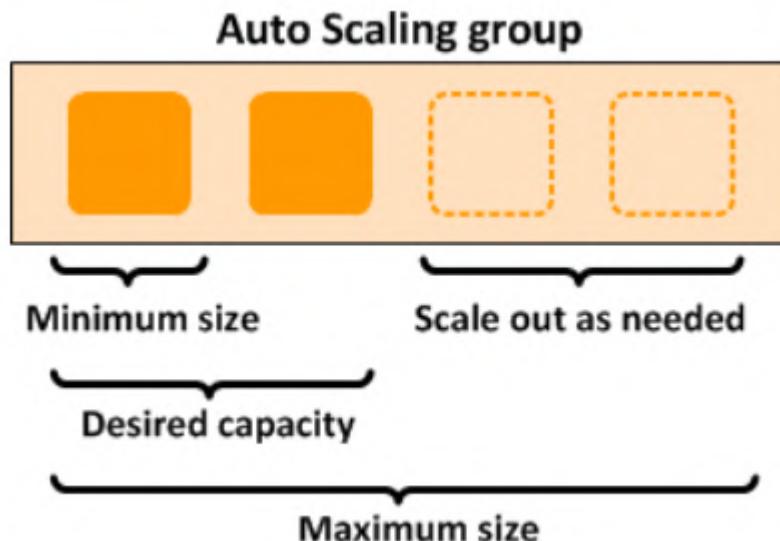
Fig.Amazon Aurora DB Clusters

AWS Autoscaling:

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size

of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



Auto scaling benefits

Adding Amazon EC2 Auto Scaling to your application architecture is one way to maximise the benefits of the AWS Cloud. When you use Amazon EC2 Auto Scaling, your applications gain the following benefits:

- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save

money by launching instances when they are needed and terminating them when they aren't.

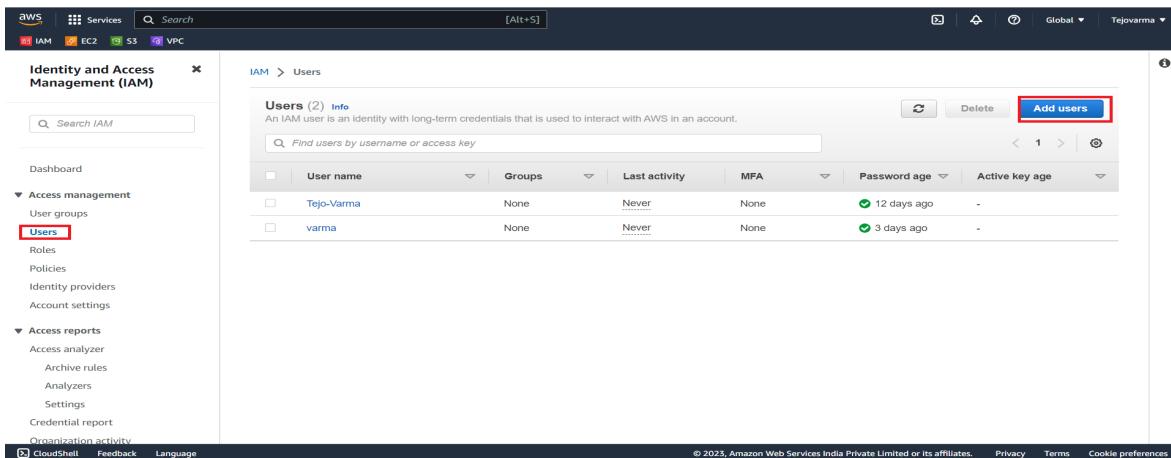
IMPLEMENTATION

Steps to perform:

Create an IAM user with ec2 full access and cloudfront full access

Step-1:

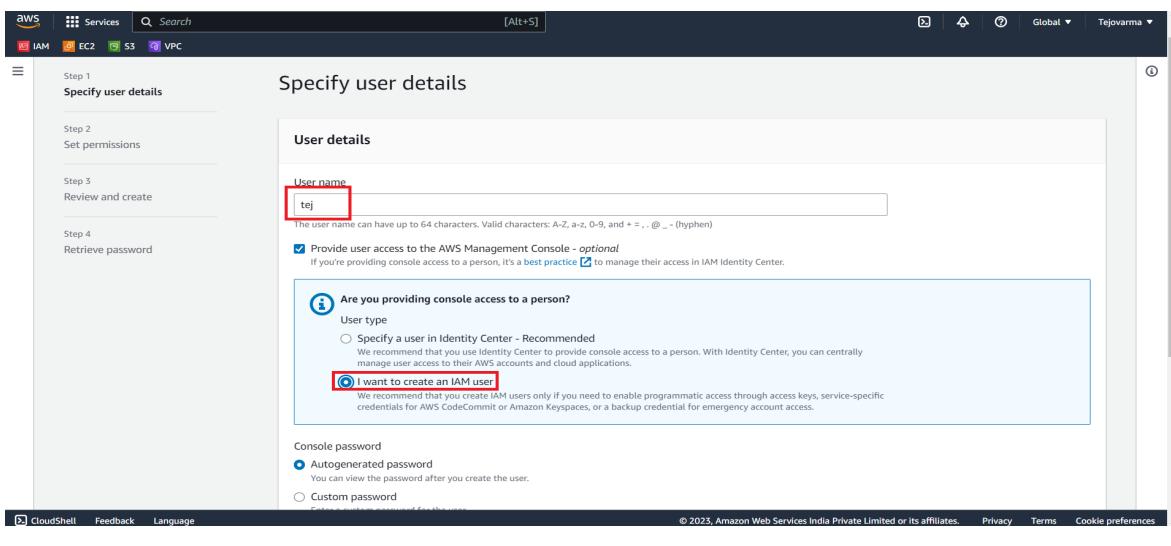
Click on IAM and click on users and click on Add users.



