Gopinath S B
28.01.2025

Aws Scenario

---

1. Frontend (React)

- **Amazon S3 + CloudFront**:
    - Store your React app's static files (HTML, CSS, JavaScript) in an S3 bucket.
    - Use Amazon CloudFront to distribute the content globally with low latency. CloudFront can cache static assets at edge locations, improving load times for end users.
    - S3 allows for easy scalability and high availability since it is designed for durability and availability. Ensure that the S3 bucket is private and set permissions correctly.

2. Backend API (Python Flask/Django)

- **Amazon EC2 (Elastic Compute Cloud)**:
    - Deploy the Flask/Django backend on EC2 instances. Choose an instance type based on expected load
    - Set up an Auto Scaling group to automatically adjust the number of EC2 instances based on demand. This ensures your backend remains scalable.
    - Utilize Elastic Load Balancer (ELB) to distribute traffic across your EC2 instances, ensuring even traffic distribution and fault tolerance.
- **Amazon ECS or EKS** (optional):
    - For containerized applications, you could use Amazon ECS (Elastic Container Service) or EKS (Elastic Kubernetes Service) to run your Flask/Django app in Docker containers. This simplifies deployment, scaling, and management of the application.

3. Database (MySQL)

- **Amazon RDS (Relational Database Service)**:

    - Host the MySQL database using Amazon RDS for a fully managed database solution. RDS handles backups, patching, and scaling for you.
    - Multi-AZ deployment: Enable Multi-AZ for RDS to ensure high availability. This automatically replicates your primary database to a secondary standby instance in a different Availability Zone.
    - Read Replicas: If you anticipate high read traffic, use Read Replicas to distribute read queries and improve performance.
- **Amazon Aurora (optional):**
    - If you need a more performant MySQL-compatible database, consider Amazon Aurora, which offers better scalability and fault tolerance than standard MySQL.

4. Networking and Security

- **Amazon VPC (Virtual Private Cloud):**
    - Set up your application in a VPC with private and public subnets.
    - Place your EC2 instances in private subnets and expose only the load balancer (ELB) to the public internet, ensuring better security.
- **Security Groups:**
    - Use security groups to control inbound and outbound traffic to your EC2 instances and database.
    - For the EC2 instances, allow only necessary ports and restrict database access to the backend API only.
- **AWS WAF (Web Application Firewall):**
    - Protect your web application from common web exploits and attacks by using AWS WAF.
- **AWS Shield:**
    - Use AWS Shield (Standard or Advanced) for DDoS protection to safeguard your application from large-scale attacks.

5. Monitoring and Logging

- **Amazon CloudWatch:**
    - Use CloudWatch for logging and monitoring your EC2 instances, RDS, and load balancers. Set up custom CloudWatch Alarms for any performance issues.
- **AWS X-Ray:**

    - Use AWS X-Ray for tracing the requests through your application and identifying performance bottlenecks.
- **CloudTrail**:
    - Enable AWS CloudTrail for auditing API activity within your AWS environment to maintain security and compliance.

6. CI/CD Pipeline

- **AWS CodePipeline** or **GitHub Actions**:
    - Set up a CI/CD pipeline using AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy for automated deployment of new application versions.
    - Ensure that your React frontend and Python backend are built, tested, and deployed efficiently to reduce human error and downtime.

7. Scaling and High Availability

- **Auto Scaling for EC2:**
    - Set up Auto Scaling for your EC2 instances to ensure that you can handle increases in traffic and reduce the number of instances during low traffic periods, saving costs.

8. Backup and Disaster Recovery

- **RDS Automated Backups**:
    - Enable automated backups on RDS to keep daily snapshots of your database and retain backups for a configurable period.
- **S3 Backup**:
    - For important files, use Amazon S3 for backup, or Amazon Glacier for long-term archival storage.
- **Cross-Region Replication (optional):**
    - Set up cross-region replication for your S3 bucket and RDS for added resilience in case of regional failure.