EXPERT INSIGHT

Solutions Architect's Handbook

Kick-start your career as a solutions architect by learning architecture design principles and strategies

Foreword by:
Rajesh Sheth
General Manager, Messaging and Streaming, AWS
Rohan Karmarkar
Director, Solutions Architecture, AWS

Second Edition

Saurabh Shrivastava
Neelanjali Srivastav

Packt>

Solutions Architect's Handbook

Second Edition

Chapter 1
The Meaning of Solution Architecture

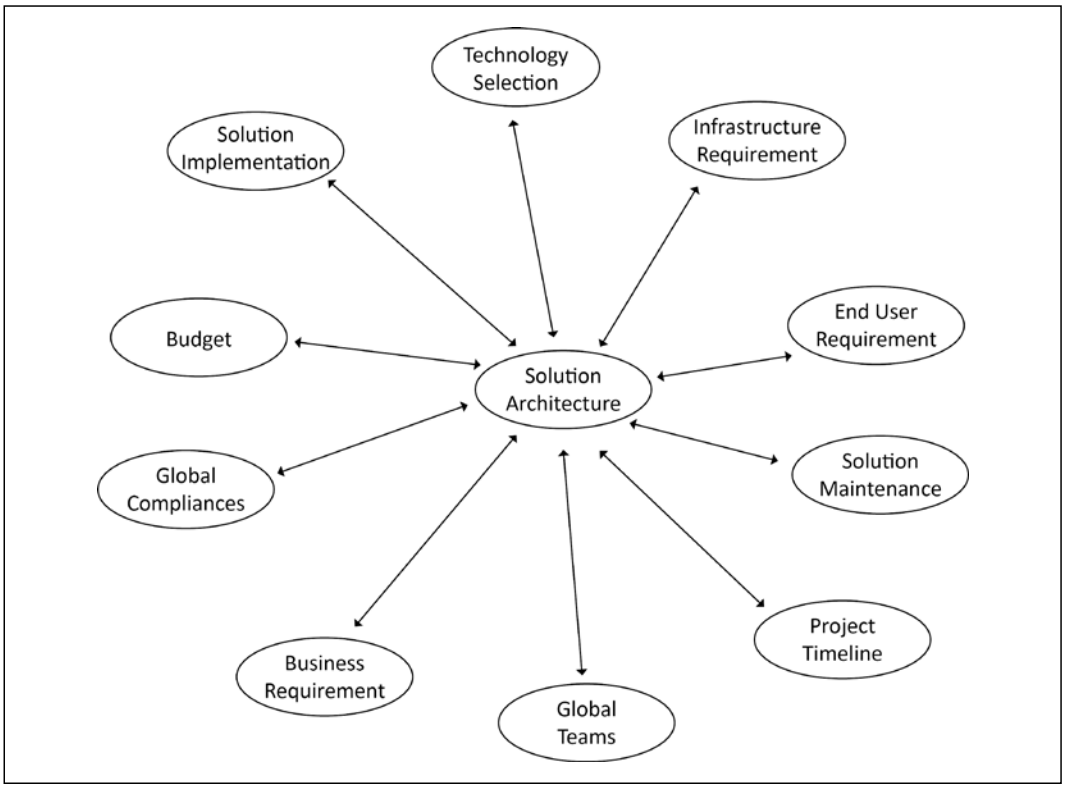


Figure 1.1: Circle of solution architecture

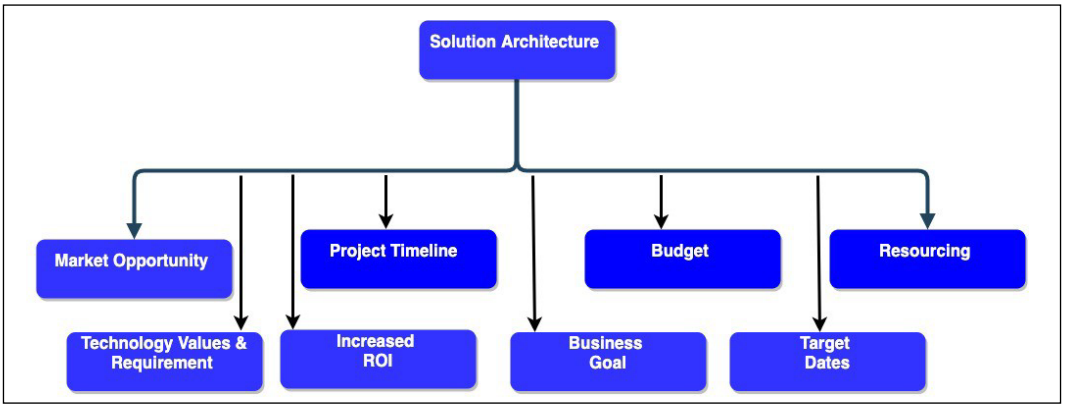


Figure 1.2: A solution architecture's beneficial attributes

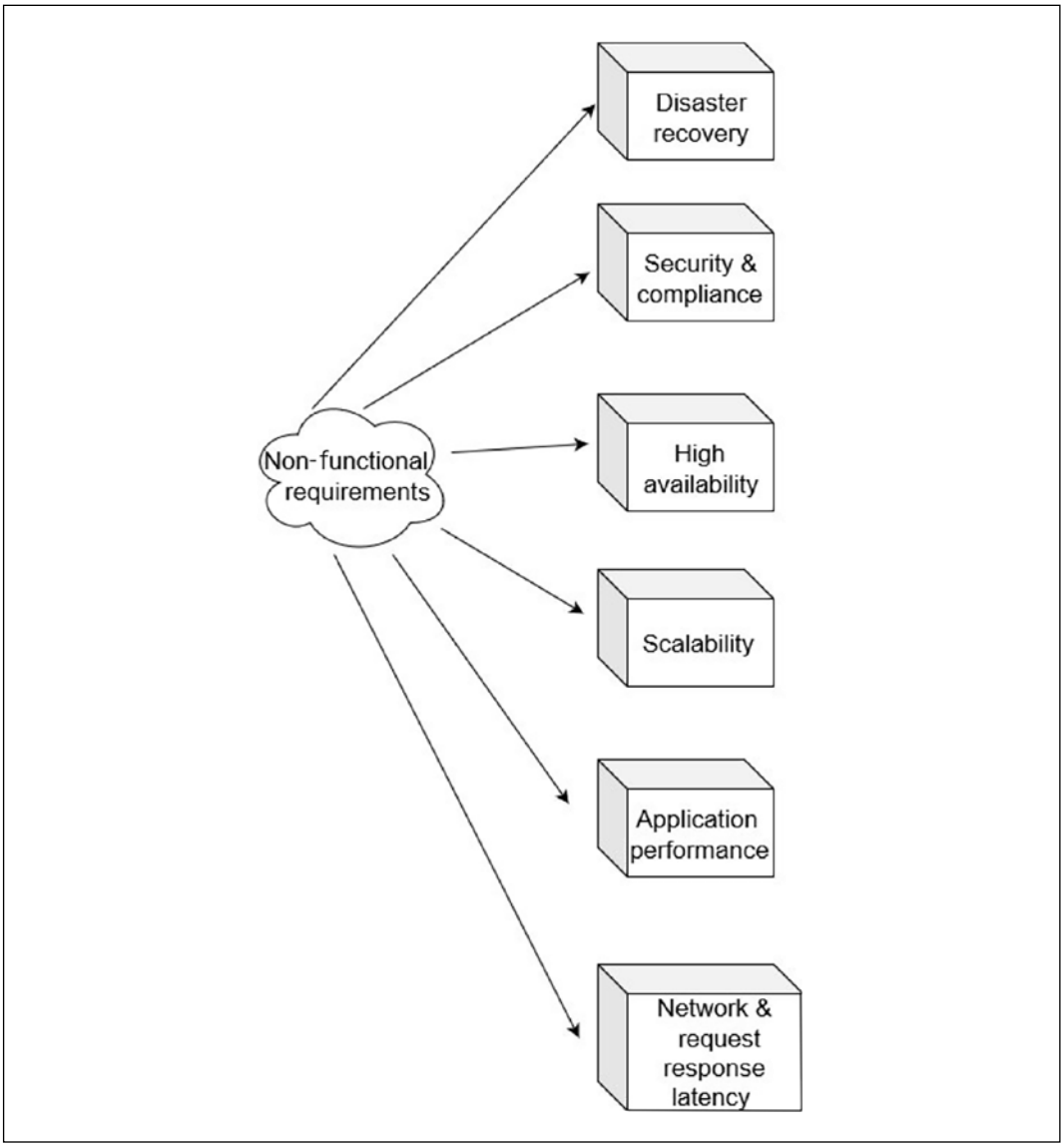


Figure 1.3: Non-functional attributes of solution architecture

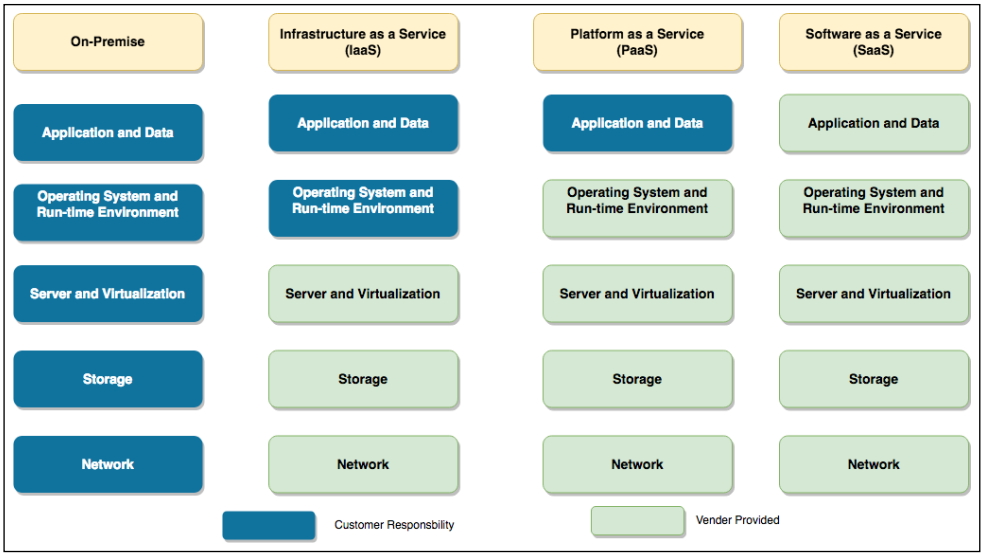


Figure 1.4: Types of cloud computing models

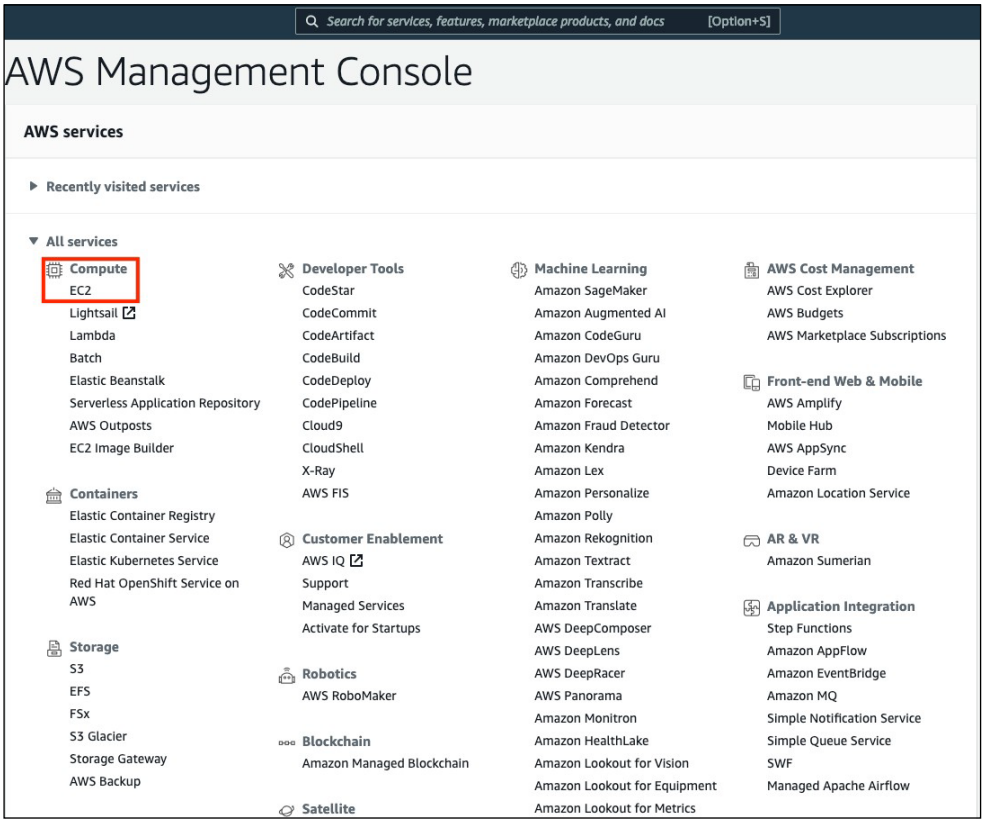


Figure 1.5: AWS console and service offerings

Chapter 2

Solution Architects in an Organization

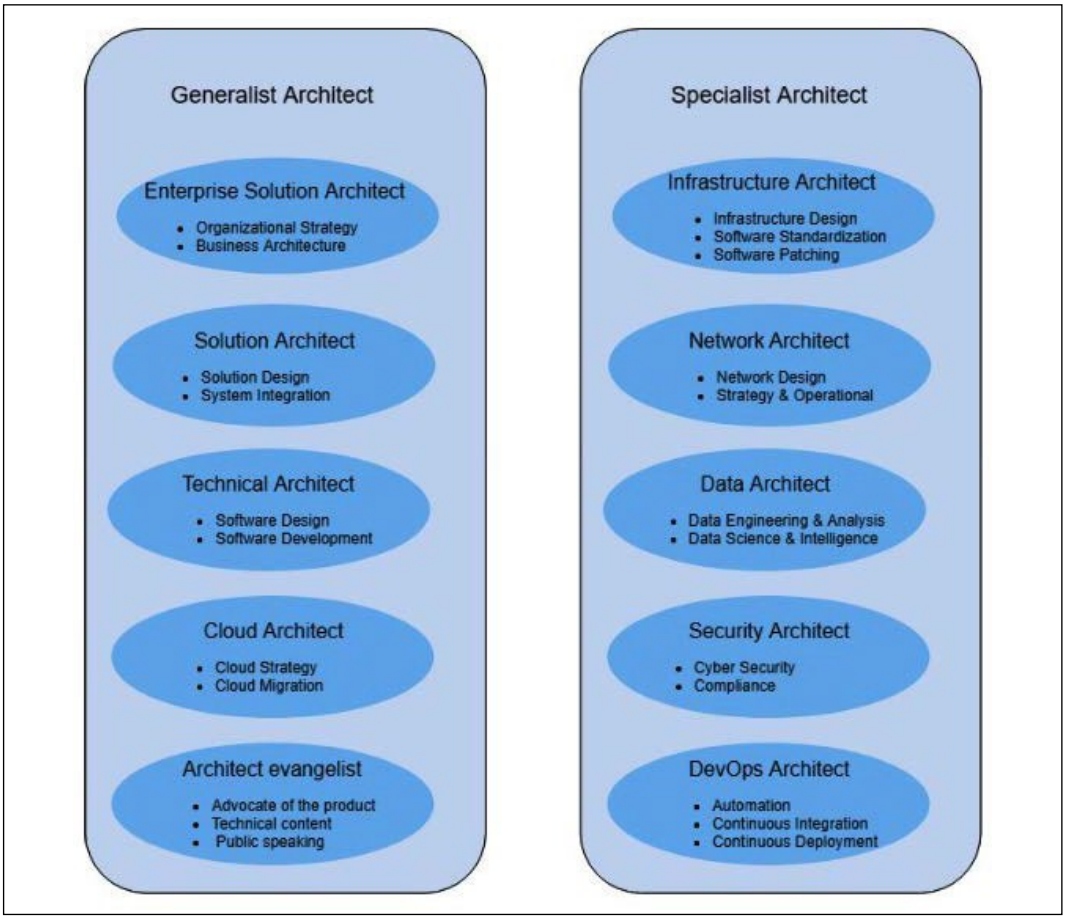


Figure 2.1: Types of solution architect

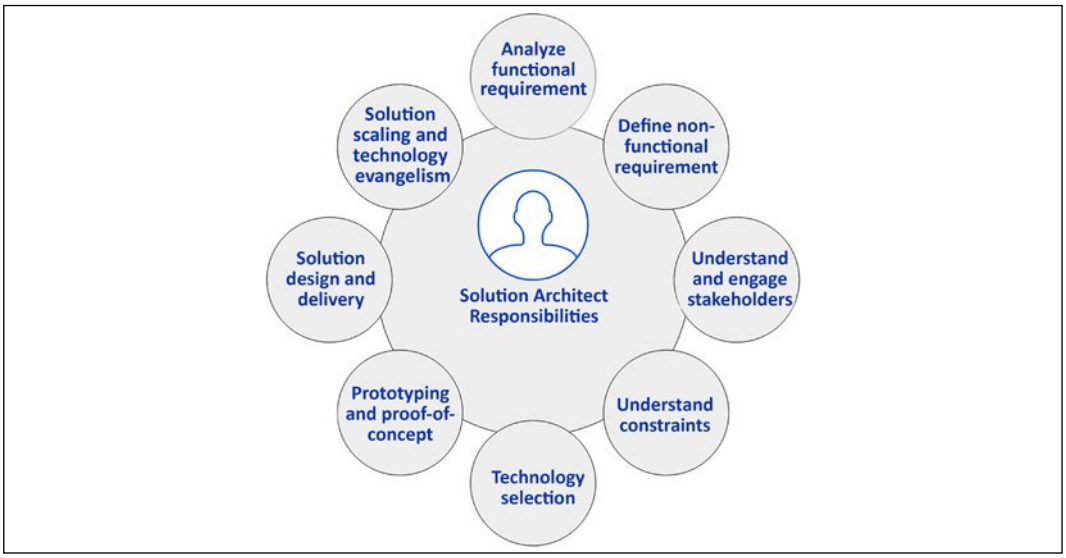


Figure 2.2: Solution architect's responsibility model

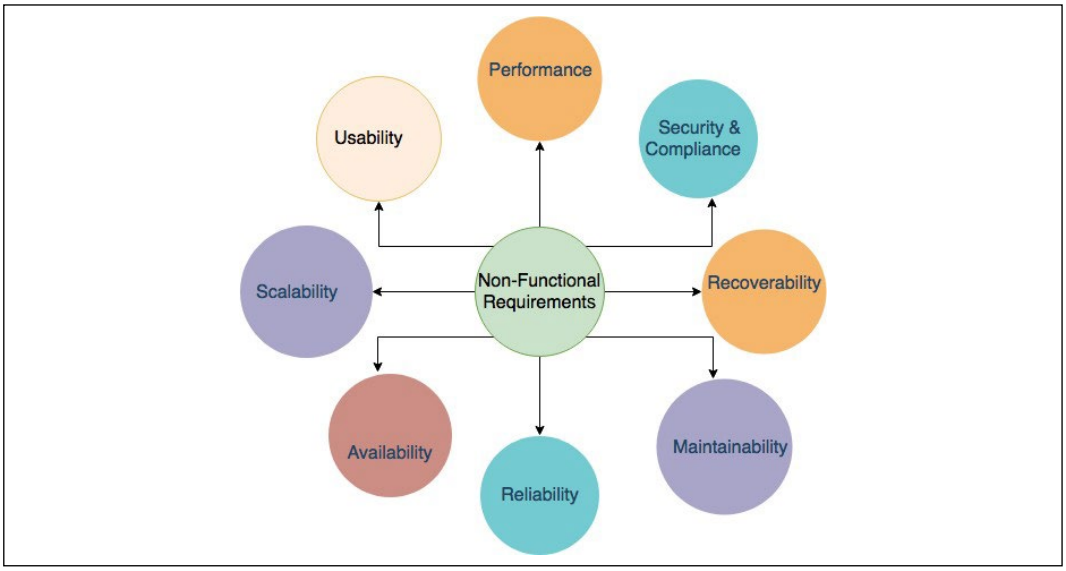


Figure 2.3: NFRs in a solution design

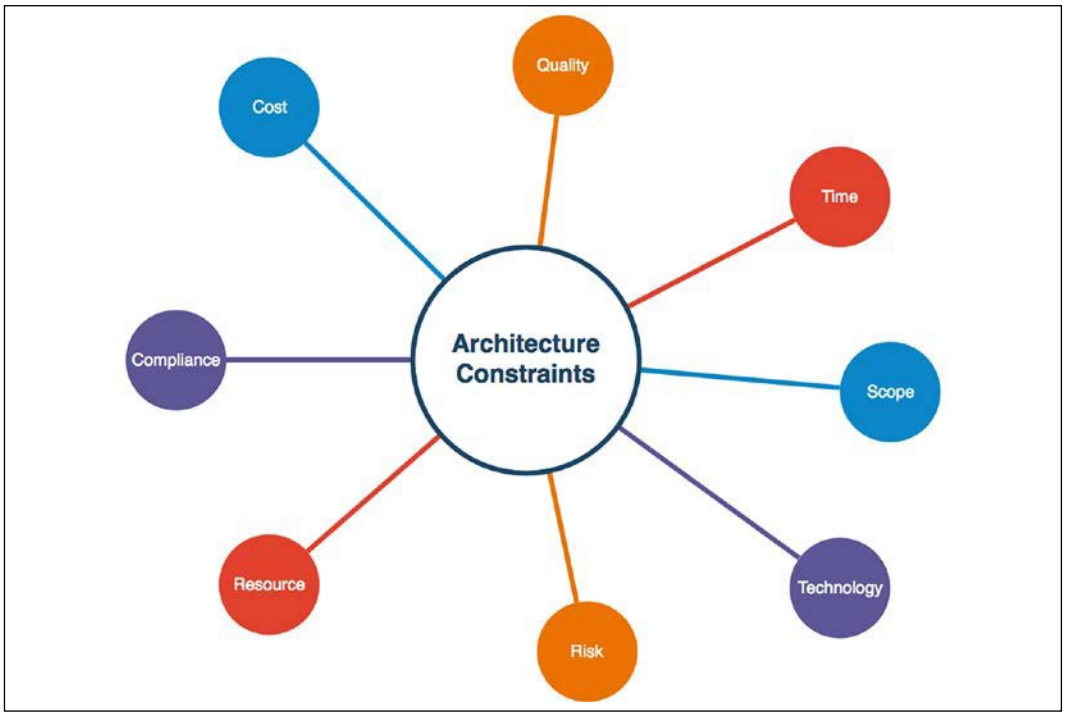


Figure 2.4: Architectural constraints in a solution design

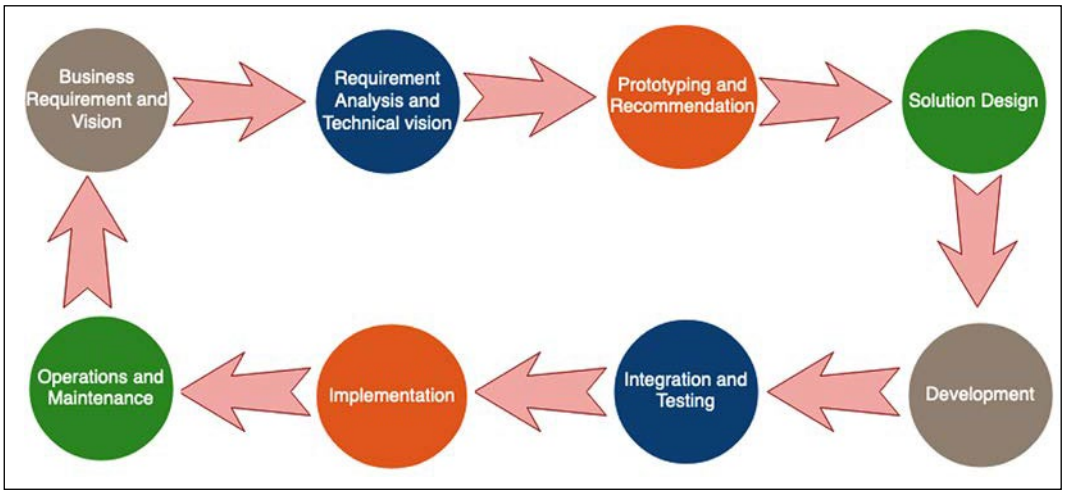


Figure 2.5: Solution delivery life cycle

Chapter 3

Attributes of the Solution Architecture