The screenshot shows the AWS IAM service interface. On the left, there's a navigation pane with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Users' is highlighted with a red box. The main area displays a table of users with columns for User name, Groups, Last activity, MFA, Password age, and Active key age. Two users are listed: 'Tejo-Varma' and 'varma'. At the top right of the table, there's a red box around the 'Add users' button.

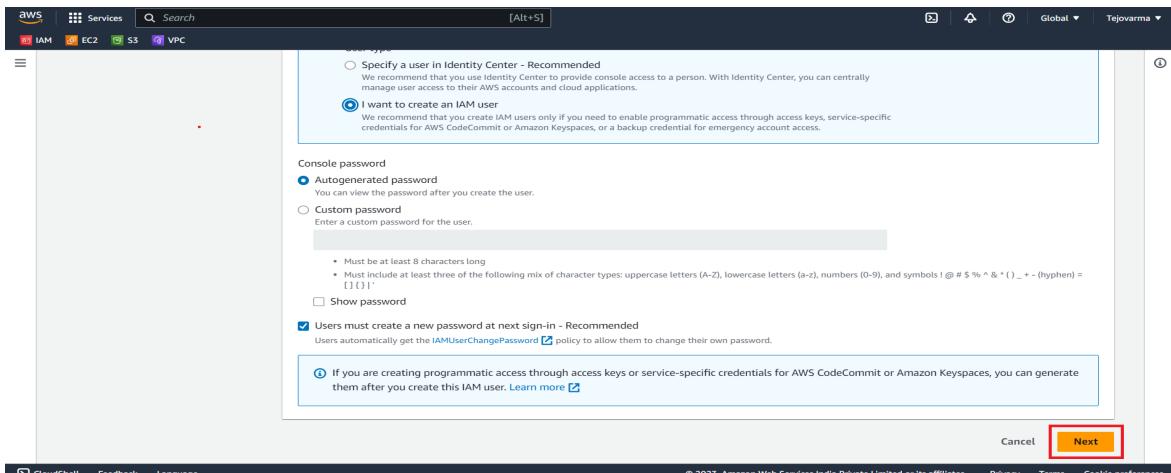
Step-2:

Give the user name and select i want to create an IAM user.

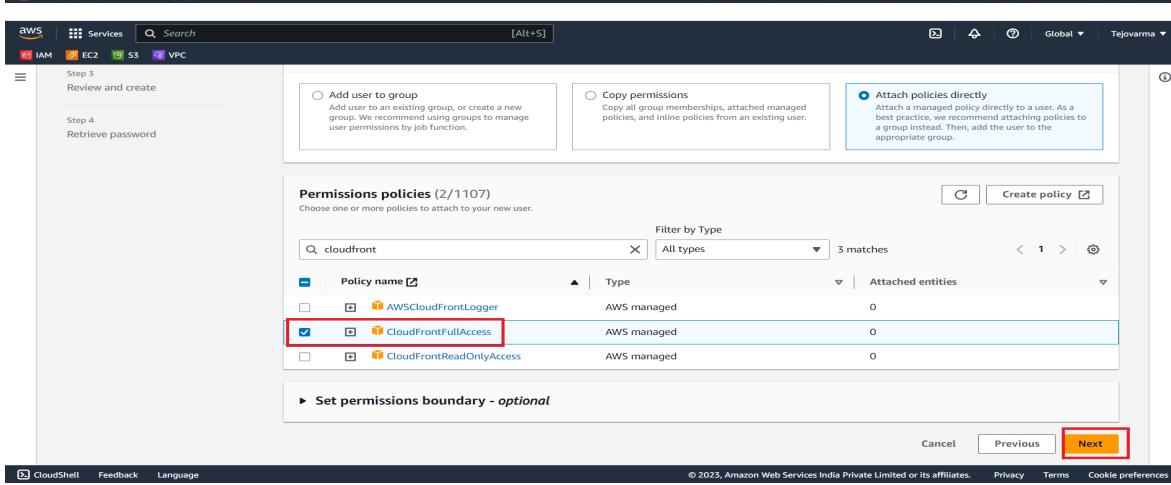
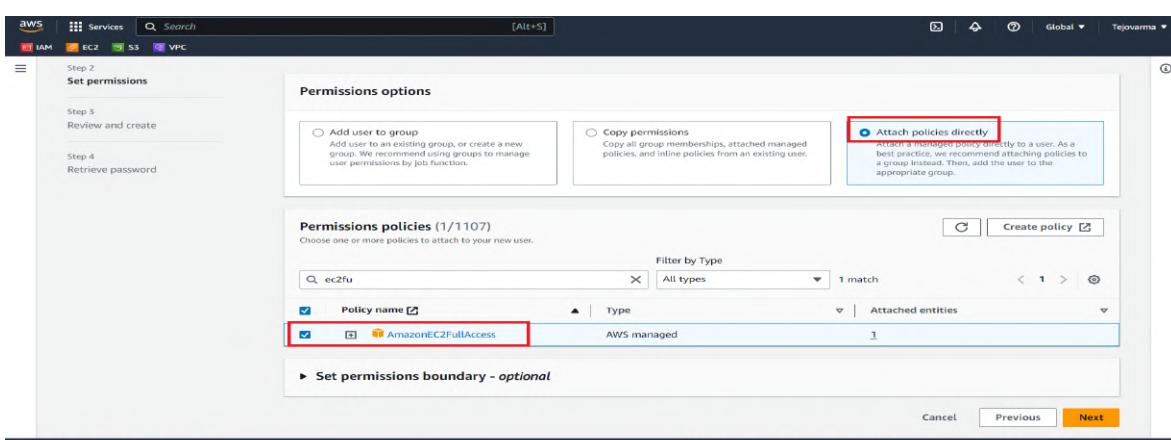


The screenshot shows the 'Specify user details' step of the IAM user creation wizard. On the left, a sidebar lists steps: Step 1 (Specify user details), Step 2 (Set permissions), Step 3 (Review and create), and Step 4 (Retrieve password). The main area has a 'User details' section. In the 'User name' field, 'tej' is entered. Below it, a note says 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and _ = . @ - (hyphen)'. A checkbox 'Provide user access to the AWS Management Console - optional' is checked. A callout box highlights the 'Are you providing console access to a person?' section, which includes a radio button 'I want to create an IAM user' (selected) and a note: 'We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keypairs, or a backup credential for emergency account access.' At the bottom, there are options for 'Console password', 'Autogenerated password' (selected), and 'Custom password'.

