

Designing Cloud Infrastructure

Table of Contents

- Business Case 2
- Architectural diagram 5
- Proposed Architecture 6
- Adherence to AWS Well-Architected Framework 7
- Cost Analysis 8

- ### Business Case

Online food ordering is a procedure that brings food or take away, from home cooks, nearby eateries and other food cooperatives through an app or a website. Since 2020 and the outbreak of Covid-19, this method of food delivery is becoming increasingly popular among a growing number of individuals, particularly the younger generation, who are embracing food ordering apps, thus revolutionizing the way food is delivered and collected.

The aim of the food ordering and delivery app is to disrupt the market by offering a smooth and effective platform for customers make orders and have food brought right to their door. The focus of the business case is on utilizing AWS cloud services to construct an adaptable, protected, and economical infrastructure that guarantees exceptional performance and fulfils the ever-changing needs of the food delivery industry.

Objectives

1. High Performance - Ensure that users receive low-latency response times during peak hours.
Optimize data transfer and processing to ensure quick order processing.
2. Scalability - Allow for varying workloads and seasonal fluctuations in demand.
Auto Scaling dynamically adjusts resources based on traffic.
3. Security - Safeguard customer information and financial transactions.
Implement strong access control, encryption, and threat detection.
4. Cost-Effectiveness - Optimize infrastructure costs without sacrificing performance.
Utilize AWS pricing models effectively, such as Reserved Instances.
5. Modularity and Ease of Development –
Enable a microservices architecture for agility and ease of development.
Support continuous integration and deployment for rapid feature updates.

Target User Base

The primary target users are tech-savvy individuals looking for a simple and efficient way to order food. The app aims to appeal to a wide range of audiences, including busy professionals, students, and families, by offering a diverse selection of restaurants and an easy-to-use interface.

Revenue Model

The revenue model is based on commissions from partner restaurants for each successful order placed via the platform. Furthermore, premium subscription plans could be introduced to provide exclusive benefits to frequent users.

Benefits

- Technology Stack

Utilize AWS services to achieve scalability and robust infrastructure.
Leverage microservices to increase flexibility and simplify maintenance.

- User Experience

Build an intuitive and user-friendly app interface.
Implement efficient order tracking and real-time delivery updates.

- Partner Ecosystem

Establish strong partnerships with a diverse range of restaurants.
Introduce features like promotions and loyalty programs to boost collaboration.

Future Expansion

The cloud infrastructure is designed to be easily scalable, allowing for future growth and expansion into new markets. The architecture enables the addition of new features and services without significant disruption to the existing system.

Risk Mitigation

Security Measures

- Continuous surveillance for potential security threats.
- Penetration testing and vulnerability assessments are conducted on a regular basis.

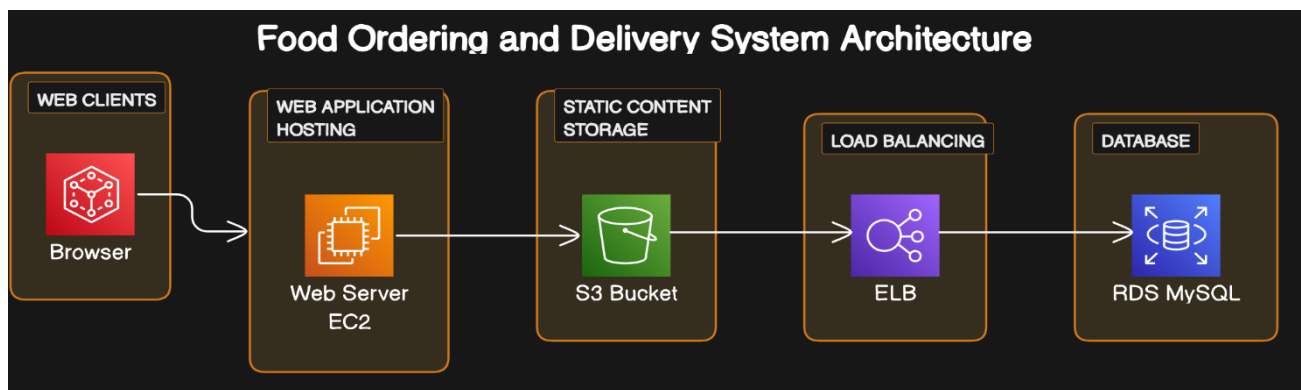
Redundancy and Disaster Recovery

- Multi-AZ deployments and data backups ensure high availability.
- Regularly test and update disaster recovery plans.