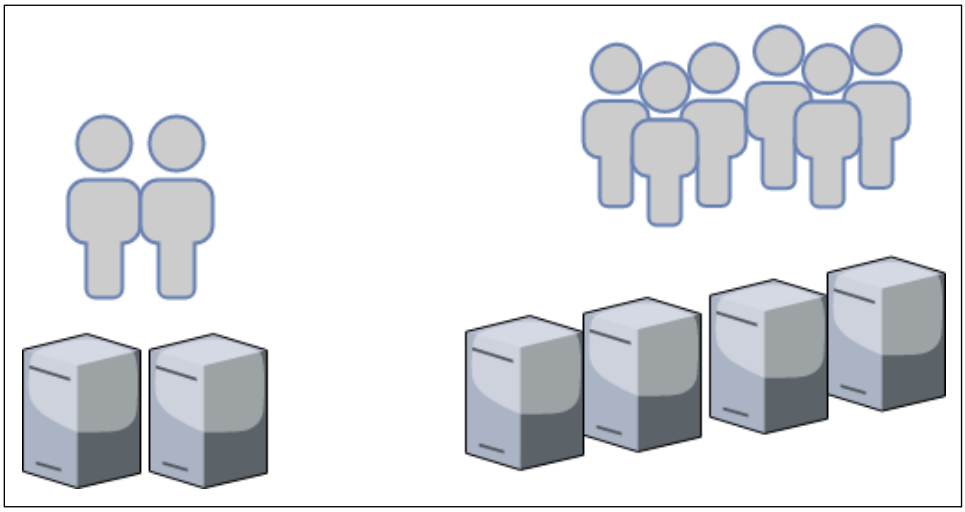


Figure 3.1: Horizontal scaling

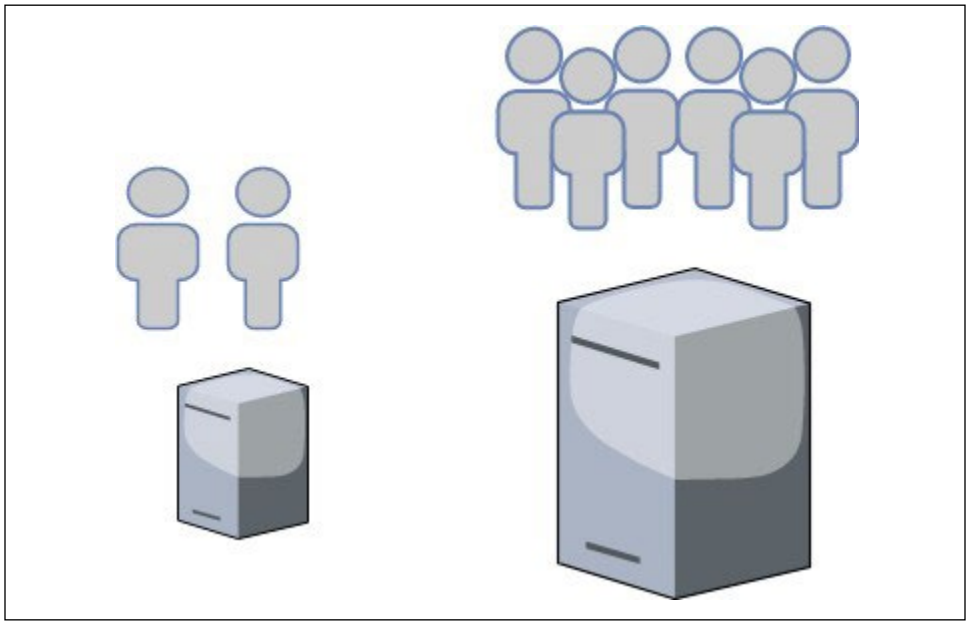


Figure 3.2: Vertical scaling

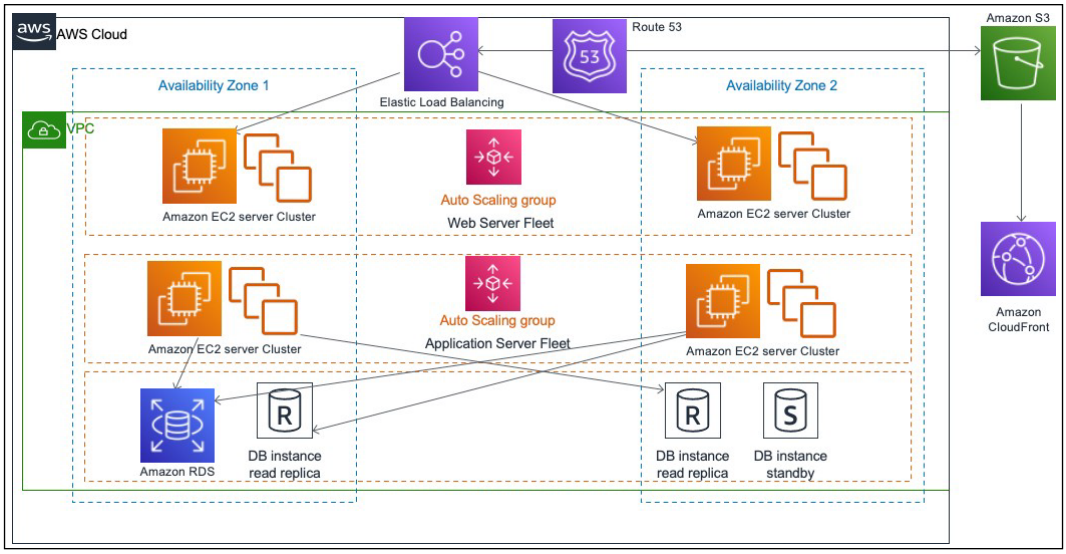


Figure 3.3: Scaling three-tier architecture

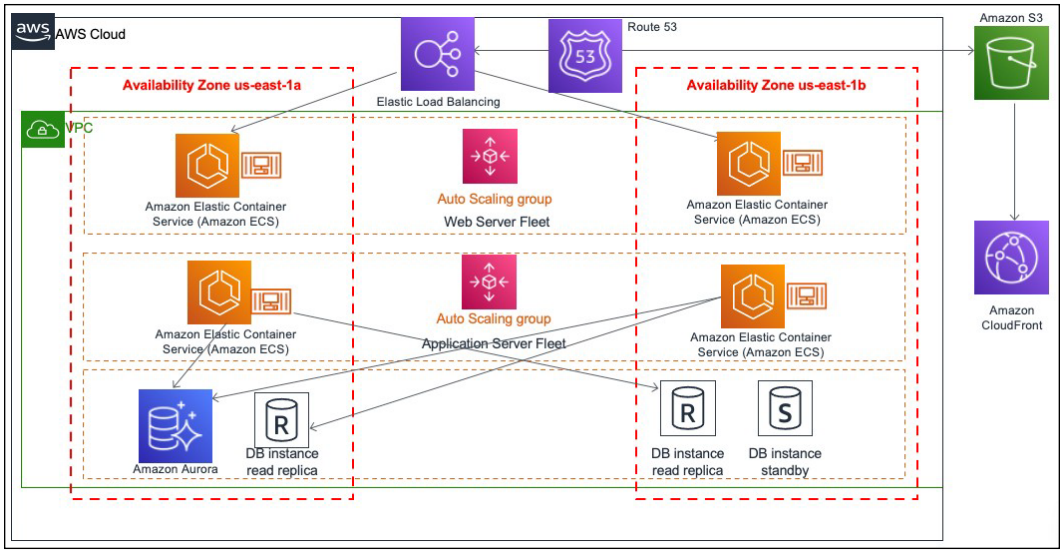


Figure 3.4: High availability and resilience architecture

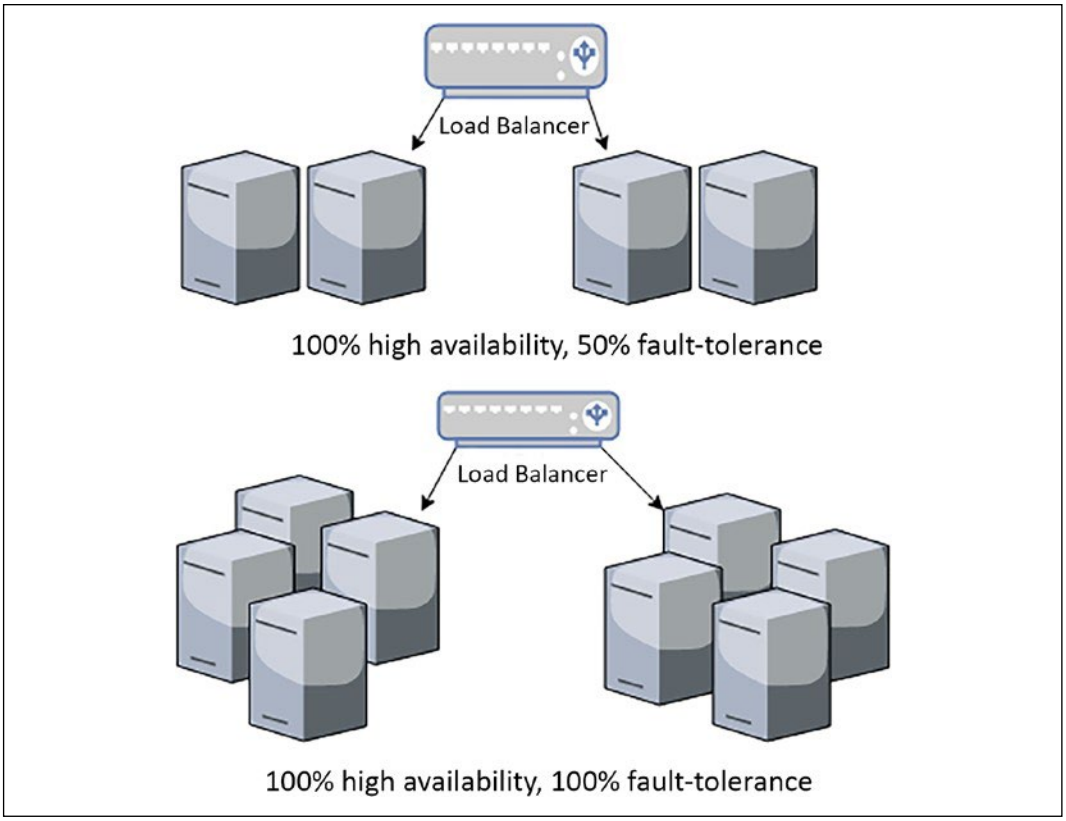


Figure 3.5: Fault-tolerance architecture

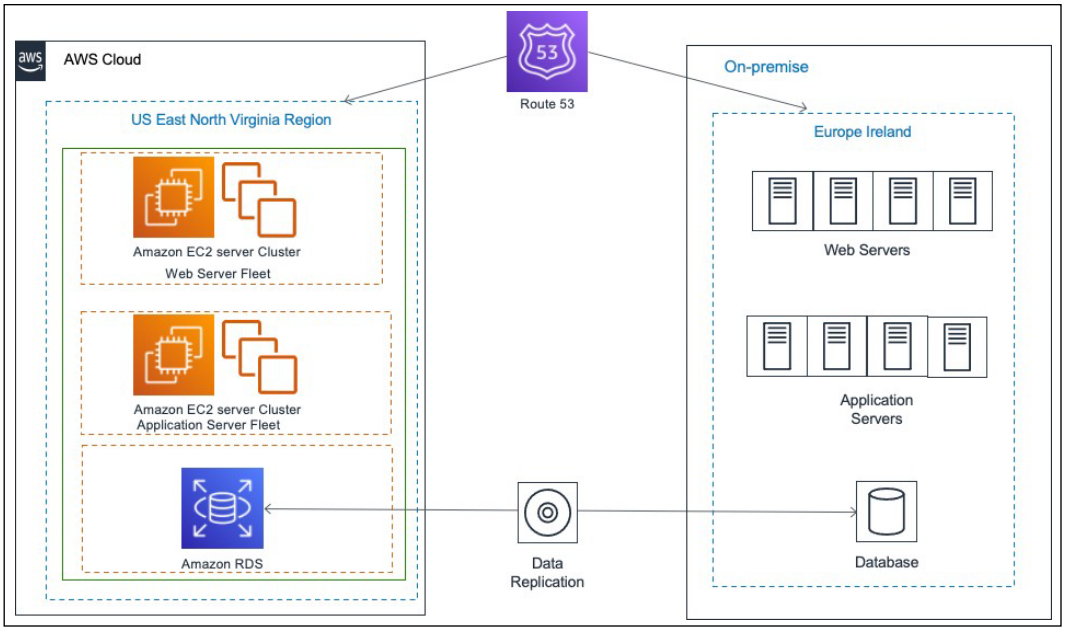


Figure 3.6: Hybrid multi-site disaster recovery architecture

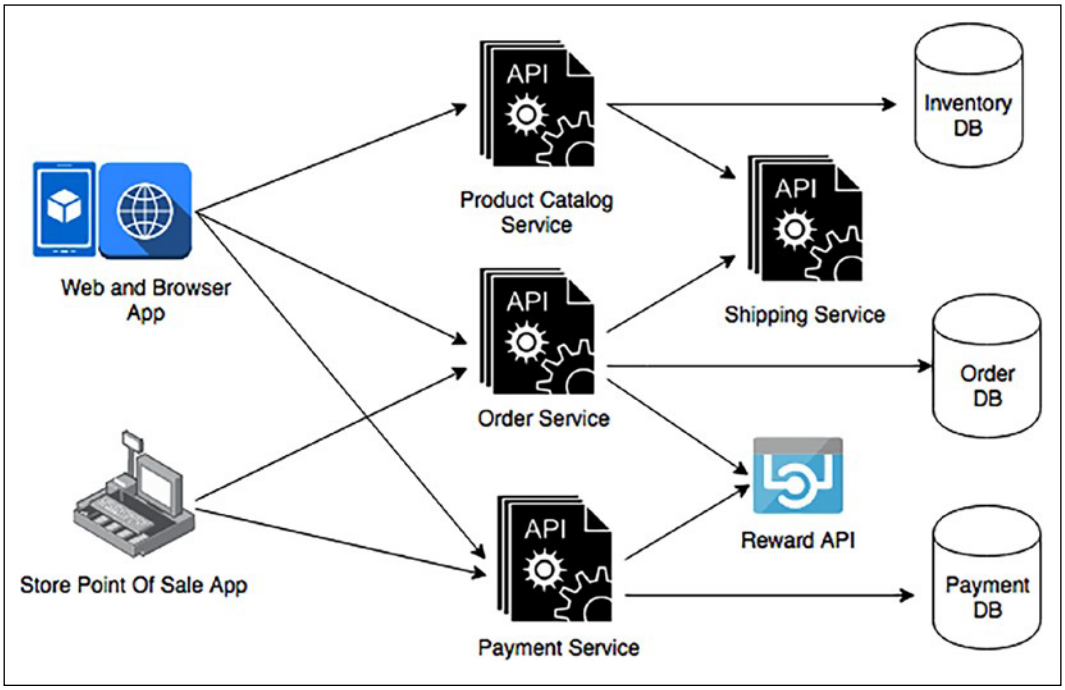


Figure 3.7: Extensible API-based architecture

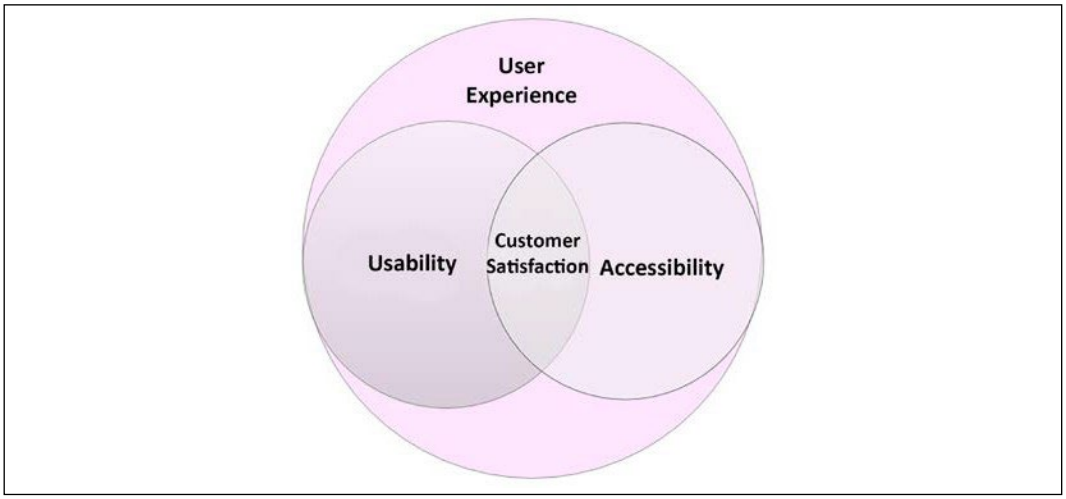


Figure 3.8: Customer satisfaction with usability and accessibility

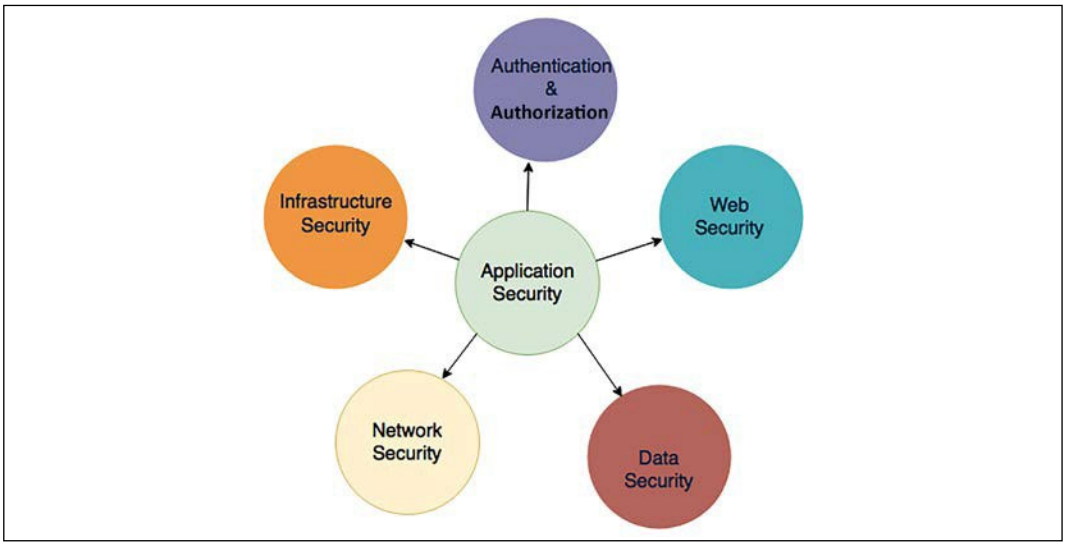


Figure 3.9: Security aspects in solution design

Chapter 4
Principles of Solution Architecture Design

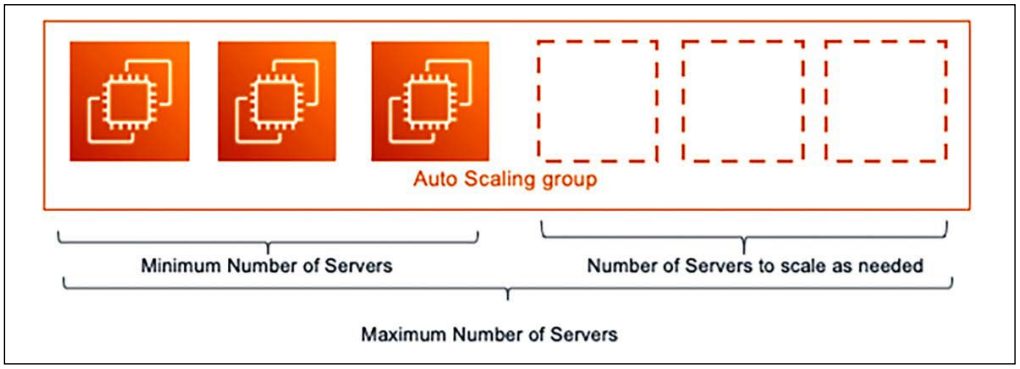


Figure 4.1: Server Auto Scaling

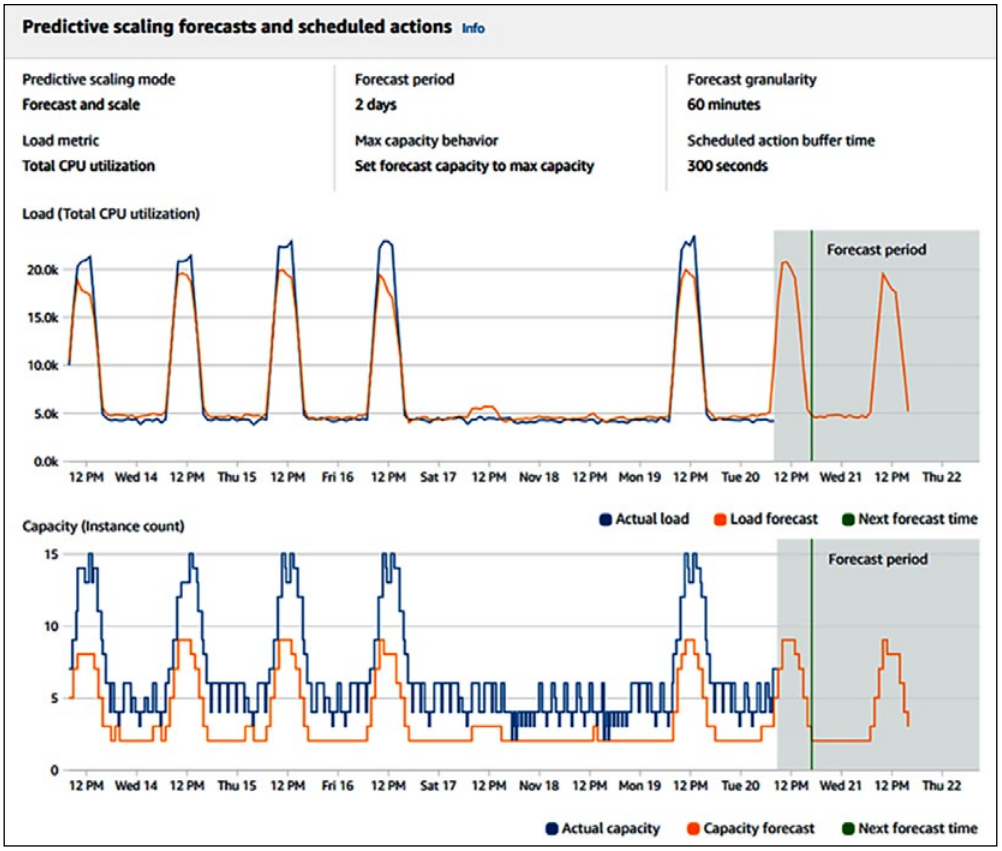


Figure 4.2: Predictive scaling forecast

Scheduled scaling actions (32)

1 2 3 4

Start time	Min capacity	Max capacity
2018-11-20 08:55:00 UTC-0800	7	15
2018-11-20 09:55:00 UTC-0800	9	15
2018-11-20 11:00:00 UTC-0800	9	15
2018-11-20 12:00:00 UTC-0800	9	15
2018-11-20 13:00:00 UTC-0800	8	15
2018-11-20 14:00:00 UTC-0800	7	15
2018-11-20 15:00:00 UTC-0800	5	15
2018-11-20 16:00:00 UTC-0800	3	15
2018-11-20 17:00:00 UTC-0800	2	15
2018-11-20 18:00:00 UTC-0800	2	15