Step-3:
Select Autogenerated password and select next.



Step-4:
Select Attach policies directly in permissions give policy name as AMAZON EC2 full Access and also CLOUDFRONT full access and select next.



Step-5: Select create user.

User name: tej
Console password type: Autogenerated
Require password reset: Yes

Name	Type	Used as
AmazonEC2FullAccess	AWS managed	Permissions policy
CloudFrontFullAccess	AWS managed	Permissions policy
IAMUserChangePassword	AWS managed	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
No tags associated with the resource.
[Add new tag](#)

[Cancel](#) [Previous](#) **Create user**

Step-6: User created successfully and click on return to users list.

User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

[View user](#)

Retrieve password
You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details	
Console sign-in URL:	https://376512929915.signin.aws.amazon.com/console
User name:	tej
Console password:	***** Show

[Download .csv file](#) **Return to users list**

Continue without viewing or downloading console password?

⚠️ You haven't viewed or downloaded this user's password. If you continue, you cannot retrieve the password in the future. Instead you must create a new password for the user.

[Cancel](#) **Continue**

The screenshot shows the AWS IAM console. A green banner at the top indicates "User created successfully". The main table lists three users: "tej" (selected), "Tejo-Varma", and "varma". The "tej" row shows "None" for Groups, Last activity, MFA, Password age, and Active key age.

User name	Groups	Last activity	MFA	Password age	Active key age
tej	None	Never	None	None	-
Tejo-Varma	None	Never	None	12 days ago	-
varma	None	Never	None	3 days ago	-

Launch an ec2 instance in two different zones and configure a web server in the instance.

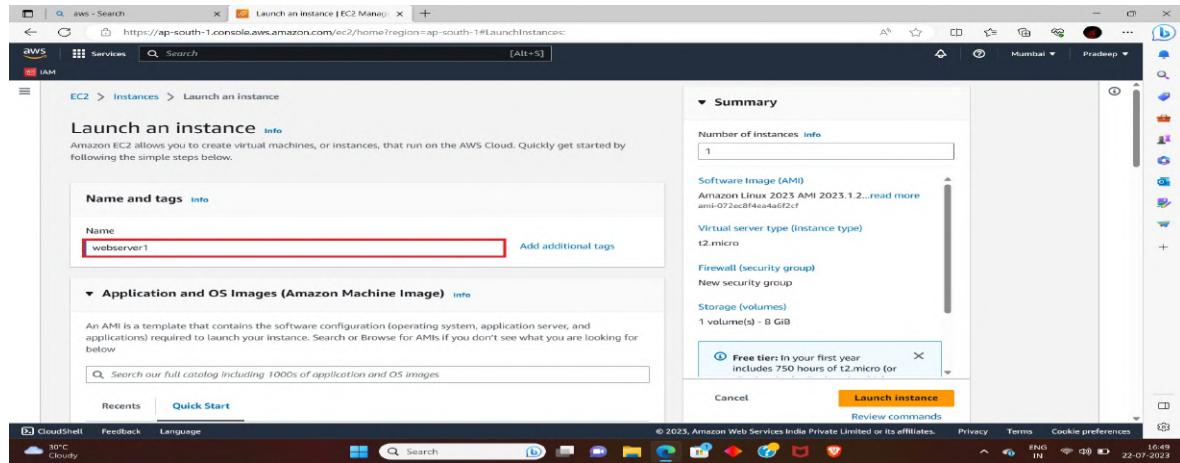
Step-1:

Click on ec2, and then click on launch instance.

The screenshot shows the AWS EC2 Management Console. The top navigation bar has "ec2" selected. The left sidebar shows "Instances" under "Instances". The main search results for "ec2" show "EC2" as the top result. The "Instances" page below shows a table with no instances, with a "Launch Instances" button at the bottom.

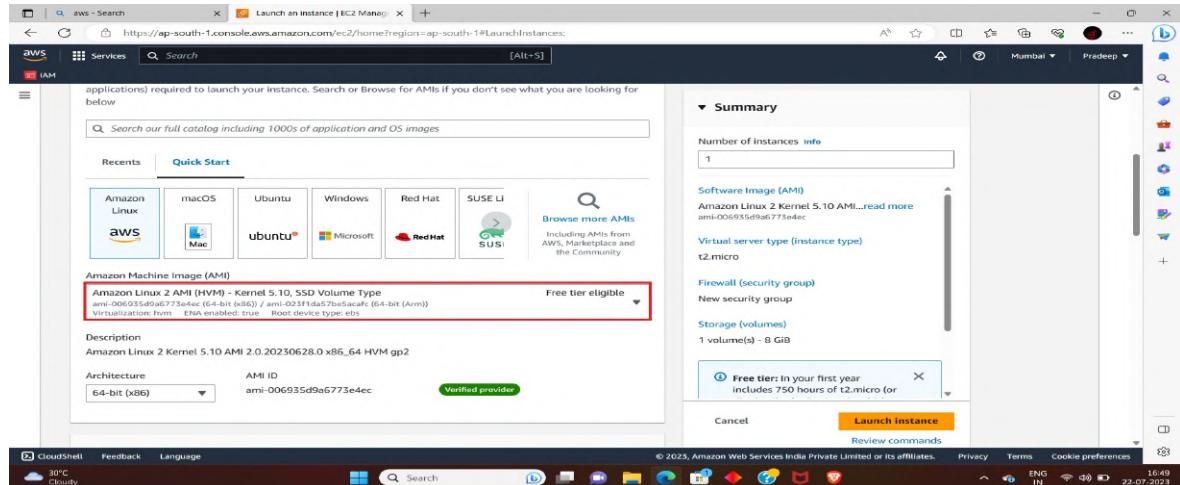
Step-2:

Give name for the EC2 Instance.



