Final Project
On
Cloud System Administration

# DEVELOPING A DYNAMIC QUIZ WEBSITE WITH MySQL & PHP THROUGH AWS

Industry-linked skill-development Summer Training Program


KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

Submitted by,


| S.No. | Name of Student | Roll number |
|---|---|---|
| **1.** | INDRAJIET GHOSAL | 1804027 |
| **2.** | SARWAM SINHA | 1804529 |
| **3.** | PIYUSH AGARWAL | 1905545 |
| **4.** | ROHIT GHOSH | 1929038 |

Under the guidance of,
**MR. ANJAN CHATTERJEE**

Career Advisory & Augmentation Services (CAAS),
Training & Placement Department.
**Kalinga Institute of Industrial Technology**
(Deemed to be University)
Bhubaneswar, Odisha


**MAY - JUNE, 2021.**

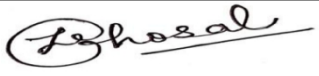
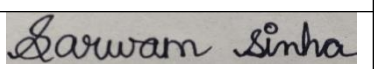
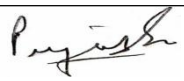
# ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

This project would not have been possible without the kind support and help of **Dr. Suman Bhattacharya, Dr. Saranjit Singh, Dr. Prachet Bhuyan, Dr. Deepak Raut** and all other esteemed faculties from **CAAS** under T&P Department of KIIT DU. We would like to extend our sincere thanks to all of them. Nonetheless, we express our gratitude towards our parents for their kind co-operation and encouragement which helped us in completion of this project amid the COVID-19 pandemic.

We express our sincere gratitude to **Mr. Anjan Chatterjee**, our project-in-charge, under the able guidance of whom, we have successfully accomplished this project. He helped us understand the intricate issues involved in the project-making. With deep sense of gratitude, we acknowledge the encouragement and guidance received from him.

## STUDENT SIGNATURE

| Roll Number | Name of Student | Signature |
|---|---|---|
| 1804027 | Indrajiet Ghosal |  |
| 1804529 | Sarwam Sinha |  |
| 1905545 | Piyush Agarwal |  |
| 1929038 | Rohit Ghosh |  |

# PROJECT ABSTRACT

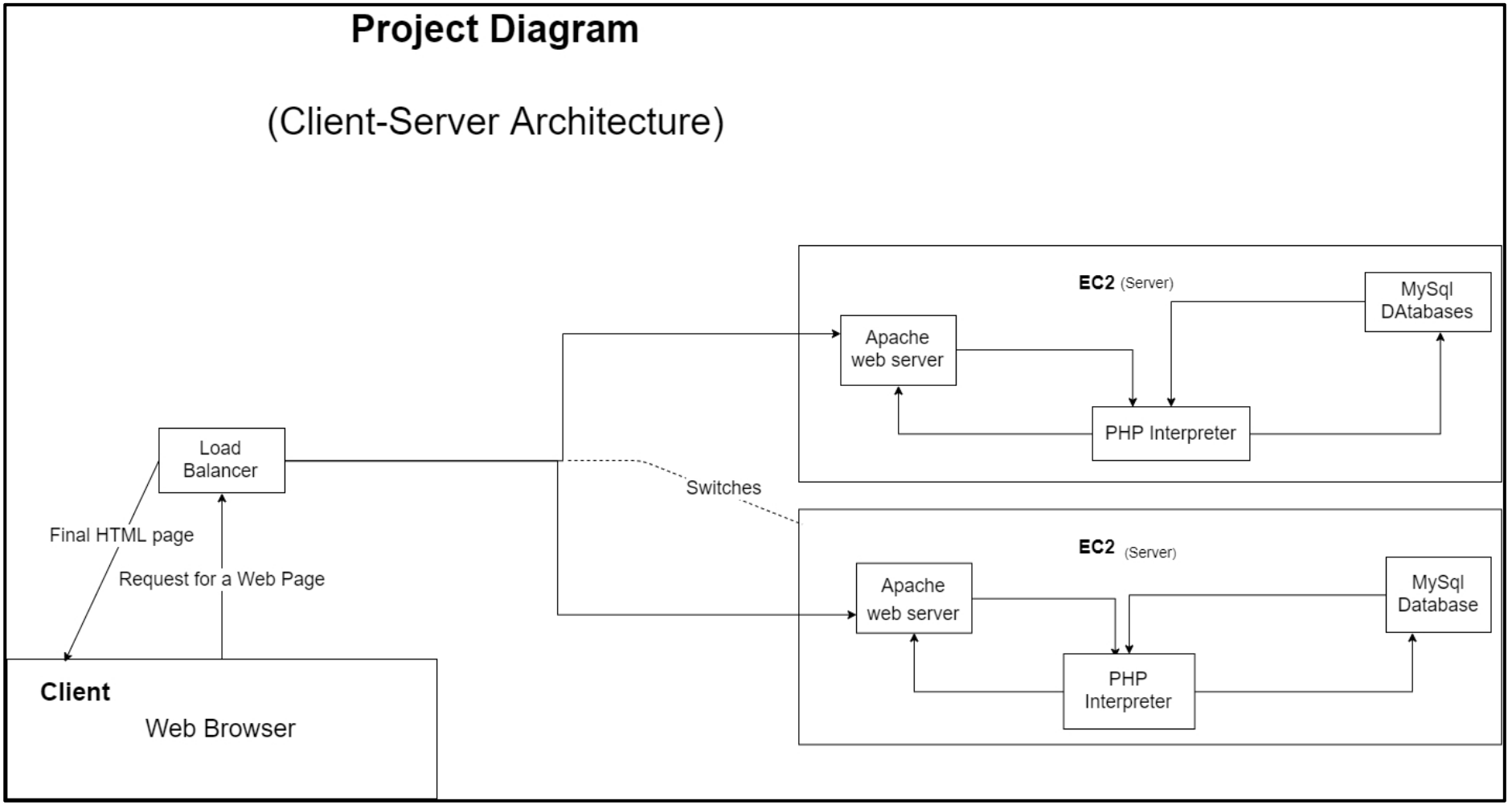
Amazon Web Services (AWS) is one of the most popular and efficient cloud platforms for administering and deploying applications to make them resilient and robust. AWS for System Administrators helped us to learn several advanced cloud administration concepts for deploying, managing, and operating highly available systems on AWS.

Starting with the fundamentals of identity and access management (IAM) for securing the environment, our project gradually parses through AWS networking and monitoring tools. Going forward, this project grips with VPC, EC2, S3, load balancer, Auto Scaling, RDS database, and data management. Nonetheless, we have initiated AWS automated backups and store and keep track of log files and also maintained the versioning of the uploaded files. Later, we implemented and worked with AWS APIs and understood their usage along with CloudFormation, Hypertext Preprocessor (PHP form), and MySQL to automate infrastructure.

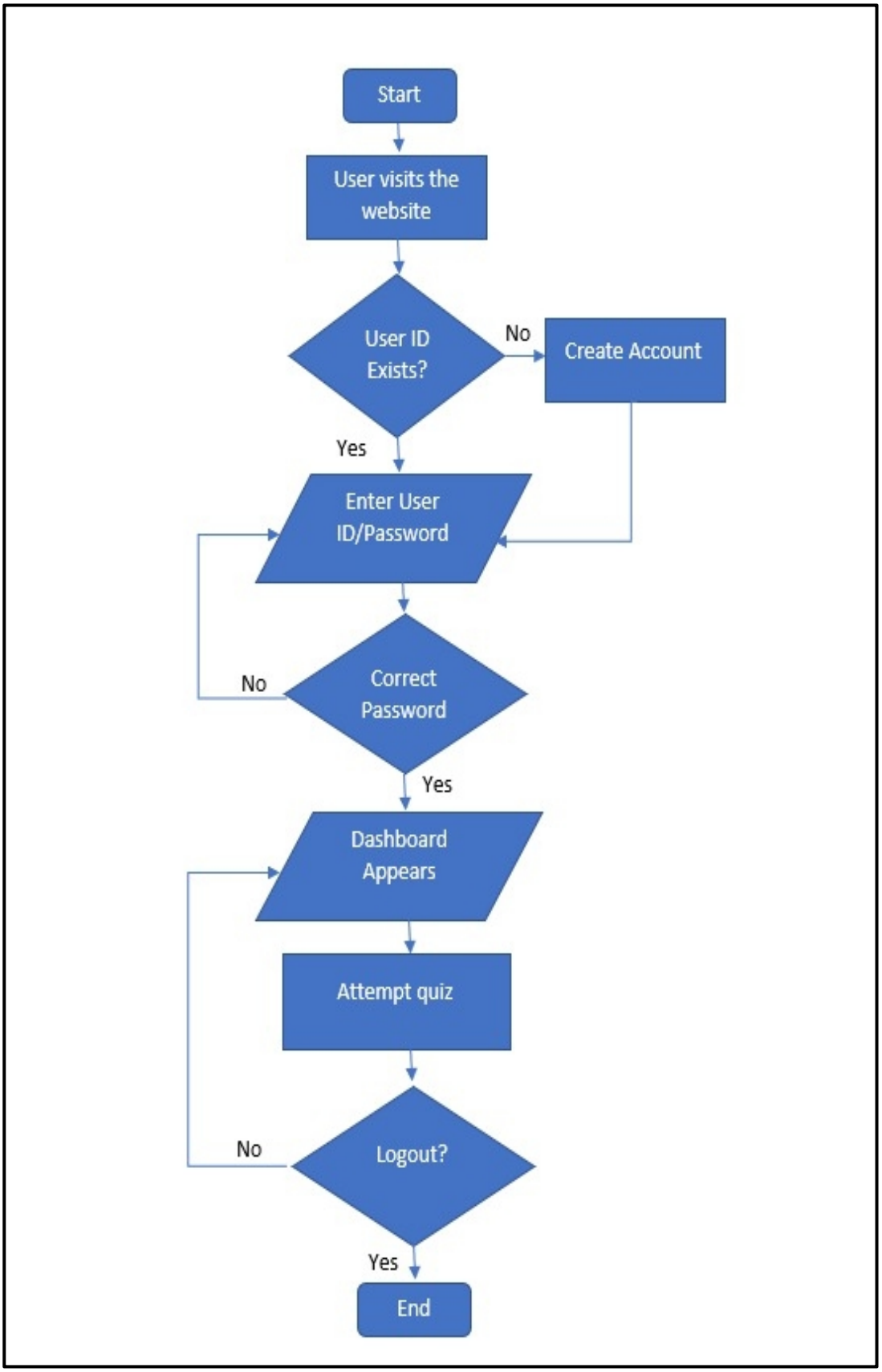
By the end of this project, we are ready to deploy a dynamic website which takes real-time user input in the web browser, where the user will login to the system which is saved in the database simultaneously. Subsequently the user then goes on to attempt the quiz prepared using PHP. This project is built with all the necessary infrastructure, monitoring, and logging components in place.

# TABLE OF CONTENTS

## PROJECT TOTAL DIAGRAM



# PROJECT FLOWCHART



# INDIVIDUAL CONTRIBUTION

## 1. INDRAJIET GHOSAL

*KIIT ROLL NUMBER* : **1804027**

*EMAIL ID :* *1804027@kiit.ac.in*

*PHONE NUMBER :* **8017440496**

*CLASS :* **ETC-1**

*SEMESTER :* **7th**

*ROLE :*

- ✧ Created S3 bucket and maintained the files uploaded with versioning.
- ✧ Took part in front-end and UI design.
- ✧ Created Elastic Load Balancer for the instances.
- ✧ Secured the IP Address using CloudFront.
- ✧ Made IAM roles and shared the authorized others to use the AWS resources, creating policies and attaching them.
- ✧ Spent time guiding other team members on specific tasks of the project to assist the team in reaching project goals.
- ✧ Project ideation, documentation and editing.

## 2. SARWAM SINHA

*KIIT ROLL NUMBER* : **1804529**

*EMAIL ID : 1804529@kiit.ac.in*

*PHONE NUMBER : 7609884328*

*CLASS :* **ETC-8**

*SEMESTER :* **7th**

*ROLE :*

- ✧ Created VPC peering connection.
- ✧ Made IAM roles and shared the authorized others to use the AWS resources.
- ✧ Took part in front-end and UI design.
- ✧ Implemented the AWS Backup & Restore facility.
- ✧ Allocated the Elastic IP.
- ✧ Perform constructive peer review.
- ✧ Project ideation and documentation.

## 3. PIYUSH AGARWAL

*KIIT ROLL NUMBER* : **1905545**

*EMAIL ID : 1905545@kiit.ac.in*

*PHONE NUMBER :* **9939526800**

*CLASS :* **CSE-8**

*SEMESTER :* **5th**

*ROLE :*

- ✧ Created the front-end and back-end of login and sign-up system.
- ✧ Design and implementation of the database.
- ✧ Structuring the document.

## 4. ROHIT GHOSH

*KIIT ROLL NUMBER* : **1929038**

*EMAIL ID :* *1929038@kiit.ac.in*

*PHONE NUMBER :* **8420105422**

*CLASS :* **CSCE-1**

*SEMESTER :* **5th**

*ROLE :*

- ✧ Designed the front-end and back-end of quiz pages.
- ✧ Design and implementation of the database.
- ✧ Parts of documentation.

# CHAPTER 1
# BACKGROUND

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, one 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). Large clouds, predominant today, often have functions distributed over multiple locations from central servers. Clouds may be limited to a single organization, or be available to multiple organizations (public cloud). Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Advocates of public and hybrid clouds note that cloud computing allows companies to avoid or minimize up-front IT infrastructure costs.

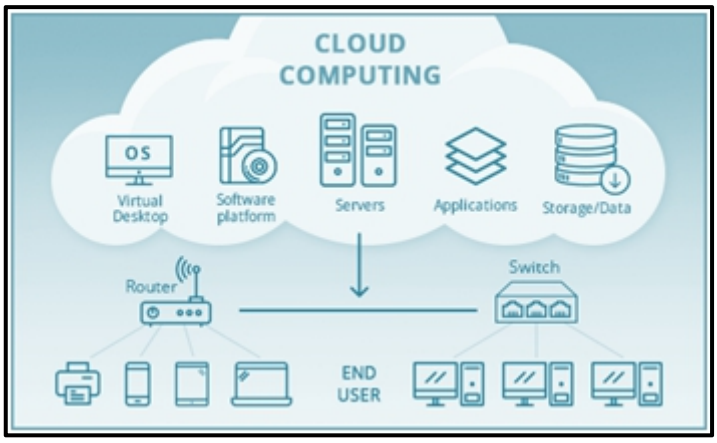



Fig. 1 : Cloud computing architecture

Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand,providing the burst computing capability: high

computing power at certain periods of peak demand. Cloud providers typically use a "pay-as-you-go" model, which can lead to unexpected operating expenses if administrators are not familiarized with cloud-pricing models.

The availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture and autonomic and utility computing has led to growth in cloud computing. As of 2017, most cloud computers run a Linux-based operating system.

Cloud computing was popularized with Amazon.com releasing its Elastic Compute Cloud product in 2006. References to the phrase "cloud computing" appeared as early as 1996, with the first known mention in a Compaq internal document.The cloud symbol was used to represent networks of computing equipment in the original ARPANET by as early as 1977, and the CSNET by 1981—both predecessors to the Internet itself. The word cloud was used as a metaphor for the Internet and a standardized cloud-like shape was used to denote a network on telephony schematics. With this simplification, the implication is that the specifics of how the endpoints of a network are connected are not relevant to understanding the diagram. The genesis of AWS was when in the early 2000s, experience with building Merchant.com, Amazon's e-commerce-as-a-service platform for third-party retailers to build their own web-stores, made them pursue service-oriented architecture as a means to scale their engineering operations led by the then CTO, Allan Vermeulen.

Around the same time frame, Amazon sought out to create "a shared IT platform" so its engineering organizations which were spending 70% of their time on "undifferentiated heavy-lifting" such as IT and infrastructure problems could focus on customer-facing innovation instead. Besides, in dealing with unusual peak traffic patterns especially during the holiday season, migrating services to commodity Linux hardware, and reliance on open source software already had Amazon's Infrastructure team, led by Tom Killalea, Amazon's first CISO, run their data centers and associated services in a "fast, reliable, cheap" way.