Figure 4.3: Predictive scaling capacity plan

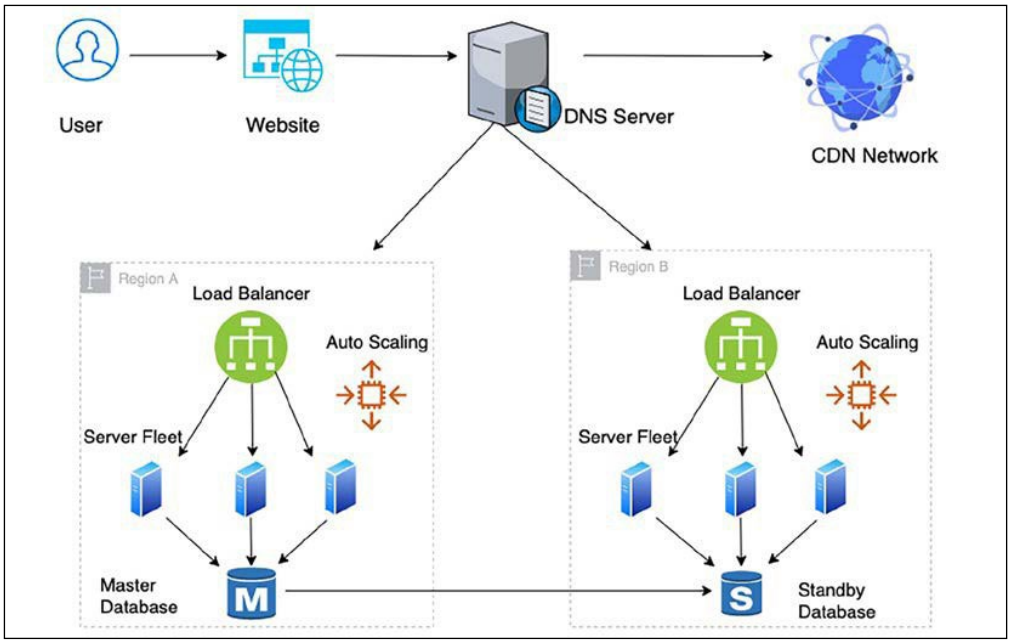


Figure 4.4: Application architecture resiliency

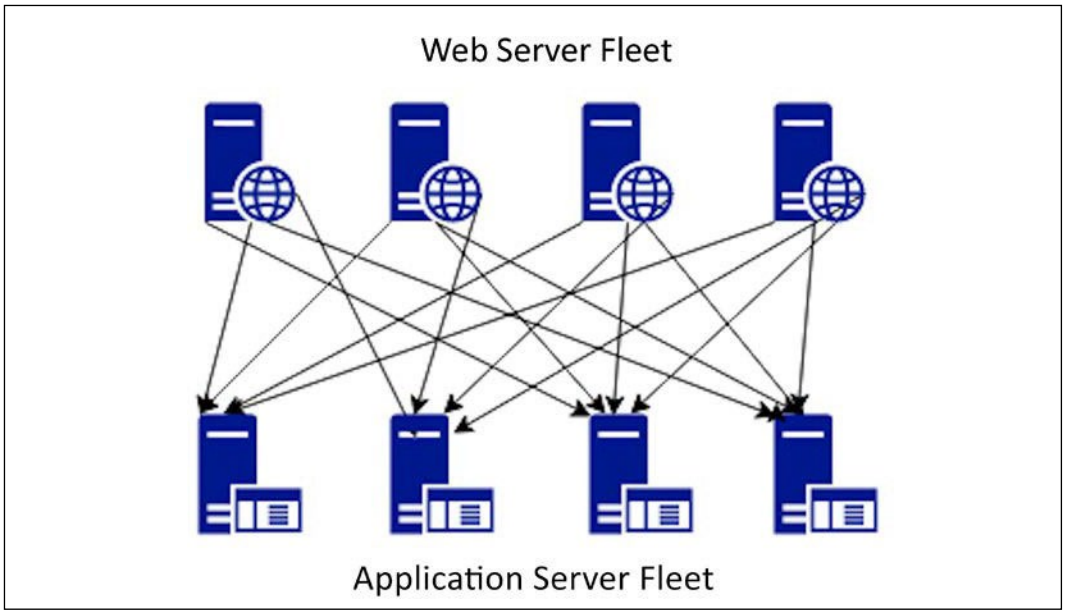


Figure 4.5: Tightly coupled architecture

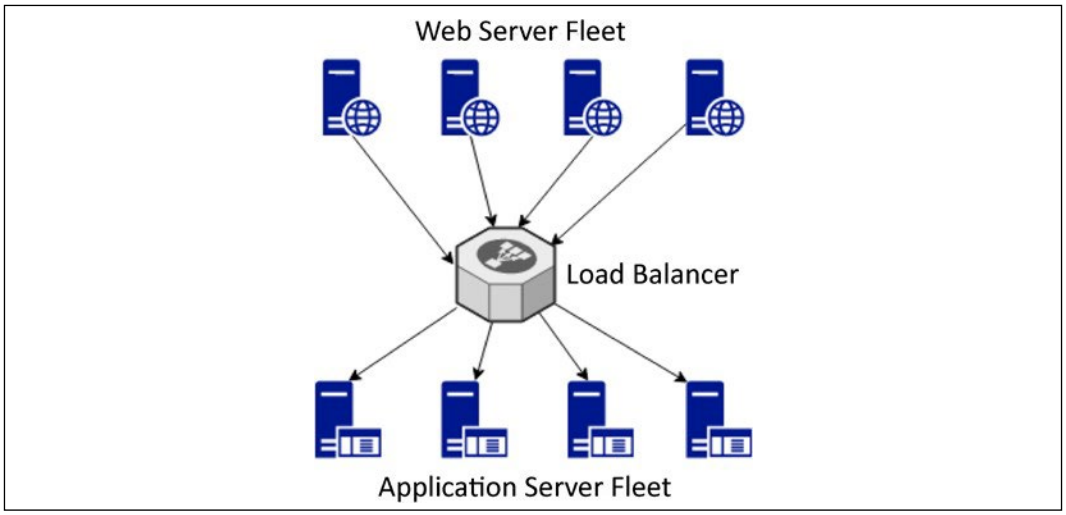


Figure 4.6: Load balancer-based, loosely coupled architecture

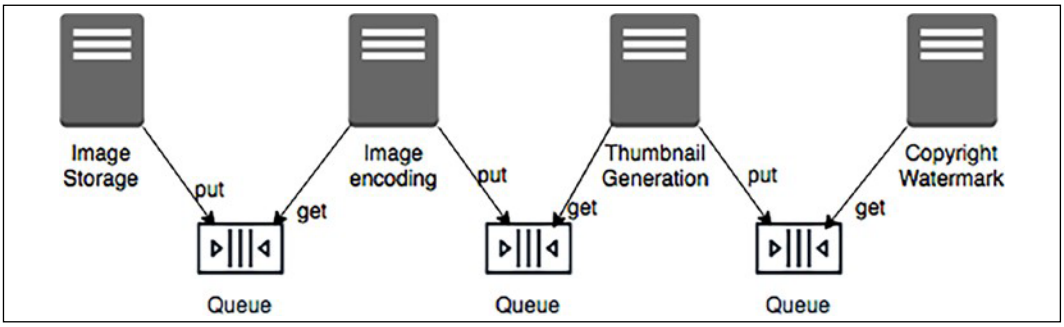


Figure 4.7: Queue-based, loosely coupled architecture

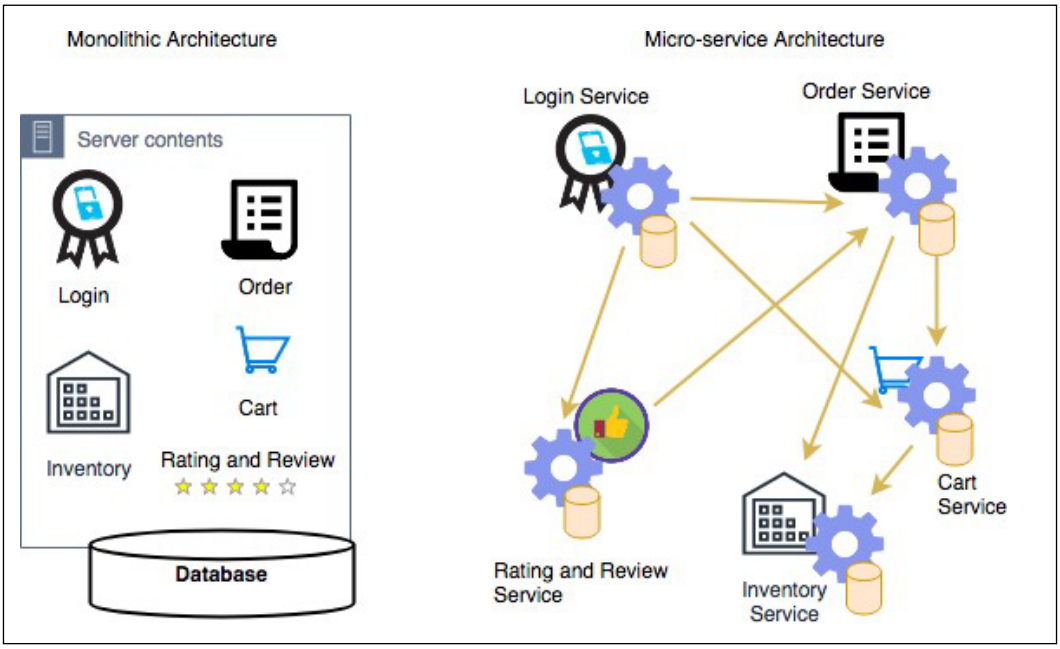


Figure 4.8: Monolithic and microservice architectures

Data Type	Data Example	Storage Type	Storage Example
Transactional, structured schema	User order data, financial transaction	Relational database	Amazon RDS, Oracle, MySQL, Amazon Aurora PostgreSQL, MariaDB, Microsoft SQL Server

Key/value pair, semi-structured, unstructured	User session data, application log, review, comments	NoSQL	Amazon DynamoDB, MongoDB, Apache HBase, Apache Cassandra, Azure Tables
Analytics	Sales data, supply chain intelligence, business flow	Data warehouse	IBM Netezza, Amazon Redshift, Teradata, Greenplum, Google BigQuery
In-memory	User home page data, common dashboard	Cache	Redis cache, Amazon ElastiCache, Memcached
Object	Image, video	File-based	SAN, Amazon S3, Azure Blob Storage, Google Storage
Block	Installable software	Block-based	NAS, Amazon EBS, Amazon EFS, Azure Disk Storage
Streaming	IoT sensor data, clickstream data	Temporary storage for streaming data	Apache Kafka, Amazon Kinesis, Spark Streaming, Apache Flink
Archive	Any kind of data	Archive storage	Amazon Glacier, magnetic tape storage, virtual tape library storage
Web storage	Static web contents such as images, videos, and HTML pages	CDN	Amazon CloudFront, Akamai CDN, Azure CDN, Google CDN, Cloudflare
Search	Product search, content search	Search index store and query	Amazon Elastic Search, Apache Solr, Apache Lucene
Data catalog	Table metadata, data about data	Metadata store	AWS Glue, Hive metastore, Informatica data catalog, Collibra data catalog
Monitoring	System log, network log, audit log	Monitor dashboard and alert	Splunk, Amazon CloudWatch, SumoLogic, Loggly

Table 4.1: Data types and their optimal storage requirements

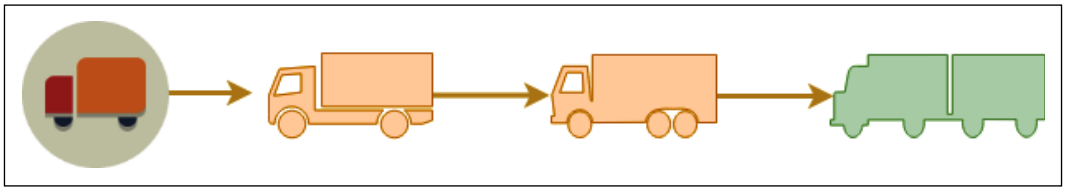


Figure 4.9: MVP approach to building the solution

Chapter 5
Cloud Migration and Hybrid Cloud Architecture Design

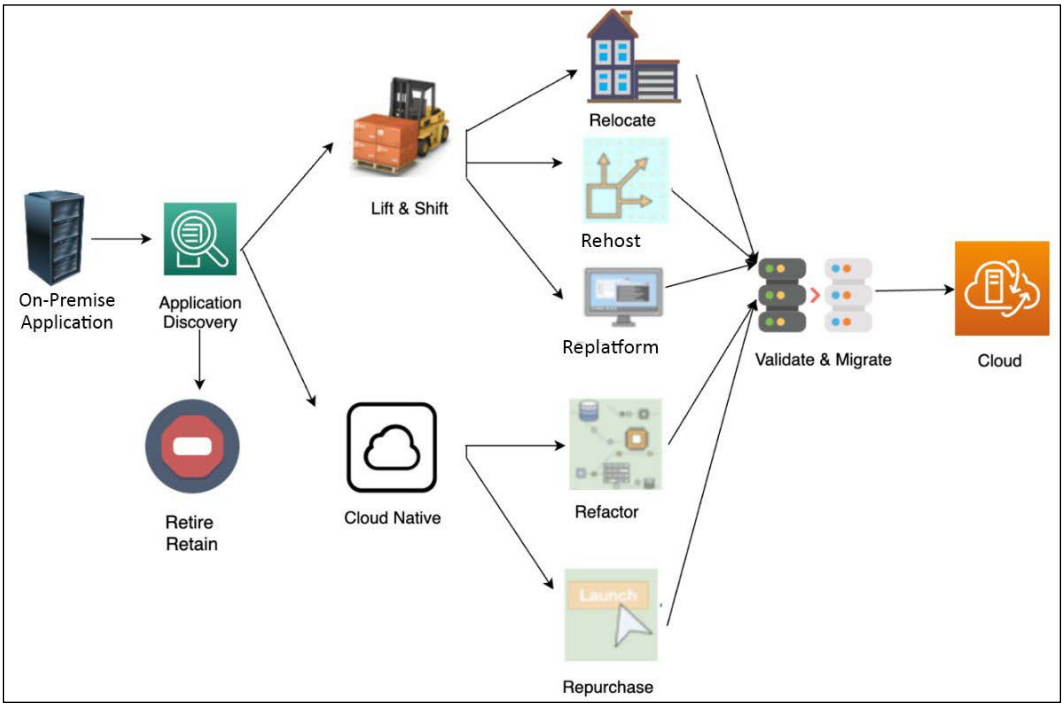


Figure 5.1: Cloud migration strategy

Migration Strategy	Description	Time and Cost	Optimization Opportunities
Refactor	Re-architect application into more modularized such as monolithic to microservice		
Replatform	Migrate application to upgraded platform without changing core architecture, such traditional database to cloud or higher operating system version		
Repurchase	Replacing your current enviroment by purchasing a cloud-based solution		
Rehost	Quickly lift and shift your applications to the cloud without architecture changes		
Retain	Leaving the application on-premises for now at least		NA
Relocate	Quickly relocate applications to the cloud without changing them, such as container-based applications		NA

Retire	Identify the assests that are no longer useful and remove them entirely	NA	NA

Table 5.1: Migration strategies based on effort and opportunity

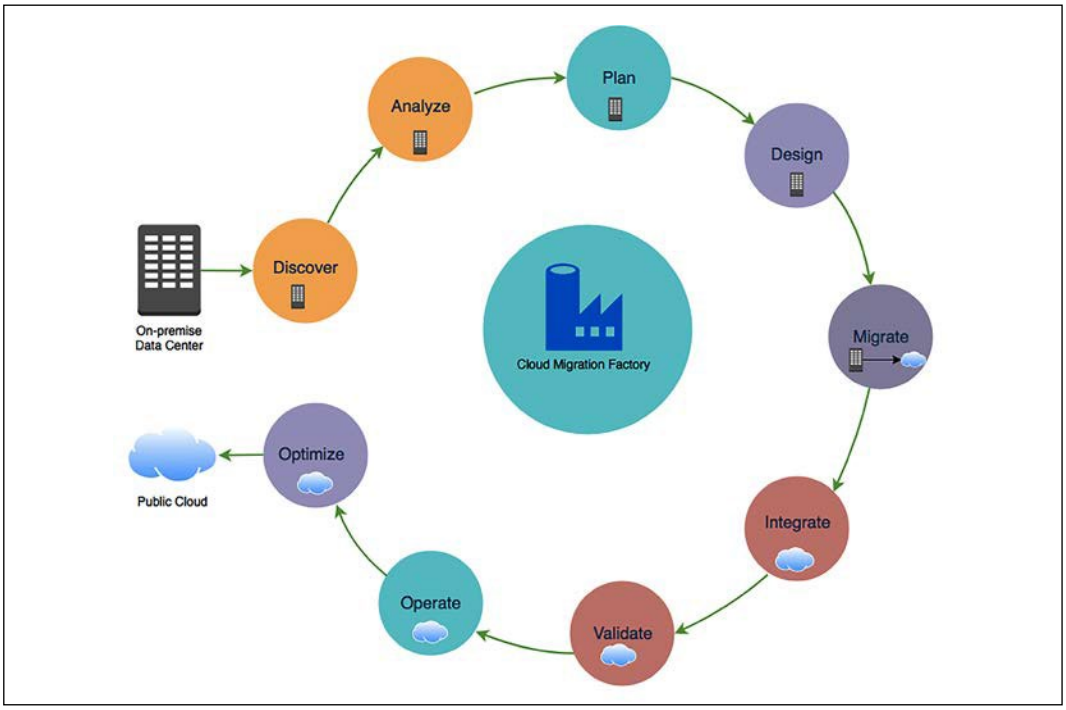


Figure 5.2: Cloud migration steps

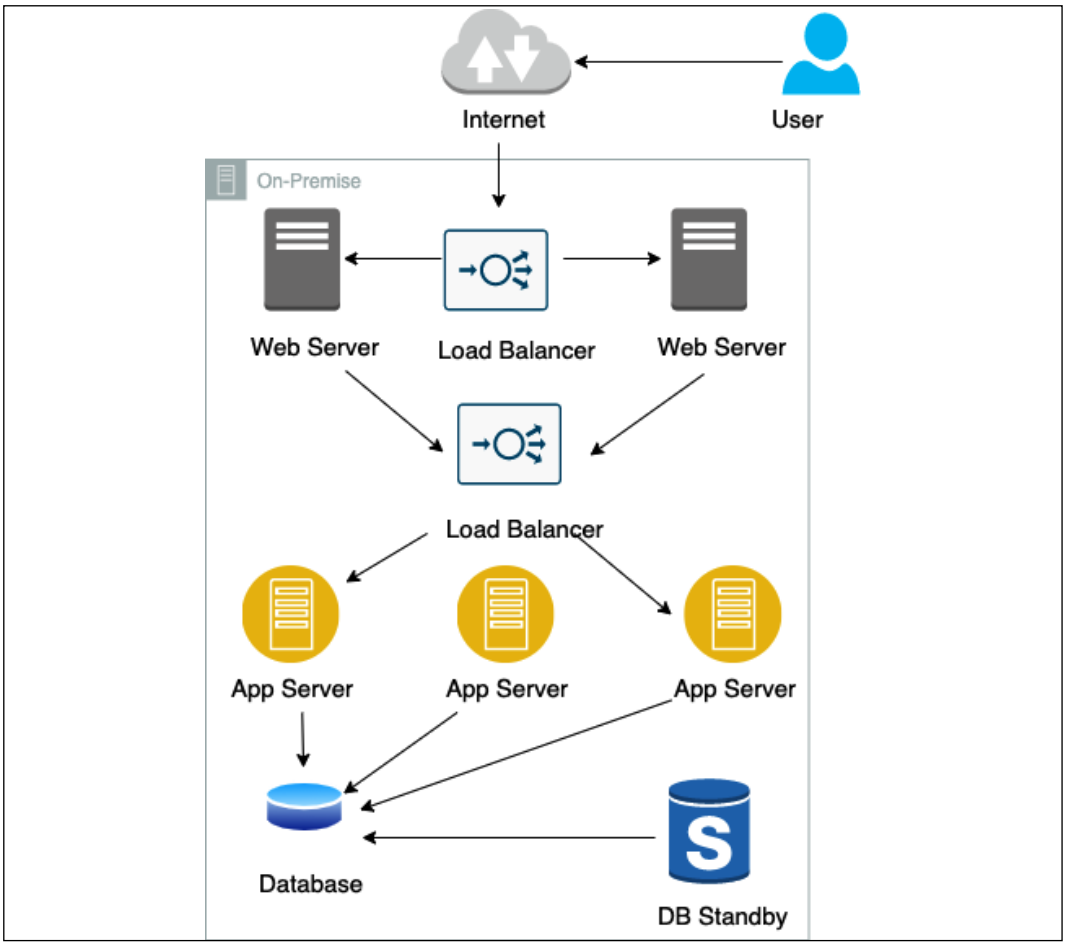


Figure 5.3: On-premise architecture mapping

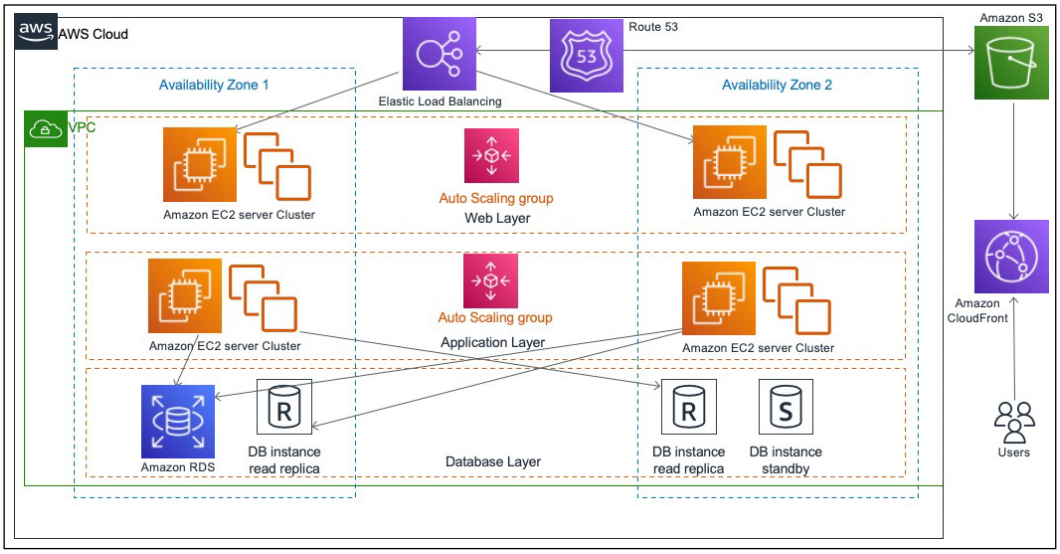


Figure 5.4: On-premise to AWS cloud architecture mapping

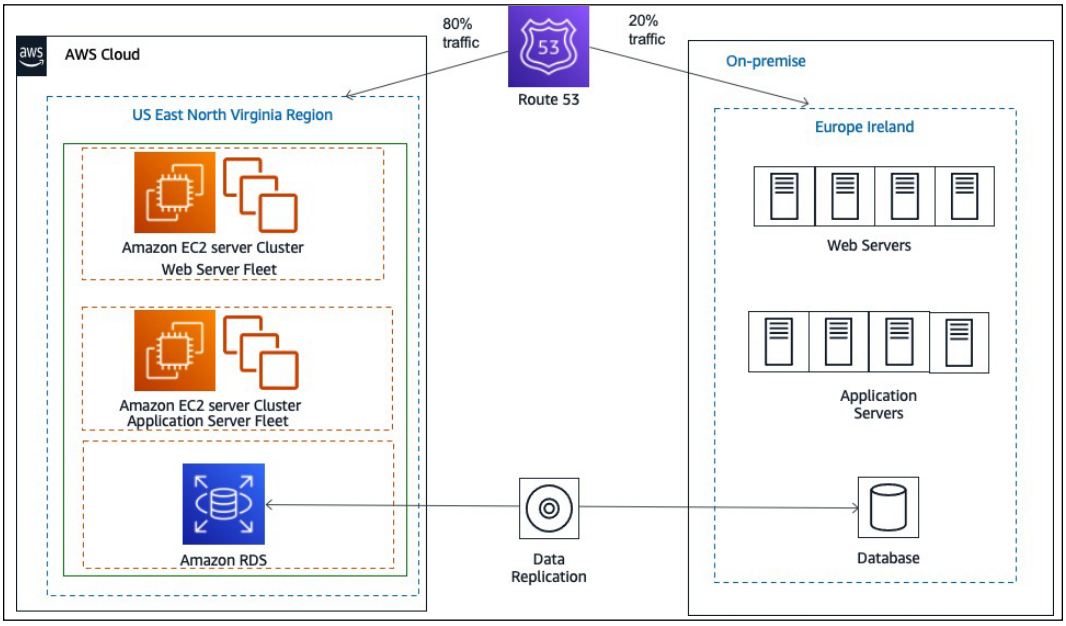


Figure 5.5: Live migration cutover using blue-green deployment

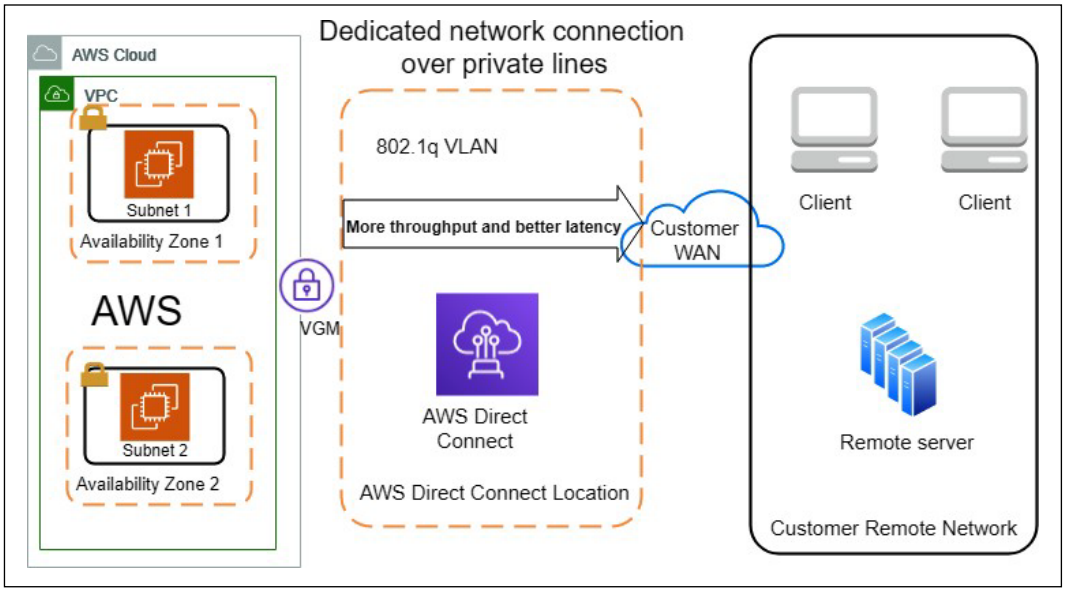


Figure 5.6: Hybrid cloud architecture (on-premises to cloud connectivity)

