LocalBazaar

A new way of supporting and
connecting with your community in an
increasingly fragmented world.

Soham Bapat

1 Introduction

1.1 Motivation

To create an e-commerce platform for small businesses to sell goods and services targeted towards local communities that connect sellers and buyers through meaningful interactions and unique goods.
To put into practice the knowledge we have gained during the course. To think like a cloud architect and gain exposure to different cloud computing platforms.

1.2 Executive summary

This project aims to revolutionize the e-commerce landscape by introducing a dynamic and community-focused platform, aptly named "LocalBazaar." LocalBazaar is designed to empower small businesses and artisans, enabling them to thrive in the digital age while fostering deeper connections within local communities. In a world increasingly dominated by faceless online marketplaces, LocalBazaar stands out as a unique and impactful solution.

LocalBazaar leverages cloud technology to provide small businesses with a user-friendly and cost-effective platform to showcase their goods and services. However, what truly sets us apart is our unwavering commitment to meaningful interactions that will drive business growth. Our platform prioritizes local connections, enabling buyers to discover not just products, but the stories and faces behind them. Through LocalBazaar, we aim to create a vibrant and interconnected ecosystem that promotes community engagement, supports local entrepreneurship, and brings back the personal touch that traditional commerce provides that is missing from current e-commerce platforms. By offering a range of innovative features like message the seller and community-emphatic search results, LocalBazaar is poised to transform the way small businesses engage with their communities, creating a thriving and vibrant marketplace where buyers and sellers truly connect.

2 Problem Specification

2.1 The problem

In the current e-commerce landscape, there exists a significant challenge for small businesses and artisans to thrive and connect with their local communities. Traditional commerce, with its personal touch and community engagement, has been largely overshadowed by faceless online marketplaces. This shift has led to a need for more meaningful interactions, more support for local entrepreneurship, and a disconnection between buyers and sellers. To address this problem, we are introducing "LocalBazaar," a dynamic and community-focused platform to revolutionize e-commerce and revitalize the connection between small businesses and their communities.

The main problem areas that we hope to solve are

1. Limited Local Engagement: Small businesses struggle to connect with their local customer base through existing online marketplaces, leading to a lack of community engagement.
2. Absence of Personal Touch: Current e-commerce platforms often lack the personal touch and human stories behind the products and services, which diminishes the overall shopping experience.
3. Underrepresentation of Small Businesses: Small businesses and artisans are underserved by existing e-commerce platforms, which do not adequately support their needs or provide a cost-effective and user-friendly solution.

By effectively addressing these objectives, "LocalBazaar" aims to mitigate the identified problem, rejuvenating community connections and local commerce in the e-commerce landscape.

2.2 Business Requirements

BR1. High Availability: We want services to be available 24/7. Customers should not experience any downtime.

BR2. Security and Compliance: Implement measures to safeguard sensitive data, intellectual property, and customer information from theft, breaches, or unauthorized access. Security measures reassure customers that their data is safe, promoting trust and loyalty.

BR3. User Management and Authentication: Allow users to access their resources.

BR4. Monitoring and Alerting: Send timely alerts when predefined thresholds have been exceeded or anomalies are detected. The IT team should be able to identify and address issues before they escalate into critical problems.

BR5. Disaster Recovery and Backup: Ensure operational continuity, resilience, and security in case of failures.

BR6. Documentation and Support: Establish and uphold comprehensive documentation practices encompassing infrastructure, configurations, and operational procedures.

BR7. Low Latency: Ensure the application has faster and more responsive experiences. Slow loading times may lead to cart abandonment and reduced sales.

BR8. Hybrid Tenancy (Single and Multi-Tenancy): Support both single-tenancy and multi-tenancy deployment options. Single tenancy for high-profile customers requiring dedicated, isolated environments, and multi-tenancy for cost-effective resource sharing among multiple customers.

BR9. Billing and Subscription Management: Implement a robust billing and subscription management system to support various pricing models, such as subscription-based, usage-based, and tiered pricing. Customers should have access to transparent billing information and the ability to manage their subscriptions easily.

BR10. High Performance and Efficiency: Deliver high performance, ensuring low latency, fast response times, and efficient use of resources. Performance metrics should be monitored, and performance optimization measures should be implemented to meet customer expectations.

2.3 Technical Requirements

TR1.1 Use redundant servers and load balancers. Deploy EC2 instances in multiple Availability Zones with an Elastic Load Balancer (ELB).

TR1.2 Implement real-time data replication and regular backups. Use Amazon RDS Multi-AZ deployments and automate backups for Amazon S3 using AWS Backup.

TR1.3 Distribute systems geographically and automate failover. Deploy resources across multiple AWS regions with Route 53 for DNS-based failover.

TR1.4 Utilize robust monitoring tools with automated alerting. Employ Amazon CloudWatch for monitoring and set up CloudWatch Alarms for automated alerts.

TR1.5 Implement auto-scaling to adapt to varying workloads. Use Auto Scaling Groups to automatically adjust the number of EC2 instances.

TR2.1. Implement data encryption at rest and in transit. Use Amazon S3 server-side encryption and enable SSL/TLS for data in transit.

TR2.2. Conduct regular security audits to identify vulnerabilities and ensure compliance. Use AWS Inspector for automated security assessments and compliance checks.

TR2.3. Enforce MFA for enhanced user authentication. Enable MFA for AWS Identity and Access Management (IAM) user accounts.

TR2.4. Implement network segmentation for access control. Use Virtual Private Cloud (VPC) and Security Groups to control network traffic.

TR2.5. Implement IAM for granular access control and least privilege. Use IAM policies to define and enforce fine-grained access controls.

TR3.1. Enforce MFA for enhanced user authentication. Enable MFA for AWS Identity and Access Management (IAM) user accounts.

TR3.2. Implement IAM for granular access control and least privilege. Use IAM policies to define and enforce fine-grained access controls.

TR3.3. Enforce strong password policies for user accounts. Use IAM to set and enforce password policies for AWS accounts.

TR4.1 Implement real-time monitoring of AWS resources, logs, and application performance using Amazon CloudWatch.

TR4.2 Aggregate and centralize logs for efficient troubleshooting and analysis. Use Amazon CloudWatch Logs for centralized log management and analysis.

TR4.3 Retain and analyze historical monitoring data for trend analysis and capacity planning. Use CloudWatch Metrics and CloudWatch Logs for historical data storage and analysis.

TR5.1 Implement automated and regular backups of critical data to prevent data loss. Use AWS Backup for automated backups and versioning to protect against accidental deletions.

TR5.2 Store backup data in geographically separated and secure locations for disaster recovery preparedness. Use Amazon S3 cross-region replication to create copies of data in different AWS regions.

TR5.3 Enable versioning for backup data to facilitate recovery to specific points in time. Leverage Amazon S3 versioning to maintain multiple versions of objects for data recovery flexibility.

TR6.1 Create and maintain thorough documentation covering infrastructure, configurations, and procedures. Use AWS Systems Manager Documents to document and automate operational procedures for AWS resources.

TR6.2 Implement version control for documentation to track changes and maintain accurate information. Use AWS CodeCommit for version control and documentation changes.

TR6.3 Provide clear and detailed documentation for APIs and SDKs to assist developers and integrators. Refer to the AWS API documentation and SDK developer guides for programming resources.

TR6.4 Conduct regular reviews of documentation to ensure accuracy, relevance, and completeness. Periodically review and update documentation to align with best practices of AWS Well-Architected Framework.

TR7.1 Establish data center presence in multiple regions to reduce latency for diverse user bases. Deploy resources across multiple AWS regions to bring services closer to end-users.

TR7.2 Employ load balancing to distribute traffic across servers and prevent latency issues. Use Elastic Load Balancing to distribute incoming application traffic across multiple targets for optimal performance.

TR7.3 Implement predictive scaling to anticipate demand and proactively allocate resources for low-latency responses. Use AWS Auto Scaling with predictive scaling to automatically adjust the number of EC2 instances based on anticipated demand.

TR8.1 Use AWS Identity and Access Management (IAM) to manage user and resource access, ensuring isolation in single tenancy environments.

TR8.2 Employ AWS Organizations to manage multiple AWS accounts for single and multi-tenancy configurations.

TR8.3 Implement AWS Virtual Private Cloud (VPC) with strict security groups and network access controls to isolate and secure single tenancy environments.