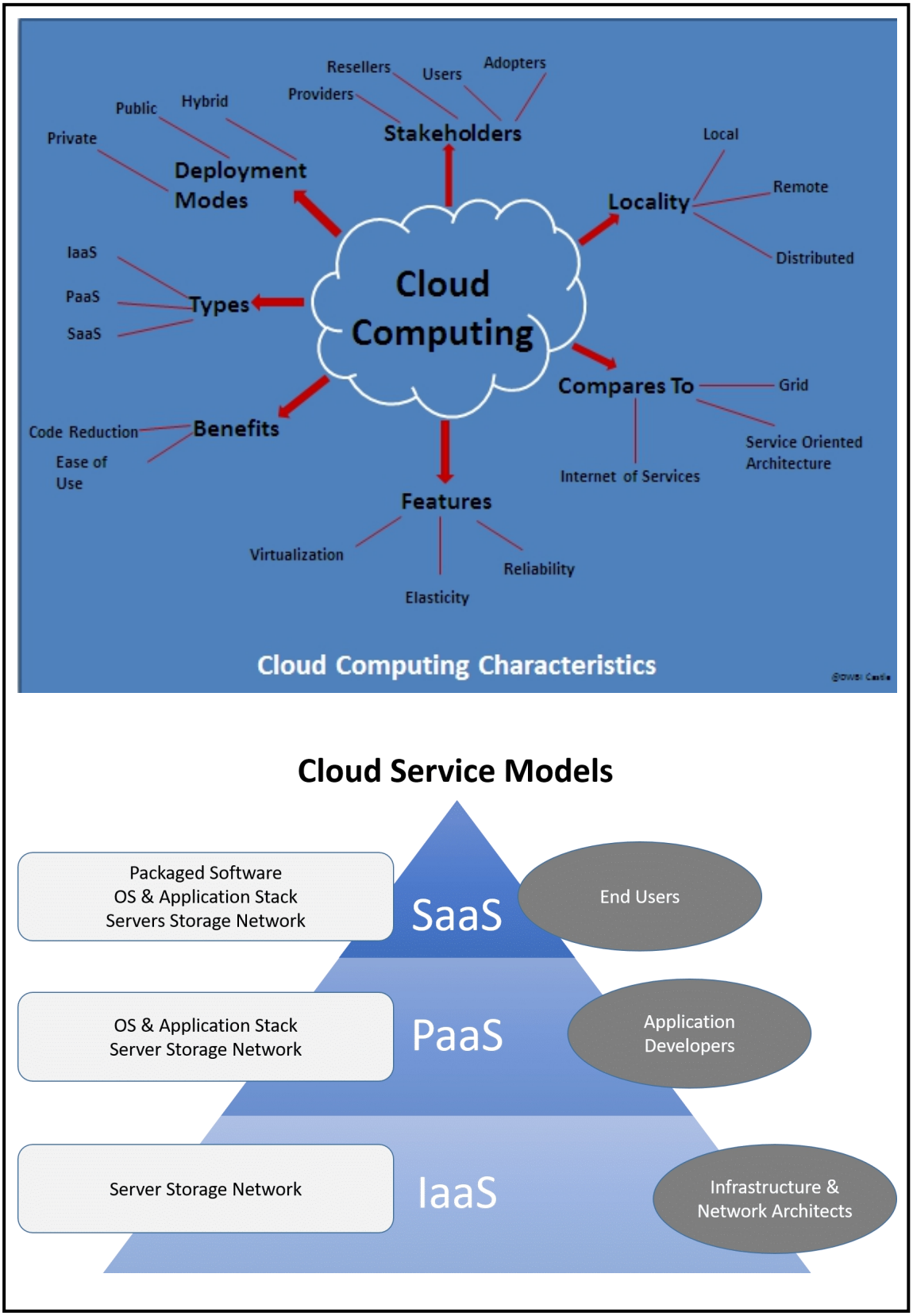



Fig. 2: Basics of cloud computing

# CHAPTER 2

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

## 2.1 : Introduction to EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows us to obtain and configure capacity with minimal friction. It provides us with complete control of our computing resources and lets us run on Amazon's proven computing environment.

Fig. 3 : Amazon Elastic Compute Cloud (EC2)

More SAP, HPC, Machine Learning, and Windows workloads run on AWS than any other cloud. EC2 provides users with control over the geographical location of

instances that allows for latency optimization and high levels of redundancy. As of January 2019, the following instance types were offered:

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

### 2.1.1 : Cost

As of December 2010 Amazon offered a bundle of free resource credits to new account holders. The credits are designed to run a "micro" sized server, storage (EBS), and bandwidth for one year. Unused credits cannot be carried over from one month to the next.

### 2.1.2 : EC2 Free Tier

As of December 2010 Amazon offered a bundle of free resource credits to new account holders. The credits are designed to run a "micro" sized server, storage (EBS), and bandwidth for one year.Unused credits cannot be carried over from one month to the next.

### 2.1.3 : Reserved Instance

Reserved instances enable EC2 or RDS service users to reserve an instance for one or three years.

### 2.1.4 : Spot Instances

Amazon EC2 Spot instances are spare compute capacity in the AWS cloud available at up to 90% discount compared to On-Demand prices.

### 2.1.5 : Savings Plans

EC2 Instance Savings plans provide the lowest prices but are less flexible meaning a user must commit to individual instance families within a region to take advantage, but with the freedom to change instances within the family in that region.

## 2.2 : Features of EC2

- Amazon EC2 provides the following features:
- Virtual computing environments, known as instances.
- Pre-configured templates for our instances, known as Amazon Machine Images (AMIs), that package the bits we need for our server (including the operating system and additional software).
- Various configurations of CPU, memory, storage, and networking capacity for our instances, known as instance types.
- Secure login information for our instances using key pairs (AWS stores the public key, and we store the private key in a secure place).
- Storage volumes for temporary data that's deleted when we stop, hibernate, or terminate our instance, known as instance store volumes.
- Persistent storage volumes for our data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes.

# CHAPTER 3

# AMAZON RELATIONAL DATABASE SERVICE (RDS)

## 3.1 : Introduction to RDS

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks such as hardware provisioning, database setup, patching and backups. It frees us to focus on our applications so we can give them the fast performance, high availability, security and compatibility they need. Amazon RDS is available on several database instance types - optimized for memory, performance or I/O.

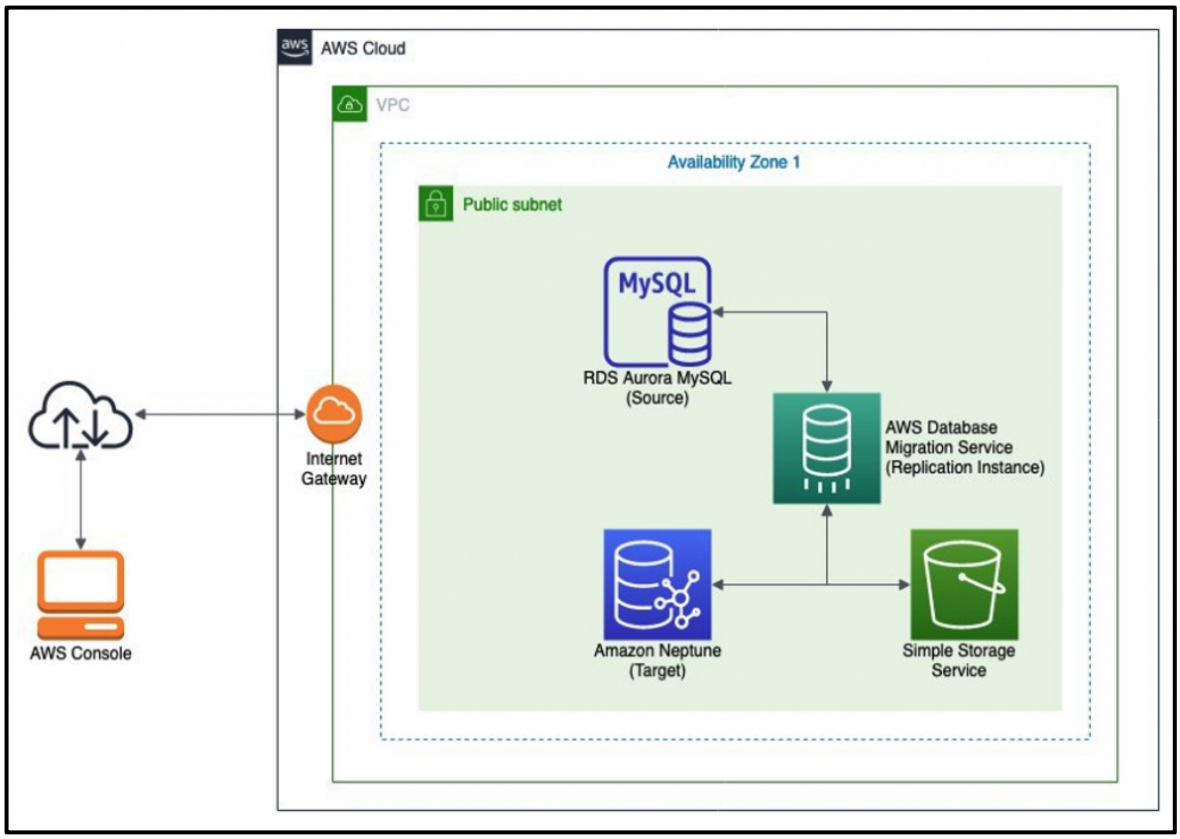



Fig. 4 : Populating a graph from a relational database using AWS.

## 3.2 : Benefits of Amazon RDS

- Easy to administer
- Highly scalable
- Fast
- Secure
- Inexpensive

## 3.3 : Working with Amazon RDS

- **AWS Management Console** : The AWS Management Console is a simple web-based user interface. One can manage our DB instances from the console with no programming required.

- **Command line interface (CLI)** : We can use the AWS Command Line Interface (AWS CLI) to access the Amazon RDS API interactively.

- **Programming with Amazon RDS** : The AWS SDKs handle low-level details such as authentication, retry logic, and error handling, so that we can focus on our application logic. AWS also provides libraries, sample code, tutorials, and other resources to help usget started more easily.

### 3.3.1 : Charging of RDS

- 750 hours of Amazon RDS Single-AZ db.t2.micro Instance usage running MySQL, MariaDB, PostgreSQL, Oracle BYOL or SQL Server (running SQL Server Express Edition) – enough hours to run a DB Instance continuously each month.

- Oracle BYOL db.t3.micro Single-AZ Instance usage is also included as part of the Amazon RDS free tier. If running both a db.t2.micro Single-AZ Instance and

a db.t3.micro Single-AZ Instance on Oracle BYOL, usage is aggregated across Instance classes.

### 3.3.2 : Automatic backups

Amazon RDS creates and saves automated backups of RDS DB instances. The first snapshot of a DB instance contains the data for the full DB instance and subsequent snapshots are incremental, maximum retention period is 35 days.

| | Self-Managed | Amazon RDS |
|---|---|---|
| Point-and-click deployment in minutes with pre-configured parameters | | ✔ |
| Scale compute resources with a few clicks or single API call | | ✔ |
| Automated backups and disaster recovery | | ✔ |
| Managed database snapshots for backup or database cloning | | ✔ |
| Access to hardware and complete environment control | ✔ | |
| Automatic software patching | | ✔ |
| Compatibility with existing applications and tools | ✔ | ✔ |
| Metrics on CPU, disk and memory utilization provided in a dashboard at no additional cost | | ✔ |

Fig. 5 : Key features of Amazon RDS

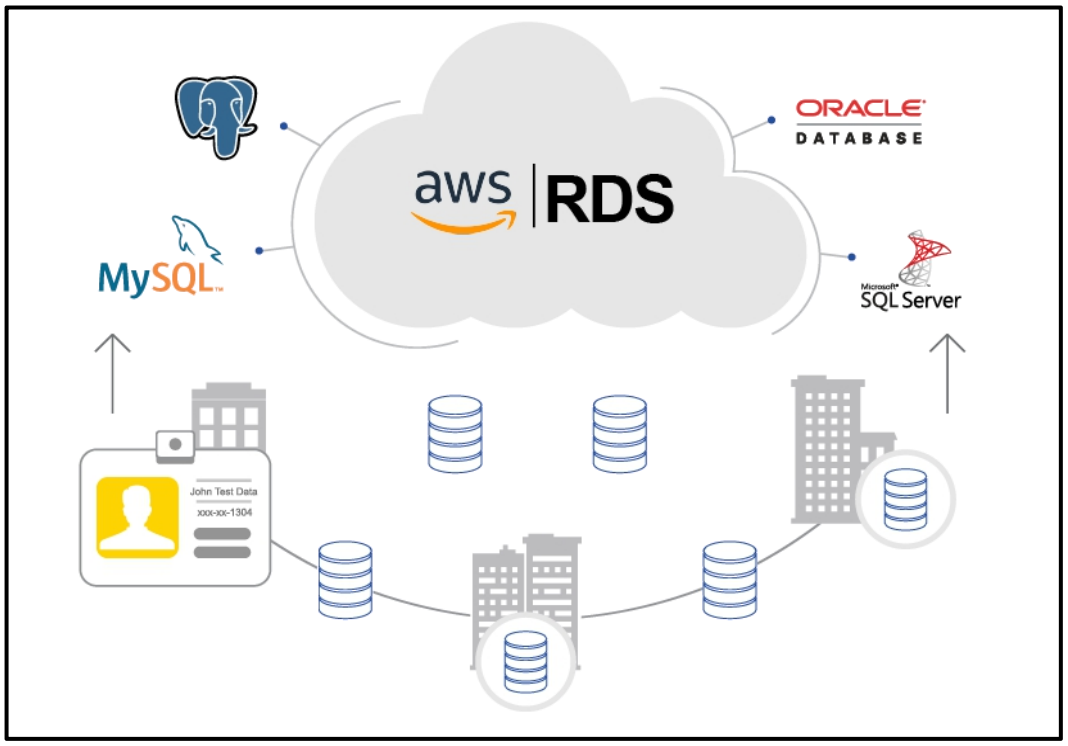



Fig. 6 : Intelligent Cloud Services Integration for Amazon RDS

# CHAPTER 4

# AMAZON SIMPLE STORAGE SERVICE (AMAZON S3)

## 4.1 : Introduction to S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is designed to make web-scale computing easier. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so we can organize our data and configure finely-tuned access controls to meet our specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.

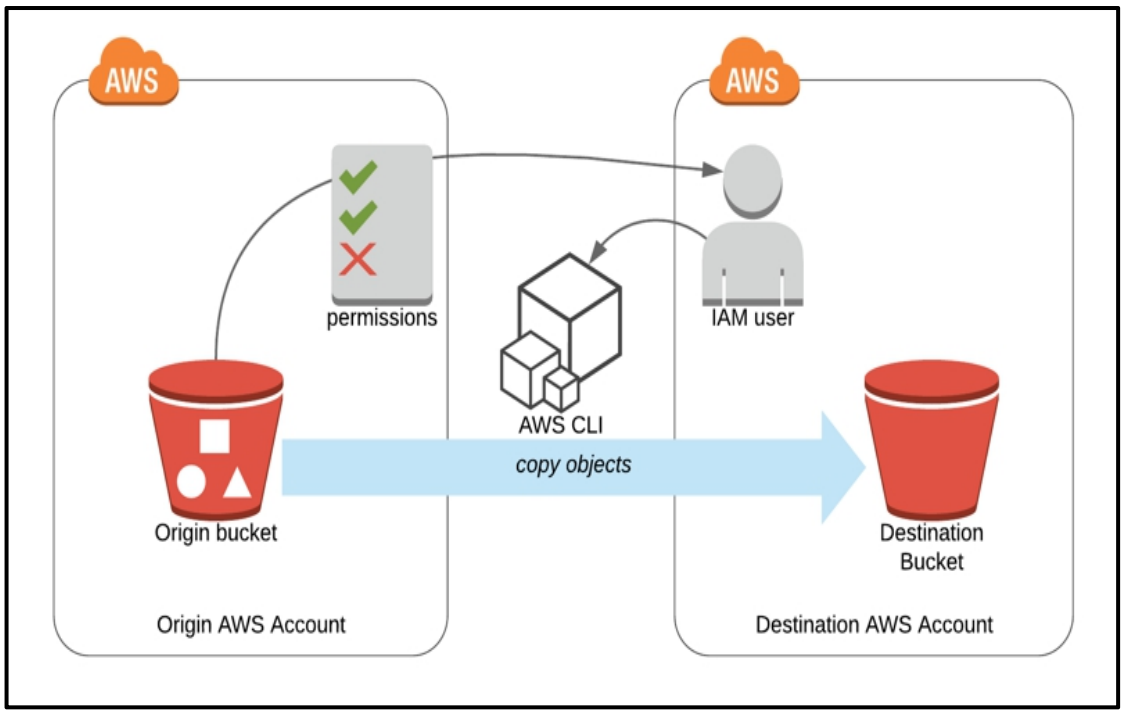



Fig. 7 : Copying S3 bucket objects to another AWS account

## 4.2 : Benefits of S3

- Industry-leading performance, scalability, availability, and durability
- Wide range of cost-effective storage classes
- Unmatched security, compliance, and audit capabilities
- Easily manage data and access controls
- Query-in-place and process on-request
- Most supported cloud storage service

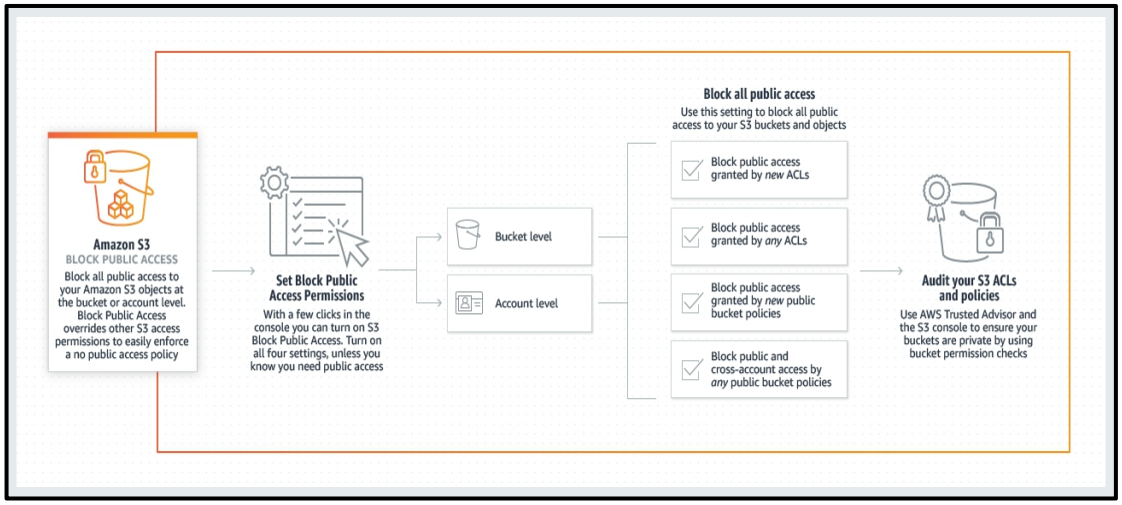



Fig. 8 : Blocking all public access to our S3 data

## 4.3 : Amazon S3 concepts

- **Buckets** :- A bucket is a container for objects stored in Amazon S3. Every object is contained in a bucket.
- **Objects** :- Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. The metadata is a set of name-value pairs that describe the object.
- **Keys** :- A key is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, key, and version ID uniquely identify each object.

- **Regions** :- We might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Objects stored in a Region never leave the Region unless we explicitly transfer them to another Region.
- **Amazon S3 data consistency model** :- Amazon S3 provides strong read-after-write consistency for PUTs and DELETEs of objects in our Amazon S3 bucket in all AWS Regions.