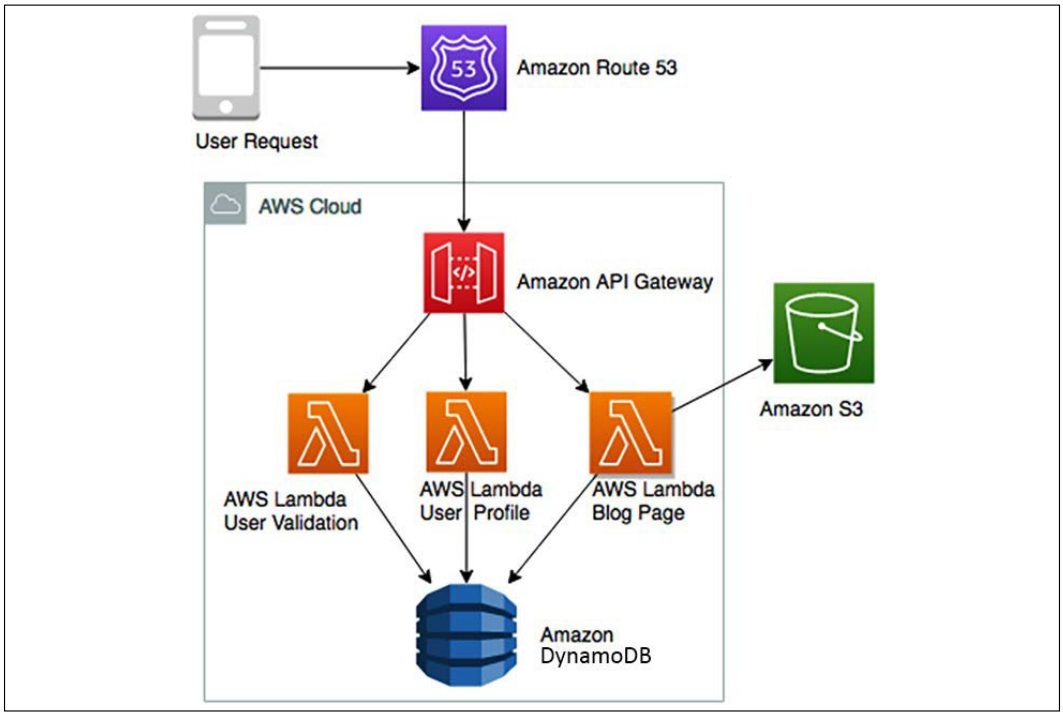


Figure 5.7: Cloud native micro-blogging application architecture

Chapter 6
Solution Architecture Design Patterns

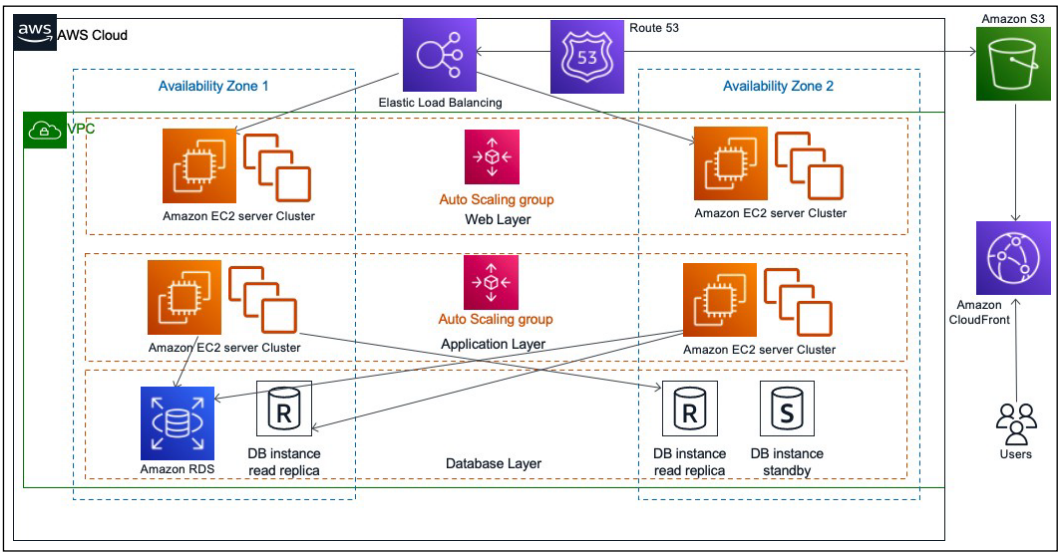


Figure 6.1: Three-tier website architecture

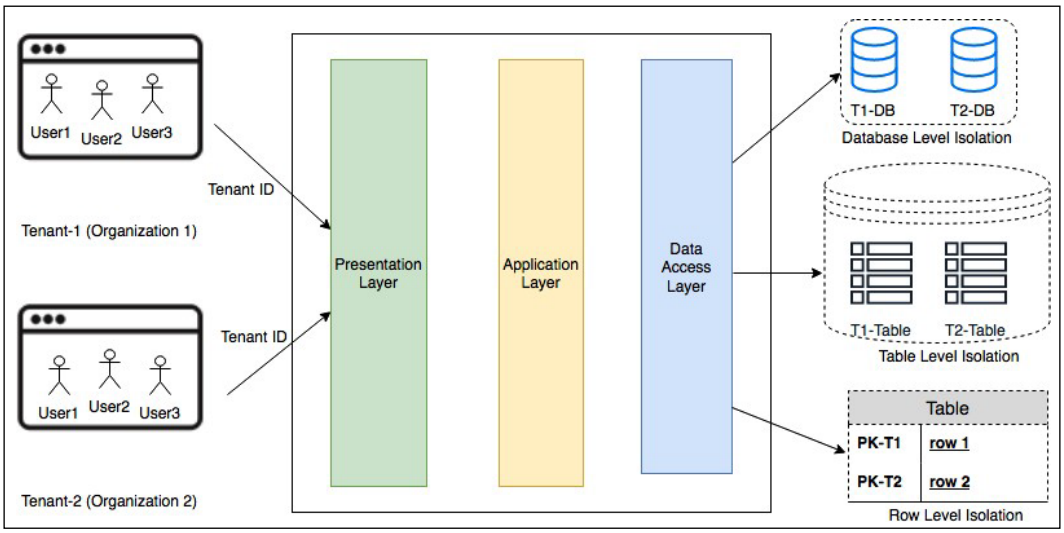


Figure 6.2: Multi-tenant SaaS architecture

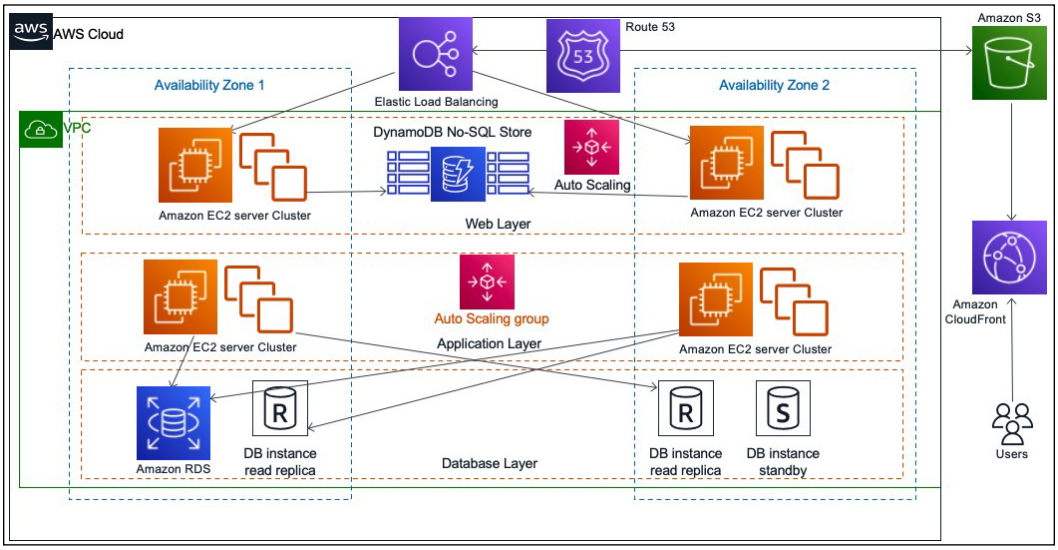


Figure 6.3: A stateless application architecture

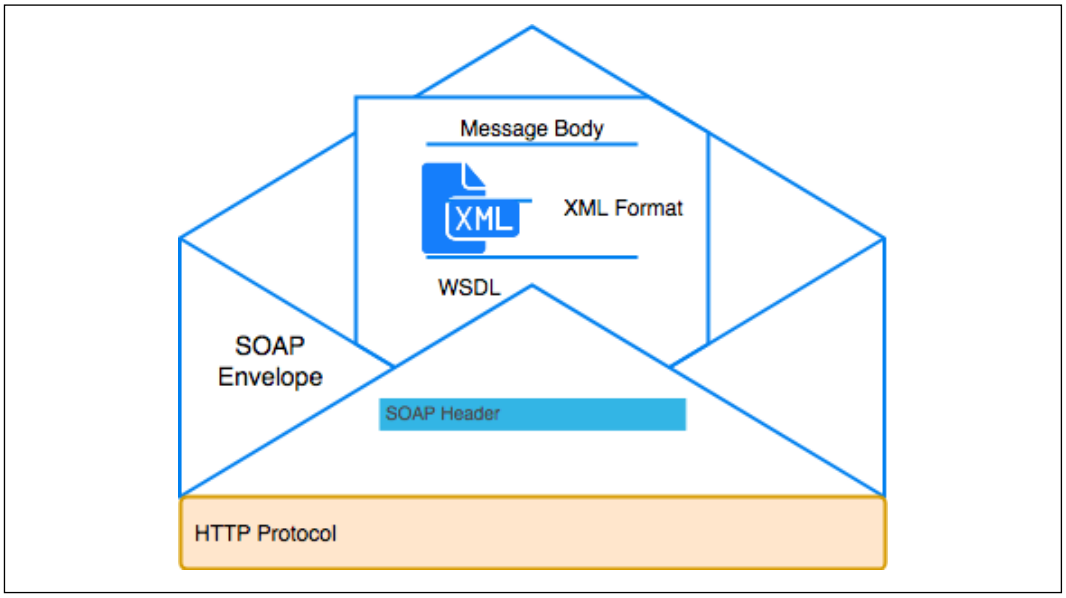


Figure 6.4: SOAP envelope for web service data exchange

```xml
<?xml version="1.0"?>
<definitions name="Order"
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
    <n:orderinfo xmlns:n="http://exampleorder.org/orderinfo">
        <n:priority>1</n:priority>
        <n:expires>2019-06-30T16:00:00-09:00</n:expires>
    </n:orderinfo>
</env:Header>
<env:Body>
    <m:order xmlns:m="http://exampleorder.org/orderinfo">
        <m:getorderinfo>
            <m:orderno>
            12345</m:oderno>
        </m:getorderinfo>
    </m:order>
</env:Body>
```

Code listing 6.1: Example SOAP envelope XML

```xml
targetNamespace=http://example.com/order.wsdl
    xmlns:tns=http://example.com/ order.wsdl
    xmlns:xsd1=http://example.com/ order.xsd
    xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
    xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
    <schema targetNamespace="http://example.com/ order.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
        <element name="PlaceOrder">
            <complexType>
                <all>
                    <element name="itemID" type="string"/>
                </all>
            </complexType>
        </element>
        <element name="ItemPrice">
            <complexType>
                <all>
                    <element name="price" type="float"/>
                </all>
            </complexType>
        </element>
    </schema>
</types>
<message name="GetOrderInfo">
    <part name="body" element="xsd1:GetOrderRequest"/>
</message>
```

```
<message name="GetItemInfo">
    <part name="body" element="xsd1:ItemPrice"/>
</message>
<portType name="OrderPortType">
    <operation name="GetOrderInfo">
        <input message="tns: GetOrderInfoInput "/>
        <output message="tns: GetOrderInfoOutput"/>
    </operation>
</portType>
<binding name="OrderSoapBinding" type="tns:OrderPortType">
    <soap:binding style="document" transport="http://schemas.
xmlsoap.org/soap/http"/>
    <operation name="GetOrderInfo">
        <soap:operation soapAction="http://example.com/GetOrderInfo
            "/>
        <input>
        <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
<service name="OrderService">
    <documentation>My first Order</documentation>
    <port name="OrderPort" binding="tns:OrderBinding">
        <soap:address location="http://example.com/order"/>
    </port>
</service>
</definitions>
```

Code listing 6.2: Example SOAP contract in WSDL

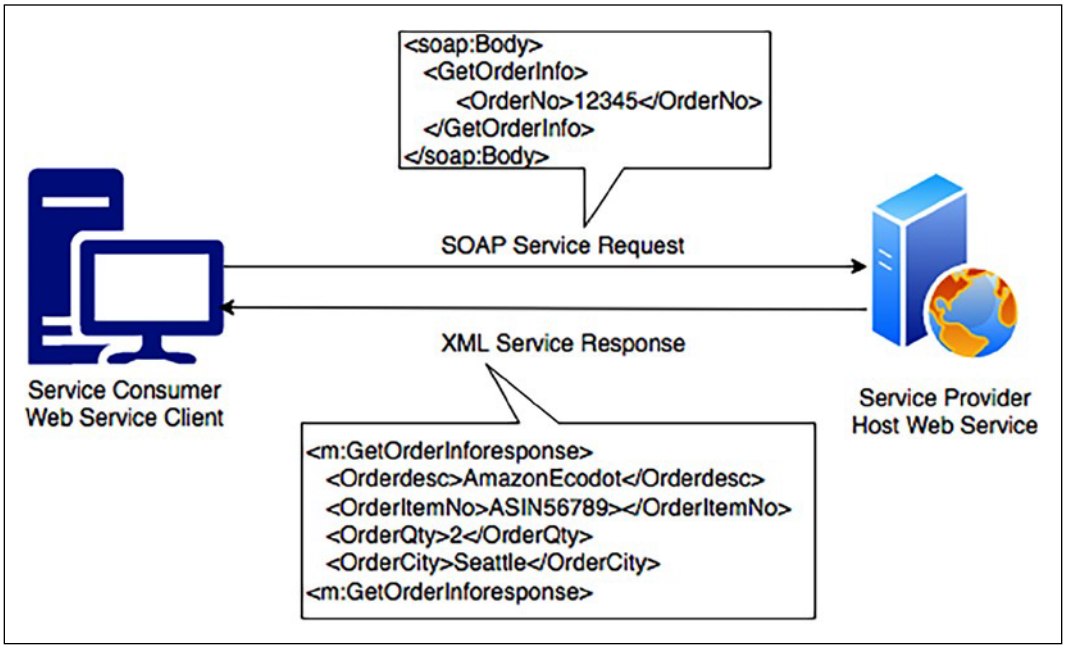


Figure 6.5: SOAP-based web service

Attributes	**REST**	**SOAP**
Design	Architectural style with an informal guideline	Predefined rules with a standard protocol
Message Format	JSON, YAML, XML, HTML, plaintext, and CSV	XML
Protocol	HTTP	HTTP, SMTP, and UPD
Session State	Default stateless	Default stateful
Security	HTTPS and SSL	Web Services Security and ACID compliance
Cache	Cached API calls	Cannot cache API calls
Performance	Fast with fewer resources	Needs more bandwidth and compute power