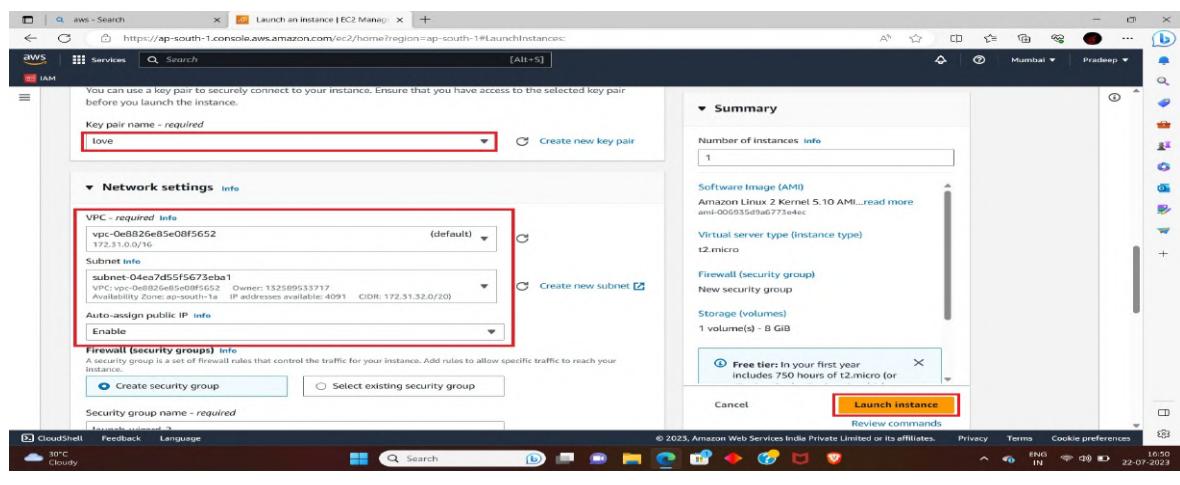
Step-3:

Select the desired AMI for your EC2 instance.



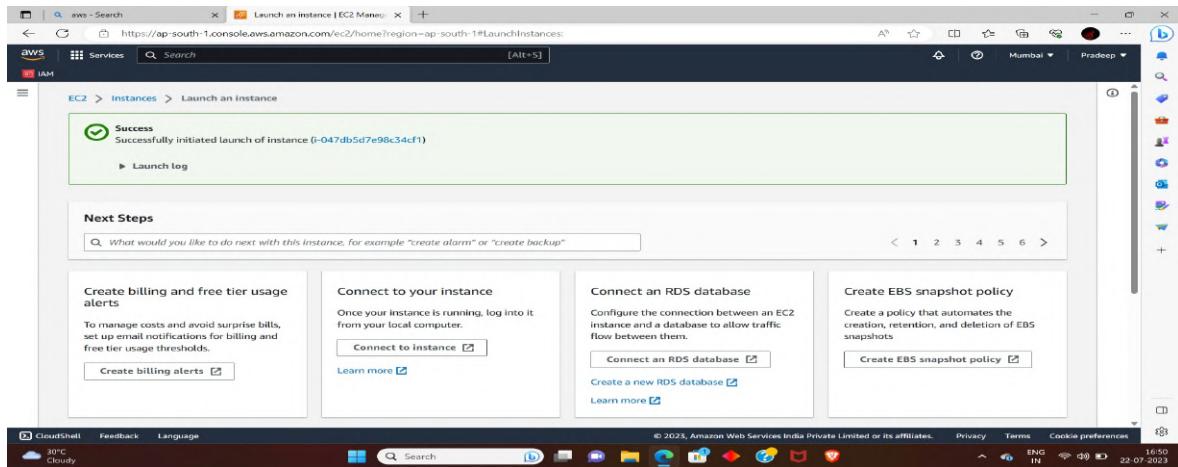
Step-4:

Enter key pair name and select the zone to ap-south-1a, and click on launch instance.



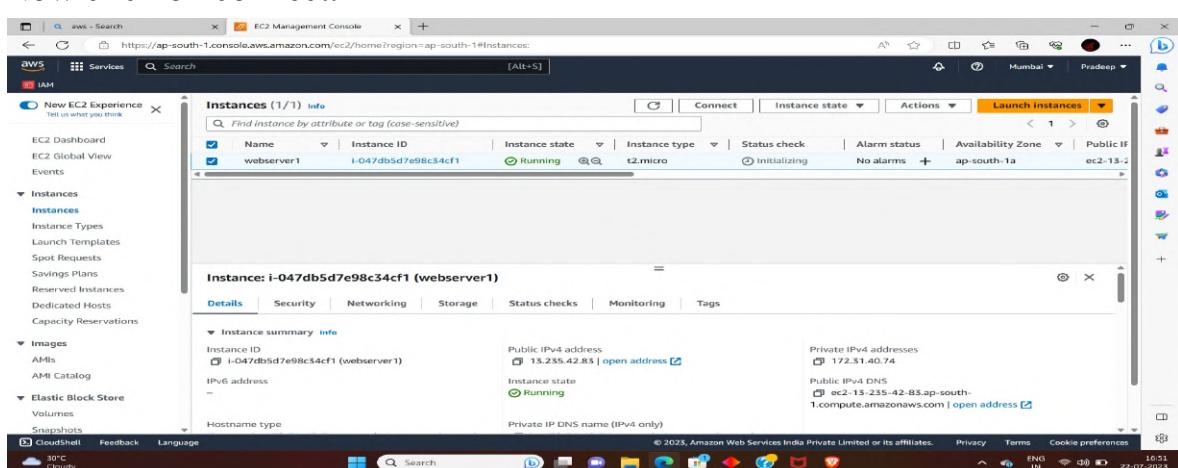
Step-5:

Instance created successfully.