## 4.4 : Amazon S3 features

- **Storage classes** :- These include Amazon S3 STANDARD for general-purpose storage of frequently accessed data, Amazon S3 STANDARD_IA for long-lived, but less frequently accessed data, and S3 Glacier for long-term archive.
- **Bucket Policies** :- Bucket policies provide centralized access control to buckets and objects based on a variety of conditions.
- **AWS Identity and Access Management** :- We can use AWS Identity and Access Management (IAM) to manage access to our Amazon S3 resources.
- **Access control lists** :- We can control access to each of our buckets and objects using an access control list (ACL).
- **Versioning** :- We can use versioning to keep multiple versions of an object in the same bucket.
- **Operations** :- Following are the most common operations that we can run through the API - create a bucket; write an object; read an object; delete an object.

## 4.5 : Paying for Amazon S3

If we exceed that capacity, our service is shut off or we are charged high overage fees. If we do not exceed that capacity, we pay as though we used it all. Amazon S3 charges us only for what we actually use, with no hidden fees and no overage charges. This gives developers a variable-cost service that can grow with their business while enjoying the cost advantages of the AWS infrastructure. Before storing anything in Amazon S3, we must register with the service and provide a payment method that is charged at the end of each month.

# CHAPTER 5

# AMAZON VIRTUAL PRIVATE CLOUD (AMAZON VPC)

## 5.1 : Introduction to VPC

Amazon Virtual Private Cloud (Amazon VPC) is a service that lets us launch AWS resources in a logically isolated virtual network that we define. We have complete control over our virtual networking environment, including selection of our own IP address range, creation of subnets, and configuration of route tables and network gateways.

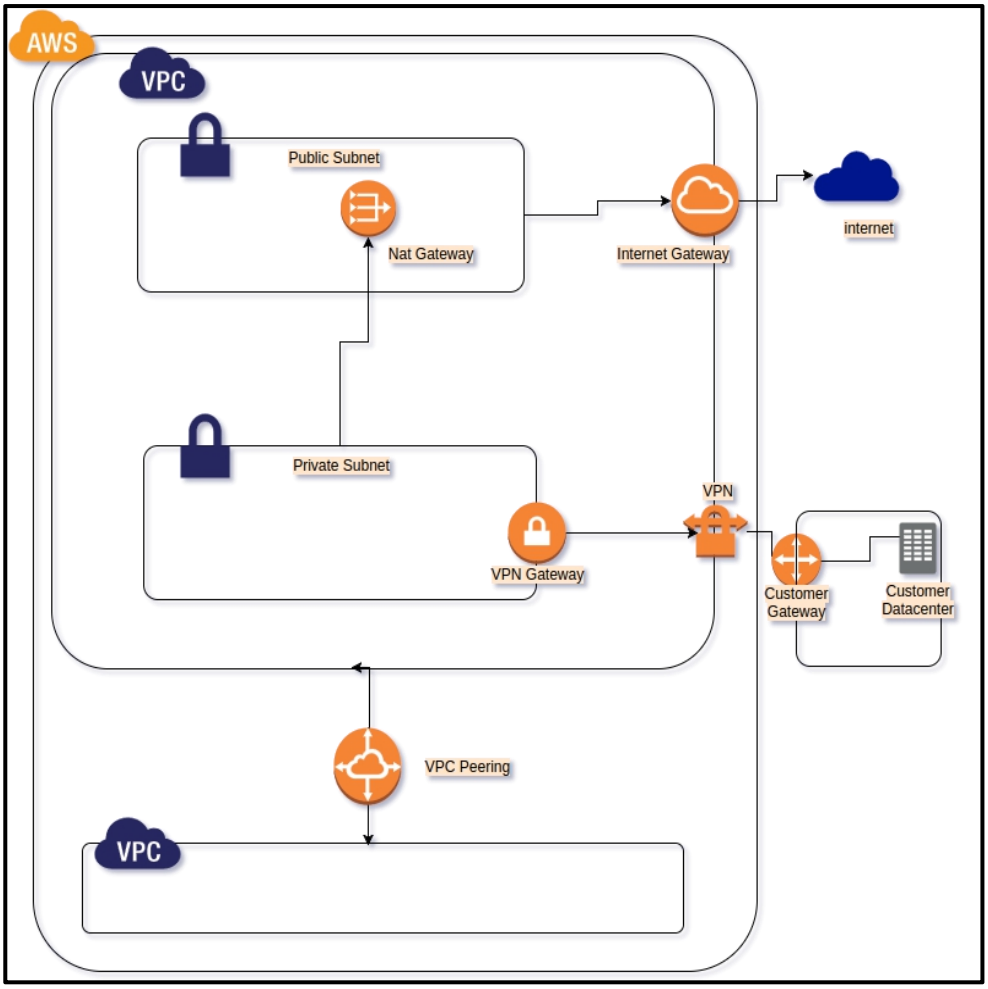



Fig. 9 : Amazon VPC diagram

As one of AWS's foundational services, Amazon VPC makes it easy to customize our VPC's network configuration. We can create a public-facing subnet for our web servers that have access to the internet. It also lets us place our backend systems, such as databases or application servers, in a private-facing subnet with no internet access. Amazon VPC lets us to use multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet.

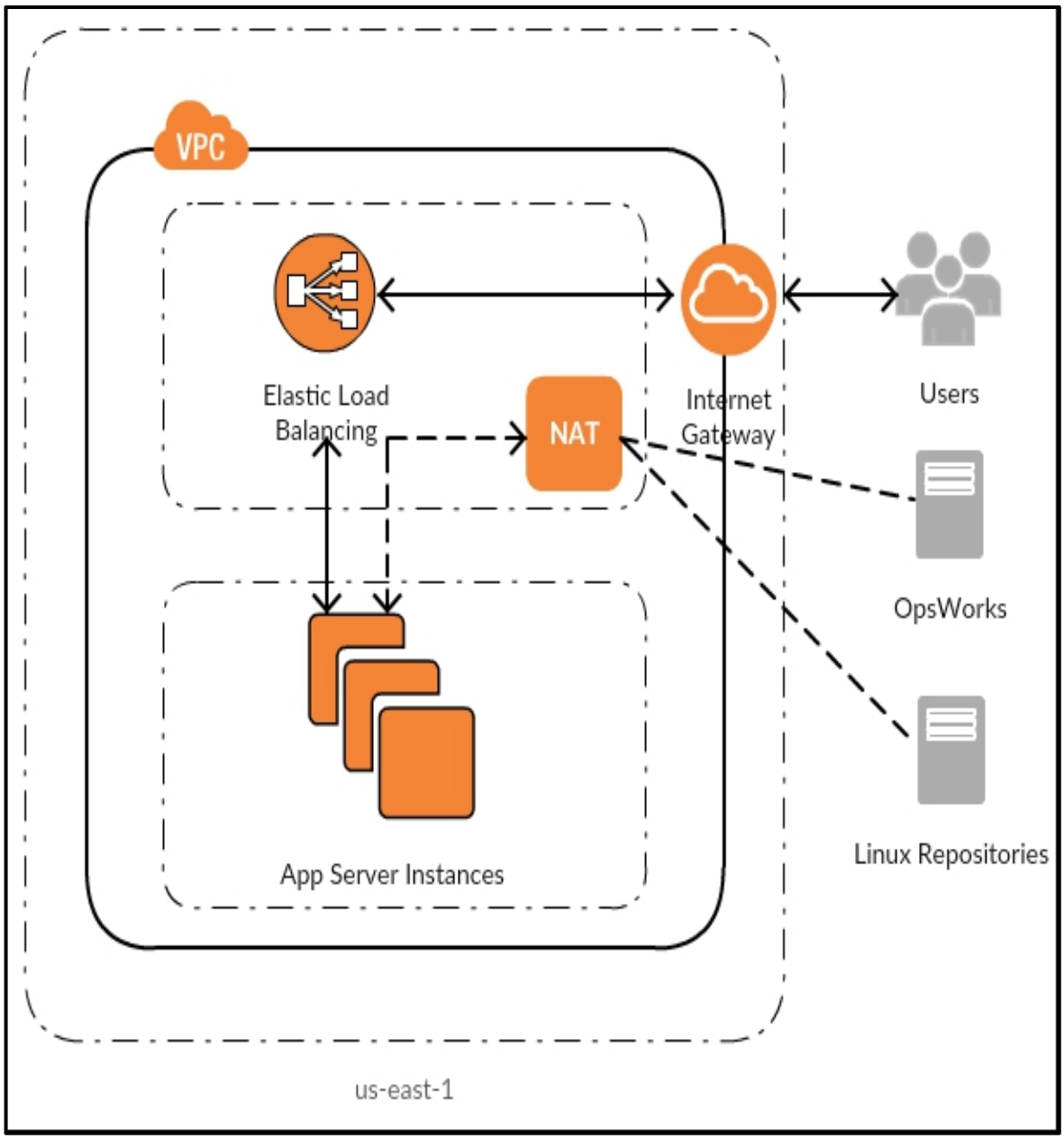



Fig. 10 : AWS architecture of VPC

## 5.2 : Amazon VPC concepts

- Virtual private cloud (VPC)
- Subnet
- Route table
- Internet gateway
- VPC endpoint
- CIDR block

## 5.3 : Benefits of Using Amazon Virtual Private Cloud (Amazon VPC)

- Secure and monitored network connections
- Simple set-up and use
- Customizable virtual network
- Host a simple, public-facing website
- Host multi-tier web applications
- Back up and recover our data after a disaster
- Extend our corporate network into the cloud
- Securely connect cloud applications to our data center

## 5.4 : Amazon VPC pricing

Amazon Virtual Private Cloud (Amazon VPC) lets us provision a logically isolated section of the Amazon Web Services (AWS) cloud where we can launch AWS resources in a virtual network that we define. We have complete control over our virtual networking environment, including selection of our own IP address range, creation of subnets, and configuration of route tables and network gateways. We can use both IPv4 and IPv6 in our VPC for secure and easy access to resources and applications. We can easily customize the network configuration for our Amazon Virtual Private Cloud. For example, we can create a public-facing subnet for our web servers that has access to the Internet, and place our backend systems such as databases or application servers in a private-facing subnet with no Internet access.

# CHAPTER 6

# AMAZON CLOUDFRONT

## 6.1 : Introduction to CloudFront

Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment**.** It is designed so we don't have to pay any up-front fees or commit to how much content we'll have. As with the other AWS services, we pay ' as you go' and pay only for what we use.

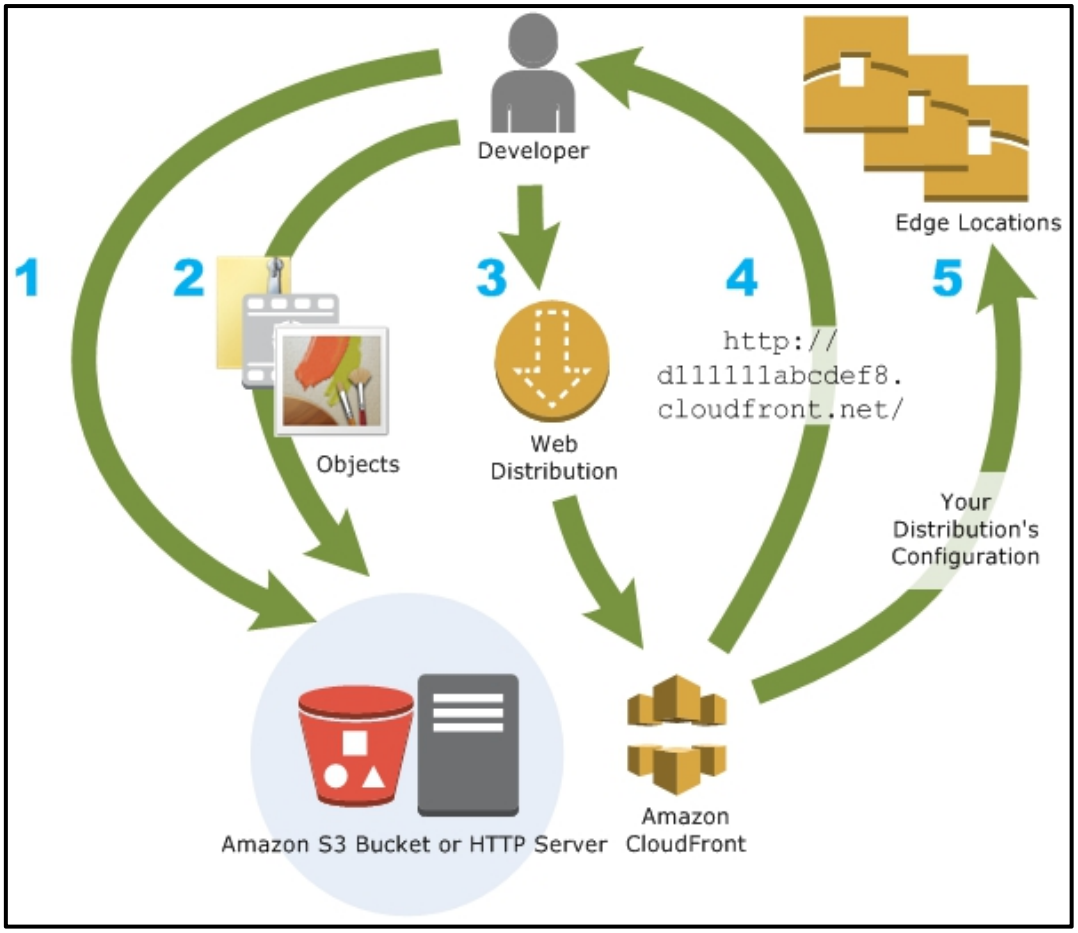



Fig. 11 : Amazon CloudFront

CloudFront offers the most advanced security capabilities, including field level encryption and HTTPS support, seamlessly integrated with AWS Shield, AWS Web Application Firewall and Amazon Route 53 to protect against multiple types of attacks including network and application layer DDoS attacks. These services co-reside at edge networking locations – globally scaled and connected via the AWS network backbone – providing a more secure, performant, and available experience for our users.

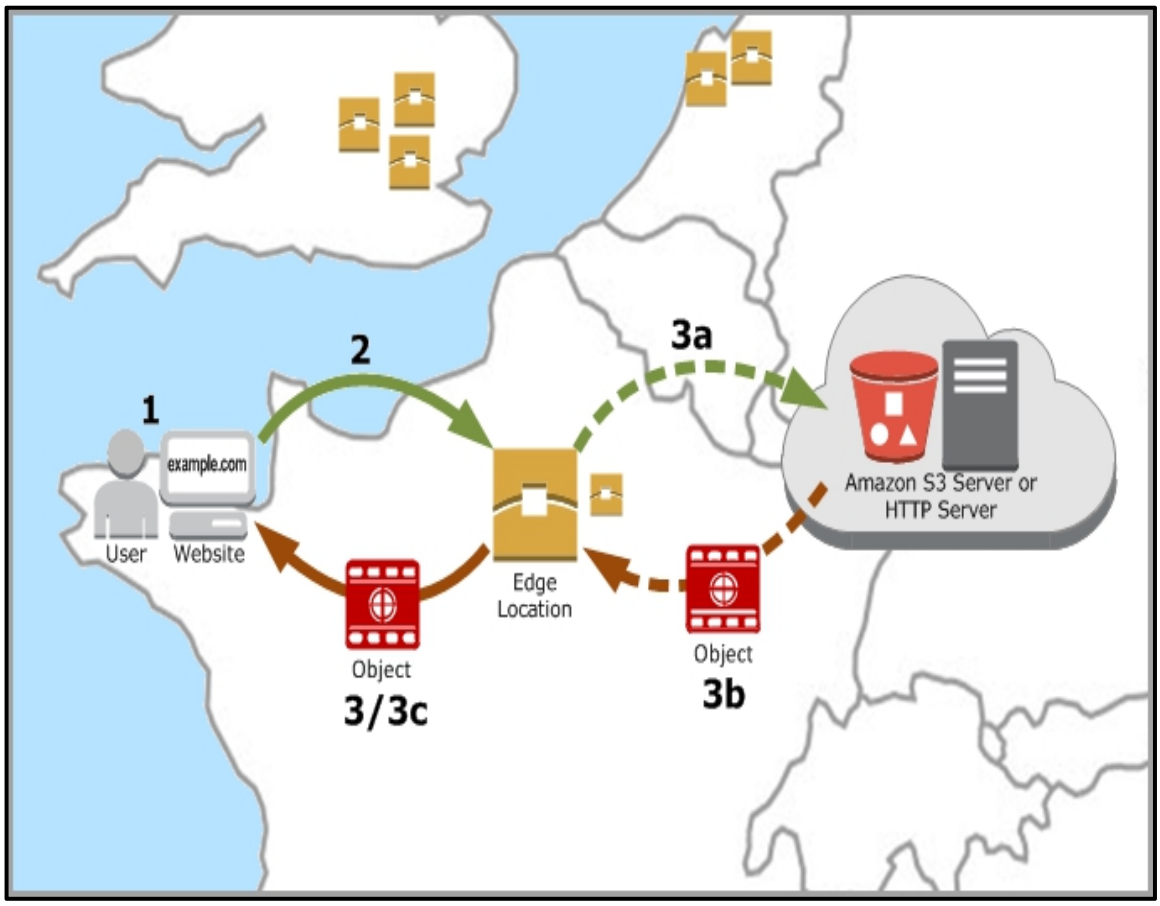



Fig. 12 : Setting up a Cloud Front CDN

## 6.2 : Benefits of CloudFront

- Cost-Effective
- Global Scaled Network for Fast Content Delivery
- Security at the Edge
- Highly Programmable and Secure Edge Computing
- Deep Integration with AWS