Table 6.1: REST and SOAP architectural attributes

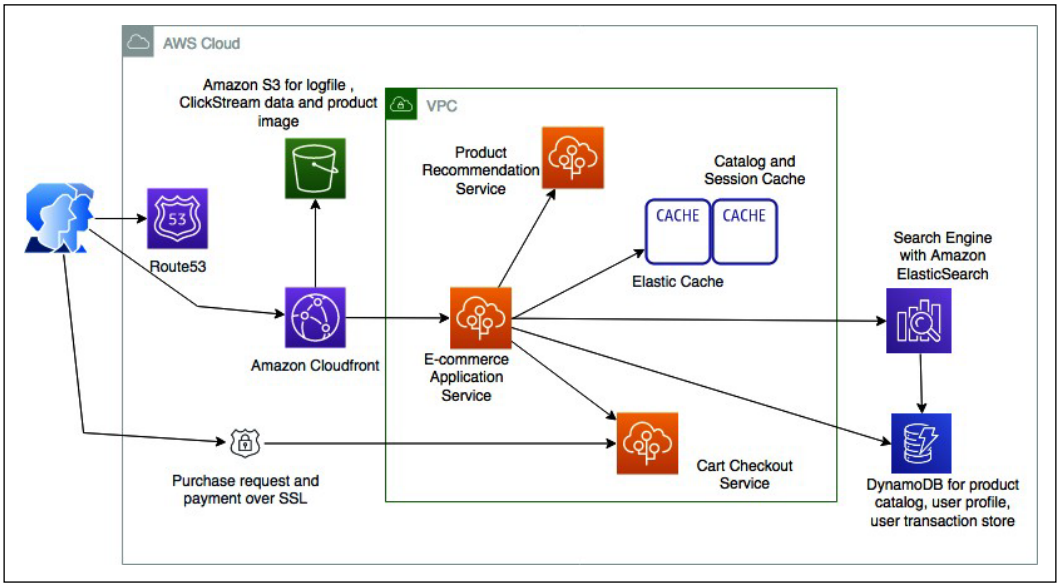


Figure 6.6: E-commerce website SOA

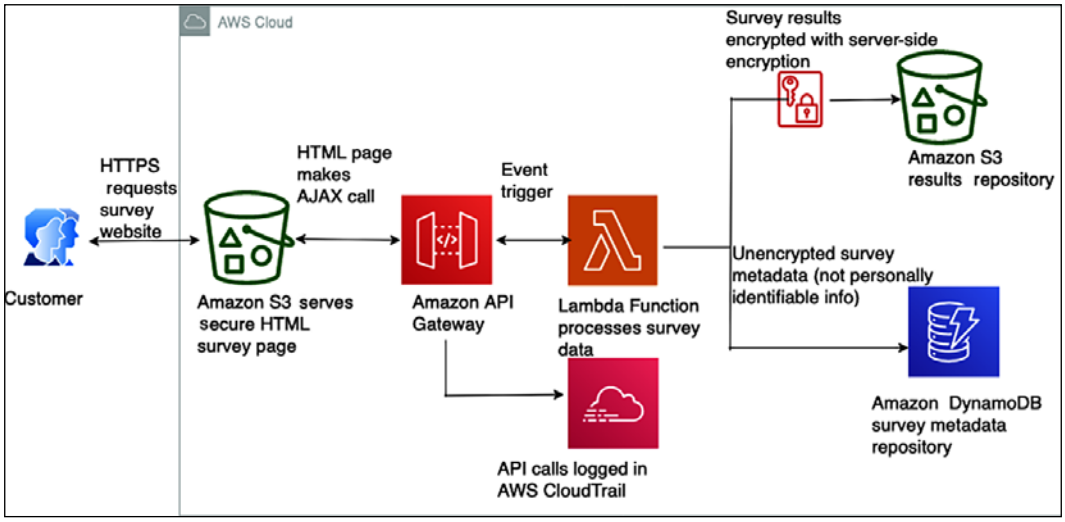


Figure 6.7: Serverless architecture for a secure survey delivery

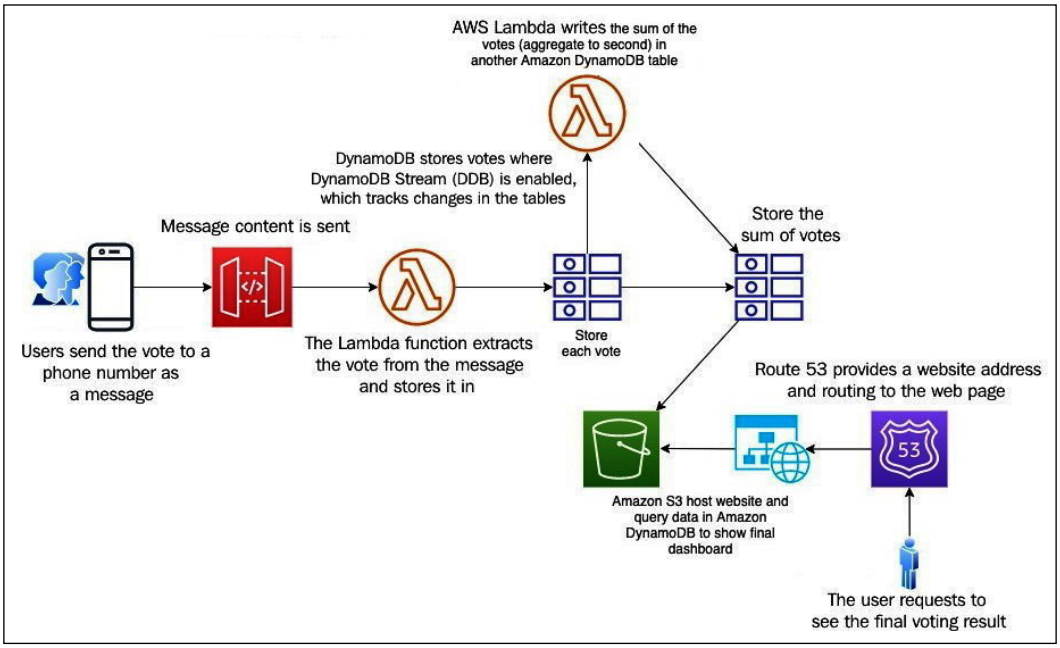


Figure 6.8: Microservice-based real-time voting application architecture

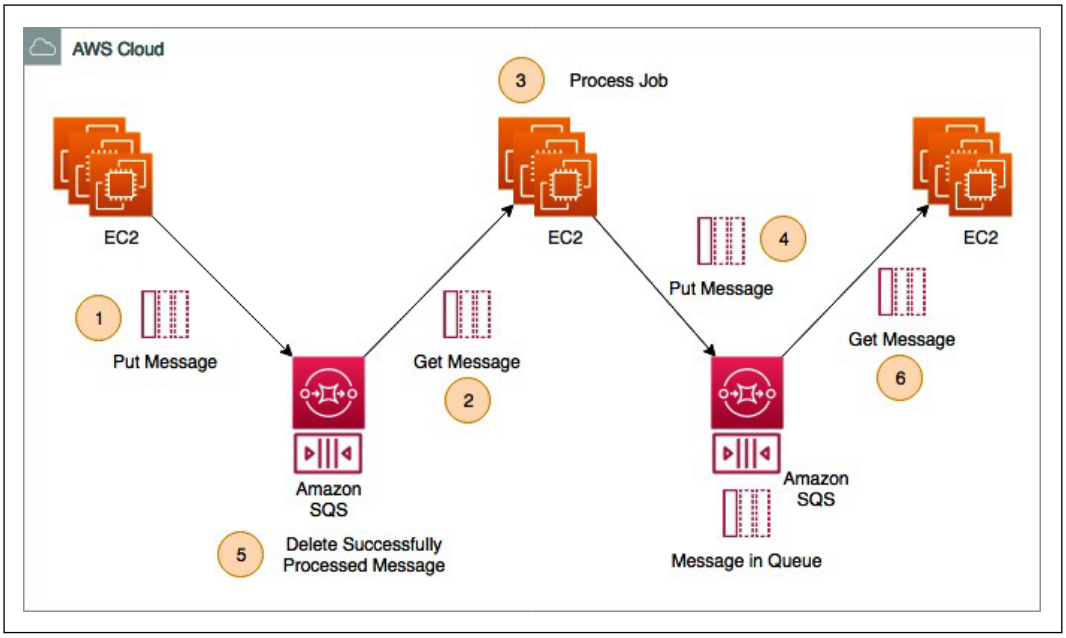


Figure 6.9: Queuing chain pattern architecture

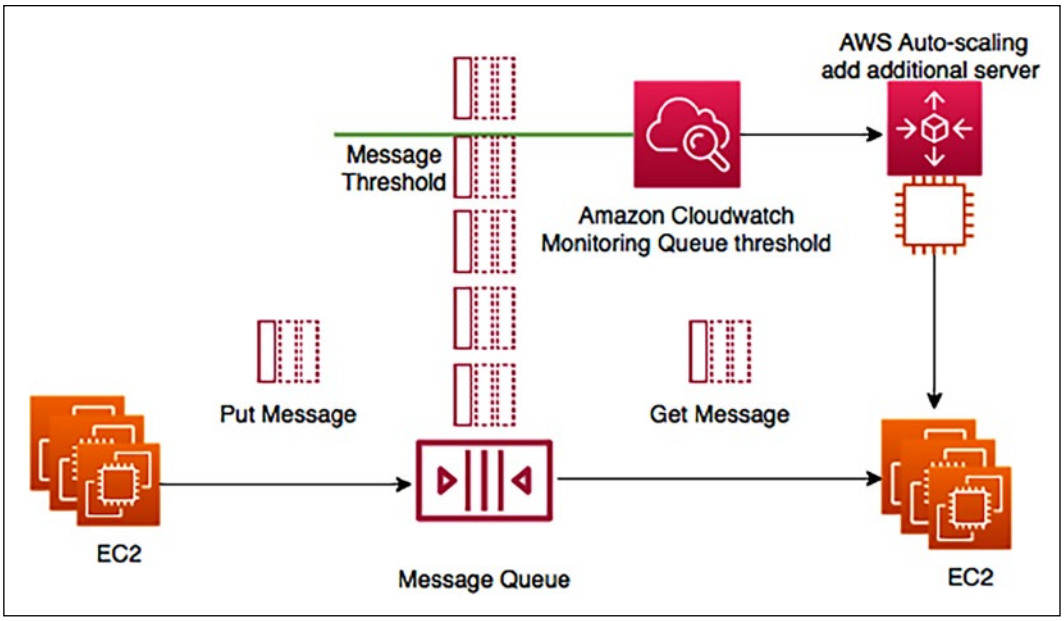


Figure 6.10: Job observer pattern architecture

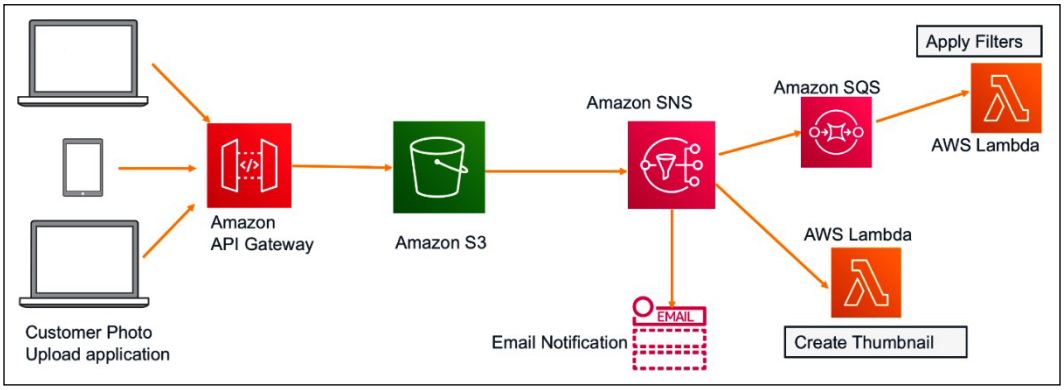


Figure 6.11: Photo studio application pub/sub event-driven architecture

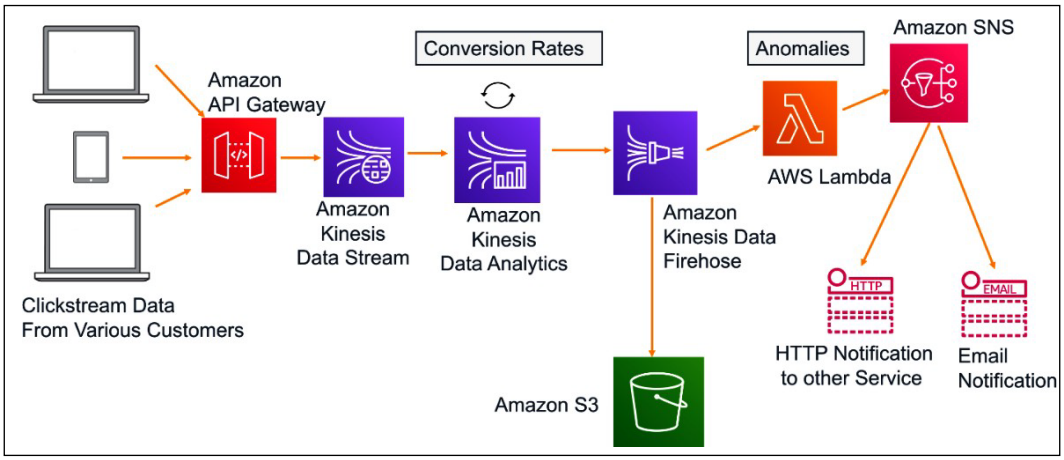


Figure 6.12: Clickstream analysis event stream architecture

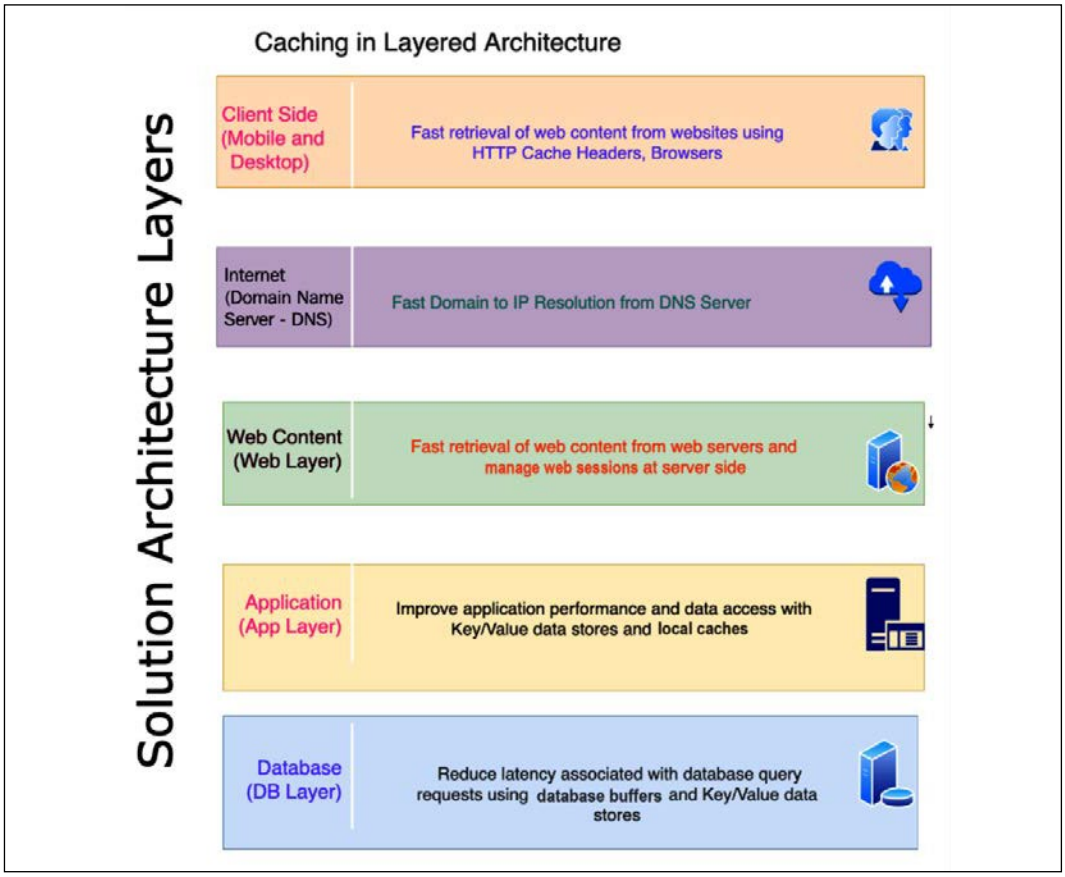


Figure 6.13: Caching at the architecture layers

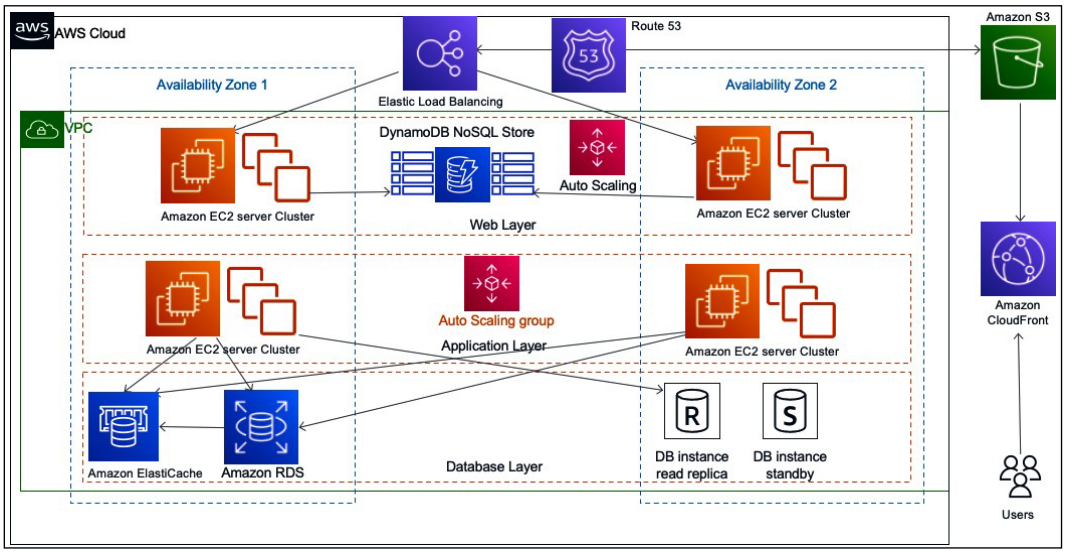


Figure 6.14: Cache distribution pattern architecture

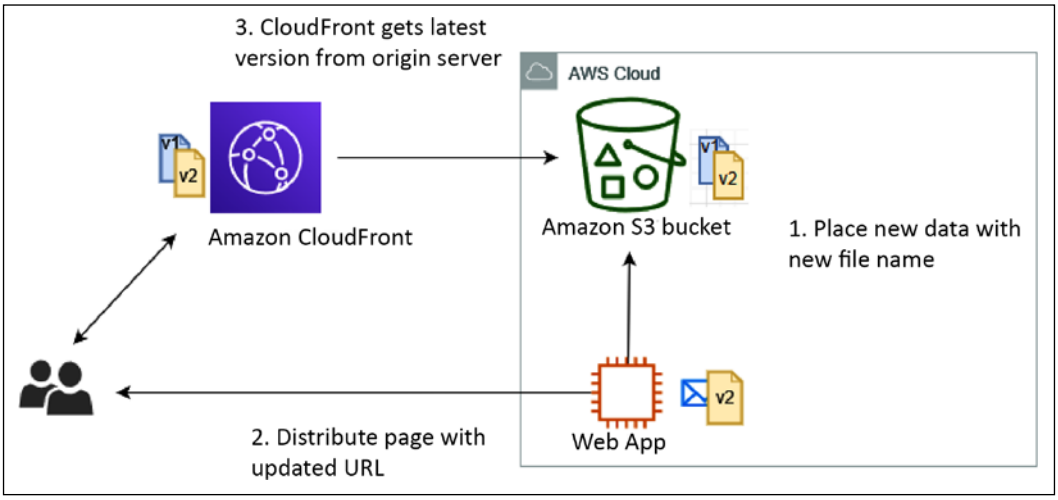


Figure 6.15: Rename distribution pattern architecture

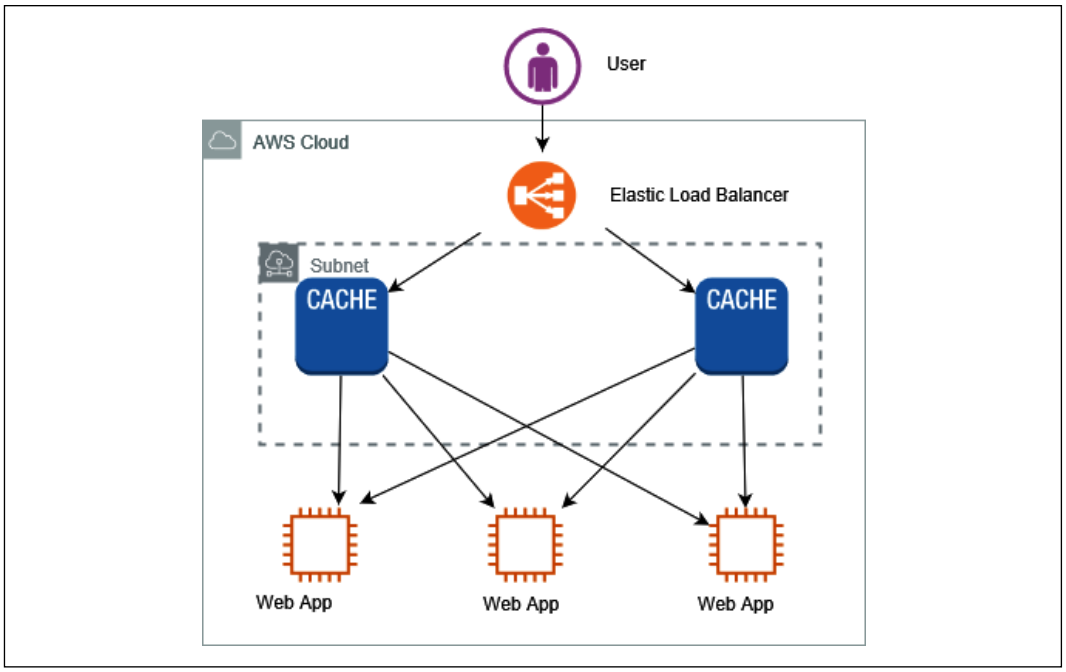


Figure 6.16: Cache proxy pattern architecture

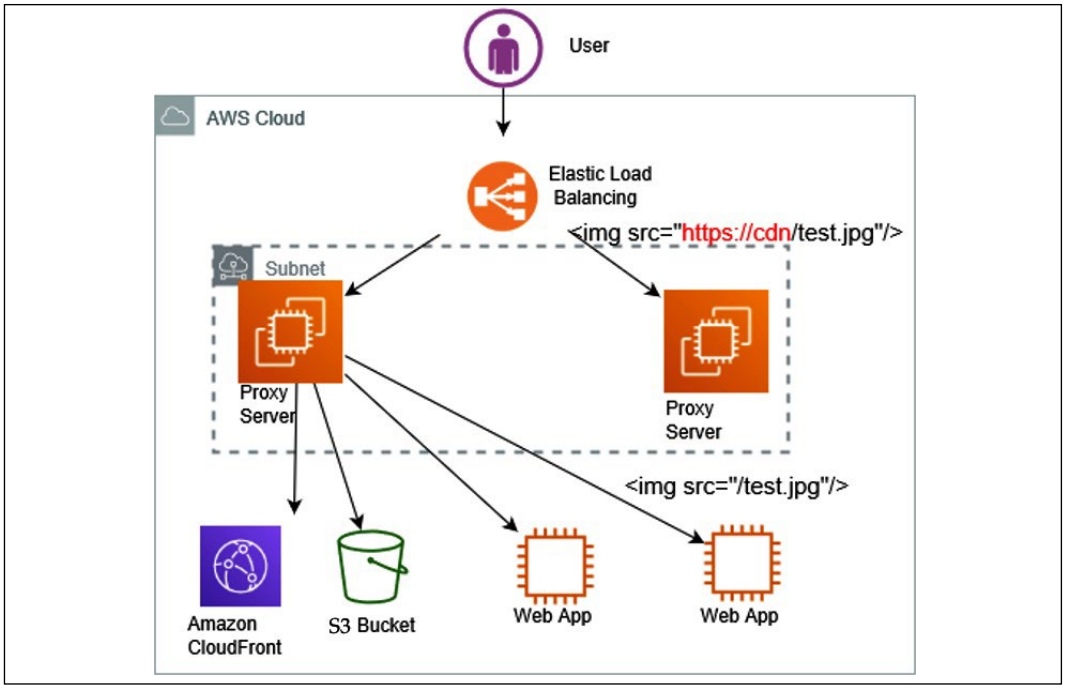


Figure 6.17: Rewrite proxy pattern architecture

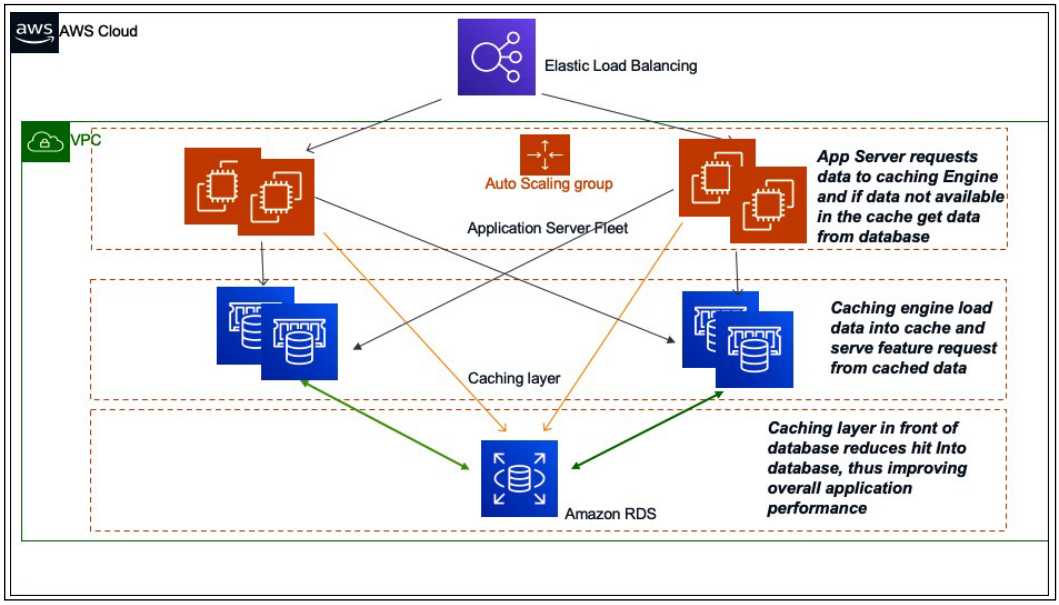


Figure 6.18: Application caching pattern architecture

Memcached	**Redis**
Offers multithreading	Single-threaded
Able to use more CPU cores for faster processing	Unable to utilize multi-core processor, which results in comparatively slow performance
Supports key-value style data	Supports complex and advanced data structures
Lacks data persistence; loses the data stored in cache memory in the event of a crash	Data can persist using built-in read replicas with failover
Easy maintenance	More complexity involved owing to the need to maintain the cluster
Good to cache flat strings such as flat HTML pages, serialized JSON, and more	Good to create a cache for a gaming leaderboard, a live voting app, and more