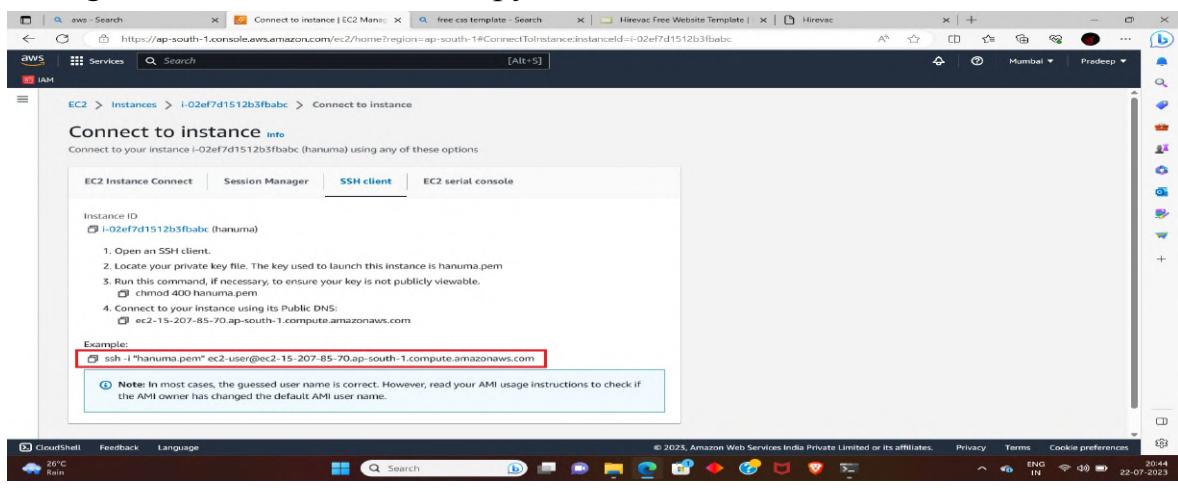
Step-6:

Now click on connect.



Step-7:

Now go to the SSH client and copy the command link.



Step-8:

Now open the command prompt and paste the link.