TR8.4 Use Amazon Elastic Kubernetes Service (EKS) for container orchestration to support both single and multi-tenancy deployment models.

TR8.5 Provide customers with options to choose their tenancy model during account setup, with clear documentation explaining the differences between single and multi-tenancy.

TR9.1 Integrate with AWS Identity and Access Management (IAM) to control customer access to billing and subscription data.

TR9.2 Develop a customer portal using AWS Amplify or Amazon API Gateway for customers to view and manage their subscriptions.

TR10.1 Use AWS Auto Scaling to dynamically adjust resources to meet performance demands.

TR10.2 Employ Amazon CloudWatch for real-time monitoring and alerting of system performance metrics.

TR10.3 Optimize database performance using Amazon RDS to ensure fast database access.

TR10.4 Implement AWS Performance Insights to analyze and optimize the database queries and transactions.

2.4 Tradeoffs

1. Isolation vs. Resource Efficiency
    a. Single tenancy offers isolation but may be less resource-efficient compared to multi-tenancy. Multi-tenancy is more cost-efficient but may have shared resource implications. Organizations need to strike a balance between the two based on customer needs.
2. Security vs. User-Friendliness

    a. Implementing stringent security measures can sometimes lead to a less user-friendly experience. Complex authentication processes or frequent security checks may inconvenience users.
    b. Increasing user authentication security, such as requiring multi-factor authentication, can enhance security but may make the login process more cumbersome for users.
3. Resource Utilization vs. Performance
    a. Implementing performance optimization measures might consume additional resources.
4. Cost
    a. Performance: Achieving low latency might involve using higher-cost infrastructure or content delivery networks.
    b. Data Integrity: Regular backups and robust disaster recovery systems can be costly but are essential for data integrity.
    c. Availability: Achieving higher availability often requires redundant systems and increased infrastructure costs. We may need to decide the appropriate level of availability based on budget constraints.
5. Resource Utilization
    a. Monitoring itself consumes resources; balancing the level of monitoring with system performance.
6. Alert Volume vs. Efficiency
    a. Receiving too many alerts for minor issues can overwhelm IT teams. Balancing the threshold for alerts to minimize false positives while not missing critical issues is important.

Tradeoff	**Description**	**Conflicting TRs**
Isolation vs Resource Efficiency	- Single tenancy offers isolation but may be less resource-efficient compared to multi-tenancy. - Multi-tenancy is more cost-efficient but may have shared resource implications. Organizations need to strike a balance between the two based on customer needs.	TR8.1, TR10.7
Security vs. User-Friendliness	- Implementing stringent security measures can sometimes lead to a less user-friendly experience. Complex authentication processes or frequent security checks may inconvenience users. - Increasing user authentication security, such as requiring multi-factor authentication, can enhance security but may make the login process more cumbersome for users.	TR2.3

Resource Utilization vs. Performance	- Implementing performance optimization measures might consume additional resources.	TR10.1 to TR10.7
Cost vs Performance and Latency	- Performance: Achieving low latency might involve using higher-cost infrastructure or content delivery networks. - Data Integrity: Regular backups and robust disaster recovery systems can be costly but are essential for data integrity. - Availability: Achieving higher availability often requires redundant systems and increased infrastructure costs. We may need to decide the appropriate level of availability based on budget constraints.	TR7.1 to 7.3, TR10.1 to 10.7
Resource Utilization	- Monitoring itself consumes resources; balancing the level of monitoring with system performance.	TR4.1 to 4.3, TR 10.7
Alert Volume vs. Efficiency	- Receiving too many alerts for minor issues can overwhelm IT teams. Balancing the threshold for alerts to minimize false positives while not missing critical issues is important.	TR4.1 to 4.3, TR10.5

3 Providers

3.1 Criteria for choosing a provider

#	**Criteria**	**TR**	
1	Availability	T.R1.1, TR1.2, TR1.3, TR1.4, TR1.5, TR7.1, TR7.2	The provider should ensure redundancy through load balancing, guaranteeing consistent traffic distribution for uninterrupted service. Offering data replication, backup mechanisms, and automated failover across multiple locations enhances overall system resilience. Comprehensive monitoring with automated alerting, coupled with adaptive features like auto-scaling, ensures optimal performance and responsiveness to varying workloads.
2	Data Replication and Backup	TR1.2, TR5.1, TR5.2, TR5.3	The provider should offer a solution for real-time data replication and regular automated backups to prevent data loss. This includes storing backup data in secure, geographically separated locations for effective disaster recovery preparedness. Additionally, the provider should enable versioning for backup data, allowing users to recover to specific points in time for increased flexibility.
3	Failover	TR1.2, TR1.3, TR7.1, TR7.2, TR7.3	The provider should offer a solution for real-time data replication and regular automated backups to ensure data integrity and facilitate failover. Geographical distribution of resources across multiple locations and the implementation of load balancing are essential for automated failover and latency prevention. Additionally, the provider should provide predictive scaling capabilities, anticipating demand and proactively allocating resources for optimal performance during failover scenarios.
4	Monitoring and Alerting	TR1.4, TR4.1, TR4.2, TR4.3	The provider should offer a robust monitoring solution with automated alerting, enabling real-time tracking of system metrics, logs, and application performance. Centralized log management and integration with notification systems should be provided for efficient troubleshooting and timely communication of critical

			alerts. Additionally, the solution must facilitate the retention and analysis of historical monitoring data, supporting trend analysis and capacity planning.
5	Auto-Scaling	TR1.5, TR7.2, TR7.3	The provider should offer an auto-scaling solution that dynamically adjusts resources to meet varying workloads, utilizing predictive scaling to anticipate demand and ensure proactive resource allocation for optimal performance. Additionally, the service must incorporate load balancing to evenly distribute traffic across servers, preventing latency issues and enhancing overall system responsiveness. This comprehensive auto-scaling offering ensures adaptability to changing demands and efficient resource utilization.
6	Disaster Recovery	TR1.2, TR5.2, TR5.3	The provider should offer real-time data replication and automated backups, ensuring data resilience. Additionally, they must store backup data in geographically separated and secure locations to enhance disaster recovery preparedness. Enabling versioning for backup data allows for flexible recovery to specific points in time, enhancing overall data recovery capabilities.
7	Data Encryption	TR2.1	The provider should offer comprehensive data encryption solutions, covering both data at rest and in transit. This includes robust mechanisms for server-side encryption of stored data and the implementation of SSL/TLS protocols for secure data transmission. Ensuring encryption at these critical points enhances overall data security.
8	Security Audits and Compliance	TR2.2, TR2.3, TR3.1, TR3.2, TR3.3	The provider should offer regular security audits and compliance checks to identify vulnerabilities, utilizing automated assessment tools. Additionally, they should provide multi-factor authentication (MFA) for enhanced user authentication, granular access controls through Identity and Access Management (IAM), and the enforcement of strong password policies to ensure robust security measures. These offerings collectively contribute to a secure and compliant environment.
9	Authentication and Access Control	TR3.1, TR3.2, TR3.3	The provider should offer enhanced user authentication by implementing multi-factor authentication (MFA). Additionally, they should provide a robust Identity and Access Management (IAM) system for granular access control, allowing organizations to enforce least privilege

			principles. The offering should also include the implementation and enforcement of strong password policies for user accounts, enhancing overall security.
10	Network Segmentation	TR8.3	The provider should support network segmentation and access control through features like Virtual Private Cloud (VPC) and Security Groups.
11	Single and Multi-Tenancy Support	TR8.1, TR8.2, TR8.3, TR8.4, TR8.5	The provider should offer robust support for both single and multi-tenancy environments, implementing comprehensive access management through identity and resource controls. This involves utilizing organizational structures for managing multiple accounts and deploying secure virtual private clouds (VPCs) with stringent security measures for single tenancy. Additionally, the provider should allow customers to choose their preferred tenancy model during account setup, providing clear documentation to elucidate the distinctions between single and multi-tenancy options.
12	Billing and Subscription Management	TR9.1, TR9.2	The provider should offer integration with identity and access management systems to control customer access to billing and subscription data. Additionally, they should provide a customer portal, facilitated through services like API Gateways, allowing customers to easily view and manage their subscriptions. This comprehensive offering enhances transparency and control over billing and subscription-related information for end-users.
13	Documentation and Support	TR6.1, TR6.2, TR6.3, TR6.4	The provider should offer comprehensive documentation covering infrastructure, configurations, and operational procedures, utilizing tools for automated documentation. This includes implementing version control to track changes and ensure accuracy. Additionally, the provider should furnish clear and detailed documentation for APIs and SDKs to facilitate developers and regularly review and update documentation to align with industry best practices.

