CA2 INT330 student - It is just a format to give your Project title.

Managing cloud solutions (Lovely Professional University)

INT 330 -- MANAGING CLOUD SOLUTIONS

CA 2: PROJECT

Designing a Simple AWS Architecture.

Description:
The goal of this project is to design a simple and efficient architecture using Amazon Web Services (AWS) to demonstrate the key components of cloud infrastructure. The architecture will focus on scalability, reliability, and cost-effectiveness, utilizing fundamental/advance AWS services. The project will provide hands-on experience in deploying cloud solutions and understanding AWS best practices.
This project will serve as an introduction to building cloud architectures, providing a foundation for more complex AWS solutions in the future..

Basic AWS Services to be used

- Amazon S3 – **As Storage**
- Amazon EC2 **– As Compute service**
- AWS Identity and Access Management (IAM) (with restrictions) - **As Security service**
- AWS VPC – **As Network service**
- AWS RDS – **As Database service**

Advanced AWS Services: (USE atleast two services)

- AWS Lambda
- Amazon CloudFront
- Amazon Elastic Load Balancer
- Amazon CloudWatch.
- AWS Auto Scaling to handle variable traffic loads.
- AWS Elastic Beanstalk
- AWS EBS

EXAMPLE

1. Scenario:

An e-commerce website regularly experiences unpredictable traffic spikes during promotional campaigns, leading to potential downtime, performance issues, and increased operational costs. The goal is to design an AWS-based architecture that ensures seamless scalability, high availability, secure operations, and cost efficiency while maintaining an excellent user experience.

2. Problem Statement:

Design an AWS architecture to support an e-commerce website capable of handling unpredictable traffic spikes during promotional campaigns.

3. Objectives:

1. To enable the system to automatically adjust resources to handle varying traffic loads
2. To ensure the website remains operational with minimal downtime, even during peak traffic.
3. To deliver fast response times and low latency for users globally.
4. To optimize the architecture to minimize costs while meeting performance and availability goals.

4. Outcomes:

1. The system automatically scales resources up or down based on real-time demand, ensuring smooth user experiences during traffic spikes.
2. The architecture sustains high availability and fault tolerance, with minimal downtime during promotional campaigns.
3. The Efficient use of AWS services to manage costs, employing tools like AWS Auto Scaling, spot instances, and serverless options where applicable.
4. The Consistent performance with fast load times during high-traffic events.

5. Proposed AWS Components:

- **Amazon EC2**: For hosting the application.
- **Elastic Load Balancer (ELB)**: Distribute traffic across multiple EC2 instances.
- **Amazon RDS (Relational Database Service)**: For storing structured data.
- **Amazon S3**: For hosting static content like images and videos.
- **Amazon CloudFront**: As a Content Delivery Network (CDN) for faster delivery of assets.
- **AWS Auto Scaling**: Automatically increase or decrease EC2 instances based on traffic.
- **AWS IAM**: To manage user permissions.
- **AWS WAF (Web Application Firewall)**: For security against common web exploits.
- **Amazon CloudWatch**: For monitoring performance and resource utilization.

6. Solution:

Implementation 1: Traditional Server-Based Approach

This approach relies on EC2 instances for compute and other managed services for scalability and availability.

Steps:

1. **Setup Frontend and Content Delivery**:
   - Use **Amazon CloudFront** for caching and accelerating content delivery globally.
   - Configure **Elastic Load Balancer (ELB)** to distribute incoming traffic across multiple EC2 instances.

2. **Launch Compute Resources**:
   - Create an **Auto Scaling Group** for EC2 instances to dynamically adjust based on traffic.
   - Use EC2 instances to run the web application and backend services.

3. **Database Configuration**:
   - Deploy **Amazon RDS (Aurora)** with read replicas for scalable relational database operations.
   - Use **Amazon ElastiCache (Redis)** for caching frequently accessed data.

4. **Storage and Backup**:
   - Store static assets and backups in **Amazon S3**.
   - Enable **S3 Transfer Acceleration** for faster global uploads.

5. **Monitoring and Security**:
   - Configure **AWS CloudWatch** to monitor instance health and application performance.
   - Implement **AWS WAF** to secure the application against attacks.

6. **CI/CD Deployment**:
   - Use **AWS CodePipeline** to automate deployments and **AWS CodeBuild** to build and test updates.

1. Rubrics for Evaluation:

Criteria	Description	
Implementation	**Scalability:**Evaluate the system's ability to scale dynamically during traffic spikes without downtime.	
	Availability : Assesses uptime and fault tolerance, ensuring consistent operations during campaigns.	
	Security Measures: Evaluate the implementation of security measures to protect against potential threats.	
	Performance: Measure the speed, response time, and low latency of the application during high traffic.	
	Monitoring & Cost Mgmt.: Evaluate and monitor how well resources are optimized to balance performance and cost.	

Documentation	Evaluate the clarity, completeness, and technical soundness of the architecture documentation.	
Presentation	Evaluate how clearly the presenter conveys the information, ensuring the audience understands the architecture and concepts.	