Conclusion

The proposed cloud infrastructure aligns with the startup's goals and industry trends, positioning our food ordering and delivery app for success. Adherence to the AWS Well-Architected Framework ensures that our platform is reliable, secure, and efficient. With a focus on innovation, user experience, and strategic partnerships, our cloud infrastructure lays the foundation for a competitive and long-term presence in the dynamic food delivery market.

- Architectural diagram



- Proposed Architecture

Application Architecture

- Utilize a microservices architecture for modularity and scalability.
- EC2 instances to host application components.
- Auto Scaling groups can dynamically adjust capacity in response to demand.

Storage

- Amazon S3 is used to store static assets such as images and user uploads.
- Amazon RDS can host a relational database to manage orders, customers, and menus.

Networking

- Implement Amazon VPC to ensure network isolation and security.
- Use Elastic Load Balancing to distribute traffic across EC2 instances.

Security

- IAM roles enable fine-grained access control.
- Data encryption at rest and in transit is accomplished through the use of AWS Key Management Service.
- AWS WAF provides protection against common web exploits.

Monitoring and Management

- AWS CloudWatch monitors EC2 instances, database metrics, and application logs.
- AWS Config tracks changes and ensures compliance.

- Adherence to AWS Well-Architected Framework

Operational Excellence

Implementing automation scripts for infrastructure provisioning and configuration guarantees the achievement of consistent and error-free deployments, minimizing the need for manual intervention and boosting operational efficiency.

Regularly conducting operational reviews enables the identification of areas that can be improved, the optimization of workflows, and ensures that operational practices are aligned with business objectives.

Security

By implementing Identity and Access Management (IAM) roles based on the principle of least privilege, security breaches can be minimized by ensuring that users and services have only the permissions that are necessary.

Frequent security audits and continuous monitoring with AWS GuardDuty improve threat detection capabilities, providing real-time insights into potential security issues and allowing for timely mitigation.

Reliability

Auto Scaling and deployment across multiple Availability Zones (Multi-AZ) ensure high availability by automatically adjusting resources based on demand and providing redundancy across data centers.

Routine testing and updating of disaster recovery plans ensure that the system can recover quickly in the event of a failure, reducing downtime and ensuring business continuity.

Performance Efficiency

Using Auto Scaling for capacity matching ensures that resources scale in response to demand, avoiding overprovisioning and optimizing costs based on actual usage.

Efficient usage of Reserved Instances for predictable workloads results in significant cost savings by committing to a specific instance type in exchange for a lower, predictable rate over a defined time period.

- Cost Analysis

Amazon EC2

Amazon Simple Storage Service (S3)








Elastic Load Balancing

Amazon RDS for MySQL

AWS Key Management Service

Amazon GuardDuty

Amazon CloudWatch

<input type="checkbox"/>	Amazon EC2		-	3.80 USD
<input type="checkbox"/>	Amazon Simple Storage Service (S3)		-	1.15 USD
<input type="checkbox"/>	Elastic Load Balancing		-	16.43 USD
<input type="checkbox"/>	AWS Key Management Service		-	11.00 USD
<input type="checkbox"/>	Amazon GuardDuty		-	0.00 USD
<input type="checkbox"/>	Amazon CloudWatch		-	0.00 USD
<input type="checkbox"/>	Amazon RDS for MySQL		-	284.30 USD

Monthly cost
316.68 USD

- Strategies for cost optimization.

Auto Scaling - Dynamically modify the number of EC2 instances in based on demand, optimizing expenses during periods of low traffic.

Reserved Instances - Analyse patterns of usage and commit to Reserved Instances for predictable workloads to take advantage of reduced, reserved pricing.

S3 Lifecycle Policies - Automatically transition or delete objects depending on established rules, hence reducing storage costs.

Elastic Load Balancer (ELB) Optimization - Modify the settings of the Elastic Load Balancer (ELB) and make use of Application Load Balancers to enhance effectiveness and achieve optimal cost-efficiency.

Review and Optimize Data Transfer Costs- To enhance the efficiency of data transfer expenses, it is imperative to select the appropriate AWS regions, employ content delivery networks (CDNs), and optimize the architecture of applications.