3.2 Provider Comparison

We compared the cloud services from AWS, Google Cloud Platform, and Azure, all three providers offer flexible computing, storage, and networking, along with cloud benefits such as self-service, instant provisioning, and autoscaling.

We set the following baseline while comparing the three:

1. Region: US East- North Virginia
2. Operating System: Linux and Windows
3. vCPUs/Cores: 4

The criteria and comparison are as follows

a. Availability
    i. All three cloud providers provide robust availability and similar load balancing as well as scalability services.
b. Data Replication and Backup
    i. All three providers have data backup capabilities but Amazon Backup suits the needs of this project correctly with its advanced automation and centralized configuration capabilities compared to the other two providers.
c. Failover
    i. All three providers offer similar services for handling DNS failover. Overall, AWS is better integrated with its services.
d. Auto-Scaling
    i. All three providers provide equally capable auto-scaling services.
e. Disaster Recovery
    i. All three providers provide equally capable disaster recovery tools.
f. Data Encryption:
    i. All three providers provide equally capable data encryption services.
g. Security Audits and Compliance:
    i. AWS offers integration of IAM with MFA and the server-side encryption of S3 buckets. Azure and GCP offer similar services but are not as well-integrated as AWS.
h. Authentication and Access Control:
    i. All three providers offer reliable authentication and access control services.
i. Network Segmentation:
    i. All three providers have network segmentation services. AWS offers multi-region VPC and Security group deployments similar to services offered by the other two providers.
j. Single and Multi-Tenancy:
    i. All three providers offer single and multi-tenancy services

k. Billing and Subscription Management:
    i. All three providers offer similar services to integrate the customer query for personal data access. AWS integrates user requests with IAM.
l. Documentation and Support:
    i. Given that documentation is a fundamental part of any cloud operation, all three providers have created robust systems to provide documentation and support to their customers.
m. Monitoring and Alerting:
    i. AWS offers a more granular service that not only yields better service but also allows more operations to be done, for example, integration with SNS.

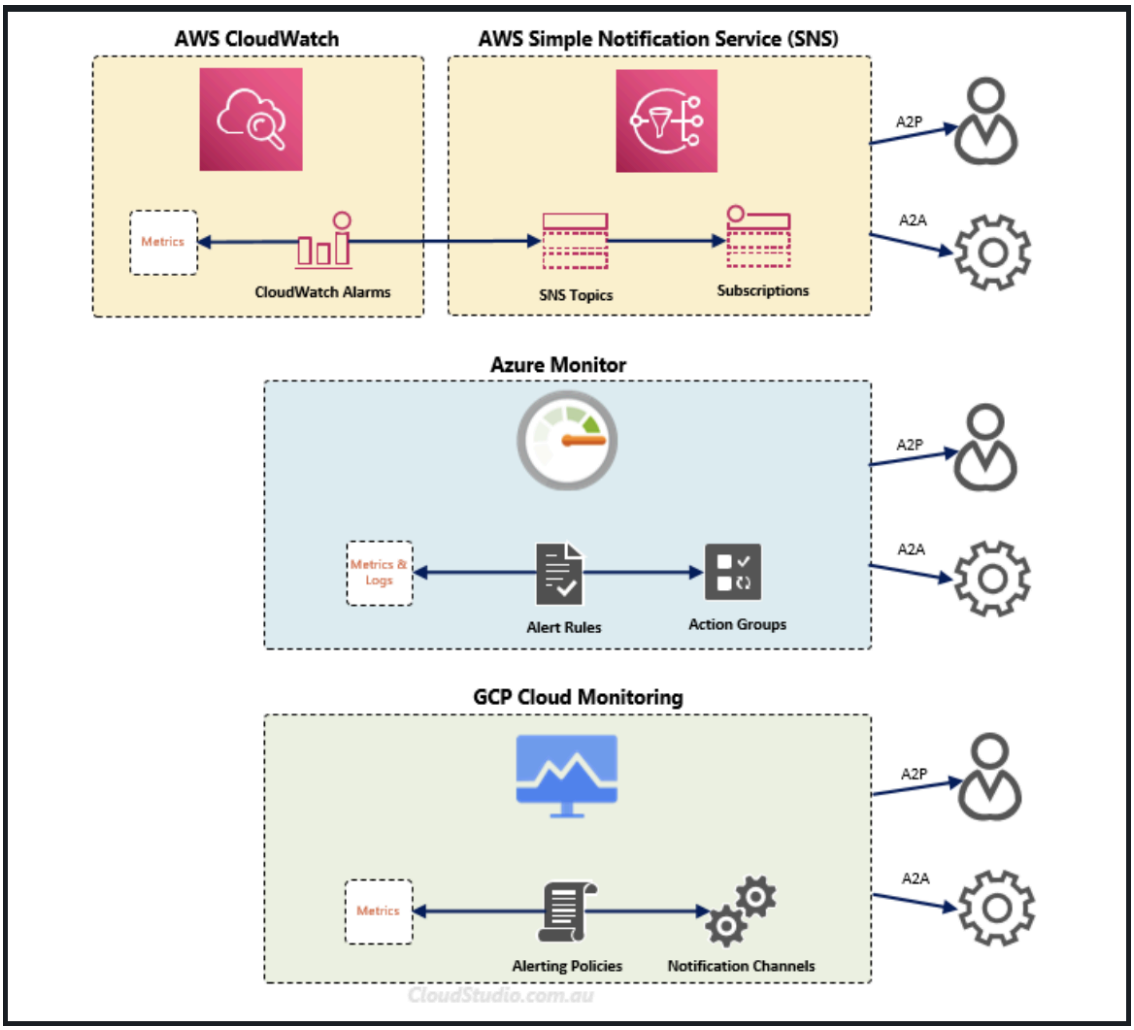

Fig 3.1 Comparison of monitoring offered by the big 3 providers

We rank the three services:

Rank	Provider	Reason
1	AWS	With its extensive global network of data centers, emphasizes breadth and depth in its service offerings, making it a preferred choice for many organizations. When compared to the other providers, it offers better monitoring tools. The rich array of tools, including databases, analytics, management, IoT, security, and enterprise applications, makes AWS the right solution for many teams.
2	Azure	Focuses on hybrid cloud solutions, especially advantageous for businesses with on-premises infrastructure. Azure also offers various services for enterprises, and Microsoft's long-standing relationship with this segment makes it an easy choice for some customers. Azure, Office 365, and Microsoft Teams enable organizations to provide employees with enterprise software while also leveraging cloud computing resources.
3	Google Cloud Platform	Distinguishes itself through its emphasis on innovation, particularly in open-source technologies like containers and its pivotal role in developing Kubernetes and Istio.

3.3 The final selection

We have chosen AWS for its wide array of service offerings, and also because we are more familiar with AWS than with any of the other providers.

3.3.1 The list of services offered by the winner

The list of services offered by AWS is mentioned in [5]. We will be using the following services by AWS as building blocks for our design.

#	Service	Details
1	EC2 (Elastic Compute Cloud)	Amazon EC2 provides scalable compute capacity in the cloud. It allows users to run virtual servers, known as instances, on-demand.
2	Elastic Load Balancer (ELB)	ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances, to ensure that

		no single instance is overwhelmed.
3	Amazon RDS (Relational Database Service)	RDS is a fully managed relational database service that simplifies database setup, operation, and scaling, while providing features like automated backups, Multi-AZ deployments for high availability, and automatic software patching.
4	Amazon S3 (Simple Storage Service)	Amazon S3 is a scalable object storage service designed to store and retrieve any amount of data. It offers high durability, availability, and security.
5	AWS Backup	AWS Backup is a centralized backup service that provides a unified backup console and supports automated backup schedules, retention policies, and data recovery.
6	Route 53	Amazon Route 53 is a scalable domain name system (DNS) web service. It translates human-readable domain names into IP addresses, and routes end-user requests.
7	Amazon CloudWatch	CloudWatch is a monitoring service that collects and tracks metrics, logs, and events from AWS resources and applications.
8	AWS Inspector	AWS Inspector is an automated security assessment service that helps users identify security vulnerabilities and compliance issues in their applications and infrastructure.
9	IAM (Identity and Access Management) and MFA	IAM is a service that enables users to manage access to AWS services securely. MFA adds an extra layer of security by requiring users to provide multiple forms of identification before gaining access to AWS resources.
10	VPC (Virtual Private Cloud)	VPC enables users to launch AWS resources into a virtual network. It provides control over the virtual networking environment, including IP address ranges, subnets, and security groups.
11	AWS Well-Architected Framework	The AWS Well-Architected Framework provides best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud.
12	AWS Auto Scaling (Predictive Scaling)	AWS Auto Scaling automatically adjusts the number of EC2 instances based on predicted demand, helping maintain performance and cost efficiency.
13	Amazon API Gateway	Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and

		secure APIs at any scale.
14	AWS Amplify	AWS Amplify is a development platform that enables developers to build full-stack serverless and static web applications with continuous deployment.
15	Amazon Elastic Kubernetes Service (EKS)	Amazon EKS is a managed Kubernetes service that simplifies the deployment, management, and scaling of containerized applications using Kubernetes.