Table 6.2: Memcached and Redis caching attributes

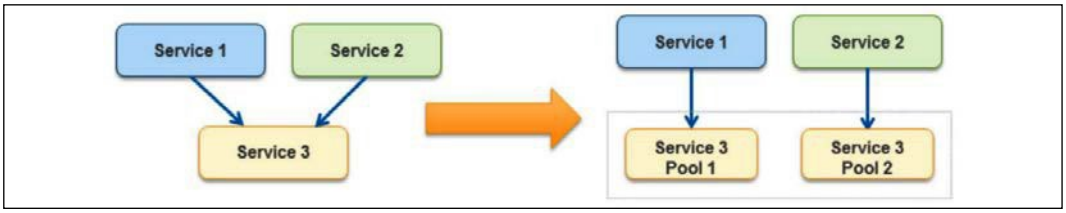


Figure 6.19: Bulkhead pattern

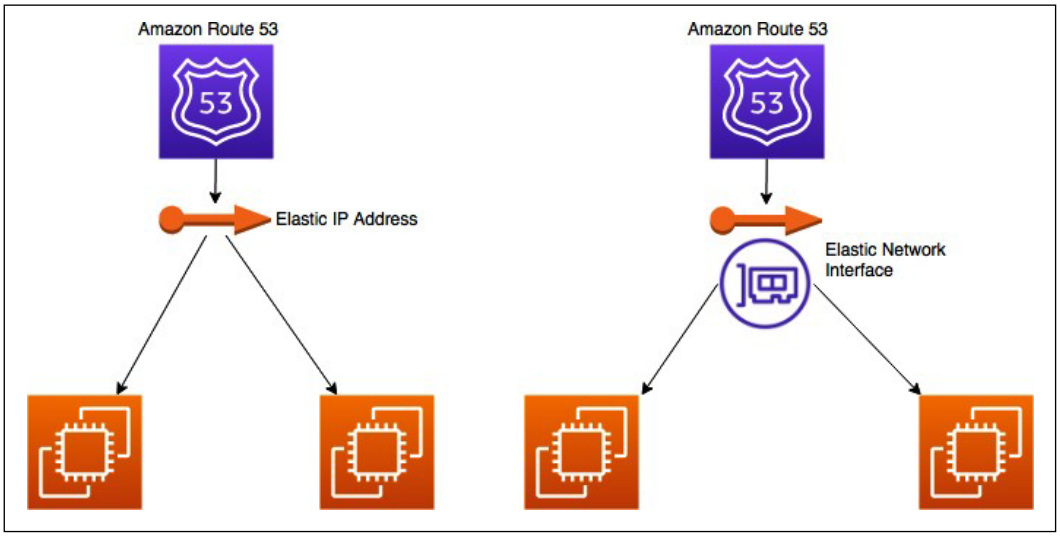


Figure 6.20: Floating IP and interface pattern

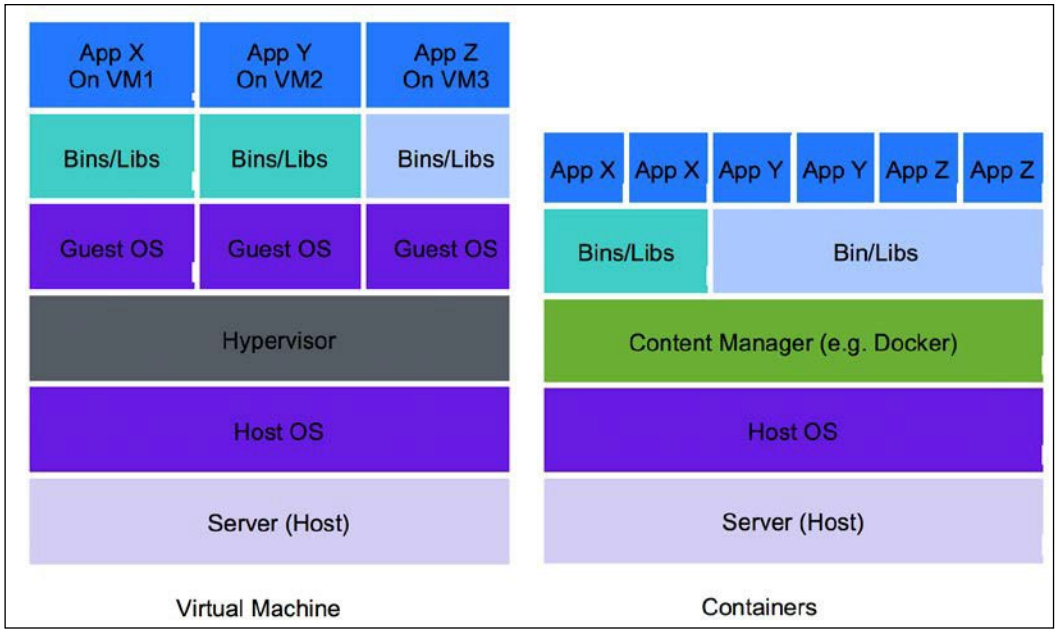


Figure 6.21: Virtual machines and containers for application deployment

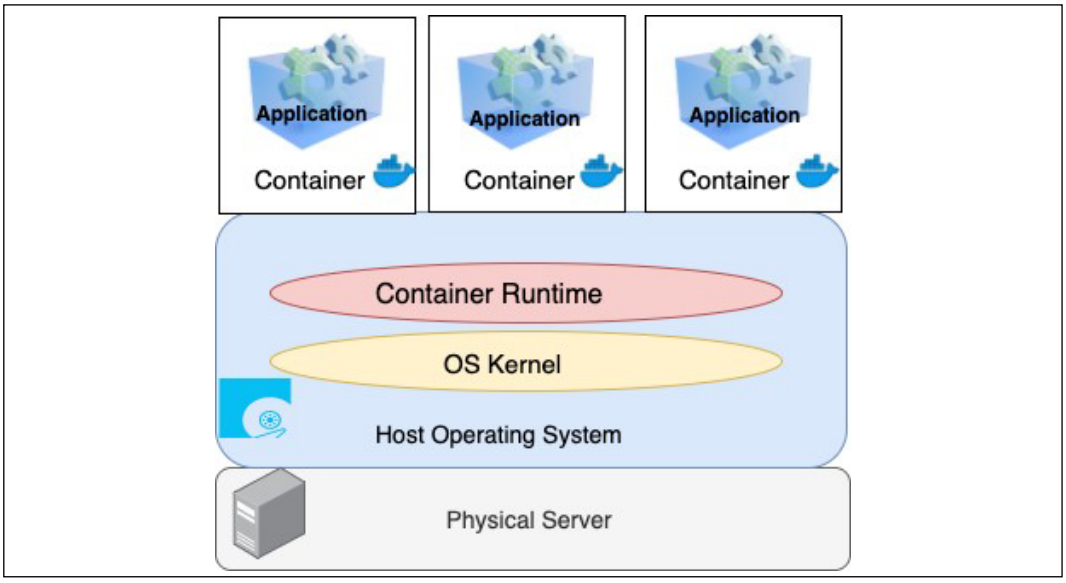


Figure 6.22: Container layer in application infrastructure

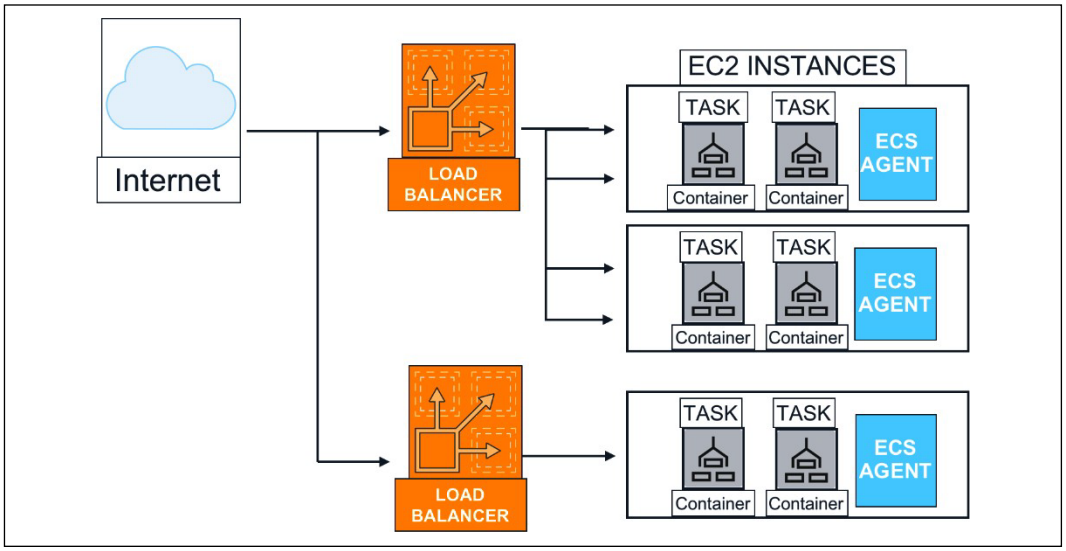


Figure 6.23: Container deployment architecture

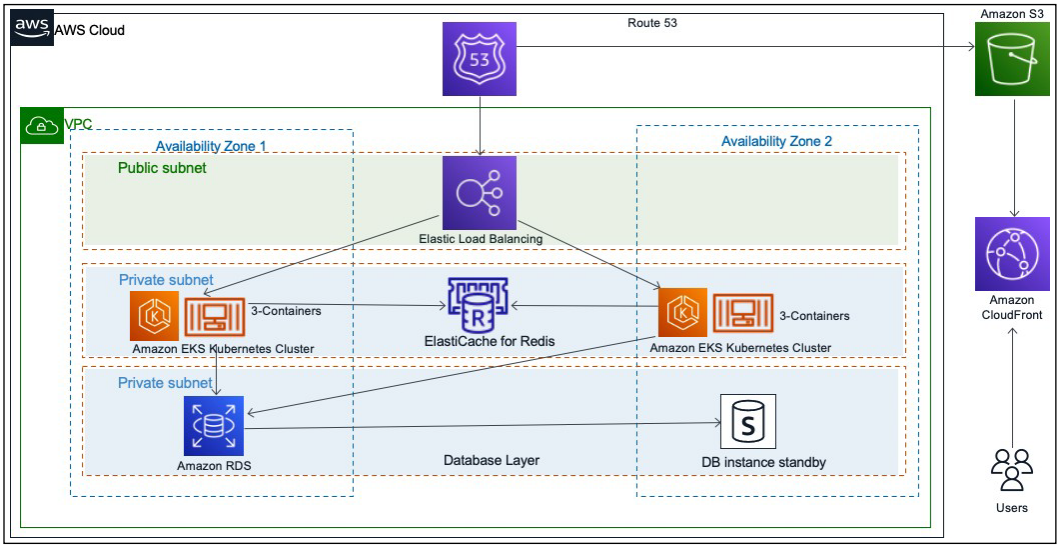


Figure 6.24: Deploying a stateful application on a container

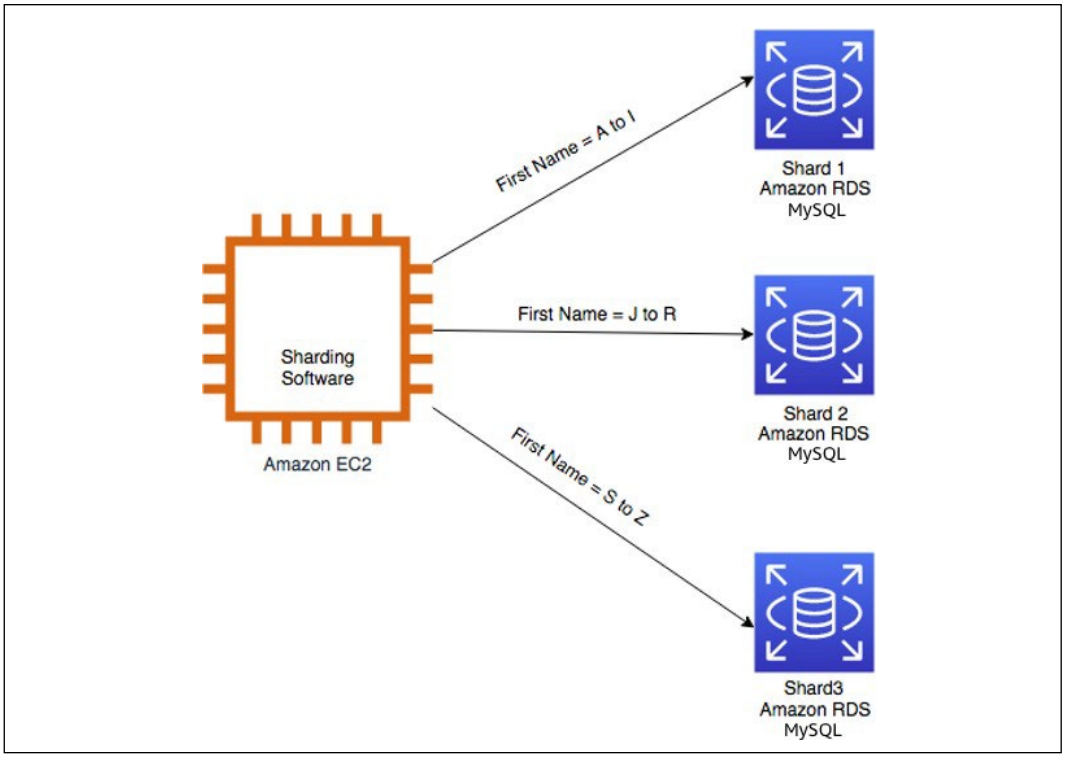


Figure 6.25: Relational database sharding

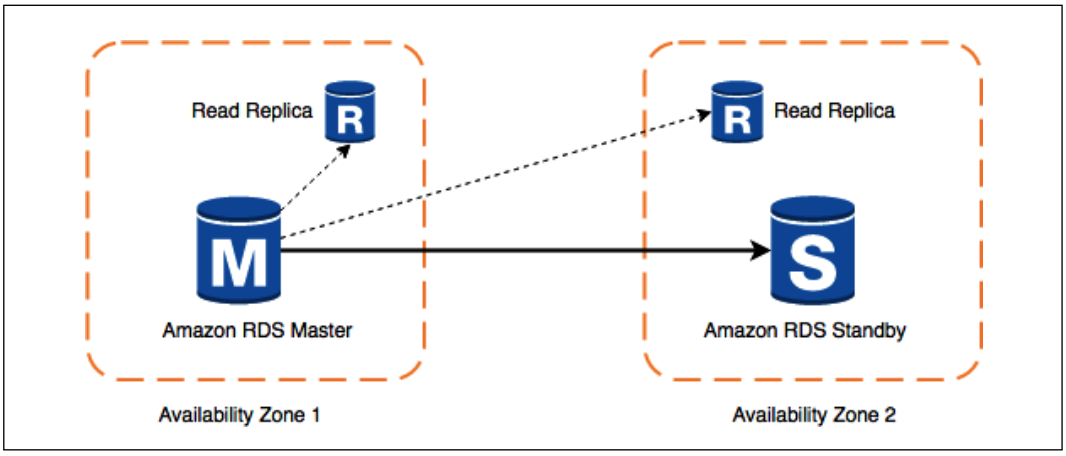


Figure 6.26: High-availability database pattern

Chapter 7
Performance Considerations

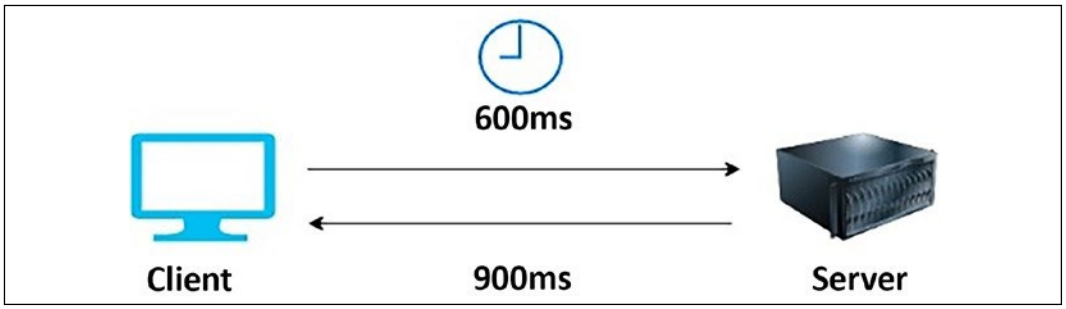


Figure 7.1: Request-response latency in a client-server model

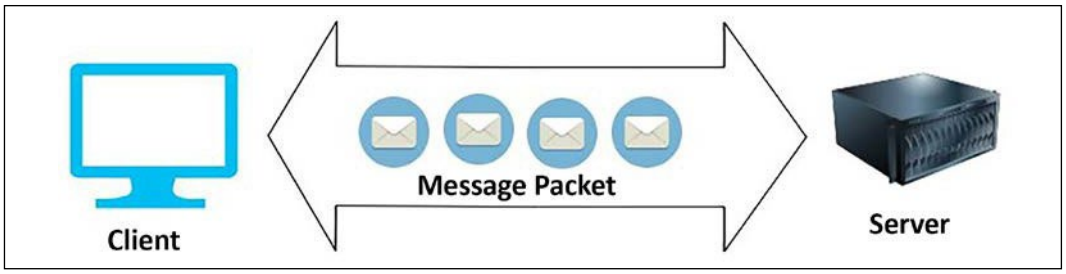


Figure 7.2: Throughput in a network

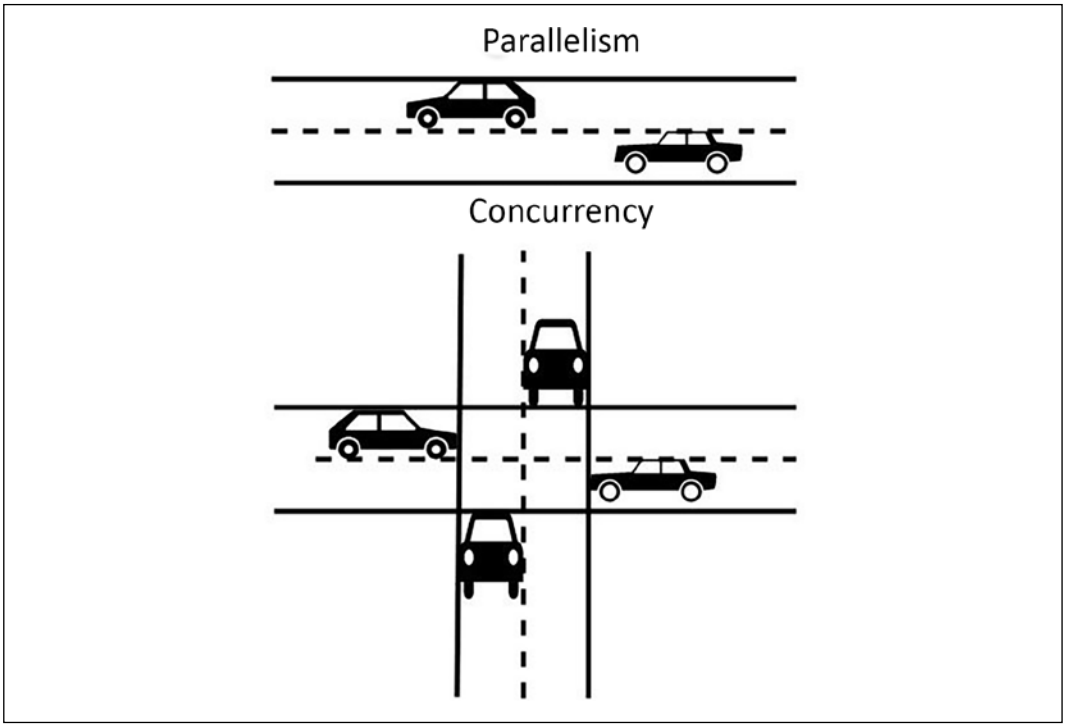


Figure 7.3: Concurrency versus parallelism

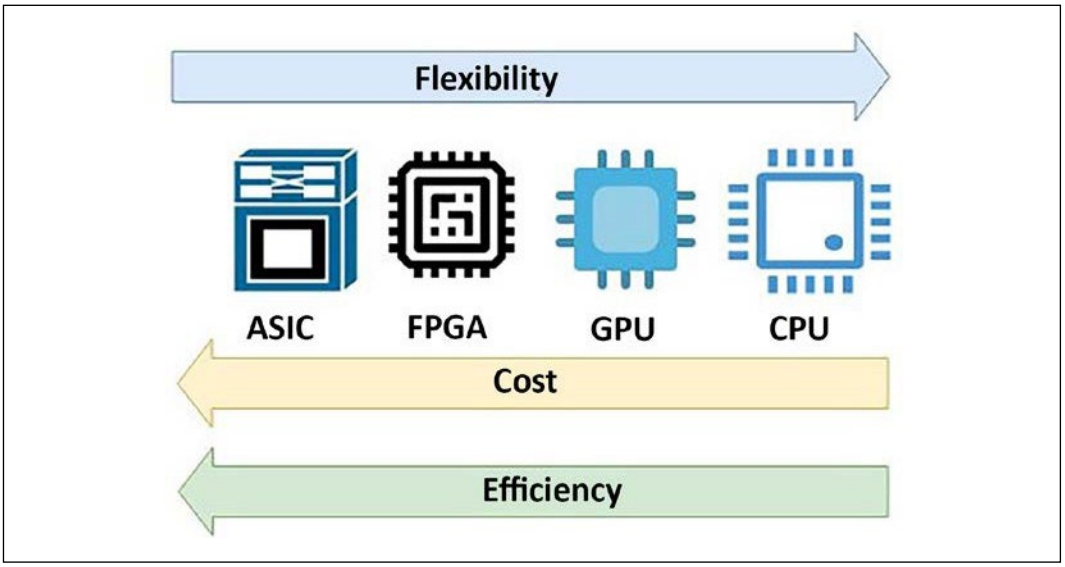


Figure 7.4: Comparison between CPUs, GPUs, FPGAs, and ASICs

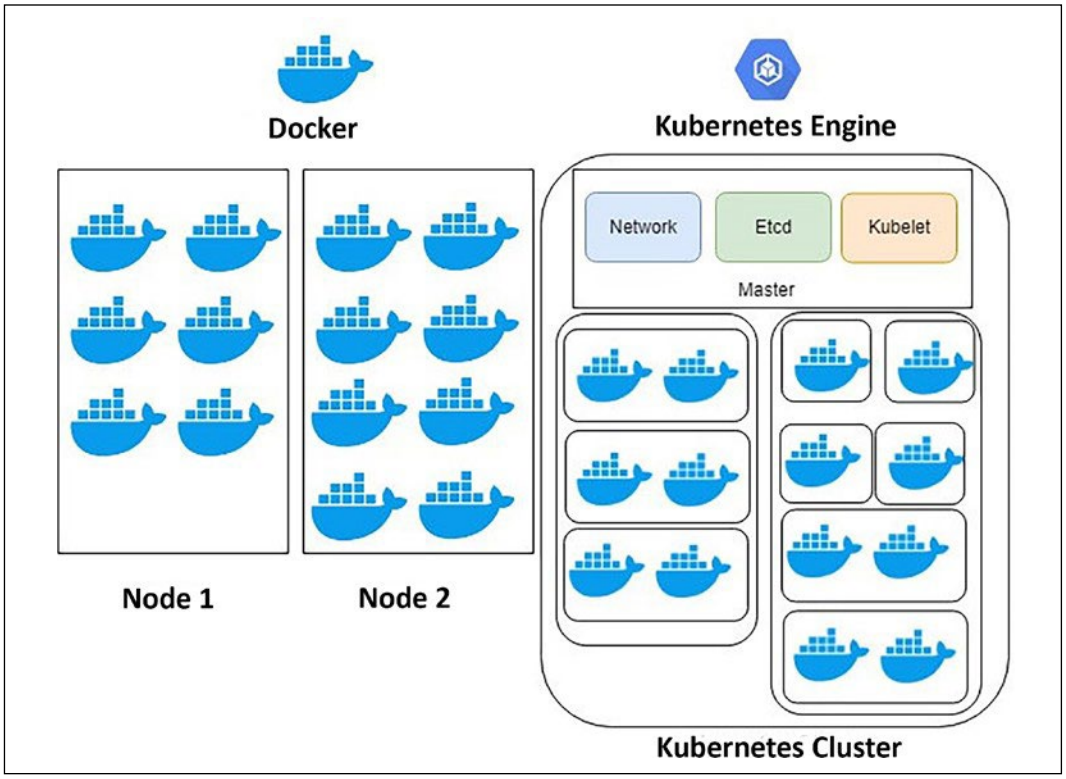


Figure 7.5: Docker and Kubernetes

Access methods	Block, file, or object
Access patterns	Sequential or random
Access frequency	Online (hot), offline (warm), or archival (cold)
Update frequency	**Write once read many** (**WORM**) or dynamic
Access availability	Availability of storage when required
Access durability	Reliability of data store to minimize any data loss
Access throughput	IOPS and data read/write per second in MBs.