aws Services Search [Alt+S] Mumbai Tejovarma

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

```
[ec2-user@ip-172-31-11-7 ~]$ sudo su
[root@ip-172-31-11-7 ec2-user]# yum update -y
Last metadata expiration check: 0:00:20 ago on Wed Jul 26 10:32:44 2023.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-11-7 ec2-user]# yum install httpd
Last metadata expiration check: 0:01:54 ago on Wed Jul 26 10:32:44 2023.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.56-1.amzn2023	amazonlinux	48 K
Installing dependencies:				
apr	x86_64	1.7.2-2.amzn2023.0.2	amazonlinux	129 k
apr-util	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	98 K
generic-logos-httpd	noarch	18.0.0-12.amzn2023.0.3	amazonlinux	19 K
httpd-core	x86_64	2.4.56-1.amzn2023	amazonlinux	1.4 M
httpd-filesystem	noarch	2.4.56-1.amzn2023	amazonlinux	15 K
httpd-tools	x86_64	2.4.56-1.amzn2023	amazonlinux	92 K

Connect to Instance | EC2 | EC2 Instance Connect | BLACKBUCKS internship pr | Untitled document - Google | Untitled document - Google | Rent4u Free Website Temp | + | Mumbai | Tejovarma

aws Services Search [Alt+S] Mumbai Tejovarma

Verifying : httpd-core-2.4.56-1.amzn2023.x86_64
Verifying : libbrotli1-0.9-4.amzn2023.0.2.x86_64
Verifying : mod_lua-2.4.56-1.amzn2023.x86_64
Verifying : httpd-tools-2.4.56-1.amzn2023.x86_64
Verifying : mod_http2-2.0.11-2.amzn2023.x86_64
Verifying : mailcap-2.1.49-3.amzn2023.0.3.noarch
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Verifying : httpd-filesystem-2.4.56-1.amzn2023.noarch

Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64 apr-util-1.6.3-1.amzn2023.0.1.x86_64 apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.56-1.amzn2023.x86_64 httpd-core-2.4.56-1.amzn2023.x86_64 httpd-filesystem-2.4.56-1.amzn2023.noarch httpd-tools-2.4.56-1.amzn2023.x86_64
libbrotli1-0.9-4.amzn2023.0.2.x86_64 mailcap-2.1.49-3.amzn2023.0.3.noarch mod_http2-2.0.11-2.amzn2023.x86_64 mod_lua-2.4.56-1.amzn2023.x86_64

Completed!
[root@ip-172-31-11-7 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-11-7 ec2-user]# systemctl start httpd
[root@ip-172-31-11-7 ec2-user]# wget https://www.free-css.com/assets/files/free-css-templates/download/page294/rent4u.zip
2023-07-26 10:38:09... https://www.free-css.com/assets/files/free-css-templates/download/page294/rent4u.zip
Receiving...[root@ip-172-31-11-7 ec2-user]# curl https://www.free-css.com/...[217.160.0.242, 2001:bd0:100ff:f000::28f
Connecting to www.free-css.com (www.free-css.com) [217.160.0.242]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 919720 (898K) [application/zip]
Saving to: 'rent4u.zip'

rent4u.zip 100%[=====] 898.16K 1.14MB/s in 0.8s
2023-07-26 10:38:11 (1.14 MB/s) - 'rent4u.zip' saved [919720/919720]
[root@ip-172-31-11-7 ec2-user]# unzip rent4u.zip
CloudShell Feedback Language Show all
hanuma.pem
26°C Cloudy 1608 26-07-2023
aws Services Search [Alt+S] Mumbai Tejovarma

inflating: rent4u-html/images/mail.png
inflating: rent4u-html/images/next.png
inflating: rent4u-html/images/prev.png
inflating: rent4u-html/images/r-1.png
inflating: rent4u-html/images/r-2.png
inflating: rent4u-html/images/r-3.png
inflating: rent4u-html/images/r-4.png
inflating: rent4u-html/images/r-5.png
inflating: rent4u-html/images/r-6.png
inflating: rent4u-html/images/right-angle-grey.png
inflating: rent4u-html/images/right-angle.png
inflating: rent4u-html/images/us-bg.png
inflating: rent4u-html/images/us-1.png
inflating: rent4u-html/images/us-2.png
inflating: rent4u-html/images/us-3.png
inflating: rent4u-html/images/us-4.png
inflating: rent4u-html/images/us-bg.jpg
inflating: rent4u-html/index.html
creating: rent4u-html/js/
inflating: rent4u-html/js/bootstrap.js
inflating: rent4u-html/js/custom.js
inflating: rent4u-html/js/jquery-3.4.1.min.js
[root@ip-172-31-11-7 ec2-user]# ls
rent4u-html rent4u.zip
[root@ip-172-31-11-7 rent4u-html]# ls
about.html blog.html car.html contact.html css images index.html js
[root@ip-172-31-11-7 rent4u-html]# mv * /var/www/html
[root@ip-172-31-11-7 rent4u-html]# ls
[root@ip-172-31-11-7 rent4u-html]# cd /var/www/html
[root@ip-172-31-11-7 html]#

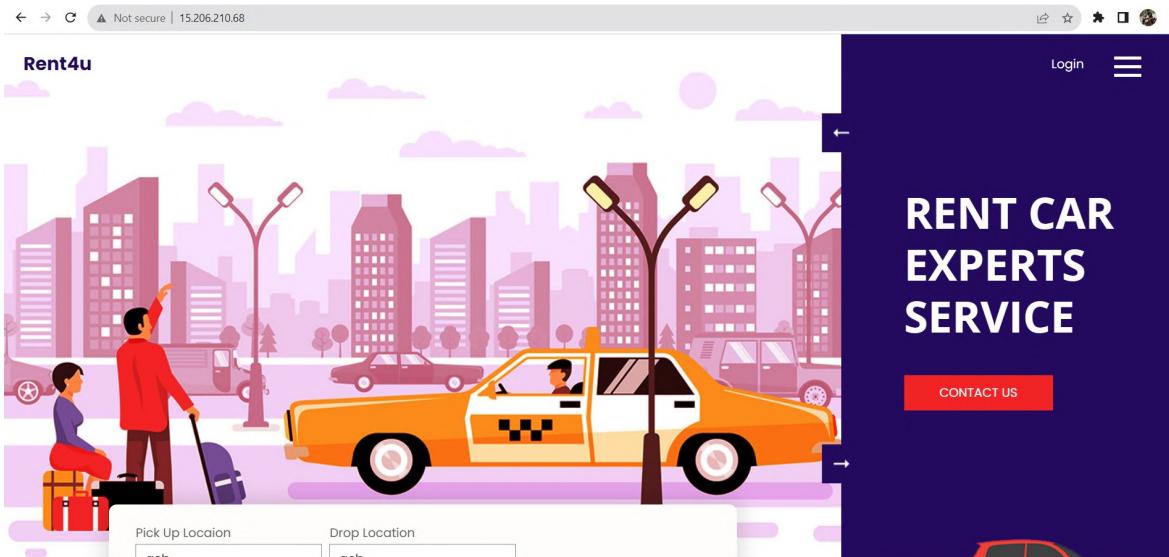
Step-9:

Go to the ec2 instance, connect and copy the IP address.