4 Design

4.1 The basic building blocks of the design

#	AWS Service	TR
1	EC2 (Elastic Compute Cloud)	TR1.1, TR1.3, TR7.1
2	Elastic Load Balancer (ELB)	TR1.1, TR7.2
3	Amazon S3 (Simple Storage Service)	TR1.2, TR10.3;
4	AWS Backup	TR1.2, TR2.1, TR5.2, TR5.3
5	Route 53	TR1.2, TR5.3
6	Amazon CloudWatch	TR1.3
7	Auto Scaling Groups	TR1.4, TR4.1, TR4.2, TR4.3, TR10.2, TR10.6
8	SSL/TLS	TR10.1, TR7.3
9	AWS Inspector	TR2.2
10	IAM (Identity and Access Management)	TR2.3, TR2.4, TR2.5, TR3.1, TR3.2, TR3.3, TR8.1, TR9.1
11	VPC (Virtual Private Cloud)	TR2.4, TR8.3
12	Security Groups	TR2.4, TR8.3
13	AWS Well-Architected Framework	TR6.4
14	Amazon Elastic Kubernetes Service (EKS)	TR8.4
15	AWS Amplify	TR9.1

4.2 Top-level, informal validation of the design

The proposed cloud architecture design for the LocalBazaar platform exhibits a robust foundation, meticulously addressing the diverse needs of small businesses. High Availability is ensured through the deployment of EC2 instances across multiple Availability Zones coupled with an Elastic Load Balancer (ELB), guaranteeing 24/7 accessibility. Security and Compliance

are meticulously handled through data encryption at rest and in transit using Amazon S3 server-side encryption and SSL/TLS. Regular security audits using AWS Inspector and the implementation of Multi-Factor Authentication (MFA) bolster user authentication.

User Management and Authentication are governed by granular access control and least privilege principles with AWS Identity and Access Management (IAM) policies, reinforcing the security posture. Monitoring and Alerting are facilitated in real-time using Amazon CloudWatch for system metrics, logs, and application performance, while Disaster Recovery and Backup leverage Amazon RDS Multi-AZ and AWS Backup for real-time data replication and automated backups. The documentation and support framework, established through AWS Systems Manager Documents and version control with AWS CodeCommit, ensures a comprehensive resource for ongoing maintenance and troubleshooting.

The design further excels in addressing Low Latency concerns, adopting a geographical distribution strategy across multiple AWS regions, employing Elastic Load Balancing for traffic distribution, and utilizing AWS Auto Scaling for predictive scaling to meet varying demand. The incorporation of Hybrid Tenancy managed through AWS IAM, Organizations, and VPC, caters to the diverse needs of both single and multi-tenancy environments. Billing and Subscription Management are streamlined with customer portals developed using AWS Amplify or Amazon API Gateway, ensuring a seamless experience for users. High Performance is achieved through dynamic resource adjustment, real-time monitoring, database optimization, caching, and regular load testing and performance tuning, showcasing a comprehensive and finely tuned architecture for LocalBazaar.

4.3 Action items and rough timeline (SKIPPED)

5 Well-Architected Framework

5.1 Use of the Well-Architected Framework

1. **Operational Excellence:** Focuses on optimizing operational processes through automation, documentation, and continuous improvement, ensuring efficient system management and incident response.
2. **Security:** Emphasizes the implementation of robust security measures, including encryption, access management, and threat detection, to safeguard data, systems, and assets.
3. **Reliability:** Aims to design fault-tolerant systems that can recover from failures, meet availability expectations, and ensure a reliable response to incidents through testing and monitoring.
4. **Performance Efficiency:** Involves optimizing resource utilization and system performance by adopting scalable architectures, efficient data handling, and responsive design.
5. **Cost Optimization:** Focuses on avoiding unnecessary costs, maximizing resource efficiency, and continuously monitoring and optimizing spending to align with business needs and achieve optimal value from AWS investments.

5.2 Discussion of pillars

1. Operational Excellence:

   Operational Excellence, as outlined in the AWS Well-Architected Framework, is manifested in the technical requirements mentioned in this document. The technical requirements advocate for redundancy and load balancing, exemplified in the deployment of EC2 instances across multiple Availability Zones with an Elastic Load Balancer (ELB). This ensures operational resilience and efficient distribution of workloads. The emphasis on real-time data replication, regular backups, and automated backups using Amazon RDS Multi-AZ and Amazon Backup aligns with operational best practices, facilitating data integrity and recovery. Furthermore, the geographical distribution of resources across AWS regions and the use of predictive scaling with AWS Auto Scaling demonstrate a commitment to proactive resource allocation and system responsiveness, contributing to overall operational efficiency.

2. Security:

   The Security pillar in the AWS Well-Architected Framework is closely aligned with the technical requirements and tradeoffs in the cloud architecture design. Technical requirements emphasize data encryption at rest and in transit using Amazon S3 server-side encryption and SSL/TLS, demonstrating a commitment to safeguarding data. The enforcement of Multi-Factor Authentication (MFA) for enhanced user authentication aligns with the tradeoff between security and user-friendliness. While MFA enhances

security, it introduces an additional layer of complexity for users, striking a balance between heightened security and user convenience.

Additionally, the technical requirement of conducting regular security audits using AWS Inspector showcases a proactive approach to identifying vulnerabilities and ensuring compliance. This aligns with the tradeoff of investing in security measures versus user-friendliness, emphasizing the importance of a secure infrastructure. The implementation of network segmentation for access control using Virtual Private Cloud (VPC) and Security Groups, along with IAM for granular access control and least privilege, illustrates a commitment to robust access management. These measures contribute to the isolation of resources in single-tenancy environments, addressing the tradeoff between isolation and resource efficiency. In summary, the technical requirements in the design document embody the Security pillar by prioritizing measures that enhance data protection, user authentication, and overall system security.