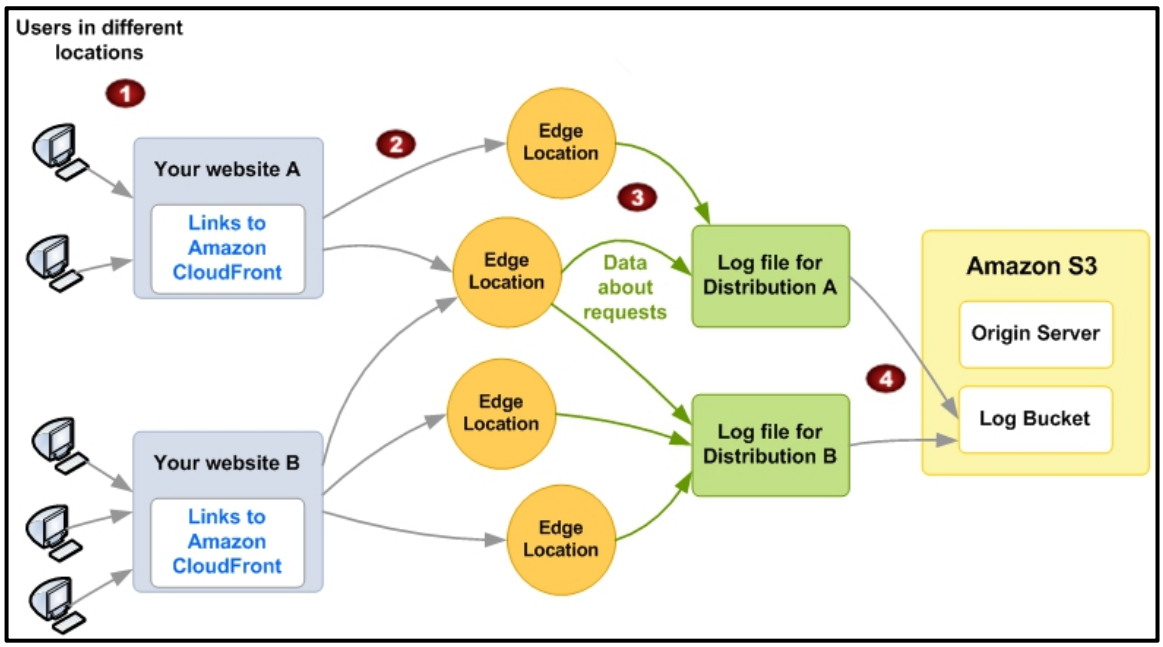



Fig. 13 : CloudFront logging information about requests for the objects

## 6.3 : CloudFront - Pricing

- Charge for storage in an Amazon S3 bucket
- Charge for serving objects from edge locations
- Charge for submitting data to our origin

### 6.3.1 : Caution for the following

- We also incur a surcharge for HTTPS requests, and an additional surcharge for requests that also have field-level encryption enabled or that use Origin Shield as an incremental caching layer.

- We do not incur any additional CloudFront charges when we use origin groups. We continue to pay the same request fees and data transfer rates as we do when we use CloudFront with any other AWS or non-AWS origin.

# CHAPTER 7

# AMAZON ELASTIC LOAD BALANCING (ELB)

## 7.1 : Introduction to Elastic Load Balancer

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, Lambda functions, and virtual appliances. It can handle the varying load of our application traffic in a single Availability Zone or across multiple Availability Zones. It helps to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed. Elastic Load Balancing offers four types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make our applications fault tolerant.

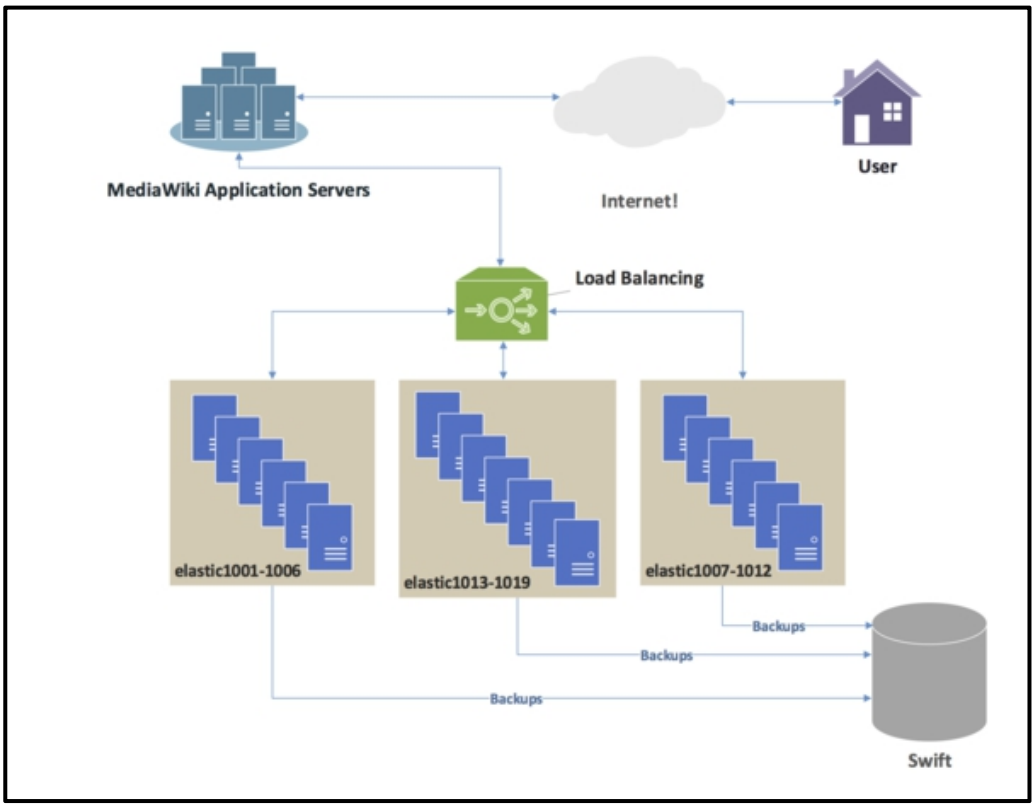



Fig. 14 : Diagram illustrating user requests to an Elastic search cluster being distributed by a load balancer.

## 7.2 : Elastic Load Balancing Types

- **Application Load Balancer** :- Routes and load balances at the application layer (HTTP/HTTPS), and supports path-based routing. An Application Load Balancer can route requests to ports on one or more registered targets, such as EC2 instances, in our virtual private cloud (VPC).

- **Network Load Balancer** :- Routes and load balances at the transport layer (TCP/UDP Layer-4), based on address information extracted from the TCP packet header, not from packet content. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.

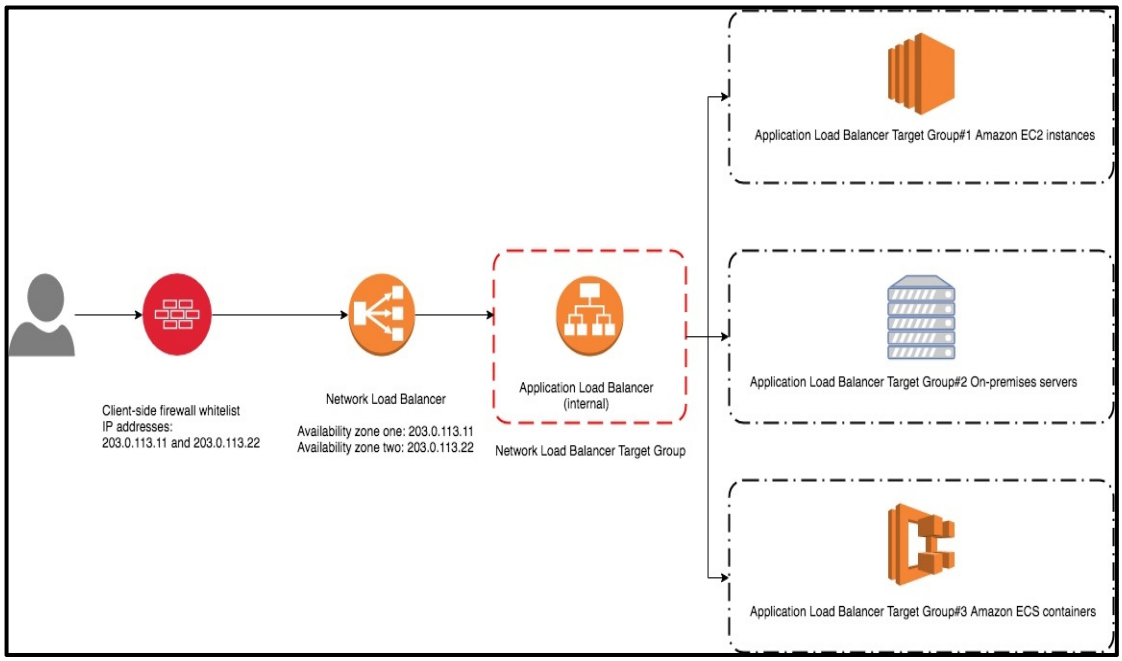



Fig. 15 : Using static IP addresses for Application Load Balancers

- **Gateway Load Balancer** :- Distributes traffic to a fleet of appliance instances. Provides scale, availability, and simplicity for third-party virtual appliances, such as firewalls, intrusion detection and prevention systems, and other appliances. Gateway Load Balancers work with virtual appliances that support the GENEVE protocol. Additional technical integration is required, so make sure to consult the user guide before choosing a Gateway Load Balancer.

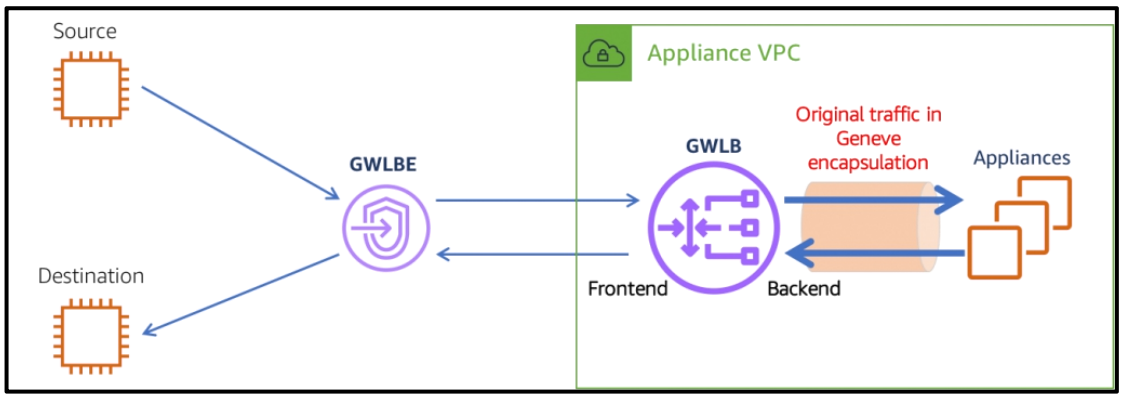



Fig. 16 : Integrating our custom logic or appliance with AWS Gateway Load Balancer

- **Classic Load Balancer** :- Routes and load balances either at the transport layer (TCP/SSL), or at the application layer (HTTP/HTTPS). A Classic Load Balancer supports either EC2-Classic or a VPC.

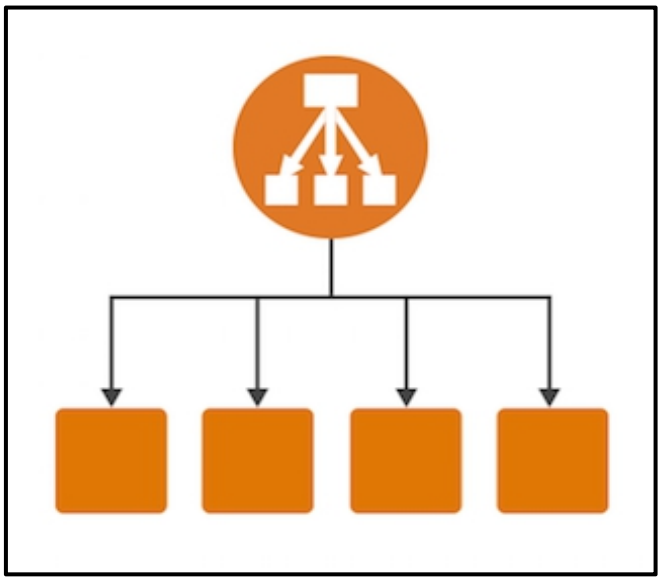


Fig. 17 : Load balancing at both the request level and the connection level by Classic Load Balancer

## 7.3 : Benefits of ELB

- Highly availability and elasticity
- Security
- Feature breadth
- Robust monitoring & visibility

# PROJECT DESCRIPTION

## *Introduction*

A website is a collection of web pages identified as a one domain name and published at least on one web server.

Any website can be roughly divided into two components :

1. **Front-end or Client side**- That is basically a User Interface through which the user can interact with the webpage.
2. **Back-end or Server-side** - Where the main focus is how will the site work and handle requests.

There are several frameworks and language to handle both the components. In this project we have used HTML, CSS, PHP, and MYSQL. Here, we have designed a simple quiz website where a user can attempt 10 simple questions and view the score. Firstly user has to login or create an account if he/she does not have it, Then a user dashboard were the users previous best score would appear, and options to attempt quiz or to logout will also be their.

## *Software used for Client-side*

**HTML** - It stands for Hyper-Text Markup Language which is an important part of any web based project because a browser can read HTML.
**CSS** - It stands for Cascading Style Sheets, which is used to style our web pages to make it look attractive.

## *Software used for Server-side*

**Apache Web Server** - It is a free, open-source web server software that accepts requests via HTTP, a network protocol created to distribute web pages.

Root folder - /var/www/html - The directory were we have to put our web pages .

HTTPD is the apache HTTP sever program which is a standalone daemon. This is a program we have installed as a server program.

**PHP** - It is a scripting language used to create dynamic page contents and decides what is to be done on when a user interact with the page.

**MySQL** - It is a database Software which is used to store data collected from the users.

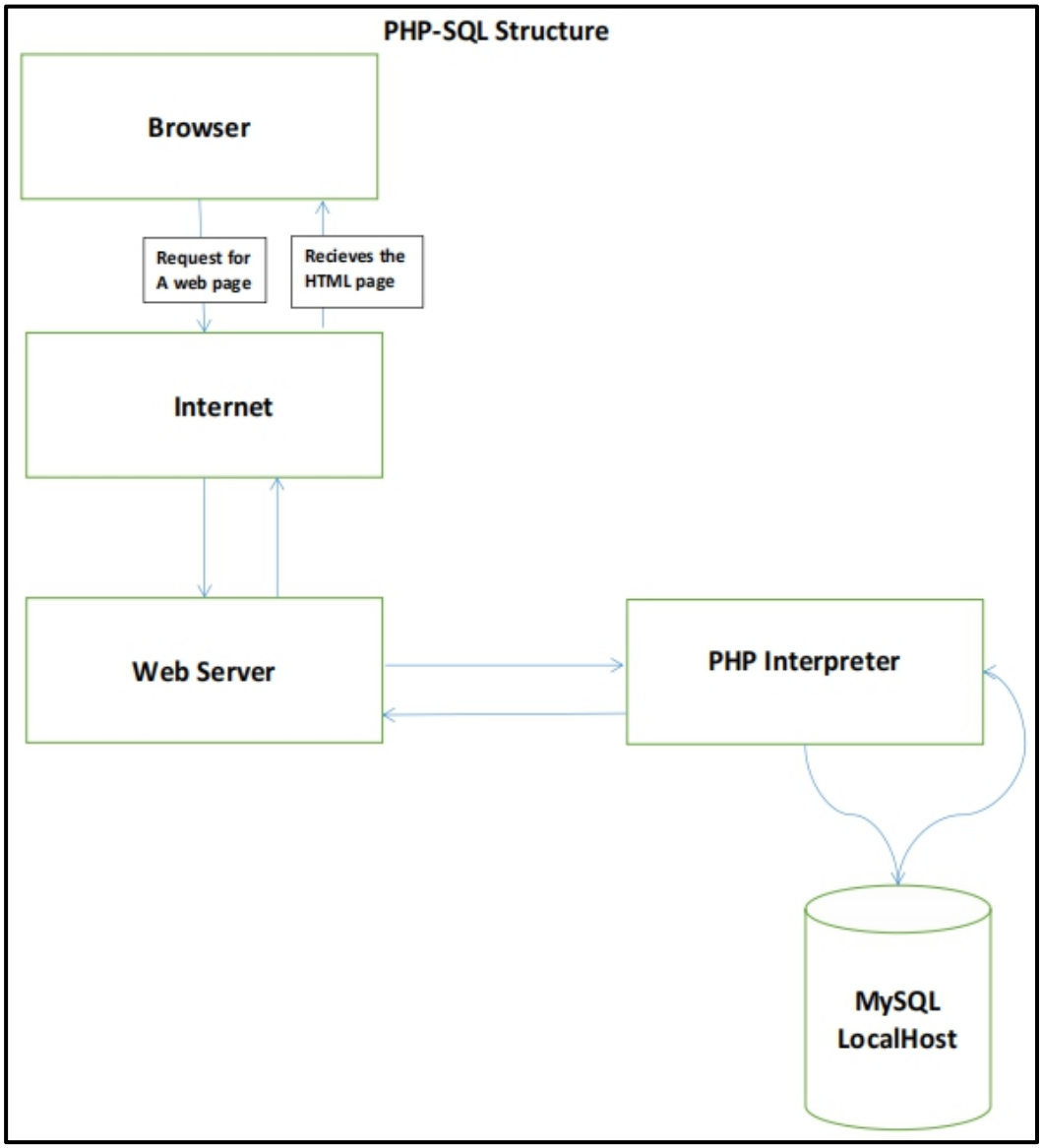



Fig. 18 : HTML-PHP implementation flow diagram

Now in order to make our website available for public we have to host our website on a server with a public IP so that other could access it. For that we have used cloud computing with Infrastructure as a Service and Platform as a Service.

## ***Cloud System***

*Platform* : Amazon Web Services (AWS)

*Resources* : Two EC2 Instances and one S3 Bucket.