The screenshot shows the AWS EC2 Instance Connect interface. At the top, there are tabs for 'EC2 Instance Connect', 'Session Manager', 'SSH client', and 'EC2 serial console'. The 'EC2 Instance Connect' tab is selected. Below it, the instance ID 'i-04ae86df517eb5ce7 (sadulla)' is listed. Under 'Connection Type', the 'Connect using EC2 Instance Connect' option is selected, with a note that it connects using a browser-based client with a public IPv4 address. The 'Public IP address copied' message is displayed. A public IP address '15.206.210.68' is shown. The 'User name' field contains 'ec2-user'. At the bottom, there is a large blue button labeled 'CONNECT'.

Step-10:

Enter the public IP address of your EC2 instance directly into the browser's address bar and press Enter.



Now the web server was created successfully.

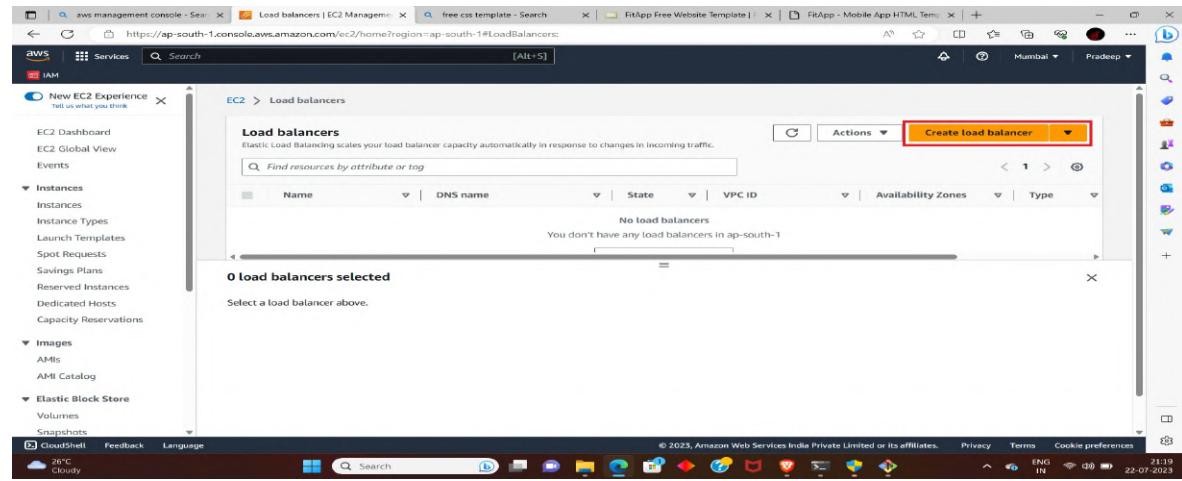
Step-11:

Similarly create another instance with a different region.

Create a load balancer in the specified region and balance traffic between the two instances.

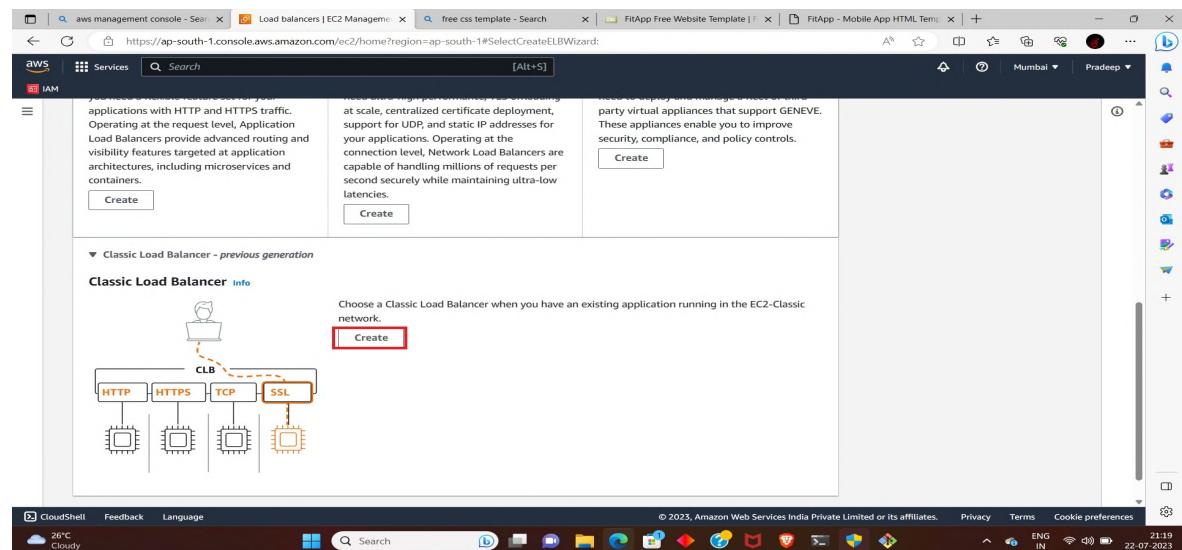
Step-1:

Search for load balancer and select create load balancer.



Step-2:

Click on create.



Step-3:

Enter load balancer name and select internet-facing.

The screenshot shows the AWS Management Console with the URL <https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateCLBWizard>. The browser tab is titled "Load balancers | EC2 Management". The main content area is titled "Create Classic Load Balancer" with a "Basic configuration" sub-section. The "Load balancer name" input field contains "pspk". Below it, a note states: "A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen." The "Scheme" section has two options: "Internet-facing" (selected) and "Internal". A tooltip for "Internet-facing" says: "An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more". The "Network mapping" section is visible at the bottom. The operating system taskbar at the bottom shows various application icons and the date/time as 22-07-2023.

Step-4:

Select VPC and subnet mapping.

The screenshot shows the AWS Management Console with the same URL as the previous step. The "Network mapping" sub-section is selected. It displays a list of VPCs. One VPC is listed: "vpc-0e8826e05ae08f5652" with "IPv4: 172.31.0.0/16". Below this, the "Mappings" section shows two subnets selected for mapping: "ap-south-1a (aps1-az1)" and "ap-south-1b (aps1-az3)". The operating system taskbar at the bottom shows various application icons and the date/time as 22-07-2023.

Step-5: Select TCP.

The screenshot shows the AWS EC2 Create Load Balancer Wizard Step 5: Listener Configuration. It displays two listeners: Listener HTTP:81 (Protocol: HTTP, Port: 81) and Instance HTTP:80 (Protocol: HTTP, Port: 80). Under Health checks, TCP is selected as the protocol with a ping port of 80 and a ping path of /index.html.

Step-6: Now select two instances and choose to enable connection draining.

The screenshot shows the AWS EC2 Create Load Balancer Wizard Step 6: Target Instances. It displays two instances selected: i-02ef7d1512b3fbabc (hanuma) and i-098fc05baff2b40c (ram), both running in the launch-wizard-1 security group. The 'Enable connection draining' checkbox is checked.

Step-7: Click on create load balancer.

The screenshot shows the AWS EC2 Create Load Balancer Wizard Step 7: Summary. It displays the configuration details and a 'Create load balancer' button.

Step-8:

Load balancer created successfully.

The screenshot shows the AWS Management Console with multiple tabs open. The active tab is 'Load balancers | EC2 Management' at the URL <https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateLBWizardSuccess:loadBalancerArn=pspk>. A green success message box at the top states 'Successfully created load balancer: pspk'. Below it, a note says 'Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.' The breadcrumb navigation shows 'EC2 > Load balancers > pspk > Create Classic Load Balancer'. A 'View load balancer' button is highlighted with a red box. The bottom of the screen shows the Windows taskbar with various pinned icons and the date/time as 22-07-2023.

Click on view load balancer.

Verify the load balancer dns on the browser.