5.3 Architecture Diagram

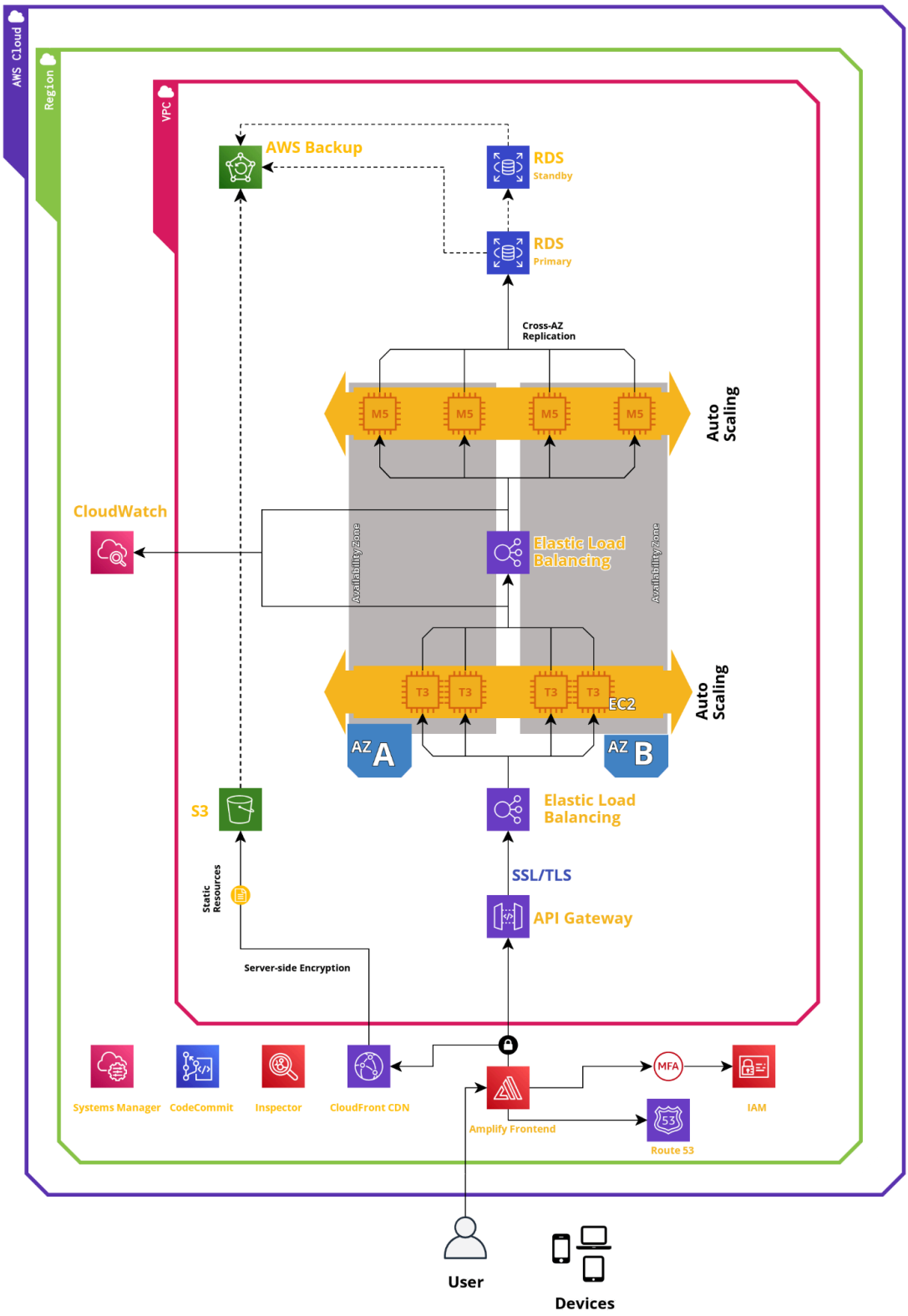

Fig. 5.1 Architecture Diagram (AWS)

5.4 Validation of the design

BR1 - High Availability: The implementation of redundant servers, load balancers, and the deployment of EC2 instances across multiple Availability Zones with an Elastic Load Balancer (ELB) ensures 24/7 availability, addressing the BR of uninterrupted services for customers.

TR1.1 - This requirement is met by deploying EC2 instances across multiple Availability Zones (AZs) and using an Elastic Load Balancer (ELB). EC2 instances are distributed across different AZs to enhance fault tolerance, ensuring that if one AZ experiences an issue, the application can continue running in other AZs. The ELB evenly distributes incoming traffic among the EC2 instances, promoting scalability and preventing any single server from becoming a bottleneck. This setup ensures high availability and reliability, meeting the business requirement of 24/7 service availability.

TR1.2 - This requirement is addressed by utilizing Amazon RDS Multi-AZ deployments for real-time data replication and automated backups using AWS Backup for Amazon S3. Multi-AZ deployments in Amazon RDS automatically replicate databases across multiple Availability Zones to provide failover support, ensuring continuous data availability. Automated backups with AWS Backup facilitate regular backups, versioning, and secure storage, reducing the risk of data loss and aligning with the business requirement for disaster recovery and backup.

TR1.3 - Geographical distribution is achieved by deploying resources across multiple AWS regions. Using Route 53 for DNS-based failover enables automatic redirection of traffic to a healthy endpoint in the event of a regional failure, ensuring continuous service availability. This setup supports the business requirement for operational continuity and resilience in case of failures.

TR1.4 - To meet the requirement for robust monitoring, the design employs Amazon CloudWatch. It monitors AWS resources, application performance, and logs in real-time. CloudWatch Alarms are configured to automatically send alerts when predefined thresholds are exceeded or anomalies are detected. This proactive approach ensures that the IT team can identify and address issues before they escalate into critical problems, aligning with the business requirement for timely alerts and issue resolution.

TR1.5 - The design incorporates Auto Scaling Groups to dynamically adjust the number of EC2 instances based on varying workloads. This enables the system to scale up during high demand and scale down during periods of lower demand, optimizing resource utilization and performance. This aligns with the business requirement for adapting to varying workloads and delivering efficient services.

BR2 - Security and Compliance: By enforcing Multi-Factor Authentication (MFA), implementing data encryption at rest and in transit, and conducting regular security audits using AWS Inspector, the design robustly safeguards sensitive data, intellectual property, and customer

information. These security measures not only meet the security requirements but also foster customer trust and loyalty.

TR2.1 - The design ensures data security by implementing encryption measures. Amazon S3 server-side encryption is employed for data at rest, and SSL/TLS is enabled for data in transit. These measures adhere to the business requirement of safeguarding sensitive data from theft, breaches, or unauthorized access.

TR2.2 - The security requirement is met through regular security audits using AWS Inspector. This tool conducts automated security assessments, identifying vulnerabilities and ensuring compliance with security standards. This proactive approach aligns with the business requirement of implementing measures to safeguard data and intellectual property.

TR2.3 - Multi-Factor Authentication (MFA) is enforced for enhanced user authentication, requiring an additional layer of verification beyond passwords. This security measure aligns with the business requirement of ensuring secure user authentication and protecting customer information.

TR2.4 - Network segmentation is achieved through the use of Virtual Private Cloud (VPC) and Security Groups, controlling network traffic and enhancing access control. This aligns with the business requirement of implementing measures to safeguard data and ensure secure access to resources.

TR2.5 - Identity and Access Management (IAM) is employed for granular access control and enforcing the principle of least privilege. IAM policies define and enforce fine-grained access controls, aligning with the business requirement of securing user access to resources.

BR3 - User Management and Authentication: The design incorporates IAM for granular access control, least privilege principles, and MFA, enabling users to securely access their resources, aligning perfectly with the user management and authentication requirements.

TR3.1 - Multi-Factor Authentication (MFA) is enforced for enhanced user authentication, requiring an additional layer of verification beyond passwords. This security measure aligns with the business requirement of ensuring secure user authentication and protecting customer information.

TR3.2 - Identity and Access Management (IAM) is employed for granular access control and enforcing the principle of least privilege. IAM policies define and enforce fine-grained access controls, aligning with the business requirement of securing user access to resources.

TR3.3 - Strong password policies are enforced for user accounts using IAM, setting and enforcing password policies for AWS accounts. This measure enhances security and aligns with the business requirement of implementing measures to safeguard data and intellectual property.

BR4 - Monitoring and Alerting: The technical requirements pertaining to real-time monitoring with Amazon CloudWatch and automated alerting through CloudWatch Alarms directly address the need for timely alerts and proactive issue resolution, aligning with the BR of avoiding downtime.

TR4.1 - Real-time monitoring is achieved through Amazon CloudWatch, which monitors AWS resources, logs, and application performance. This aligns with the business requirement of ensuring efficient system management and incident response through continuous monitoring.

TR4.2 - Logs are aggregated and centralized using Amazon CloudWatch Logs for efficient troubleshooting and analysis. This proactive approach aligns with the business requirement of continuous improvement in operational workflows.

TR4.3 - Historical monitoring data is retained and analyzed using CloudWatch Metrics and CloudWatch Logs for trend analysis and capacity planning. This ensures informed decision-making and aligns with the business requirement of optimizing operational processes.

BR5 - Disaster Recovery and Backup: The implementation of real-time data replication, automated backups using Amazon RDS Multi-AZ, and S3 backups with versioning ensures operational continuity, resilience, and security in the face of failures, aligning with disaster recovery and backup requirements.

TR5.1 - Automated and regular backups of critical data are implemented using AWS Backup, protecting against data loss and accidental deletions. Versioning is utilized to maintain data recovery flexibility, meeting the business requirement for disaster recovery and backup.

TR5.2 - Backup data is stored in geographically separated and secure locations using Amazon S3 cross-region replication. This ensures disaster recovery preparedness and aligns with the business requirement for operational continuity and resilience.

TR5.3 - Versioning is enabled for backup data using Amazon S3 versioning, facilitating recovery to specific points in time and providing data recovery flexibility. This aligns with the business requirement for disaster recovery and backup.