Table 7.1: Factors affecting optimal storage solutions

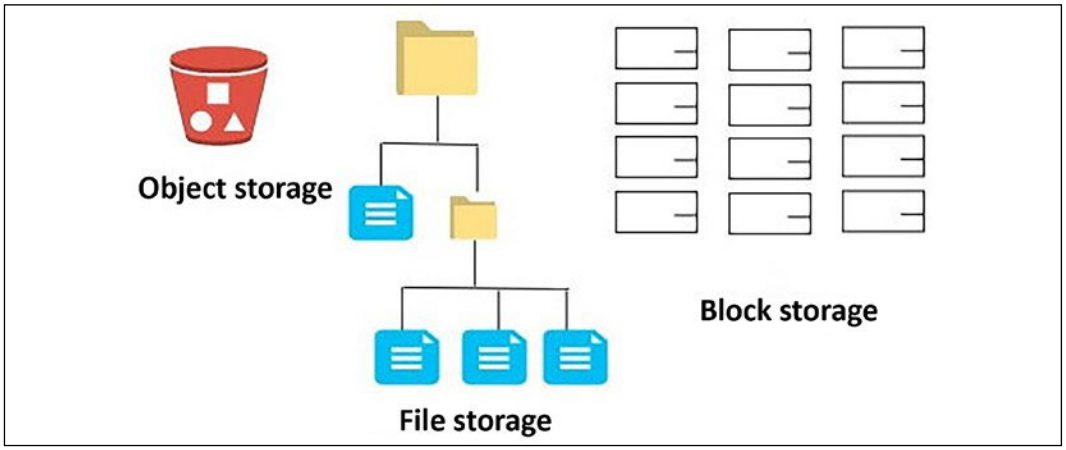


Figure 7.6: Data storage systems

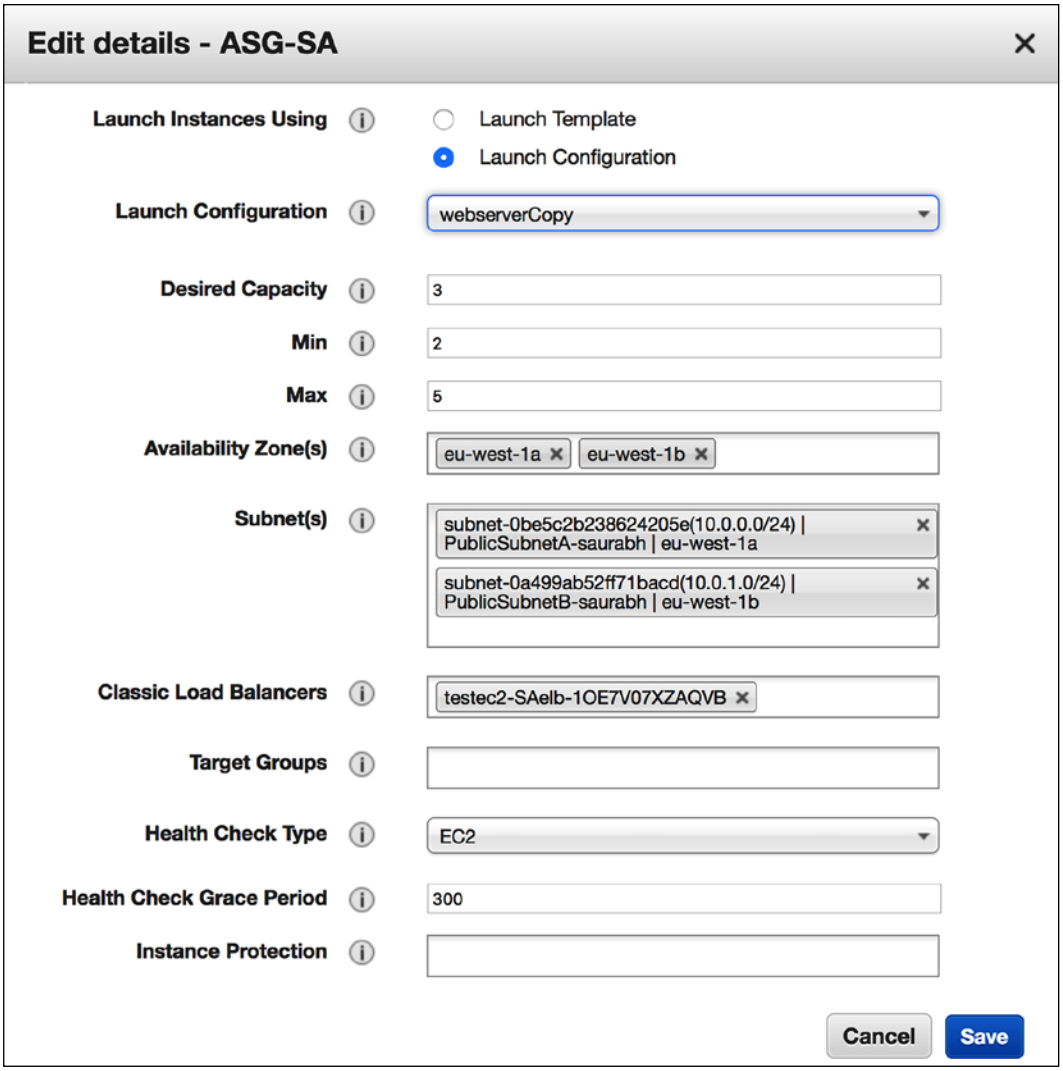


Figure 7.7: Auto-scaling configuration

Chapter 8
Security Considerations

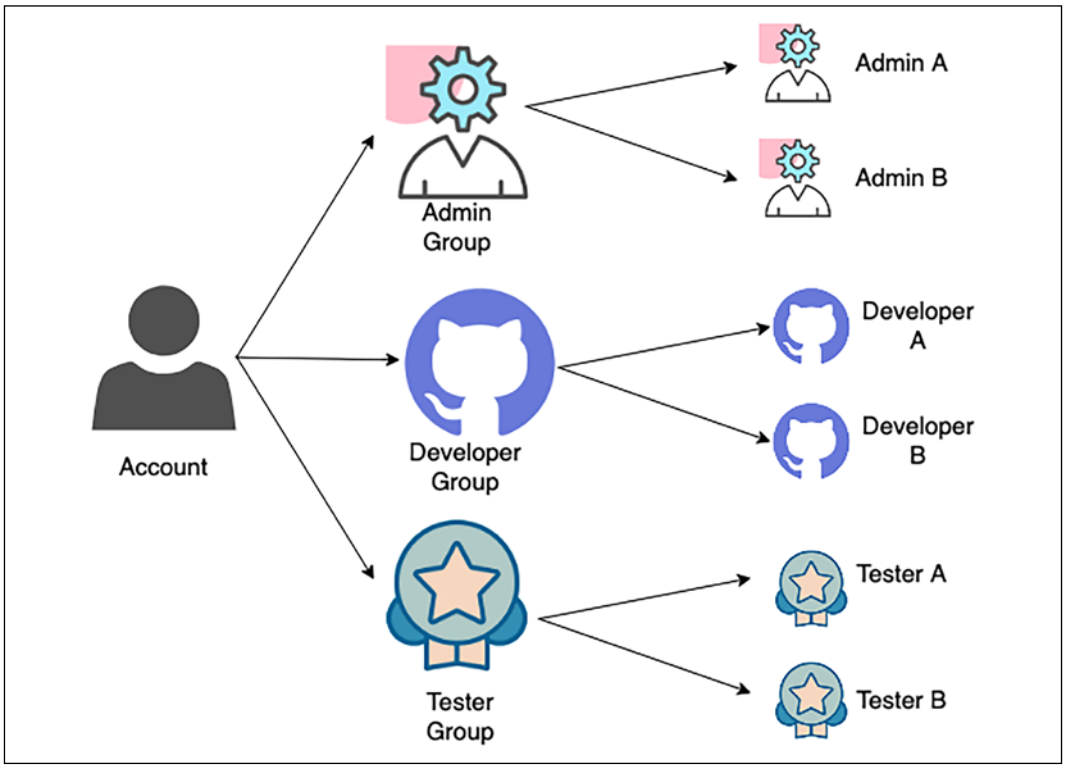


Figure 8.1: User group organization

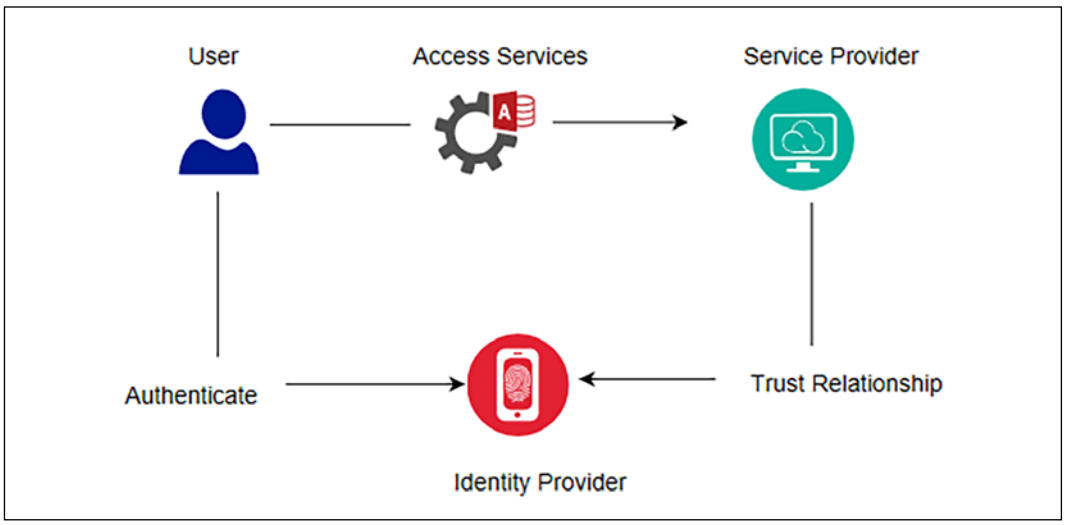


Figure 8.2: FIM authentication flow

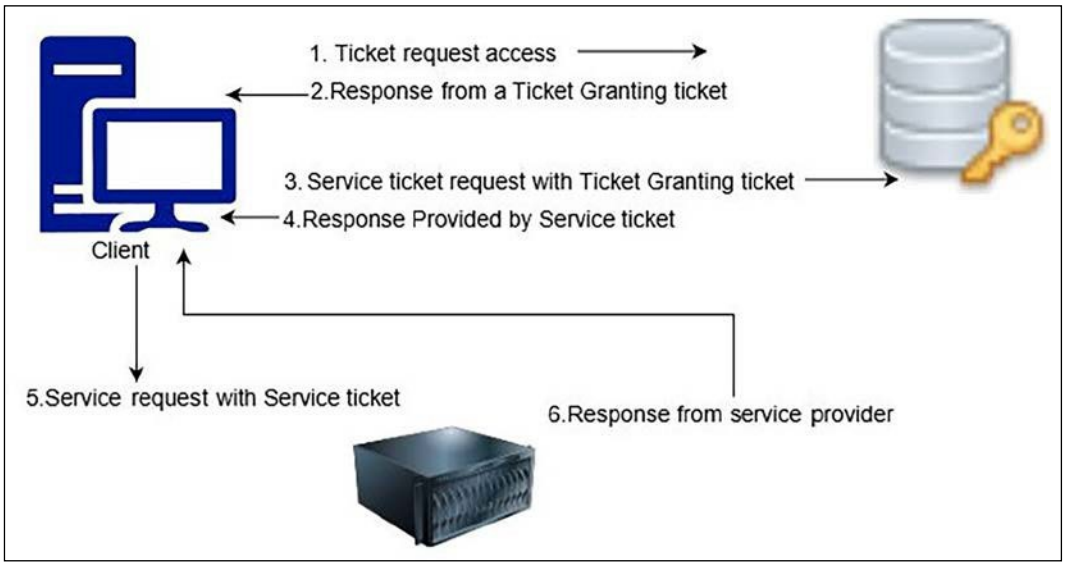


Figure 8.3: Kerberos authentication

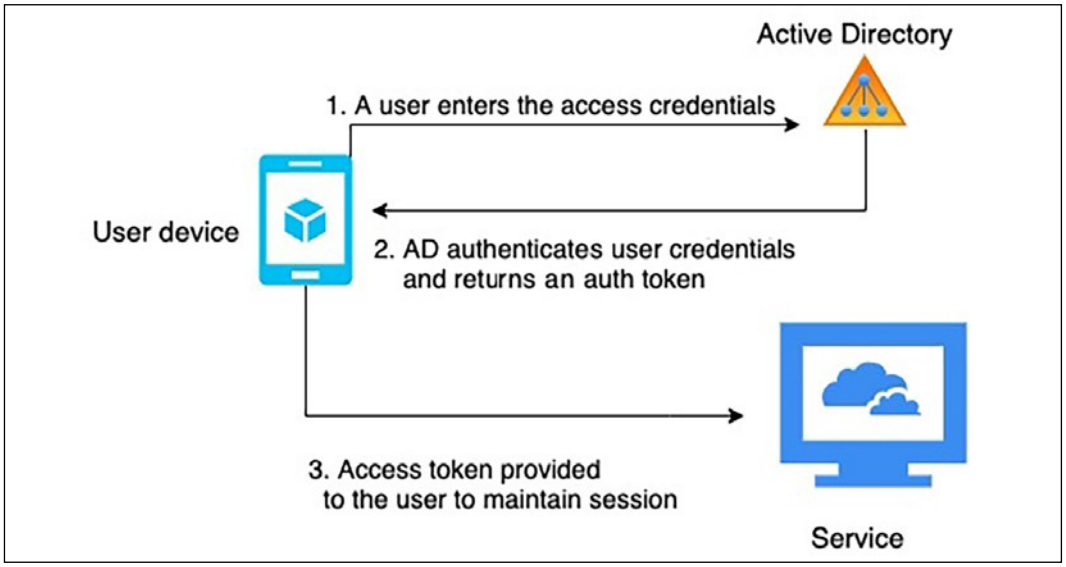


Figure 8.4: AD authentication flow

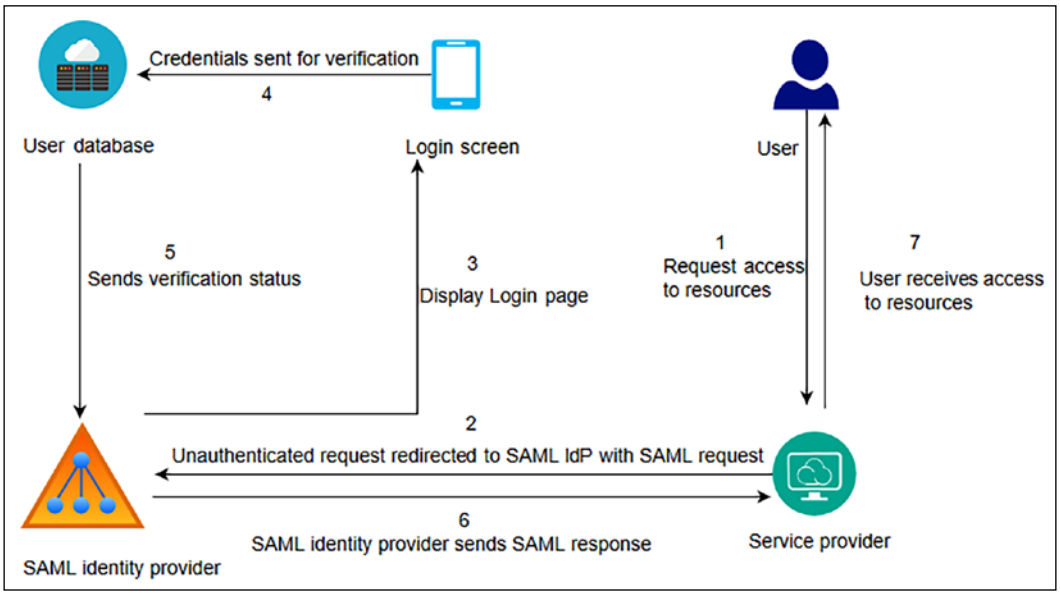


Figure 8.5: User authentication using SAML

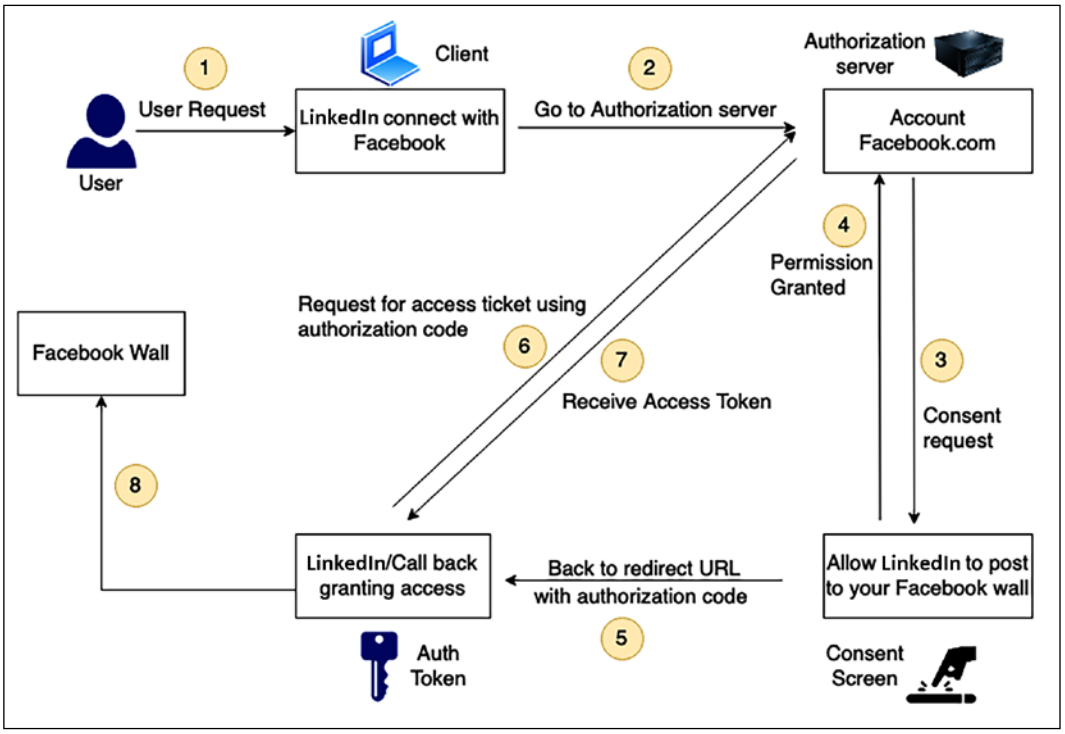


Figure 8.6: User access delegation with OAuth 2.0

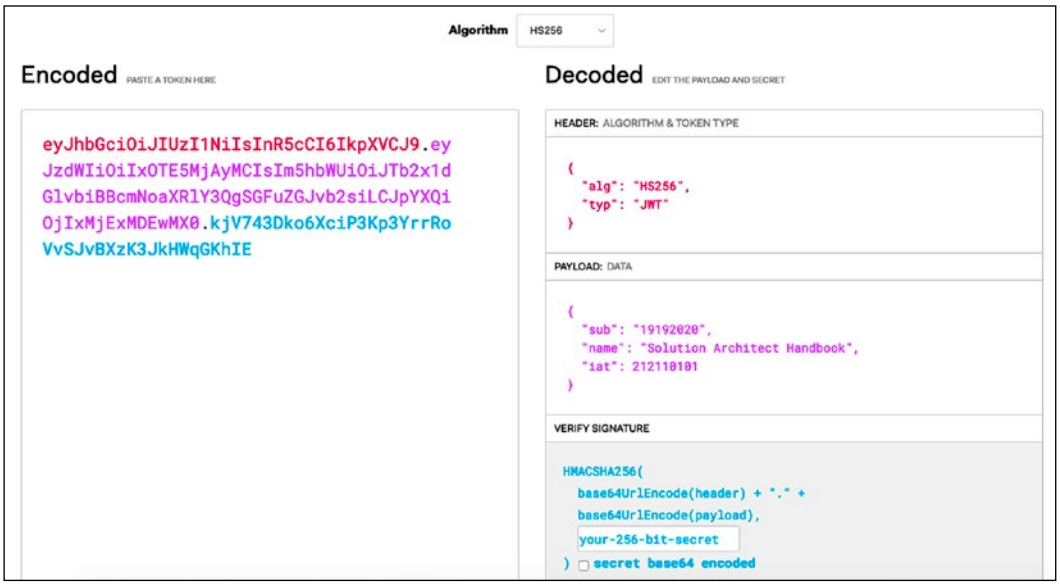


Figure 8.7: Sample JWT

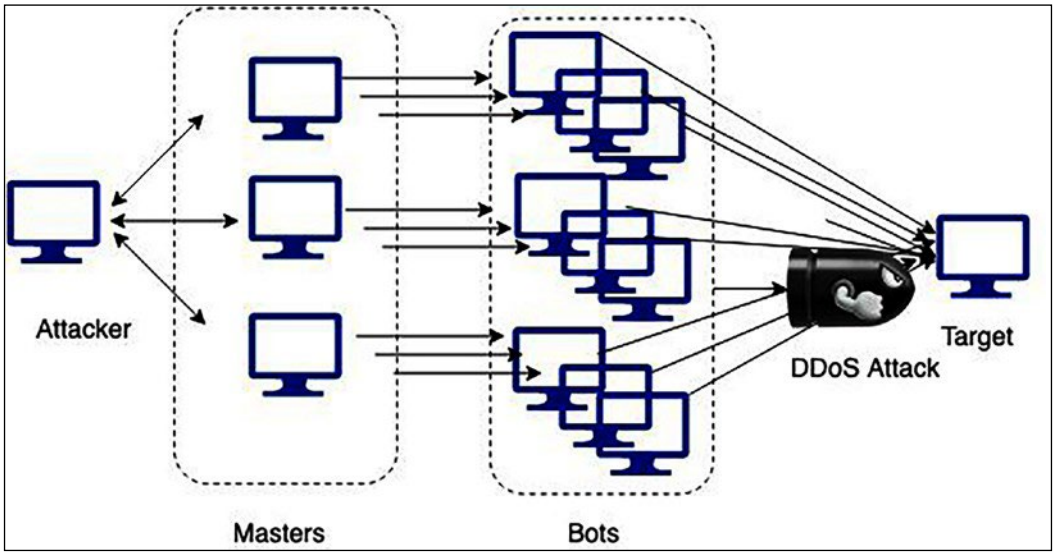


Figure 8.8: DDoS attack

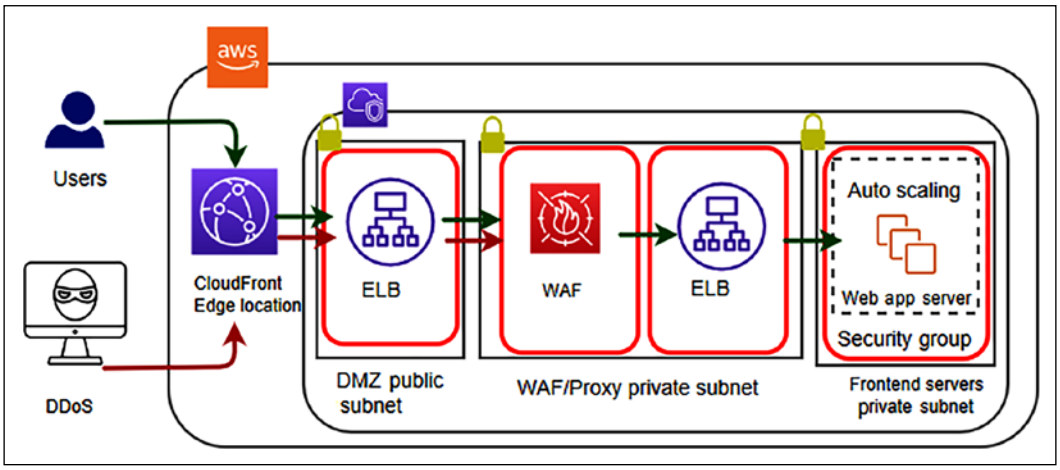


Figure 8.9: DDoS WAF sandwich mitigation strategy

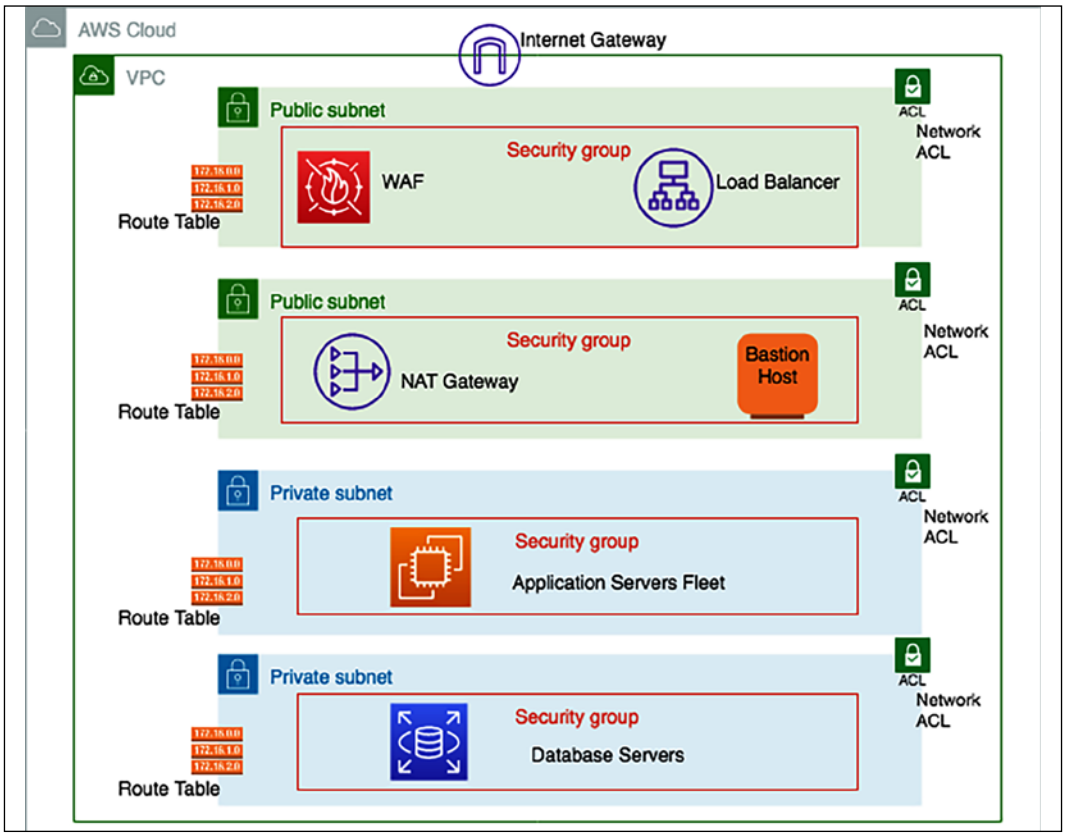


Figure 8.10: Network configuration for infrastructure security

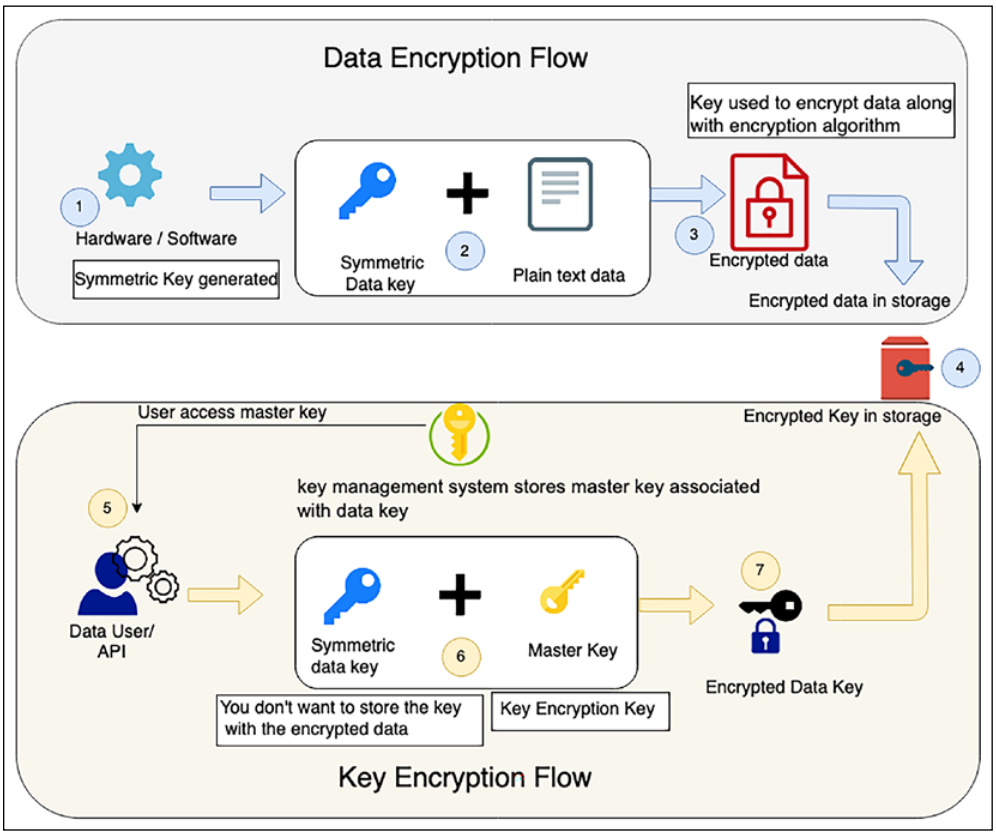


Figure 8.11: Envelope encryption

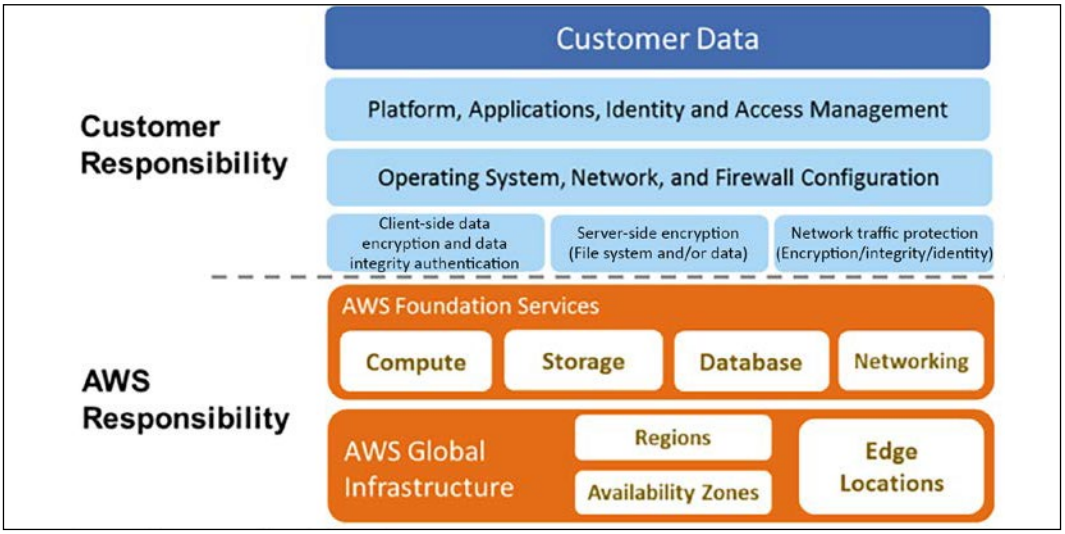


Figure 8.12: AWS cloud shared security responsibility model

Chapter 9
Architectural Reliability Considerations