## ***Elastic Compute Cloud***

*Configuration of EC2 Instances* :

- 1 vCPUs, Intel Xeon ® CPU E5-2676 v3 @ 2.40 GHZ
- Red Hat Linux Operating System
- 1 GB Memory
- 10 GB Elastic Block Storage (EBS)

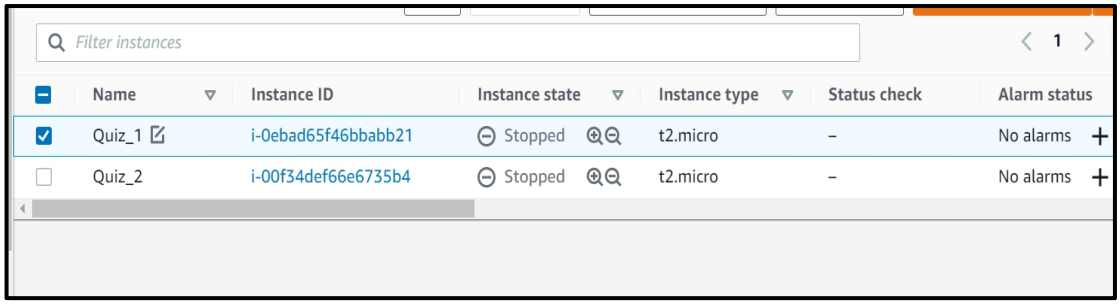



Fig. 19 : Creating instances for the dynamic quiz website

### *1. Generating a Key-Pair*

A key pair, consisting of a public key and a private key, is a set of security credentials that we use to prove our identity when connecting to an EC2 instance. Amazon EC2 stores the public key on our instance, and we store the private key. For Linux instances, the private key allows us to securely SSH into our instance.

### *2. Generating a SSH Key*

To generate a SSH key, we uses PuTTYgen, which is a key generator for creating SSH key.

## 3. *Connecting to the EC2 instance*

To connect to our server, we use PuTTY, which is a free, open-source Terminal emulator and supports several network protocol, including SSH,SCP and row socket connection.

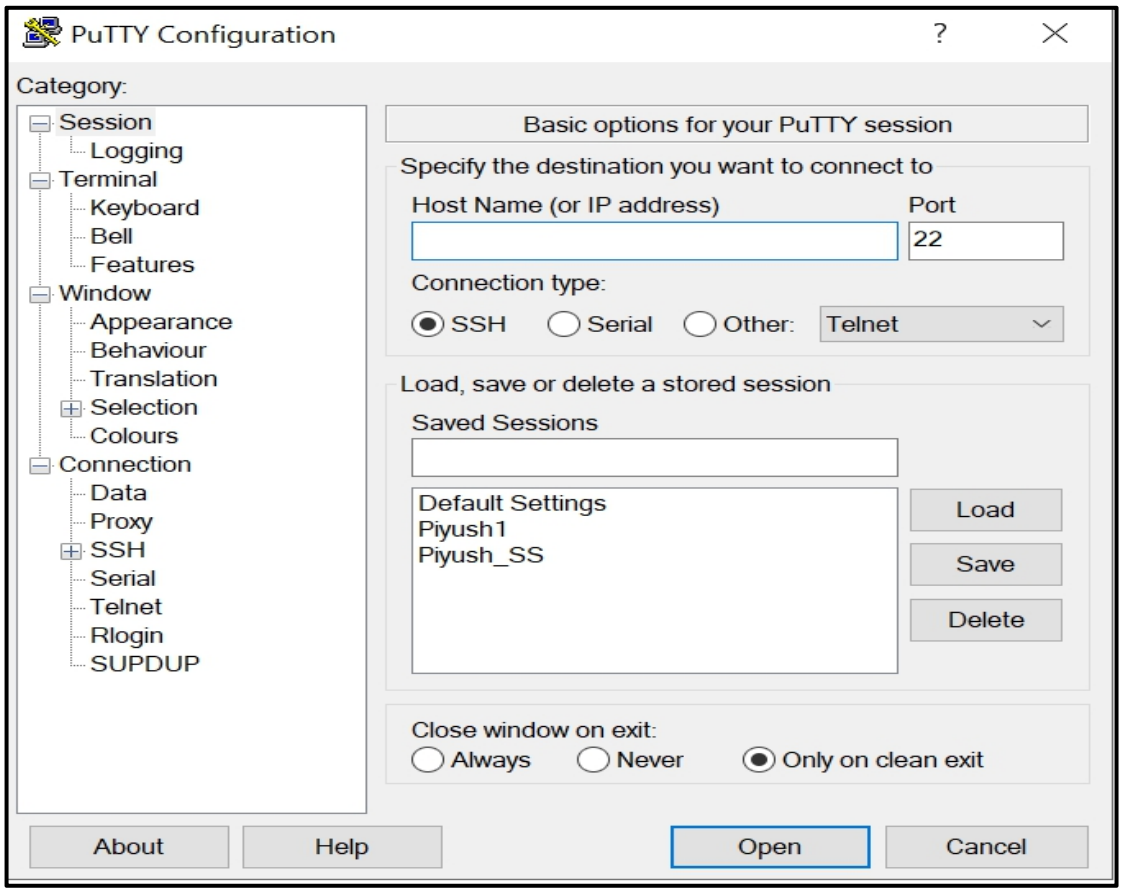



Fig. 20 : PuTTY configuration of the created instance

## 4. *Setting up the EC2 instance*

Switching to the Root User, Command used - *sudo su*

Installing the Apache HTTP Server

*yum install -y httpd*

Starting the Server

*systemctl start httpd*

Root directory of the web server - */var/www/html*

Installing PHP

*yum install -y php*

Configure PHP for Apache

Go to */etc/httpd/conf*

Open the httpd.conf and add *Include /etc/httpd/conf.d/*.conf*

Save the file

Restart the Apache Server

Installing MYSQL

*yum install -y mysql*

Installing MySQL Server

*yum install -y mysql-server*

Starting the MySql Server

*systemctl start mysqld*

Connecting PHP with MySQL

*yum install -y php-mysqlnd*

Use of *systemctl enable httpd* and *systemctl enable mysqld* - Now, if our EC2 is stopped by any running then we have to manually start the servers again to handle the HTTP request, But with this command it will automatically start the server whenever the EC2 is started.

## 5. *Security Groups*

Security groups enable us to control traffic to our instance, including which kind of traffic can reach our instance. Like as our instance is a web server then we have to allow all the IP to access the instance using HTTP.

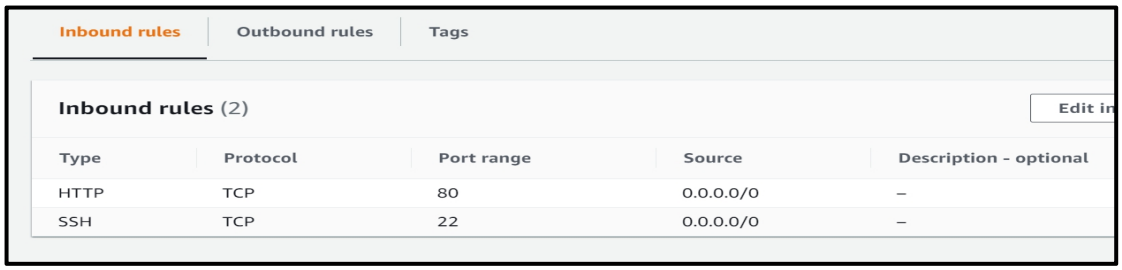



Fig. 21 : Configuring inbound rules of the created Security Group

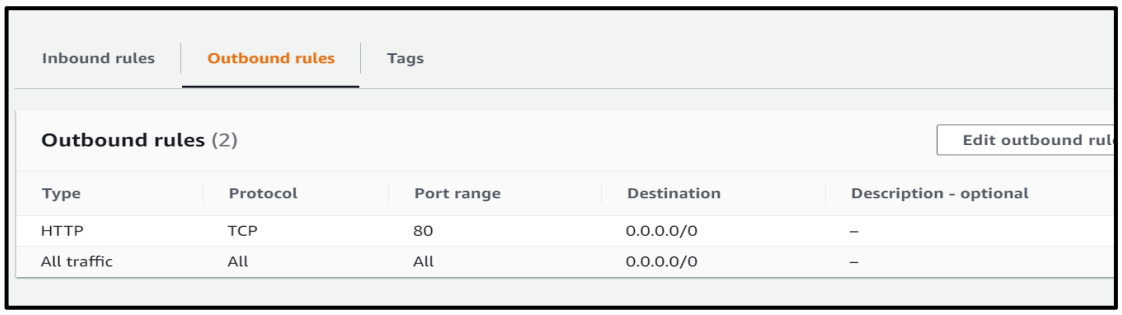



Fig. 22 : Configuring outbound rules of the created Security Group

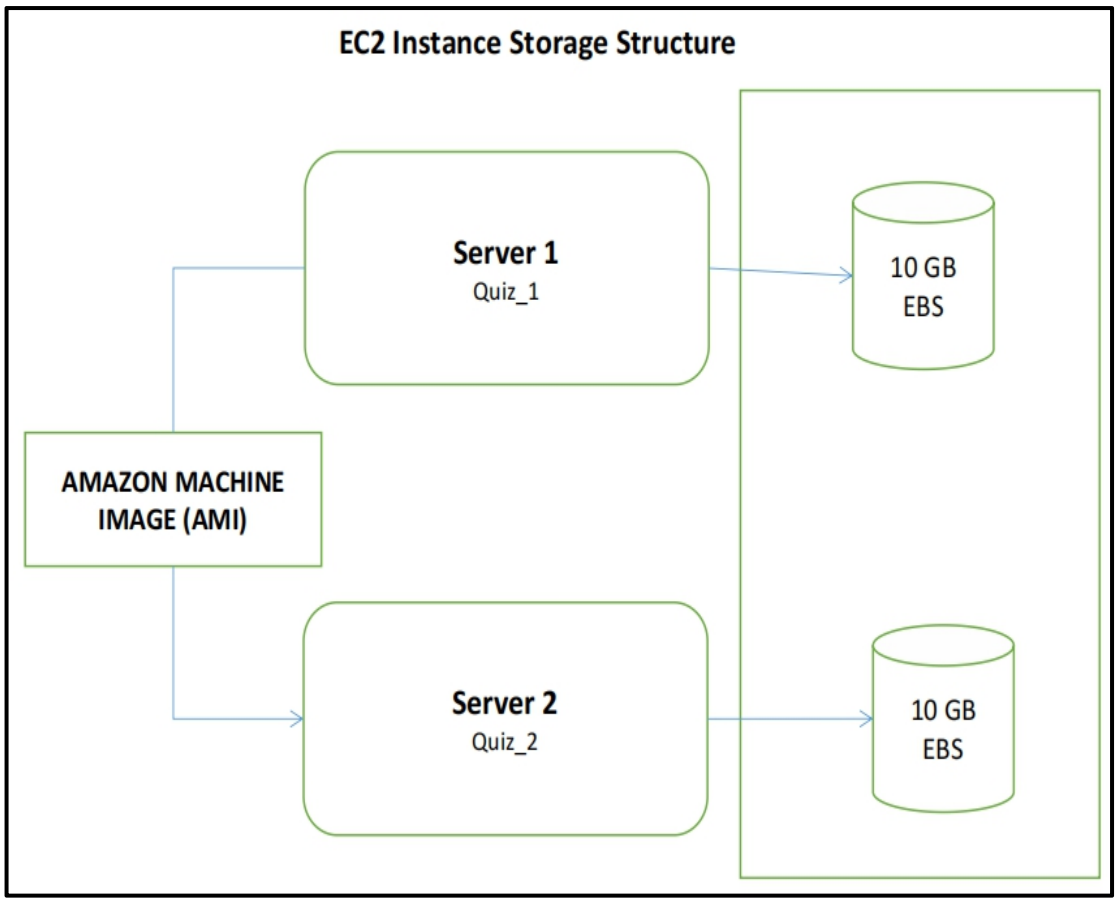



Fig. 23 : Structure of the EC2 RedHat Linux instance which have a default 10GB storage allotted to them

## ***Amazon Machine Image***

An AMI provides information needed to launch an instance and can be used to launch multiple instances using one AMI when we need multiple instances with same configuration. Here, we configured one instance and then used its image to make another one.

## ***Server Architecture***

While designing web servers, our main focus is how to manage traffic and Failover. In case if they is any fault in one of the instance or they is excess of traffic, then we need a system to manage it efficiently.

In order to do that, we have used Classic Load Balancer which helps in managing traffic. Also to deliver our content securely, we have used a Content Delivery Network called CloudFront.

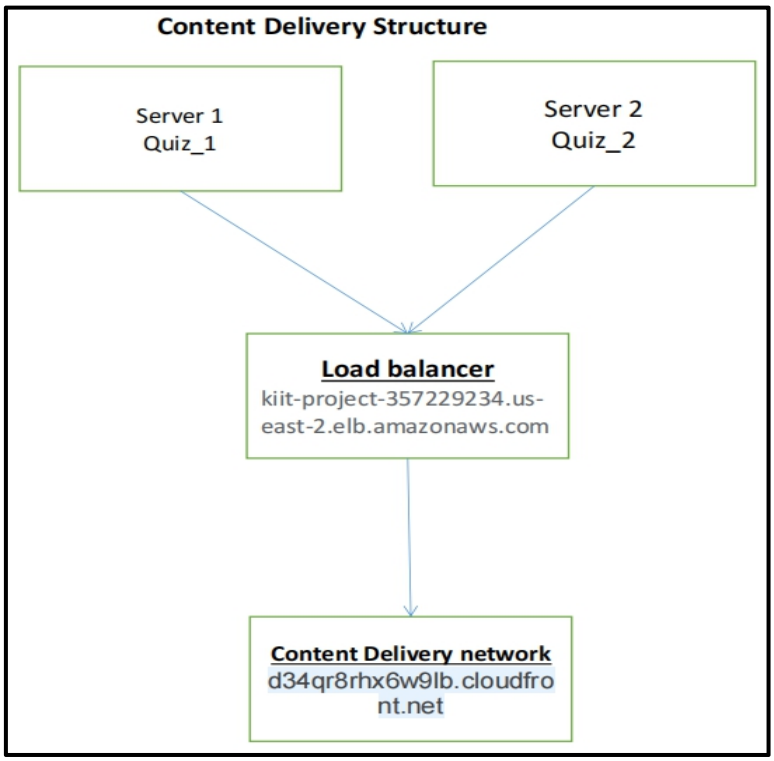



Fig. 24 : Flow Diagram of ELB and CloudFront for traffic management and shielding IPv4 Address.


Fig. 25 : Created CloudFront distributions in work


Fig. 26 : Created Elastic Load Balancer in work

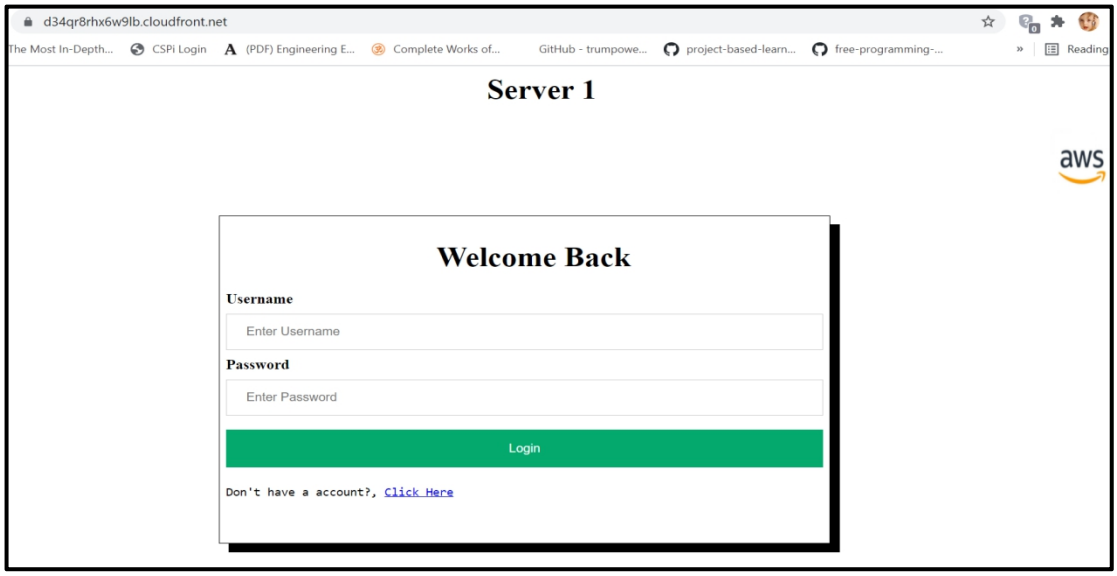



Fig. 27 : Getting the server1 html page using the DNS name of Load Balancer protected by CloudFront

