Contents:

1. *Project Overview*

This Project is focused on students and instructor. It helps to create courses , storing them , managing profiles and delivery of educational courses . By gathering all learning materials at one place , it helps create professionally structured course content that is delivered through a variety of different modes.

A key feature of our LMS is the ability to create different types of users. In our E-Leaning platform we have Administrator , Users, and Instructors who have different tasks and functionality assigned to them.

2. *Project Objective*

This project is seeking to achieve the following objectives:

- students are able to complete lessons and courses.
- Instructors are able to access the site and perform CRUD operations on the site (like create, read, update and delete courses).
- Administrators, who have complete access to everything on the site also he/she is responsible to

pass the courses that are created by instructor to be available on our platform to avoid plagrism.

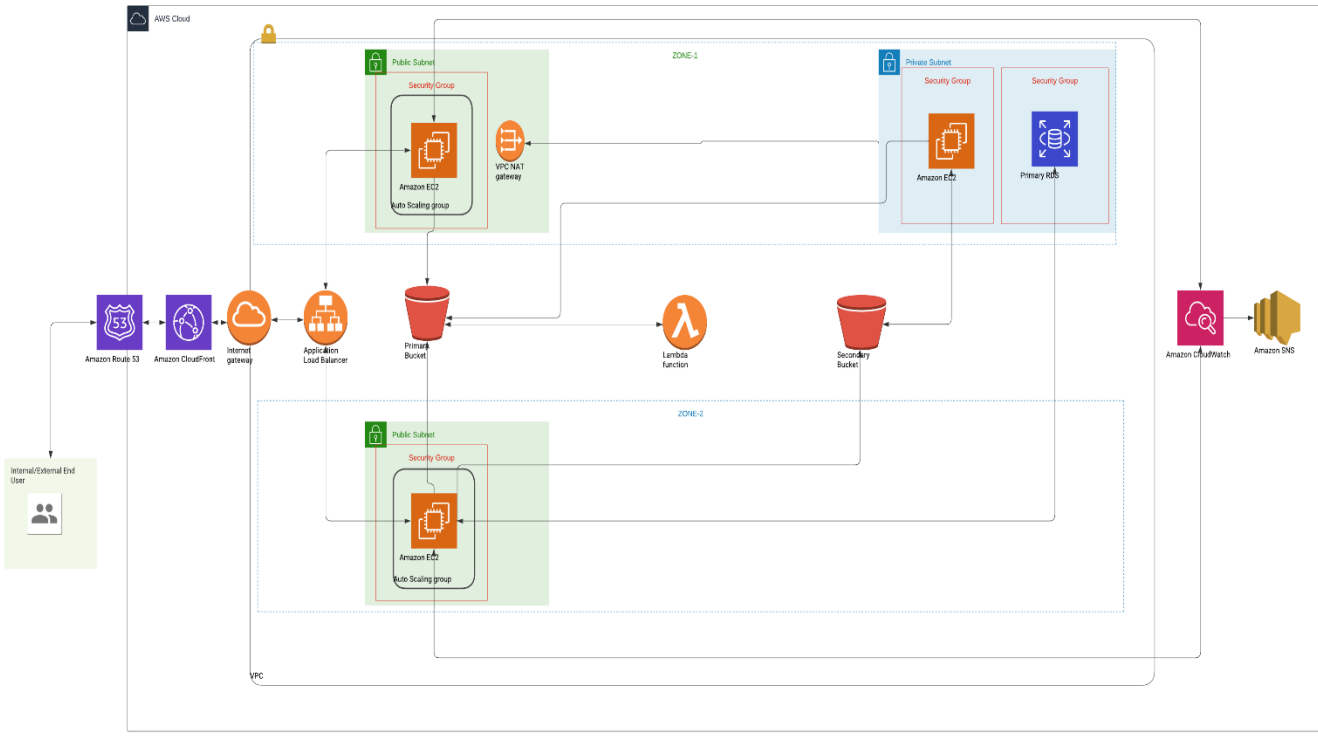
- It provides a platform for both users as well as instructors.

3. *Project Architecture and References*

AWS Solution Architecture:

Diagram

An architecture diagram is a graphical representation of a set of concepts that are part of the architecture, including their principles, elements, and components. The architecture diagram of the project is below :



Explanation:

We have created a website called “Learn it” which is a E-learning platform ,when a user comes to our website it uses elastic IP attached to load-balancer which distribute traffic to EC2 instances deployed in 2 different zone with auto scaling enabled on both the zones , this help our app to be highly available . The use of load balancer is important to distribute the load equally in both the zones so resources are utilized properly . Autoscaling help us in reducing the number of EC2 instances running on AWS when load on traffic is less and increase the EC2 instances when load is high also we can configure health checks for our EC2 instances in auto scaling group , hence we can avoid sending traffic to ec2 instances that are unhealthy and all this is automatically handled by auto scaling .