BR6 - Documentation and Support: The use of AWS Systems Manager Documents, version control with AWS CodeCommit, and regular documentation reviews exhibit a commitment to comprehensive documentation practices, fulfilling the BR of establishing and upholding thorough documentation encompassing infrastructure, configurations, and operational procedures.

TR6.1 - Thorough documentation covering infrastructure, configurations, and procedures is created and maintained. AWS Systems Manager Documents are utilized to document and automate operational procedures for AWS resources, aligning with the business requirement of comprehensive documentation practices.

TR6.2 - Version control for documentation is implemented using AWS CodeCommit, ensuring accurate information and tracking changes over time. This aligns with the business requirement of maintaining accurate documentation.

TR6.3 - Clear and detailed documentation for APIs and SDKs is provided, referencing the AWS API documentation and SDK developer guides. This facilitates developers and integrators in understanding and utilizing the system, aligning with the business requirement of supporting seamless integration.

TR6.4 - Regular reviews of documentation are conducted to ensure accuracy, relevance, and completeness. Documentation is periodically updated to align with the best practices of the AWS Well-Architected Framework, ensuring continuous improvement.

BR7 - Low Latency: The design achieves low latency through geographical distribution, load balancing with Elastic Load Balancing, and predictive scaling with AWS Auto Scaling, addressing the BR of ensuring faster and more responsive user experiences.

TR7.1 - Data center presence is established in multiple regions, reducing latency for diverse user bases. Resources are deployed across multiple AWS regions to bring services closer to end-users, meeting the business requirement for low-latency experiences.

TR7.2 - Load balancing is employed using Elastic Load Balancing to distribute traffic across servers, preventing latency issues. This ensures optimal performance and aligns with the business requirement for low-latency responses.

TR7.3 - Predictive scaling is implemented using AWS Auto Scaling to anticipate demand and proactively allocate resources for low-latency responses. This aligns with the business requirement for delivering low-latency experiences based on anticipated demand.

BR8 - Hybrid Tenancy: The use of AWS IAM, Organizations, VPC, and EKS supports both single and multi-tenancy options, providing flexibility for high-profile customers and cost-effective resource sharing, as outlined in the BR for supporting diverse deployment models.

TR8.1 - IAM is used to manage user and resource access, ensuring isolation in single tenancy environments. This aligns with the business requirement for supporting both single and multi-tenancy deployment options.

TR8.2 - AWS Organizations are employed to manage multiple AWS accounts, facilitating single and multi-tenancy configurations. This supports the business requirement for flexibility in deployment models.

TR8.3 - VPC is implemented with strict security groups and network access controls, isolating and securing single tenancy environments. This ensures security and aligns with the business requirement for secure access control.

TR8.4 - Amazon Elastic Kubernetes Service (EKS) is used for container orchestration, supporting both single and multi-tenancy deployment models. This aligns with the business requirement for flexibility in deployment options.

TR8.5 - Customers are provided with options to choose their tenancy model during account setup, and clear documentation is available to explain the differences between single and multi-tenancy. This ensures transparency and supports the business requirement for customer choice.

BR9 - Billing and Subscription Management: Integration with AWS IAM for access control and the development of a customer portal using AWS Amplify or Amazon API Gateway fulfill the BR of implementing a robust billing and subscription management system. This allows customers transparent access to billing information and easy management of their subscriptions.

TR9.1 - Integration with IAM is implemented to control customer access to billing and subscription data. This aligns with the business requirement for secure access control to sensitive customer information.

TR9.2 - A customer portal is developed using AWS Amplify or Amazon API Gateway, providing customers with the ability to view and manage their subscriptions. This meets the business requirement for implementing a robust billing and subscription management system.

BR10 - High Performance and Efficiency: The implementation of AWS Auto Scaling, real-time monitoring with Amazon CloudWatch, database performance optimization with Amazon RDS, caching with Amazon Elasticache, and performance analysis with AWS Performance Insights collectively ensure high performance, low latency, and efficient resource utilization, meeting the BR for delivering optimal performance and efficiency.

TR10.1 - AWS Auto Scaling is employed to dynamically adjust resources based on performance demands. Auto Scaling Groups are configured to automatically scale the number of EC2 instances up or down, ensuring optimal resource utilization and responsiveness to varying workloads. This aligns with the technical requirement of implementing auto-scaling to adapt to varying workloads.

TR10.2 - Amazon CloudWatch is utilized for real-time monitoring and alerting of system performance metrics. CloudWatch provides comprehensive insights into AWS resources, logs, and application performance. Alarms are configured to automatically trigger alerts when predefined thresholds are exceeded, enabling proactive issue resolution. This fulfills the technical requirement for robust monitoring tools with automated alerting.

TR10.3 - Amazon RDS is used to optimize database performance, ensuring fast database access. Multi-AZ deployments are implemented to enhance fault tolerance, and

automated backups are set up to prevent data loss. This aligns with the technical requirement of optimizing database performance to meet performance demands.

TR10.4 - AWS Performance Insights is implemented to analyze and optimize database queries and transactions. This tool provides a detailed view of database performance, allowing for the identification and resolution of bottlenecks. The insights gained from Performance Insights contribute to ongoing efforts to improve the efficiency and responsiveness of the system, aligning with the technical requirement of analyzing and optimizing database queries and transactions.

5.5 Design principles and best practices used

Business requirements:

1. **High Availability:**

   Alignment with Principles: The defense-in-depth approach enhances the system's resilience, contributing to 24/7 availability as required by the business.

   Cloud Best Practice: Continuous refinement and evolution align with the business need for a system that can adapt to changing requirements and challenges, contributing to long-term availability.

2. **Security and Compliance:**

   Alignment with Principles: The defense-in-depth strategy ensures robust security measures, addressing the need for safeguarding sensitive data as per security and compliance requirements.

   Cloud Best Practice: The architecture's adaptability supports ongoing compliance efforts as regulatory landscapes change, aligning with cloud best practices for security and compliance.

Technical Requirements:

1. **Use Redundant Servers and Load Balancers:**

   Alignment with Principles: Redundancy is a component of defense in depth, ensuring that even if one server fails, others can handle the load.

Cloud Best Practice: The continuous evolution principle supports the ongoing optimization of server configurations for efficiency and performance.

2. **Implement Real-Time Data Replication and Regular Backups:**

   Alignment with Principles: Data protection is part of the defense-in-depth approach, and regular backups align with the always be architecting principle for continuous improvement.

   Cloud Best Practice: Adaptable backup strategies consider changes in data volume and ensure data recovery flexibility, meeting technical and business requirements.

5.6 Tradeoffs revisited

Isolation vs. Resource Efficiency:
In the design, the tradeoff between isolation and resource efficiency is addressed by providing both single tenancy and multi-tenancy deployment options. Single tenancy offers heightened isolation, crucial for high-profile customers with specific security needs. On the other hand, multi-tenancy is leveraged for resource efficiency, catering to customers who prioritize cost-effectiveness. The tradeoff here aligns with the need to balance isolation for security-conscious clients and resource efficiency for those seeking cost optimization. The document "Even Swaps" emphasizes the importance of considering alternatives and making rational trade-offs, which is evident in the strategic choice between single and multi-tenancy.

Security vs. User-Friendliness:
The design acknowledges the tradeoff between security and user-friendliness by implementing strong security measures while striving to maintain a positive user experience. For instance, multi-factor authentication (MFA) is enforced to enhance security, but measures are taken to streamline the login process to minimize user inconvenience. This reflects a thoughtful trade-off, considering both security imperatives and the need for a user-friendly interface. The "Even Swaps" approach of rational decision-making supports this trade-off, emphasizing the importance of carefully evaluating the costs and benefits of different options.

Resource Utilization vs. Performance:
To address the tradeoff between resource utilization and performance, the design incorporates dynamic resource adjustment with AWS Auto Scaling. While optimization measures may consume additional resources, the trade-off is mitigated by the ability to scale resources based on demand. This aligns with the principles of making rational trade-offs and considering

alternatives, as highlighted in "Even Swaps." The design aims to achieve an optimal balance between resource utilization and performance to meet varying workloads efficiently.

Cost:
The tradeoff between achieving low latency (performance) and the associated higher costs is considered in the design. For example, the use of higher-cost infrastructure or content delivery networks may be necessary to meet performance goals. This decision aligns with the idea of rational decision-making advocated in "Even Swaps," where the trade-off involves carefully evaluating the costs and benefits of different choices, taking into account the specific requirements and constraints.