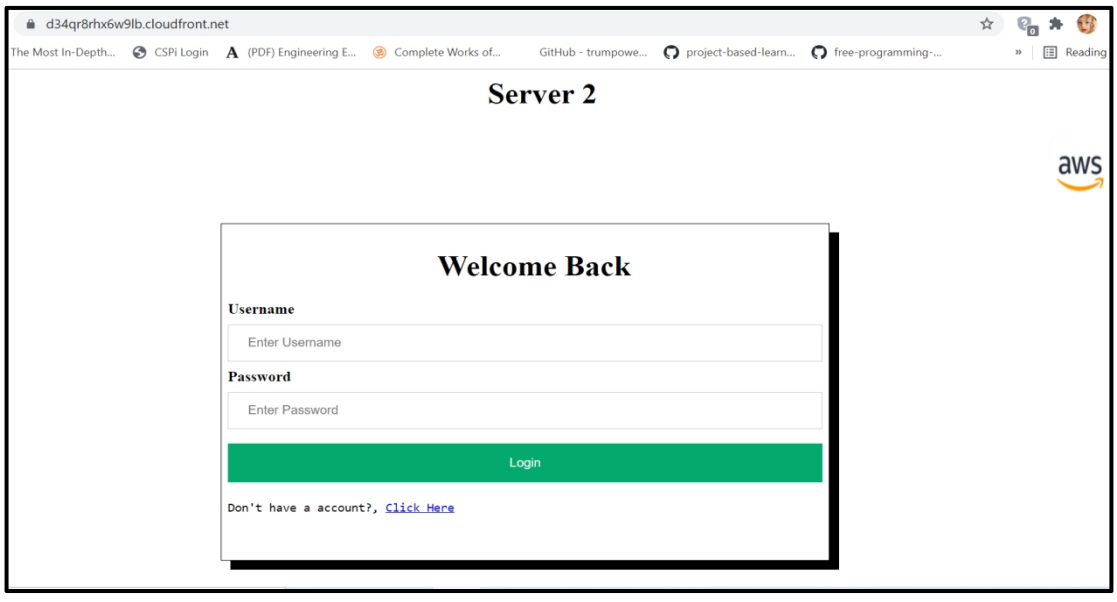
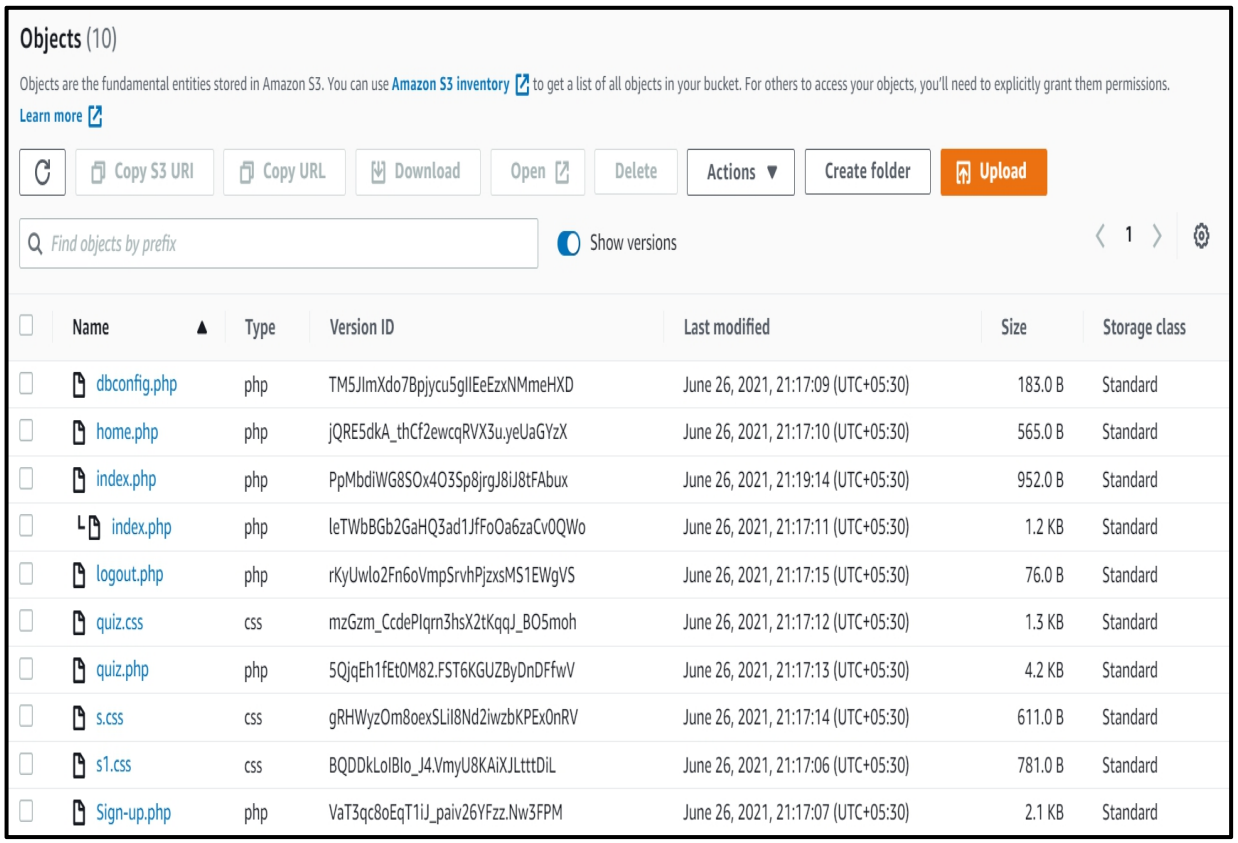



Fig. 28 : Elastic Load Balancer is working properly as server2 takes over in case of a failover

## ***Amazon S3***

Amazon S3 is a storage which enabled us to store the files and details of this projects at one secured place. Also the Bucket versioning feature of S3 helped us in making change to our pages and if required to go back to the previous one as well.



**Objects** (10)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.

Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix | Show versions | 1

| Name | Type | Version ID | Last modified | Size | Storage class |
|---|---|---|---|---|---|
| dbconfig.php | php | TM5JImXdo7Bpjycu5gIIEeEzxNMmeHXD | June 26, 2021, 21:17:09 (UTC+05:30) | 183.0 B | Standard |
| home.php | php | jQRE5dkA_thCf2ewcqRVX3u.yeUaGYzX | June 26, 2021, 21:17:10 (UTC+05:30) | 565.0 B | Standard |
| index.php | php | PpMbdiWG8SOx4O3Sp8jrgJ8iJ8tFAbux | June 26, 2021, 21:19:14 (UTC+05:30) | 952.0 B | Standard |
| index.php | php | leTWbBGb2GaHQ3ad1JfFoOa6zaCv0QWo | June 26, 2021, 21:17:11 (UTC+05:30) | 1.2 KB | Standard |
| logout.php | php | rKyUwlo2Fn6oVmpSrvhPjzxsMS1EWgVS | June 26, 2021, 21:17:15 (UTC+05:30) | 76.0 B | Standard |
| quiz.css | css | mzGzm_CcdePIqrn3hsX2tKqqJ_BO5moh | June 26, 2021, 21:17:12 (UTC+05:30) | 1.3 KB | Standard |
| quiz.php | php | 5QjqEh1fEt0M82.FST6KGUZByDnDFfwV | June 26, 2021, 21:17:13 (UTC+05:30) | 4.2 KB | Standard |
| s.css | css | gRHWyzOm8oexSLiI8Nd2iwzbKPEx0nRV | June 26, 2021, 21:17:14 (UTC+05:30) | 611.0 B | Standard |
| s1.css | css | BQDDkLoIBlo_J4.VmyU8KAiXJLtttDiL | June 26, 2021, 21:17:06 (UTC+05:30) | 781.0 B | Standard |
| Sign-up.php | php | VaT3qc8oEqT1iJ_paiv26YFzz.Nw3FPM | June 26, 2021, 21:17:07 (UTC+05:30) | 2.1 KB | Standard |

Fig. 29 : Created Amazon S3 bucket to store objects

## ***Elastic IP Addresses***

An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is allocated to our AWS account, and is ours until we release it. By using an Elastic IP address, we can mask the failure of an instance or software by rapidly remapping the address to another instance in our account. Alternatively, we can specify the Elastic IP address in a DNS record for our domain, so that we domain points to our instance.

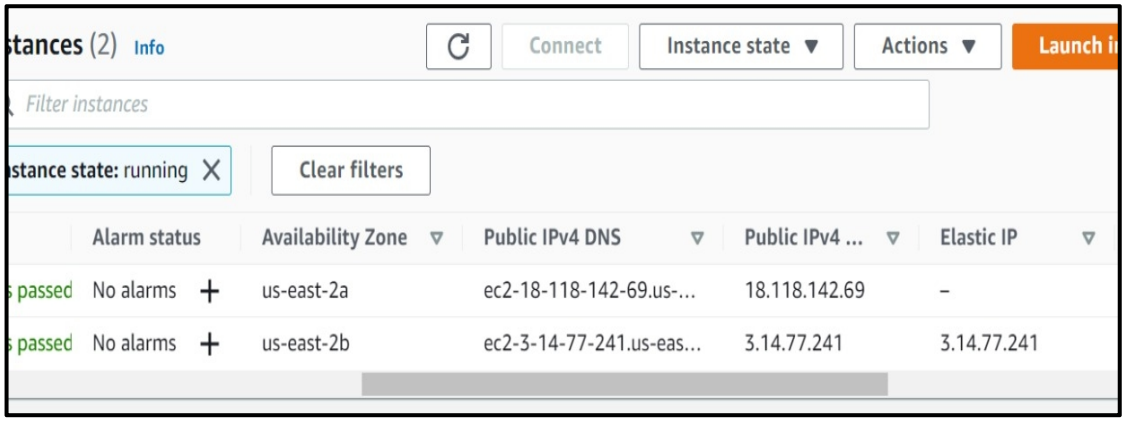



Fig. 30 : Allocated the Elastic IP Address for our instance

## ***AWS Identity and Access Management (IAM)***

AWS Identity and Access Management (IAM) is a web service that helps us securely control access to AWS resources. We use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

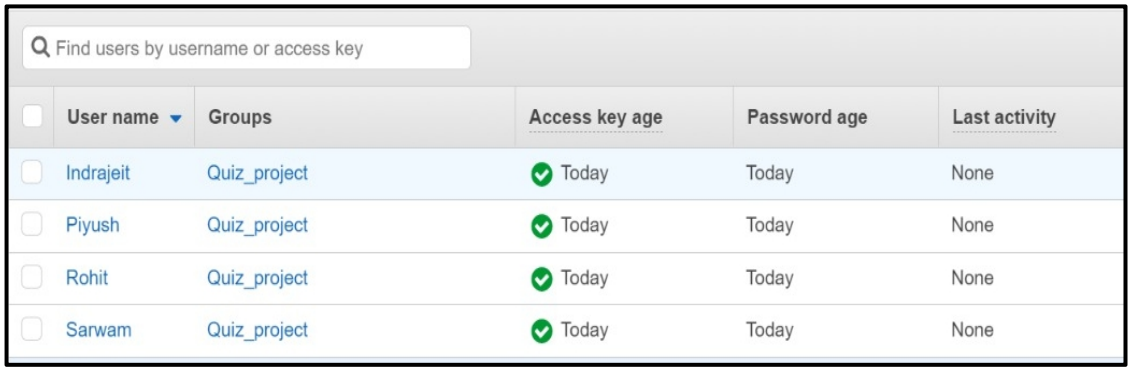



Fig. 31 : Created IAM roles for all the group members

## ***AWS Backup & Restore***

AWS Backup enables us to centralize and automate data protection across AWS services. AWS Backup offers a cost-effective, fully managed, policy-based service that further simplifies data protection at scale. AWS Backup also helps us support our regulatory compliance or business policies for data protection. Together with AWS Organizations, AWS Backup enables us to centrally deploy data protection policies to configure, manage, and govern our backup activity across our organization's AWS

accounts and resources, including Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Elastic Block Store (Amazon EBS) volumes, Amazon Relational Database Service (RDS) databases (including Amazon Aurora clusters), Amazon DynamoDB tables, Amazon Elastic File System (EFS), Amazon FSx for Lustre, Amazon FSx for Windows File Server, and AWS Storage Gateway volumes.