Every EC2 instance is placed in security group so unauthorized access to instances cannot be made . In our project we also have a private subnet where are running an EC2 instance which will be used as admin and has vital role in functioning of website . This instance is responsible for passing new courses that are launched by instructor on website hence avoiding any kind of plagiarism and error in newly launched courses .This instance being in private subnet might require to access internet for downloading new software for that we have placed a NAT gateway which this instance has access to for connecting to internet without compromising on security .

In our project we are using two S3 bucket for putting data of courses , in primary S3 bucket data that website will need to render to the users is placed and in secondary S3 bucket new

courses will be kept , once they are passed by the admin these courses content will be moved to primary bucket which website / public instances can access for showing these new courses also to the users (students) .

We are available with an RDS where we will be storing all data of students , courses , instructor etc. that will be required by our website . With databases we can easily perform CRUD operations and AWS RDS service is highly scalable and available with also options to do read replicas , backup data and encryption of data .

Now to expose the application publicly, we use a DNS service called Route53, by using Route53 we can register the domain, and that domain will be linked to the public IP address of the Elastic Load Balancer . We have enabled cloud watch to track various metrices and trigger a SNS service that will notify us whenever a threshold is reached . We have also used Lambda function to perform data processing whenever a new object is added to S3 bucket .

WEBSITE Technology :

Frontend:

- Html : HTML stands for Hyper Text Markup Language . HTML is the standard markup language for creating Web page . HTML describes the structure of a Web page
- CSS : CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media . CSS saves a lot of work. It can control the layout of multiple web pages all at once

- Javascript : JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc.

Backend:

- Python(Flask) : Flask is a micro web framework written in python. It is classified as a microframework because it does not require particular tools or libraries.[2] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

- Mysql: MySQL is one of the most recognizable technologies in the modern big data ecosystem. Often called the most popular database and currently enjoying widespread, effective use regardless of industry

WEBSITE OVERVIEW : our website will contain a login and signup page for student and teacher , as they login , into the website they are send to their respective homepage ,for student he is directed to courses page where he can see all courses available and purchase new courses and a teacher is send to a dashboard where he can see all courses that are made by him and various details , like number of student , revenue etc. also he will have another option where a button will be available to launch new courses , these courses content will be moved to secondary s3 bucket , once the courses are approved by admin these courses will be send to primary bucket which website has access to and uses to load content of courses .

Cloud Monitoring:

For Cloud Monitoring and logging we use cloud watch. Amazon CloudWatch provides real-time monitoring of Amazon Web Services (AWS) resources and applications running on AWS . It is used 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 this data to determine whether you should launch additional instances to handle the increased load. You can also use this data to stop under-used instances to save money.

Reference:

- https://www.youtube.com/watch?v=__knpcBRLHg&ab_channel=edureka%21
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Dashboards.html
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/working_with_metrics.html

CI/CD:

Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control/GIT.

Continuous Delivery (CD) is the process to build, test, configure and deploy from a build to a production environment.

To continuously deliver consistently and reliably, a team must break down the software delivery process into delivery stages and automate the movement of the code through the stages to create a delivery pipeline.

A delivery pipeline is so-named because it allows code to flow through a consistent, automated sequence of stages where each stage in the sequence tests the code from a different perspective.

Each successive stage becomes more production-like in its testing and provides more confidence in the code as it progresses through the pipeline.
While each stage is either building or testing the code, the stage must have the necessary automation to not only run the test but also to provision, deploy, set up, and configure the testing and staging environments. The code should progress through each stage automatically. The goal is to strive for unattended automation that eliminates or minimizes human intervention.

References:

- https://www.youtube.com/watch?v=NwzJCSPSPZs&ab_channel=BlockExplorer
- https://aws.amazon.com/devops/continuous-delivery/
- https://aws.amazon.com/getting-started/hands-on/set-up-ci-cd-pipeline/
- https://aws.amazon.com/getting-started/hands-on/create-continuous-delivery-pipeline/

4. Defination of Terms

Acronym	Description
AWS	Amazon Web Services
VPC	Virtual Private Network
S3	Simple storage service
EC2	Elastic Cloud Compute
SNS	Simple notification service
RDS	Relational Database service