Availability vs. Cost:
The tradeoff between achieving higher availability and the associated infrastructure costs is managed by implementing redundant systems and failover mechanisms. The design aligns with the principles of the "Even Swaps" method, which emphasizes the need to weigh alternatives and make rational choices. The appropriate level of availability is determined based on budget constraints, demonstrating a deliberate trade-off between system robustness and cost-effectiveness.

Resource Utilization:
The trade-off between resource utilization and the consumption of resources for monitoring is considered. Balancing the level of monitoring with system performance is crucial to prevent resource exhaustion. This trade-off reflects a thoughtful approach to optimizing resource utilization while ensuring effective system monitoring. "Even Swaps" principles align with this by encouraging a rational evaluation of trade-offs to achieve the best overall outcome.

Alert Volume vs. Efficiency:
The trade-off between alert volume and efficiency is addressed by setting thresholds for alerts to minimize false positives. This strategic decision aligns with the principles of rational decision-making advocated in "Even Swaps," where trade-offs involve careful consideration of the trade-offs between alert volume and operational efficiency. Balancing the alert threshold aims to avoid overwhelming IT teams while ensuring critical issues are promptly addressed.

5.7 Discussion of an alternate design (SKIPPED)

6 Experimental Validation

6.1 Experiment Design

Technical Requirements considered during the experiment design

TR	Details
TR1.1	Use redundant servers and load balancers. Deploy EC2 instances in multiple Availability Zones with an Elastic Load Balancer (ELB).
TR1.5	Implement auto-scaling to adapt to varying workloads. Use Auto Scaling Groups to automatically adjust the number of EC2 instances.
TR7.2	Employ load balancing to distribute traffic across servers and prevent latency issues. Use Elastic Load Balancing to distribute incoming application traffic across multiple targets for optimal performance.
TR8.4	Use Amazon Elastic Kubernetes Service (EKS) for container orchestration to support both single and multi-tenancy deployment models.
TR10.1	Use AWS Auto Scaling to dynamically adjust resources to meet performance demands.
TR10.5	Implement AWS Performance Insights to analyze and optimize the database queries and transactions.

Experiment

Goal: To validate our design - specifically the TRs mentioned above.

Components:

1. Flask Application

   A basic Flask app which serves the following pages:

   i. Login
   ii. Login with a specific user
   iii. Logout
   iv. Index

2. Dockerfile

   A Dockerfile is created to containerize the Flask application. The container uses the python:3.9-slim base image, installs Python dependencies from requirements.txt, and exposes port 80.

```
Unset
#Dockerfile
FROM python:3.9-slim

COPY . .

RUN pip3 install --upgrade pip
RUN pip3 install --no-cache-dir -r requirements.txt

EXPOSE 80

WORKDIR src

#Execute Flask Server
ENTRYPOINT ["python3"]
CMD ["server.py"]
```

3. Docker Image
   The Dockerfile is used to build a local Docker image. We name this image local_bazaar_api
4. MiniKube
   A tool that allows local testing and deployment of Kubernetes clusters. We installed and started MiniKube and brought up its dashboard.
5. Kubernetes Deployment
   We defined the following YAML files -
   a. deployment.yaml

```
Unset
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: localbazaar-app
spec:
 replicas: 3
 selector:
  matchLabels:
   app: localbazaar
 template:
  metadata:
   labels:
    app: localbazaar
```

```
  spec:
   containers:
   - name: localbazaar-container
    image: local_bazaar_api:latest
    imagePullPolicy: Never
    ports:
    - containerPort: 80
    volumeMounts:
    - name: localbazaar-storage
     mountPath: /app/data
```

b. service.yaml

```
Unset
# service.yaml
apiVersion: v1
kind: Service
metadata:
 name: localbazaar-service
spec:
 selector:
  app: localbazaar
 ports:
  - protocol: TCP
   port: 80
   targetPort: 80
 type: LoadBalancer
```

c. persistent-volume.yaml

```
Unset
# persistent-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
 name: localbazaar-pv
spec:
 capacity:
  storage: 1Gi
 accessModes:
  - ReadWriteOnce
```

```
 hostPath:
  path: "/app/data"
```

d. persistent-volume-claim.yaml

```
Unset
# persistent-volume-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: localbazaar-pvc
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 1Gi
```

To access the Flask Server within the Kubernetes cluster, we enabled port forwarding.

MiniKube Dashboard

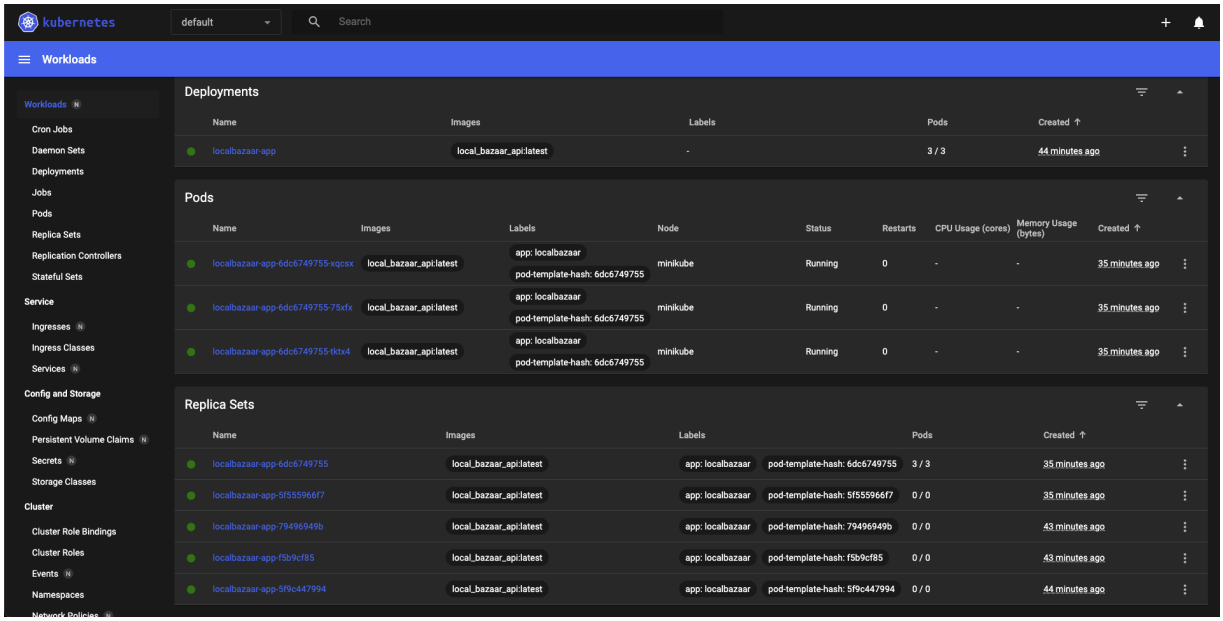

Fig. 6.1 MiniKube Cluster deployment with 3 Pods.

Output Expectations

Load Balancer service should adapt to varying workloads. Since MiniKube does not provide Auto Scaling, we will manually scale and check if the traffic is distributed and latency improves accordingly. The Kubernetes cluster should distribute the number of requests received equally across all Pods.