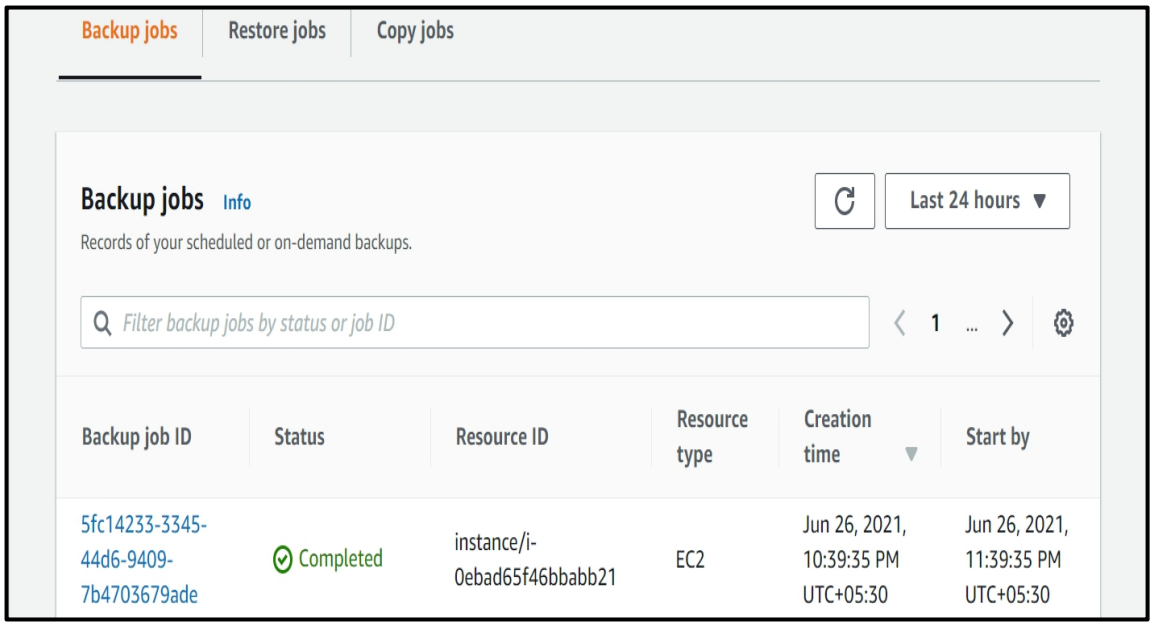



Fig. 32 : Creation of a Backup of our EC2 instance

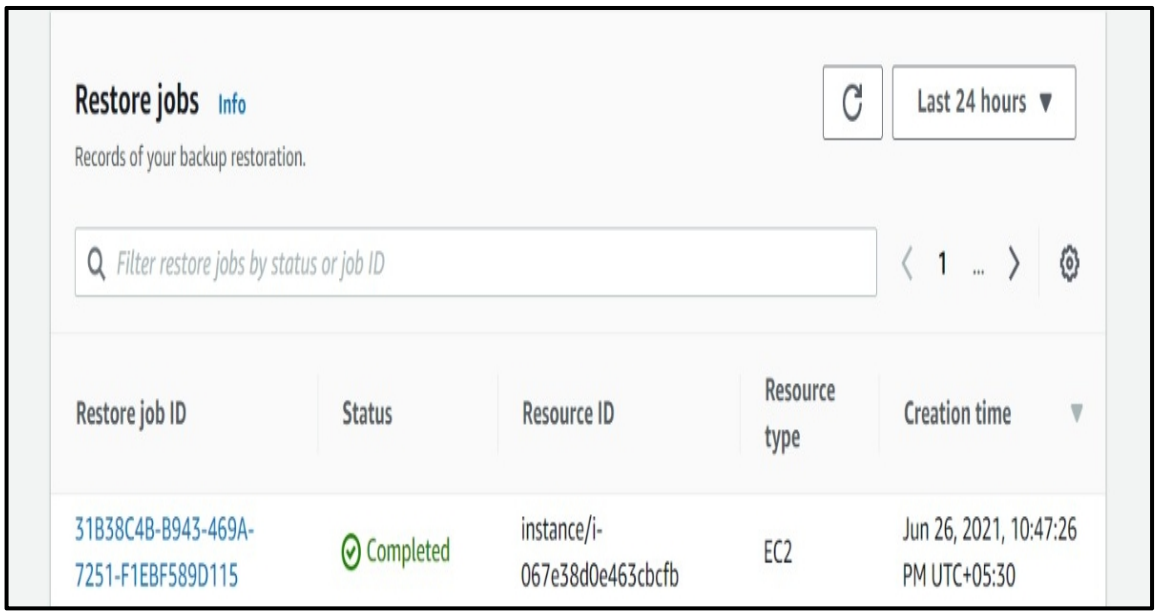



Fig. 33 : Testing the Restoration of the backed-up EC2 instance

## ***VPC Connection***

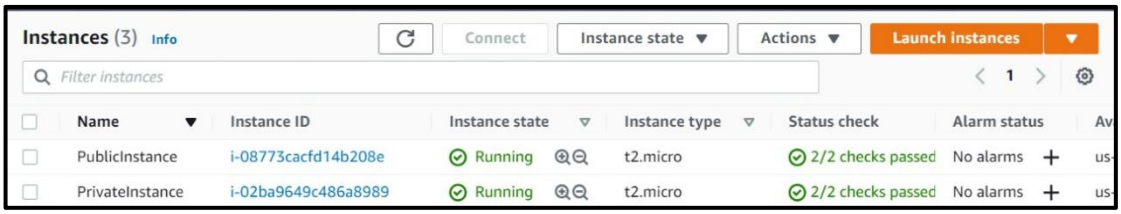



Fig. 34 : The two instances which are attached to the VPC

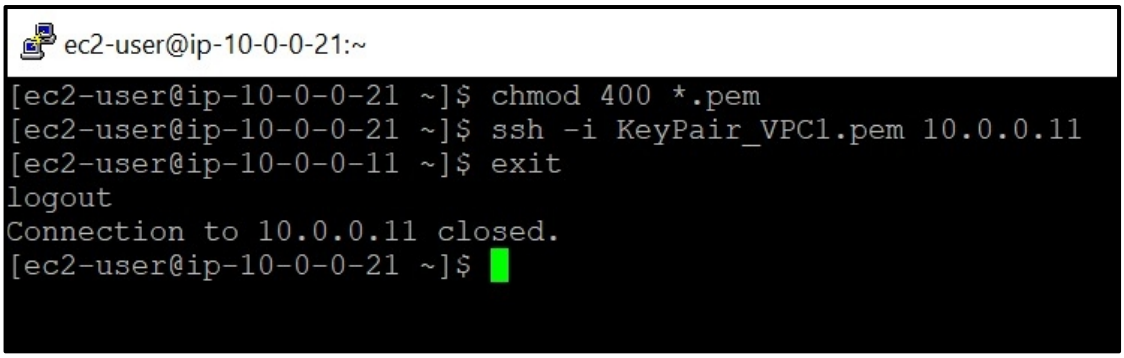



Fig. 35 : Successfully pinged the private instance inside public instance by VPC

## DYNAMIC WEBSITE FOR THE QUIZ

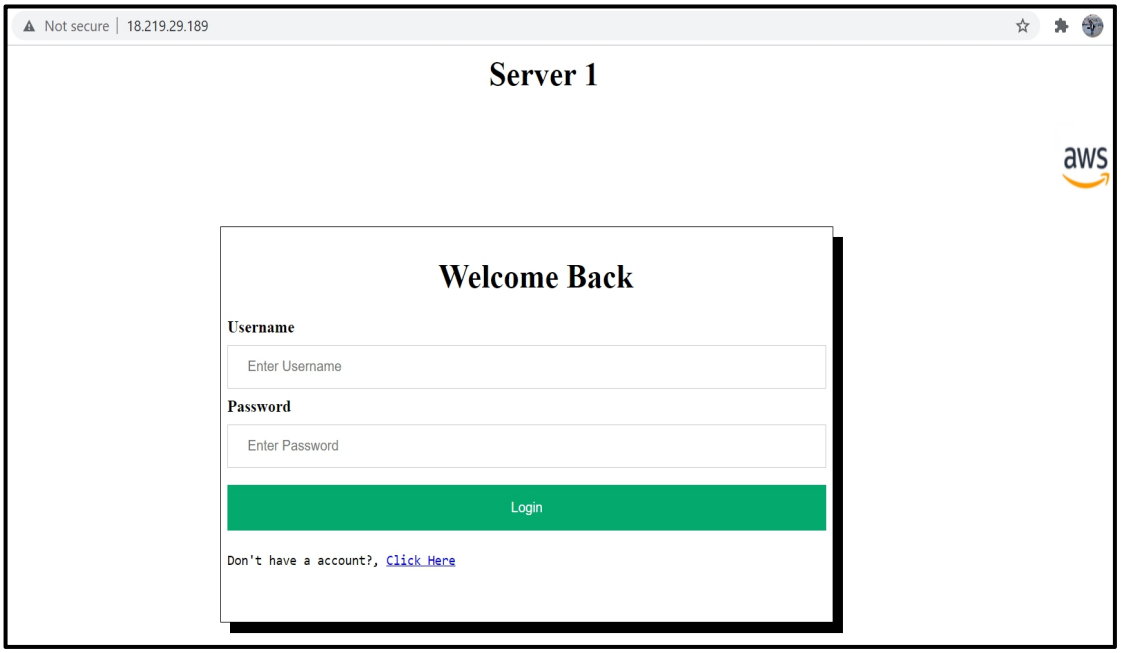



Fig. 36 : First page displayed by the browser for logging into the Quiz for existing users

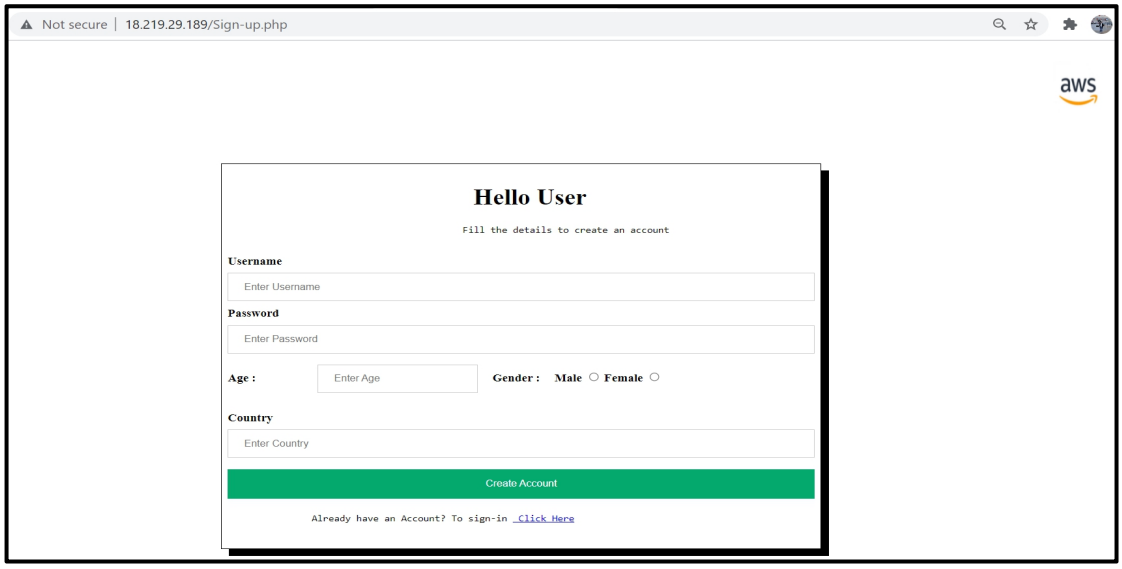



Fig. 37 : This is the page for sign-up for new users

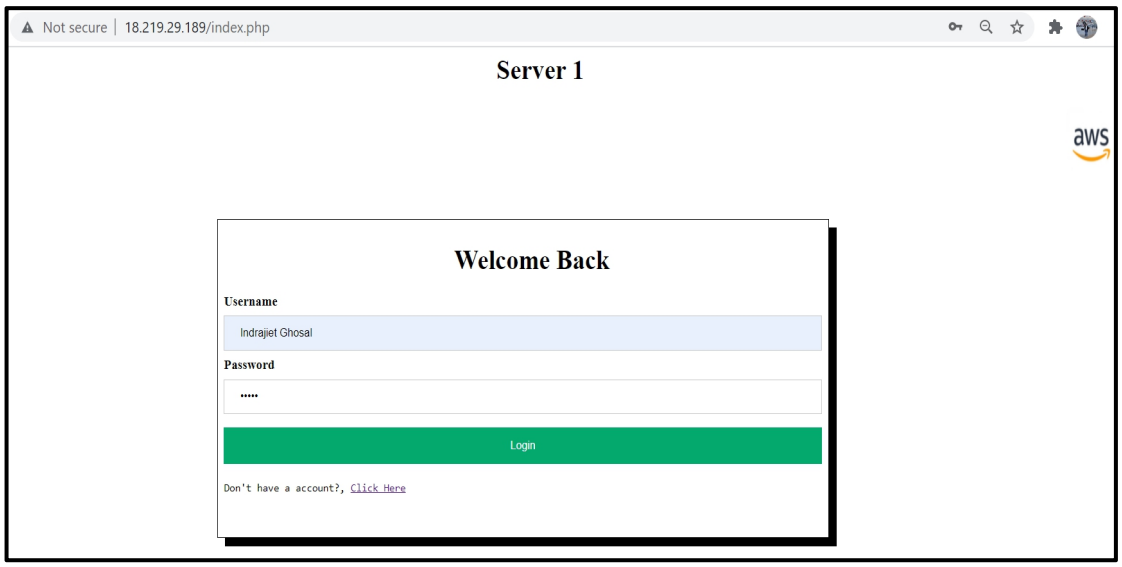



Fig. 38 : After sign-up, sign-in to give the Quiz

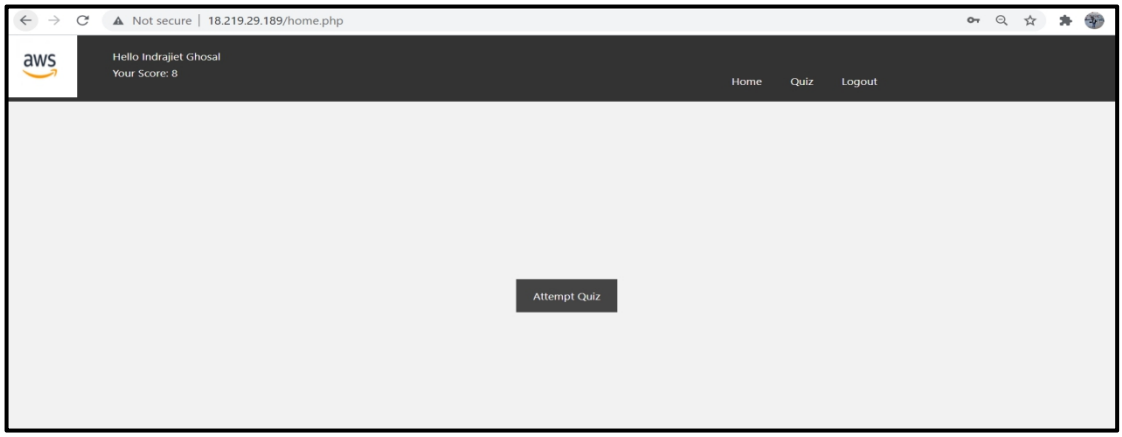



Fig. 39 : This is the website's Homepage where the user can attempt the Quiz

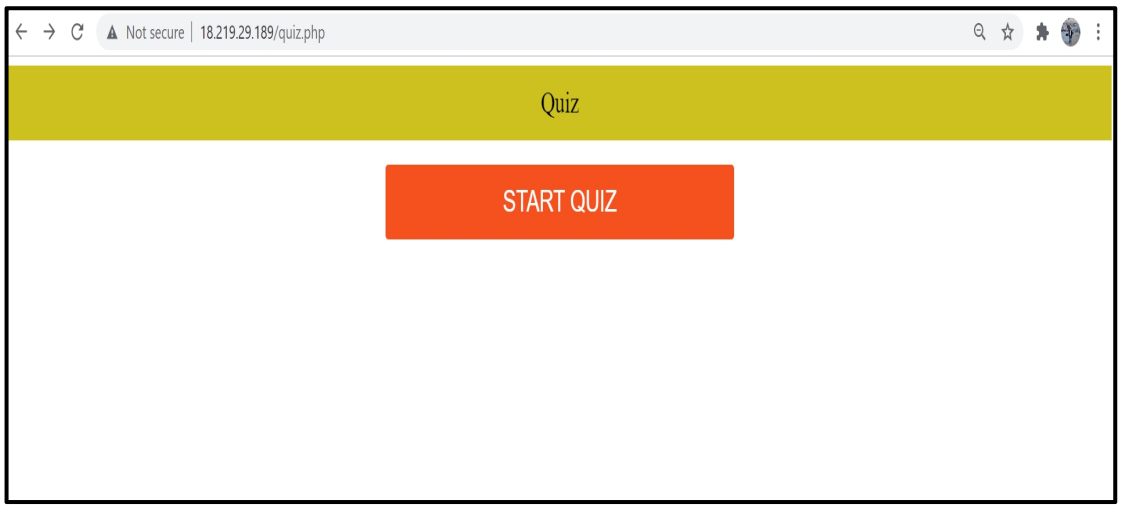



Fig. 40 : Start the Quiz here

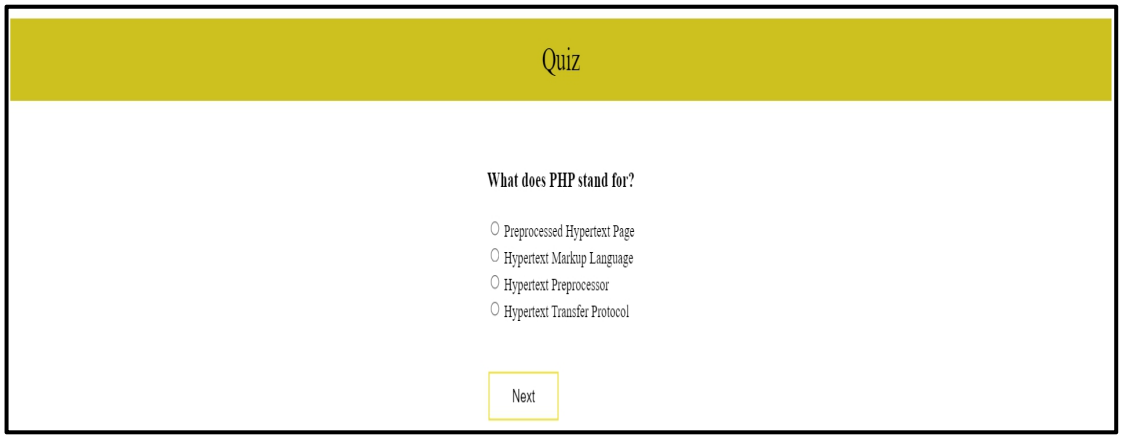
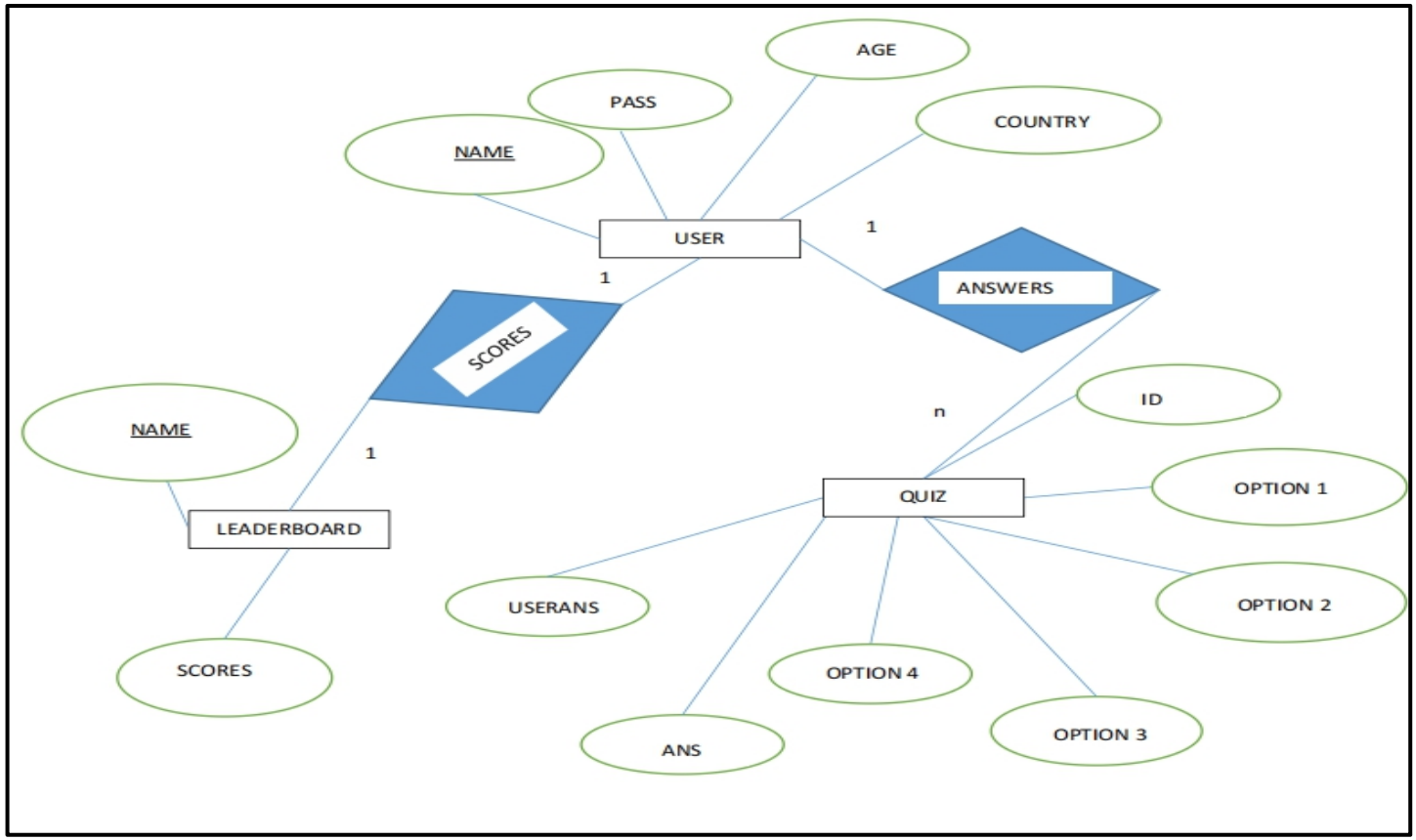



Fig. 41 : Select the correct answer and answer all 10 MCQ Quiz questions

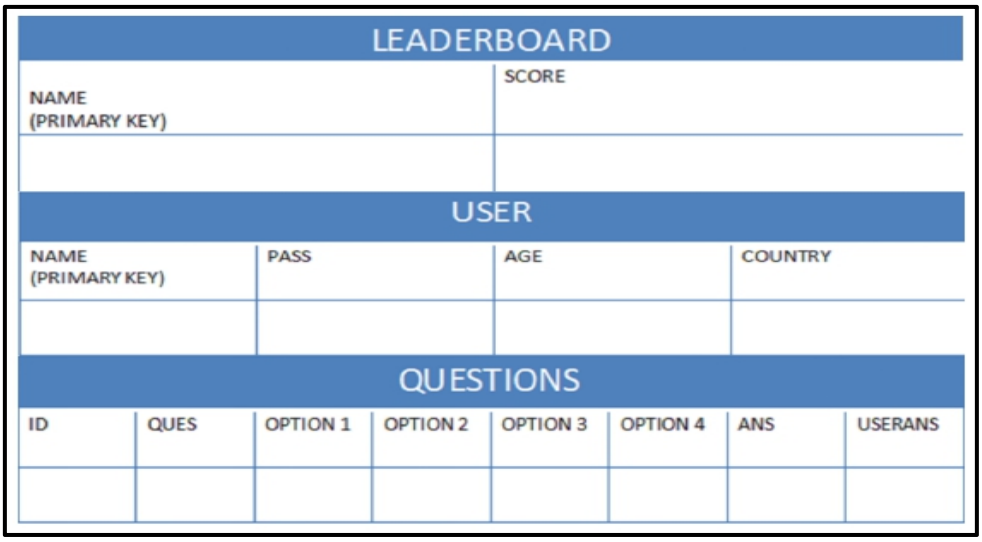

Fig. 42 : After finishing the Quiz, the result is displayed

***Database Structure*** **-** Database is used to store the user information such as username,password, score and questions.