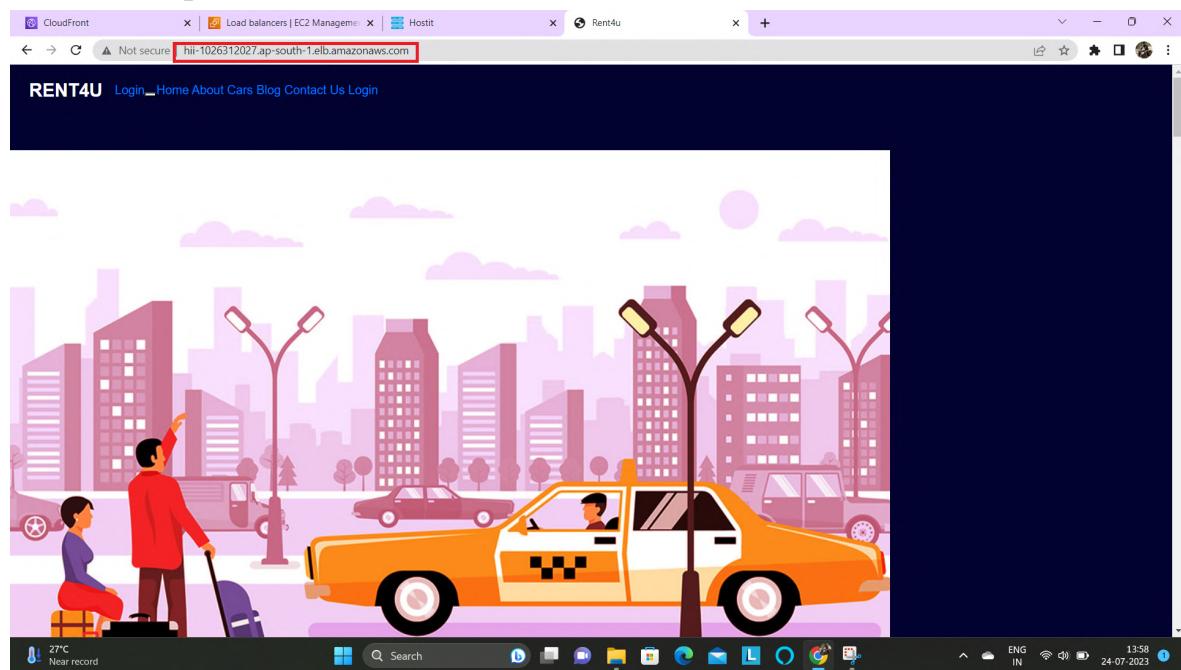
Step-1:

Copy the DNS name of the load balancer.

The screenshot shows the AWS Management Console with the 'Load balancers' page open. The left sidebar shows 'Load Balancing' selected under 'Load Balancers'. The main pane displays a table of load balancers with one entry: 'hi' (DNS name copied). A red box highlights the 'DNS name copied' status message next to the entry. The bottom of the screen shows the Windows taskbar with various pinned icons and the date/time as 22-07-2023.

Step-2:

Paste the copied DNS in the new tab.



Load balancer dns verified on the browser.

Create a cloud front distribution and push the load balancer dns to the cloud front.

CLOUDFRONT:

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centres called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

Step-1:

Search for cloudfont and open cloudfont.

The screenshot shows the AWS search interface with the query 'cloudfront'. The results are categorized under 'Services' and 'Features'. In the 'Services' section, 'CloudFront' is highlighted with a red box. A modal window for 'CloudFront' is open, showing its details: 'Global Content Delivery Network'. The 'Create target group' button is also visible in the modal.

Step-2:

Click on create distribution.

The screenshot shows the 'Distributions' page with one existing distribution listed. The 'Create distribution' button is highlighted with a red box at the top right of the table header.

Step-3:

Choose the origin domain from elastic load balancer.

The screenshot shows the 'Create distribution' wizard at the 'Origin' step. Under 'Origin domain', there is a dropdown menu labeled 'Choose origin domain' which has 'Amazon S3' selected. Below it, 'Elastic Load Balancer' is listed with 'hii' selected. This 'Elastic Load Balancer' entry is highlighted with a red box.

Step-4:

Choose legacy cache settings and also change headers to All.

The screenshot shows the 'Cache key and origin requests' section of the CloudFront configuration. The 'Legacy cache settings' radio button is selected. In the 'Headers' section, the dropdown menu is set to 'All'. A note below states: 'You are choosing to forward all headers to the origin, which means CloudFront doesn't cache objects for this cache behavior. It sends every request to the origin.' Other sections like 'Query strings' and 'Cookies' have their dropdowns set to 'None'.

Step-5:

Now Do not enable security protections in Web application firewall(WAF).

The screenshot shows the 'Web Application Firewall (WAF)' configuration. The 'Do not enable security protections' radio button is selected. A note below it says: 'Select this option if your application does not need security protections from AWS WAF.' The 'Enable security protections' option is also shown with its description.

Step-6:

Finally click on create distribution.

The screenshot shows the 'Create distribution' page. The 'Request certificate' dropdown is set to 'Choose certificate'. Under 'Supported HTTP versions', 'HTTP/2' is checked. Under 'Default root object - optional', there is a text input field. Under 'Standard logging', 'Off' is selected. Under 'IPv6', 'On' is selected. Under 'Description - optional', there is a text input field. At the bottom right, the 'Create distribution' button is highlighted with a red box.

Step-7:

Successfully created a new distribution.
Now copy the Distribution domain name.

The screenshot shows the AWS CloudFront console with a success message: "Successfully created new distribution." A red box highlights the distribution domain name "d36gbos2lxxam.cloudfront.net".

The distribution details page for "EBOQQQMPDL8G7" is displayed. The "General" tab is selected. The "Details" section shows the ARN and last modified status. The "Settings" section includes fields for Description, Alternate domain names, Standard logging, Price class, and cookie logging.

Step-8:

Paste the Distribution domain name in the new tab.

A browser window shows the RENT4U website loading at the URL "d36gbos2lxxam.cloudfront.net". The page features a cartoon illustration of a city street with a yellow taxi and a person hailing it.