6.2 Workload generation with Locust

We use Locust to simulate multiple users accessing the Flask application concurrently.

```python
from locust import HttpUser, task, between
class LocalBazaarApp(HttpUser):
    wait_time = between(1,5)
    @task(4)
    def get_index(self):
        self.client.get("/")
    @task(2)
    def get_login(self):
        self.client.get("/login")
    @task(3)
    def get_login_with_user(self):
        self.client.get("/login/CSC547")
    @task(1)
    def get_logout(self):
        self.client.get("/logout")
    @task(2)
    def get_test(self):
        self.client.get("/test")
```

Test cases

Users	Spawn Rate	Clusters
100	1	1
1000	10	3
1000	10	5
10000	50	3
10000	50	5

6.3 Analysis of the results

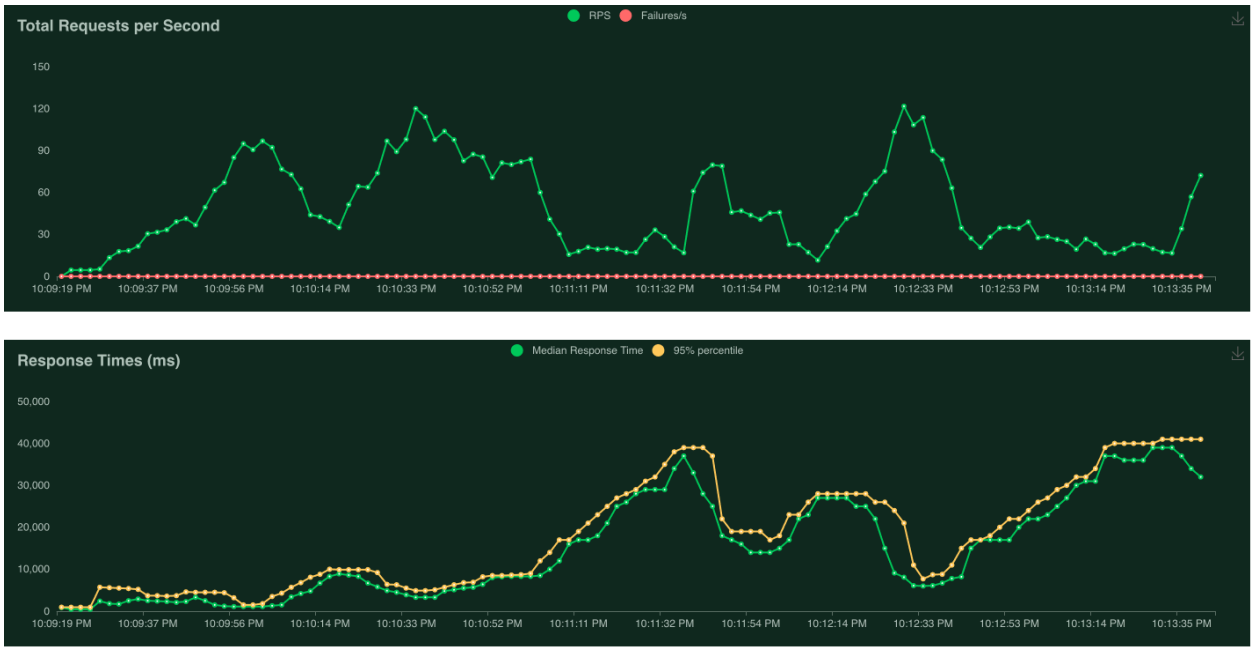

Fig 6.2 Number of requests per second at varying loads and its respective response time

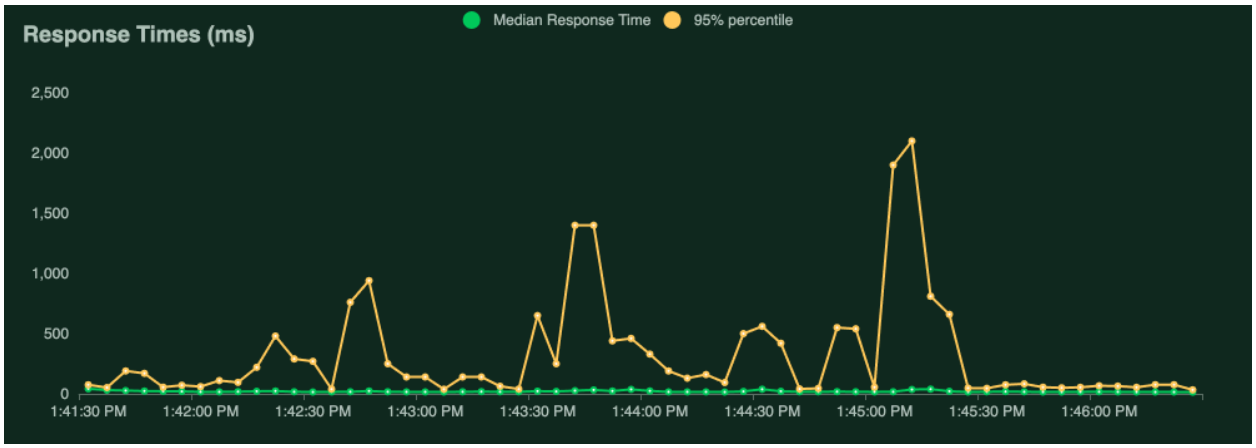

Fig 6.3 Response times for varying users and spawn rates.

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	5783	0	75	9	2563	5	19.3	0.0
GET	/login	2953	0	73	8	2303	6	9.8	0.0
GET	/login/CSC547	4322	0	74	9	2388	14	14.4	0.0
GET	/logout	1370	0	75	9	2430	4	4.6	0.0
GET	/test	3012	0	73	9	2347	4	10.0	0.0
	Aggregated	**17440**	**0**	**74**	**8**	**2563**	**7**	**58.1**	**0.0**

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	17	21	26	37	130	370	1400	2600
GET	/login	18	22	27	39	140	360	1100	2300
GET	/login/CSC547	18	21	27	38	130	360	1300	2400
GET	/logout	17	20	26	36	100	350	1400	2400
GET	/test	17	20	26	37	130	380	1100	2300
	Aggregated	**18**	**21**	**26**	**37**	**130**	**370**	**1300**	**2600**

Fig 6.3 Request statistics for 10,000 users

The test results show the cluster's ability to distribute the load relatively evenly. As the user count escalates to 10,000 with three pods, a discernible uptick in response times is observed. This momentary increase in response times is effectively mitigated when the cluster is scaled up to five pods. This demonstrates the potential effect of an efficient scaling mechanism that manages increased loads and optimizes response times.

The test results validate the design decisions of the selected TRs using the Flask application deployed on Minikube. The moderate increase in response times under heavy load suggests that the system handles normal loads well but may encounter scalability challenges during peak demand, aligning with the design expectations. The steady increase in the request rate and subsequent plateau indicates a reasonable level of scalability within current configurations, prompting further investigation into potential bottlenecks. While the low error rate under normal and moderate load demonstrates effective handling of most requests, occasional spikes in latency suggest areas for optimization. The successful auto-scaling behavior and the system's ability to recover well after peak load validate the stability and resilience of the design choices, providing valuable insights for further refinement and optimization.

7 Ansible Playbooks (SKIPPED)

8 Demo (SKIPPED)

9 Comparisons (SKIPPED)

10 Experience & Future Scope

10.1 The lessons learned

The project as a whole gave us a good overview of the different cloud components used in real world cloud projects. Some sections challenged us to evolve our understanding of what it means to create a good cloud architecture. For example, comparing tradeoffs helped us understand how to weigh limitations and find the best way out. Creating the Architecture diagram made us really introspect on the intricate interconnections and integrations among different components within the AWS services suite. Using Locust and MiniKube helped us learn performance testing and evaluate the effects of scaling and load balancing.

10.2 Possible continuation of the project

This design document is a good starting point for building a cloud based architecture that deploys a small ecommerce fullstack application.
Possible avenues for future work -

1. Develop the application
2. Deployment on AWS Infrastructure.
3. Use AWS Kubernetes.
4. Experiment in AWS instead of locally.

References

[1] YV and YP, “ECE547/CSC547 class notes”.

[2] YV and YP, “ECE547/CSC547 references”

[3] Gilad David Maayan, “Assessing Cloud Backup Solutions: AWS vs. Azure vs. Google Cloud,” https://dzone.com/articles/assessing-cloud-backup-solutions-aws-vs-azure-vs-g

[4] Laurent Gil, “Cloud Pricing Comparison: AWS vs. Azure vs. Google Cloud Platform in 2023,” https://cast.ai/blog/cloud-pricing-comparison-aws-vs-azure-vs-google-cloud-platform/

[5] “AWS Cloud Products,” https://aws.amazon.com/products/?nc2=h_ql_prod_fs_f&aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc&awsf.re%3AInvent=*all&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all

[6] “Comparing AWS, Azure, GCP,” https://www.digitalocean.com/resources/article/comparing-aws-azure-gcp

[7] “Compare AWS, Microsoft Azure and Google Cloud for Backup,” https://www.msp360.com/resources/blog/comparison-aws-azure-google/

[8] “Cloud DNS management comparison: Alibaba, Amazon, Google, IBM, Microsoft,” https://www.dailyhostnews.com/cloud-dns-providers-comparison

[9] Cody Slingerland, “The Ultimate Cloud Storage Pricing Comparison,” https://www.cloudzero.com/blog/cloud-storage-pricing/

[10] Our friend, ChatGPT