## ENTITY-RELATIONSHIP (ER) DIAGRAM



## TABLE STRUCTURE



| LEADERBOARD | |
|---|---|
| NAME (PRIMARY KEY) | SCORE |
| | |

| USER | | | |
|---|---|---|---|
| NAME (PRIMARY KEY) | PASS | AGE | COUNTRY |
| | | | |

| QUESTIONS | | | | | | | |
|---|---|---|---|---|---|---|---|
| ID | QUES | OPTION 1 | OPTION 2 | OPTION 3 | OPTION 4 | ANS | USERANS |
| | | | | | | | |

## TABLE DESCRIPTION

```
Database changed
mysql> desc user;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | NO   | PRI | NULL    |       |
| pass    | varchar(20) | YES  |     | NULL    |       |
| age     | int         | YES  |     | NULL    |       |
| country | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc leaderboard;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(20) | NO   | PRI | NULL    |       |
| score | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc quiz;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int          | NO   |     | NULL    |       |
| que      | text         | NO   |     | NULL    |       |
| option 1 | varchar(222) | NO   |     | NULL    |       |
| option 2 | varchar(222) | NO   |     | NULL    |       |
| option 3 | varchar(222) | NO   |     | NULL    |       |
| option 4 | varchar(222) | NO   |     | NULL    |       |
| ans      | varchar(222) | NO   |     | NULL    |       |
| userans  | text         | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

# CODE OF PHP

## Index.php

```
<?php
session_start();
?>
<head>
<title>Login| Quiz </title>
<link rel="stylesheet" href="s.css">
</head>
<body>
<img src="https://www.jdrf.org/wp-content/uploads/2020/12/AWS-logo-2.jpg"
width="100" height="100">
<form method="post">
<div>
<h1> Welcome Back </h1>
<label for="uname"><b>Username</b></label>
<input type="text" placeholder="Enter Username" name="uname"
required>
<label for="psw"><b>Password</b></label>
<input type="password" placeholder="Enter Password" name="psw"
required>
<button type="submit">Login</button>
<pre>Don't have a account?, <a href='Sign-up.php'>Click Here</a>
</div>
</form>
<?php
if ($_POST)
{
$t=$_POST['uname'];
$t1=$_POST['psw'];
$conn = mysqli_connect('localhost', 'root', ','quiz');
```

```
$result =mysqli_query($conn, "select pass from user where name='$t'");
if (mysqli_num_rows($result)==0){
print "<h1>You don't have an account.</h1>";
}
else
{
$row= mysqli_fetch_assoc($result);
$s=mysqli_query($conn,"select score from leaderboard where name='$t'");
$row1=mysqli_fetch_assoc($s);
$_SESSION["score1"]=$row1["score"];
$_SESSION["Admin"]=$t;
if ($t1== $row["pass"])
header("location: home.php");
else
print "<h1>You enter a wrong password</h1>";
}
}
?>
</body>
</html>
```

## Dbconfig.php

```
<?php
$con = mysqli_connect("localhost","root","","quiz");
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```

## Sign-Up.php

```
<head>
<title>Login| Quiz </title>
<link rel="stylesheet" href="s.css">
</head>
<?php
$conn = mysqli_connect('localhost', 'root', '','quiz');
?>
<body>
<div
class='main_div'><img
src
="https://www.jdrf.org/wp-
content/uploads/2020/12/AWS-logo-2.jpg" width="100" height="100">
<form method="post">
<div>
<h1> Hello User </h1>
<pre style="text-align: center;">
Fill the details to create an account
</pre>
<label for="uname"><b>Username</b></label>
<input type="text" placeholder="Enter Username" name="uname"
required>
<?php
$t=$_POST['uname'];
$row="select * from user where name='$t' ";
$result=mysqli_query($conn,$row);
if (mysqli_num_rows($result)>0)
{
print("<p class= 'u_e'>Username name already exist</p>");
}
?>
```

```
<label for="psw"><b>Password</b></label>
<input type="password" placeholder="Enter Password" name="psw"
required>
<label for="age" style="margin-right:10%;"><b>Age :</b></label>
<input type="number" placeholder="Enter Age" name="Age" required>
<label for="gender" style="margin:2%;"><b>Gender :</b></label>
<label for="male"><b>Male</b></label>
<input type="radio" placeholder="Enter Gender " name="Gender" >
<label for="gender "><b>Female</b></label>
<input
type="radio"
placeholder="Enter
Gender"
name="Gender"><br><br>
<label
for="Country"
style="margin-
right:4%;"><b>Country</b></label>
<input type="text" placeholder="Enter Country" name="Country"
required>
<button type="submit">Create Account</button>
<pre>
Already have an Account? To sign-in <a href="index.php"> Click
Here</a>
</pre>
</div>
</form>
</div>
<?php
if ($_POST)
{
$t1=$_POST['psw'];
$t2=$_POST['Age'];
$t3=$_POST['Country'];
```

```
if (mysqli_num_rows($result)==0)
{
$sql = "insert into user(name,pass,age,country) values('$t','$t1',$t2,'$t3')";
$sql1 = "insert into leaderboard(name,score) values('$t',0)";
$result = mysqli_query($conn,"select * from user");
mysqli_query($conn,$sql1);
mysqli_query($conn, $sql);
print "<h1>Account created</h1>";
header("location: index.php");
}
}
?>
</body>
</html>
```

## Logout.php

```
<?php
session_start();
session_destroy();
header("location:index.php");
?>
```

## Quiz.php

```
<!DOCTYPE>
<html>
<?php require 'dbconfig.php';
session_start(); ?>
<head>
<title>Quiz</title>
<style>
body {
```

```
background: url("bg.jpg");
background-size:100%;
background-repeat: no-repeat;
position: relative;
background-attachment: fixed;
}
/* button */
.button {
display: inline-block;
border-radius: 4px;
background-color: #f4511e;
border: none;
color: #FFFFFF;
text-align: center;
font-size: 28px;
padding: 20px;
width: 500px;
transition: all 0.5s;
cursor: pointer;
margin: 5px;
}
.button span {
cursor: pointer;
display: inline-block;
position: relative;
transition: 0.5s;
}
.button span:after {
content: '\00bb';
position: absolute;
opacity: 0;
top: 0;
right: -20px;
transition: 0.5s;
```

```
}
.button:hover span {
padding-right: 25px;
}
.button:hover span:after {
opacity: 1;
right: 0;
}
.title{
background-color: #ccc11e;
font-size: 28px;
padding: 20px;
}
.button3 {
border: none;
color: white;
padding: 10px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
margin: 4px 2px;
-webkit-transition-duration: 0.4s; /* Safari */
transition-duration: 0.4s;
cursor: pointer;
}
.button3 {
background-color: white;
color: black;
border: 2px solid #f4e542;
}
.button3:hover {
background-color: #f4e542;
color: Black;
```

```
}
</style>
</head>
<body><center>
<div class="title">Quiz</div>
<?php
if (isset($_POST['click']) || isset($_GET['start'])) {
@$_SESSION['clicks'] += 1 ;
$c = $_SESSION['clicks'];
if(isset($_POST['userans'])) { $userselected = $_POST['userans'];
$fetchqry2 = "UPDATE `quiz` SET `userans`='$userselected' WHERE
`id`=$c-1";
$result2 = mysqli_query($con,$fetchqry2);
}
} else
{
$_SESSION['clicks'] = 0;
}
//echo($_SESSION['clicks']);
?>
<div class="bump"><br><form><?php if($_SESSION['clicks']==0){ ?> <button
class="button"
name="start"
float="left"><span>START
QUIZ</span></button> <?php } ?></form></div>
<form action="" method="post">
<table><?php if(isset($c)) {
$fetchqry = "SELECT * FROM `quiz` where
id='$c'";
$result=mysqli_query($con,$fetchqry);
$num=mysqli_num_rows($result);
$row = mysqli_fetch_array($result,MYSQLI_ASSOC); }
?>
<tr><td><h3><br><?php
```

```
echo
@$row['que'];?></h3></td></tr>
<?php
if($_SESSION['clicks'] > 0 && $_SESSION['clicks'] < 11){ ?>
<tr><td><input required type="radio" name="userans" value="<?php echo
$row['option 1'];?>"> <?php echo $row['option 1']; ?><br>
<tr><td><input required type="radio" name="userans" value="<?php echo
$row['option 2'];?>"> <?php echo $row['option 2'];?></td></tr>
<tr><td><input required type="radio" name="userans" value="<?php echo
$row['option 3'];?>"> <?php echo $row['option 3']; ?></td></tr>
<tr><td><input required type="radio" name="userans" value="<?php echo
$row['option
4'];?>"> <?php
echo
$row['option
4']; ?><br><br><br></td></tr>
<tr><td><button class="button3" name="click" >Next</button></td></tr>
<?php }
?>
<form>
<?php if($_SESSION['clicks']> 10){
$qry3 = "SELECT `ans`, `userans` FROM `quiz`;";
$result3 = mysqli_query($con,$qry3);
$storeArray = Array();
while ($row3 = mysqli_fetch_array($result3, MYSQLI_ASSOC)) {
if($row3['ans']==$row3['userans']){
@$_SESSION['score'] += 1 ;
}
}
$p=$_SESSION['score'];
$p2=$_SESSION['Admin'];
$sql = "UPDATE `leaderboard` SET `score` = $p WHERE `name`='$p2'";
$r1= mysqli_query($con,$sql);
?>
```

```
<h2>Result</h2>
<span>No. of Correct Answer: <?php echo $no = @$_SESSION['score'];
unset($_SESSION['score']);?></span><br>
<span>Your Score: <?php echo $no*2; ?></span><br> <br>
<a class="button3" href="home.php" >Home</a>
<a class="button3" href="logout.php">Log Out</a>
<?php } ?>
<!-- <script type="text/javascript">
function radioValidation(){
/* var useransj = document.getElementById('rd').value;
//document.cookie = "username = " + userans;
alert(useransj); */
var uans = document.getElementsByName('userans');
var tok;
for(var i = 0; i < uans.length; i++){
if(uans[i].checked){
tok = uans[i].value;
alert(tok);
}
}
}
</script> -->
</center>
</body>
</html>
```

## Home.php

```
<?php
// Start the session
session_start();
if (!isset($_SESSION["Admin"]))
{
```

```
header("location:index.php");
}
$conn=mysqli_connect('localhost','root','','quiz');
$p=$_SESSION["Admin"];
$sql1="select * from `leaderboard` where `name`='$p'";
$result=mysqli_query($conn,$sql1);
$row=mysqli_fetch_assoc($result);
?>
<html>
<head>
<title>Home| Quiz </title>
<link rel="stylesheet" href="s2.css">
</head>
<body>
<nav id="navbar">
<div class="container">
<div id="img-container">
<img src="https://www.jdrf.org/wp-content/uploads/2020/12/AWS-logo-2.jpg"
width="100" height="100">
</div>
<div id="inside">
Hello <?php echo $_SESSION["Admin"]; ?><br>
Your Score: <?php echo $row['score']; ?>
</div>
<ul>
<li><a href='home.php'>Home</a></li>
<li><a href='quiz.php'> Quiz</a></li>
<li id='btn'><a href='logout.php'>Logout</a></li>
</ul>
</div>
</nav>
<div id='showcase'>
<a href='quiz.php' id='a_q'>Attempt Quiz</a>
</div>
```

```
</body>
</html>
```

## S2.css

```
* {
margin: 0;
padding:0;
box-sizing: border-box;
}
#inside
{
margin-left:50px;
margin-top:20px;
}
.container
{
display:flex;
margin: ;
overflow: auto;
padding: 0;
}
html,body {
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
line-height: 1.7em;
}
#navbar {
background: #333;
color: #fff;
overflow: auto;
}
#navbar a {
text-decoration:none;
```

```
text-color:#fff;
color: #fff;
}
#navbar ul {
margin-left:700px;
list-style: none;
float: right;
margin-top:40px;
}
#navbar ul li {
float: left;
}
#navbar ul li a {
display: block;
padding: 20px;
text-align: center;
}
#navbar ul li a:hover,
#navbar ul li a.current {
background: #444;
color: #f7c08a;
}
#showcase{
text-align:center;
margin:300px;
height:600px;
}
body
{
background:rgb(242,242,242);
}
#a_q:hover,
#a_q.current
{
```

```
text-decoration:none;
background:#444;
color: #f7c08a
}
#a_q
{
width:20px;
background:#444;
padding: 1em 1.5em;
color:#fff;
text-align:center;
text-decoration:none;
}
```

## S.css

```
form { border: 1px solid black; width: 50%;
margin-left: 25%; margin-top: 10%; box-shadow: 10px 10px;
}
input[type=text], input[type=password] { width: 100%; padding: 12px
20px; margin: 8px 0; border: 1px solid #ccc; box-sizing: border-box;
}
input[type=number] { width: fit-content; padding: 12px 20px; margin: 8px
0; border: 1px solid #ccc; box-sizing: border-box;
}
h1 { margin-left: 3%; text-align: center;
}
button { background-color: #04AA6D; color: white; padding: 14px 20px;
margin: 8px 0; border: #04AA6D; cursor: pointer; width: 100%;
}
div { padding: 1%;
}
.u_e
```

```
{
color:red;
}
img{
float:right;
margin: 2px 3px 15px 25 px;
}
```

## LIMITATIONS OF THIS PROJECT

It is clear that every system has advantages and disadvantages and cloud computing is not an exception. Its advantages for business such as minimizing public infrastructure investment costs so that businesses can be more focused on aspects of its functionality. SaaS services enable rapid development and implementation of applications to increase productivity, open new markets for information technology development service industries. However, everything has its weakness include cloud computing. Cloud computing pros and cons are debated to its user.

1) **Downtime :** Downtime is often cited as one of the biggest disadvantages of cloud computing. Since cloud computing systems are internet-based, service outages are always an unfortunate possibility and can occur for any reason. Best practices for minimizing planned downtime in a cloud environment Design services with high availability and disaster recovery in mind. Leverage the multi-availability zones provided by cloud vendors in the infrastructure.

2) **Security and privacy :** Although cloud service providers implement the best security standards and industry certifications, storing data and important files on external service providers always opens up risks. Any discussion involving data must address security and privacy, especially when it comes to managing sensitive data. However, our responsibilities lie in the realm of user access management, and it's up to us to carefully weigh all the risk scenarios. Best practices for minimizing security and privacy risks is to understand the shared responsibility model of our cloud provider (AWS) as we will still be liable for

what occurs within our network and in our product. Implement security at every level of our deployment. Knowing who is supposed to have access to each resource and service, and limit access to least privilege. Turn on encryption wherever we can.

3) **Vulnerability to attack :** In cloud computing, every component is online, which exposes potential vulnerabilities. Even the best teams suffer severe attacks and security breaches from time to time. Since cloud computing is built as a public service, no one at a cloud vendor checks our administration skills before granting an account. Best practices to help reduce cloud attacks are - making security a core aspect of all IT operations, keeping all teams up-to-date with cloud security practices, ensure security policies and procedures are regularly checked and reviewed, proactively classify information and apply access control, using cloud services such as AWS Inspector, AWS CloudWatch, AWS CloudTrail, and AWS Config to automate compliance controls.

4) **Limited control and flexibility :** Since the cloud infrastructure is entirely owned, managed, and monitored by the service provider, it transfers minimal control over to the customer. To varying degrees (depending on the particular service), cloud users may find they have less control over the function and execution of services within a cloud-hosted infrastructure. A cloud provider's end-user license agreement (EULA) and management policies might impose limits on what customers can do with their deployments. Customers retain control of their applications, data, and services, but may not have the same level of control over their backend infrastructure.

5) **Vendor lock-in :** Vendor lock-in is another perceived disadvantage of cloud computing. Easy switching between cloud services is a service that hasn't yet completely evolved, and organizations may find it difficult to migrate their services from one vendor to another. Differences between vendor platforms may create difficulties in migrating from one cloud platform to another, which could equate to additional costs and configuration complexities. Gaps or compromises made during migration could also expose our data to additional security and privacy vulnerabilities. Employ a multi-cloud strategy to avoid vendor lock-in.

Building applications with services that offer cloud-first advantages, such as modularity and portability of microservices and code, think containers and Kubernetes.

6) **Cost concerns :** Adopting cloud solutions on a small scale and for short-term projects can be perceived as being expensive. However, the most significant cloud computing benefit is in terms of IT cost savings. Pay-as-you-go cloud services can provide more flexibility and lower hardware costs, but the overall price tag could end up being higher than we expected. Until sure of what will work best, it's a good idea to experiment with a variety of offerings. Make use of the cost calculators made available by providers like Amazon Web Services and Google Cloud Platform. Prevent over provision of services, instead look into using auto-scaling services. Ensuring the option to scale DOWN as well as UP. Pre-pay and take advantage of reserved instances if we have a known minimum usage. Automate the process to start/stop the instances to save money when they are not being used. Create alerts to track cloud spending.

# REFERENCES


[1] https://docs.aws.amazon.com

[2] Wikipedia contributors. (2021, June 25). Amazon Web Services. In Wikipedia, The Free Encyclopedia. Retrieved 17:38, June 29, 2021, from https://en.wikipedia.org/w/index.php?title=Amazon_Web_Services&oldid=1030330967

[3] https://aws.amazon.com/elasticloadbalancing/?whats-new-cards-elb.sort-by=item.additionalFields.postDateTime&whats-new-cards-elb.sort-order=desc

[4] https://aws.amazon.com/backup/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

[5] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

[6] https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

[7] https://searchaws.techtarget.com/definition/Amazon-Relational-Database-Service-RDS

[8] https://www.simplilearn.com/tutorials/aws-tutorial/aws-s3

[9] https://www.knowledgehut.com/tutorials/aws/aws-cloudfront

[10] https://www.javatpoint.com/aws-iam