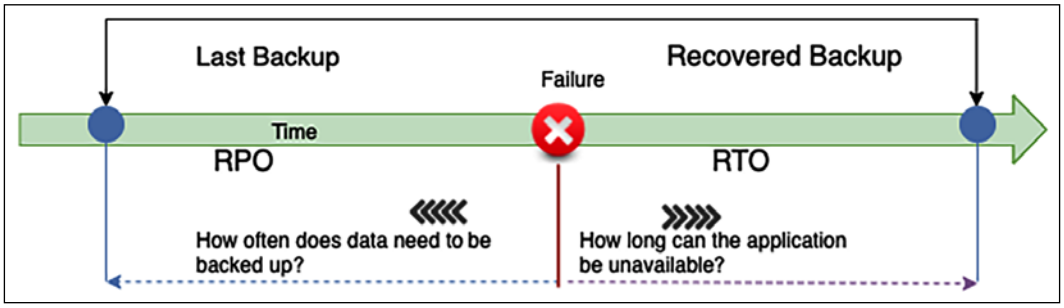


Figure 9.1: RTO and RPO

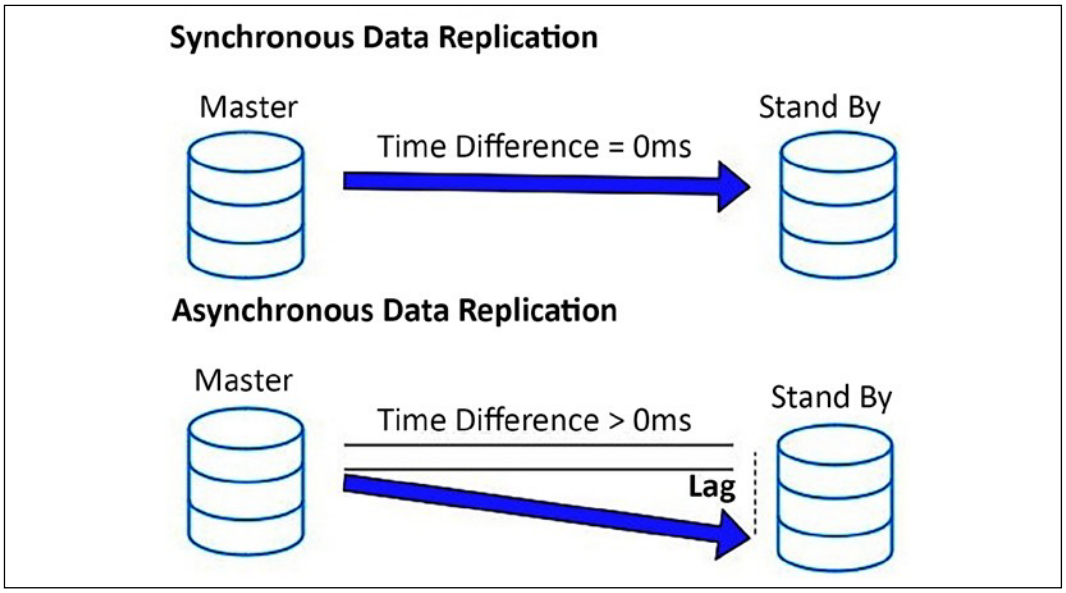


Figure 9.2: Synchronous and asynchronous data replication

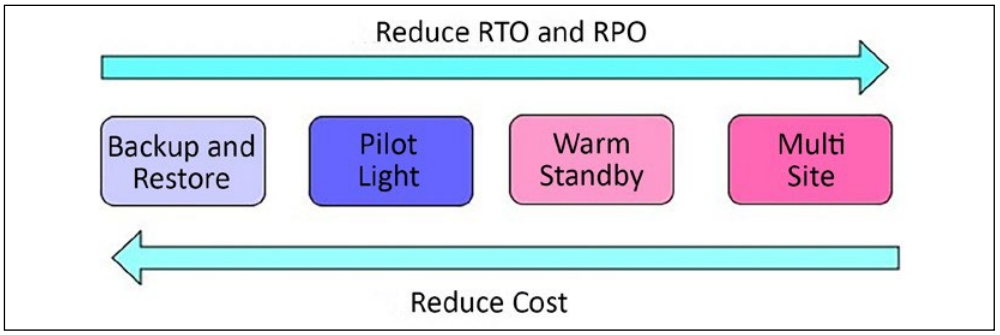


Figure 9.3: The spectrum of DR options

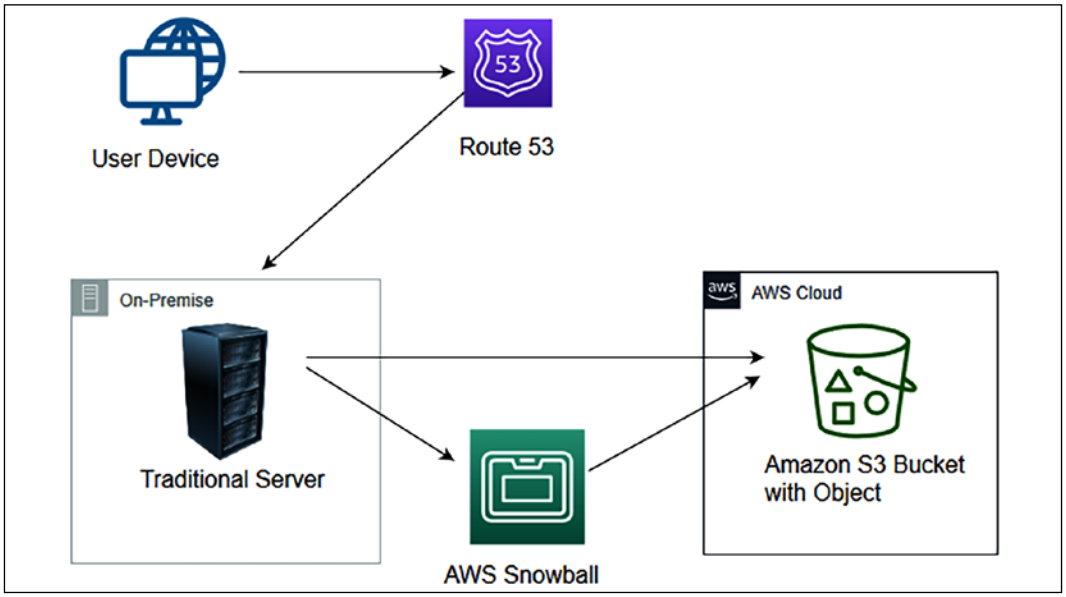


Figure 9.4: Data backup to Amazon S3 from on-premises infrastructure

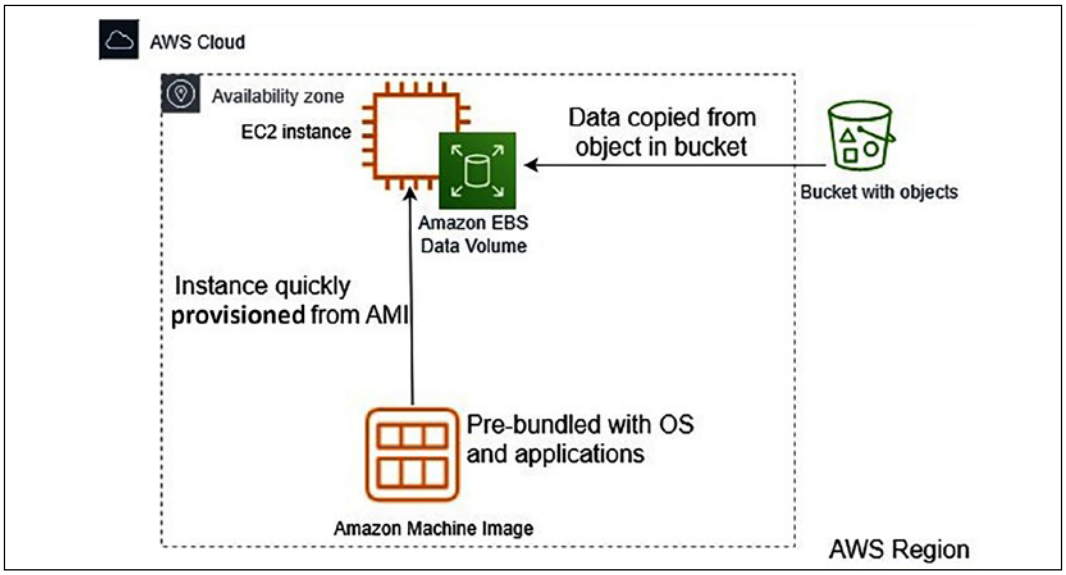


Figure 9.5: Restoring systems from Amazon S3 backups in the cloud

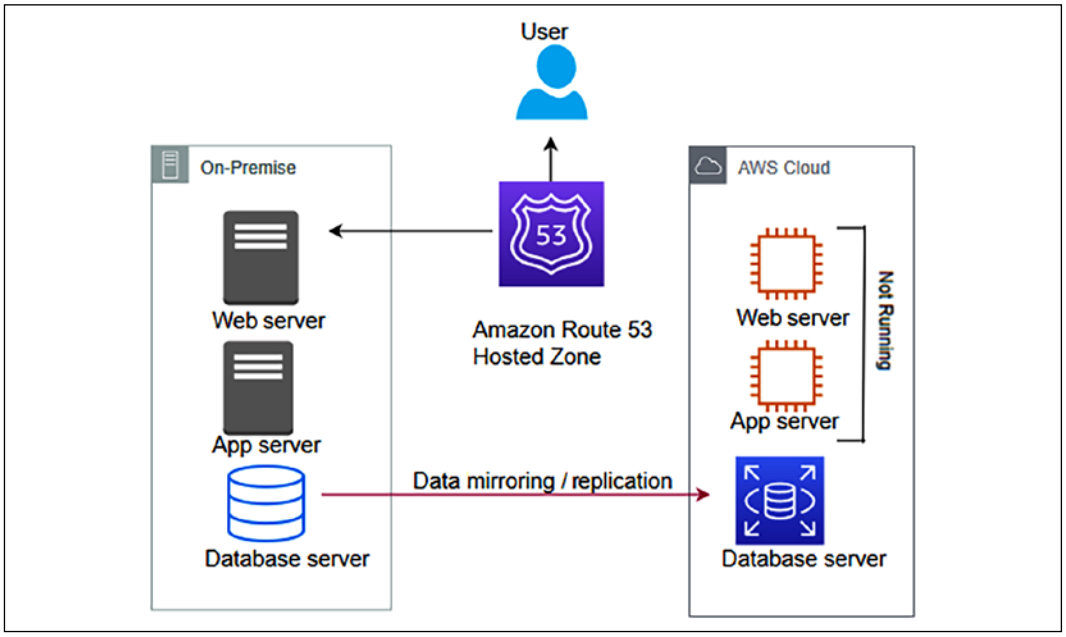


Figure 9.6: The pilot light data replication to DR site scenario

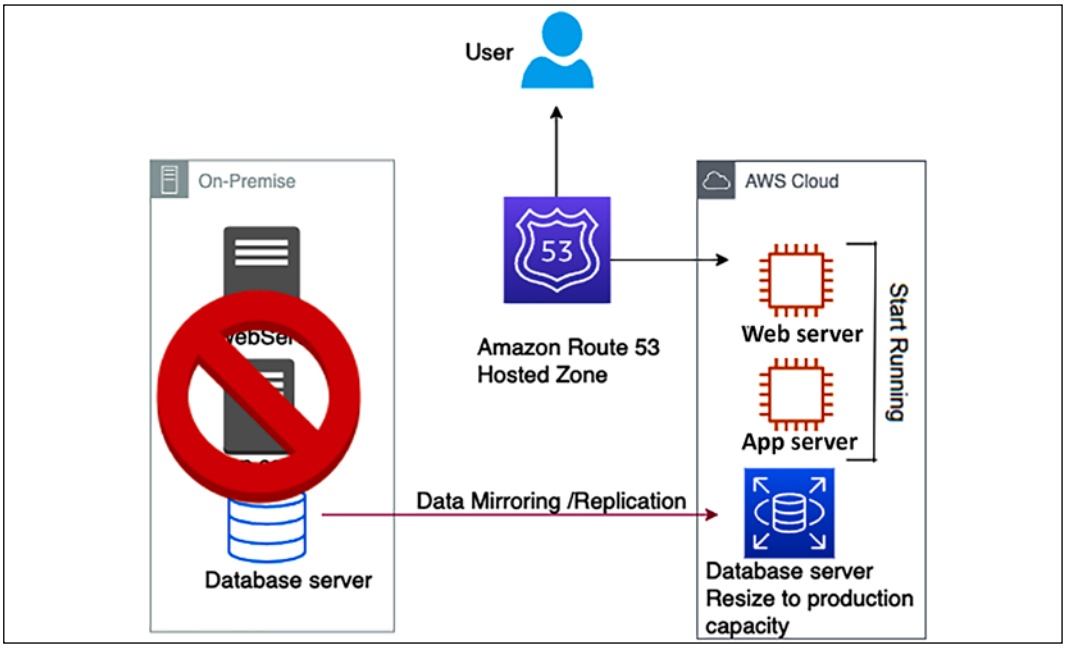


Figure 9.7: Recovery in the pilot light method

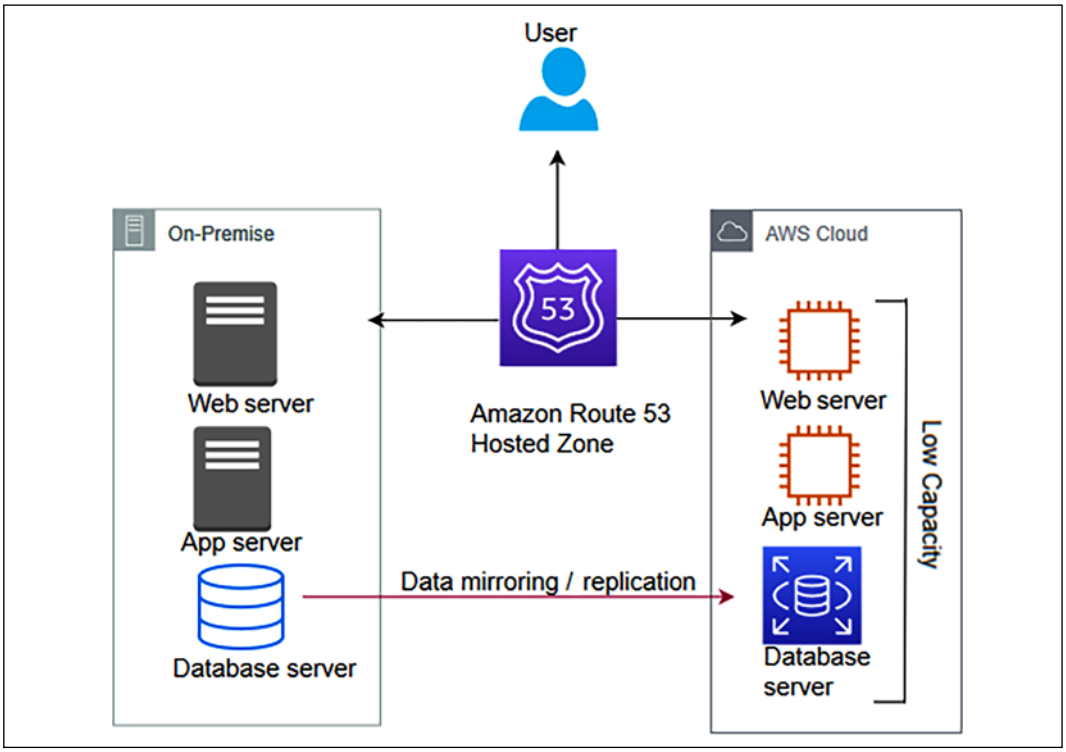


Figure 9.8: Warm standby scenario running an active-active workload with a low capacity

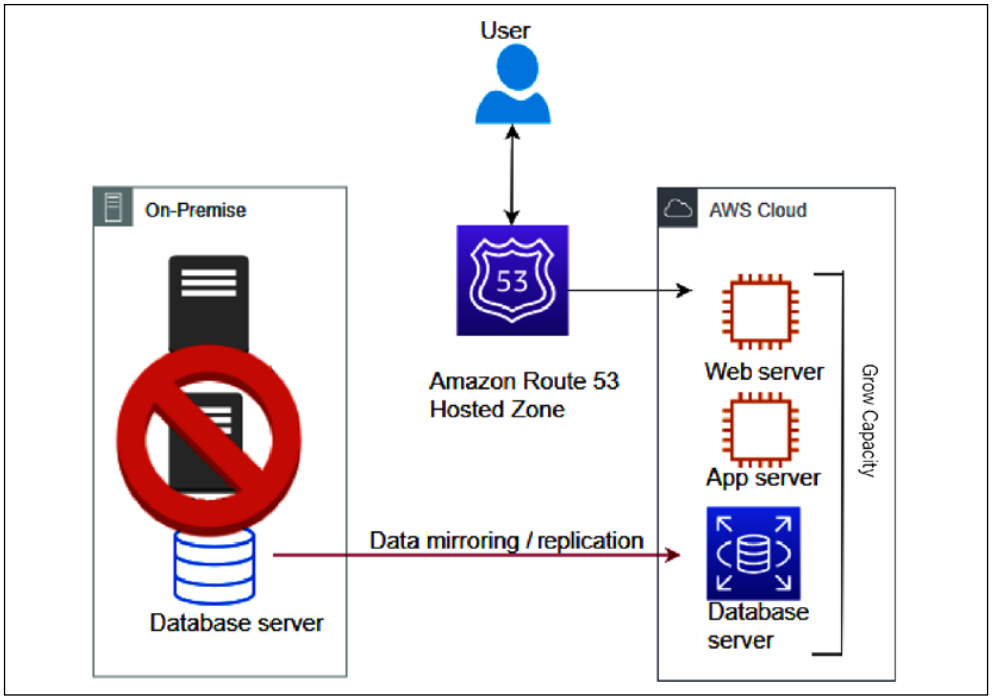


Figure 9.9: Recovery phase in the warm standby scenario

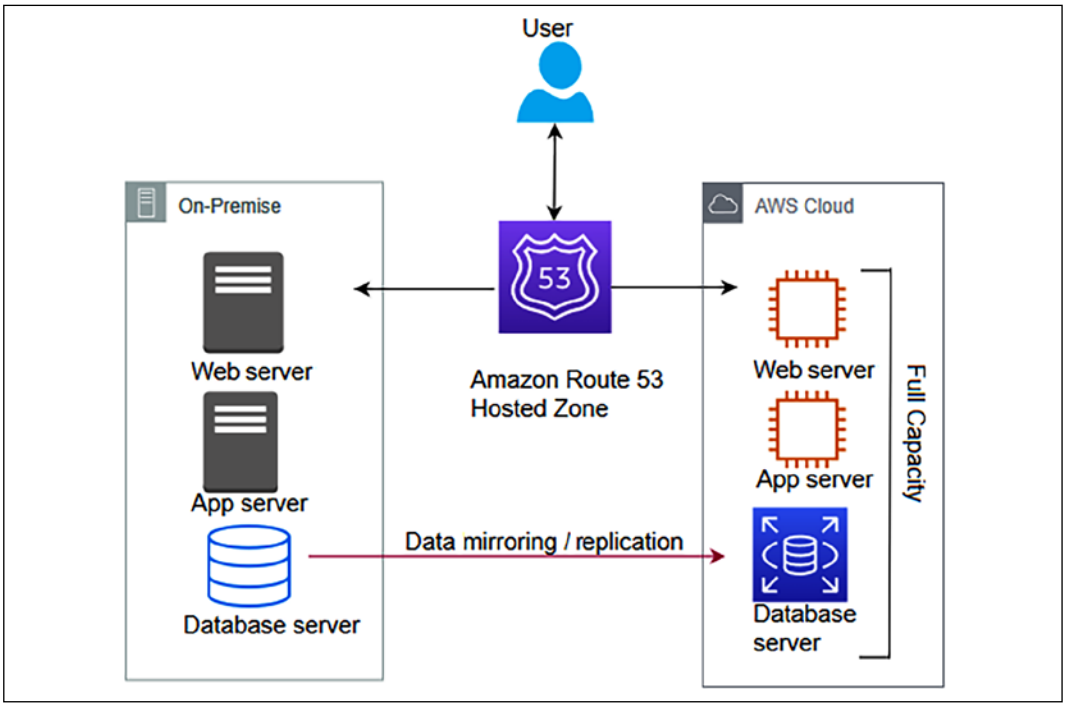


Figure 9.10: Multi-site scenario running an active-active workload with full capacity

Chapter 10
Operational Excellence Considerations

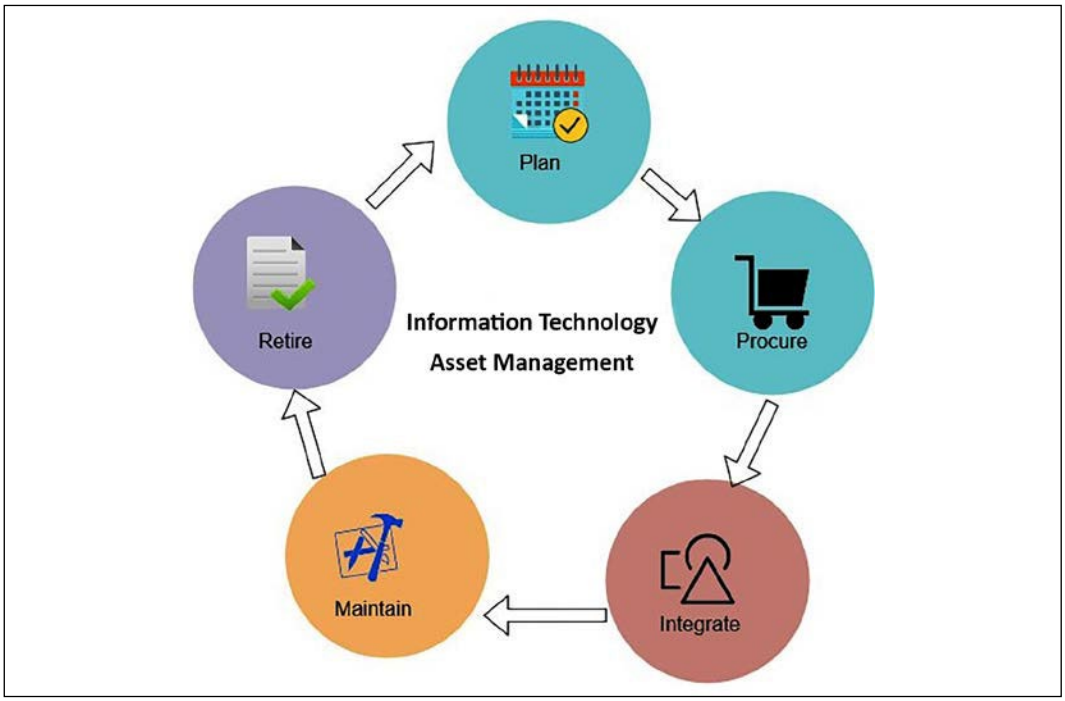


Figure 10.1: ITAM process

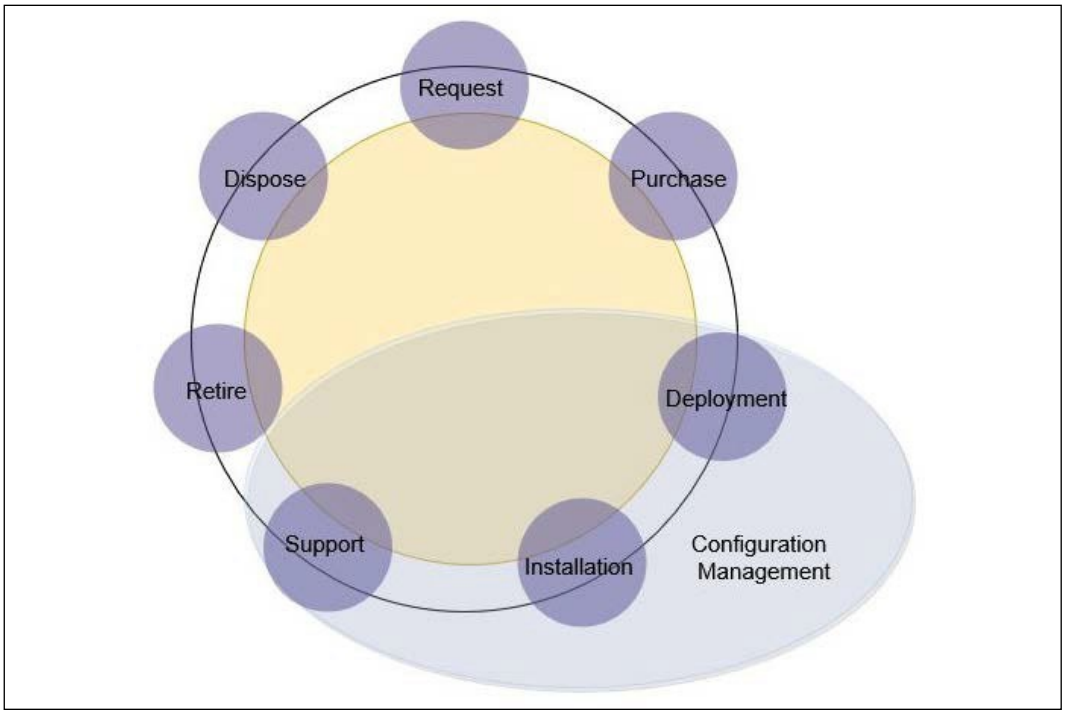


Figure 10.2: IT asset life cycle versus configuration management

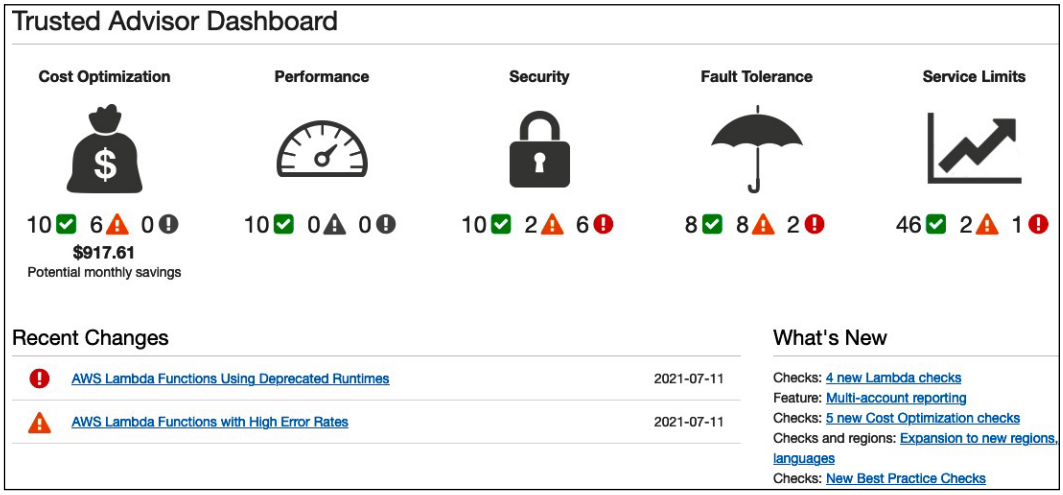


Figure 10.3: AWS Trusted Advisor dashboard

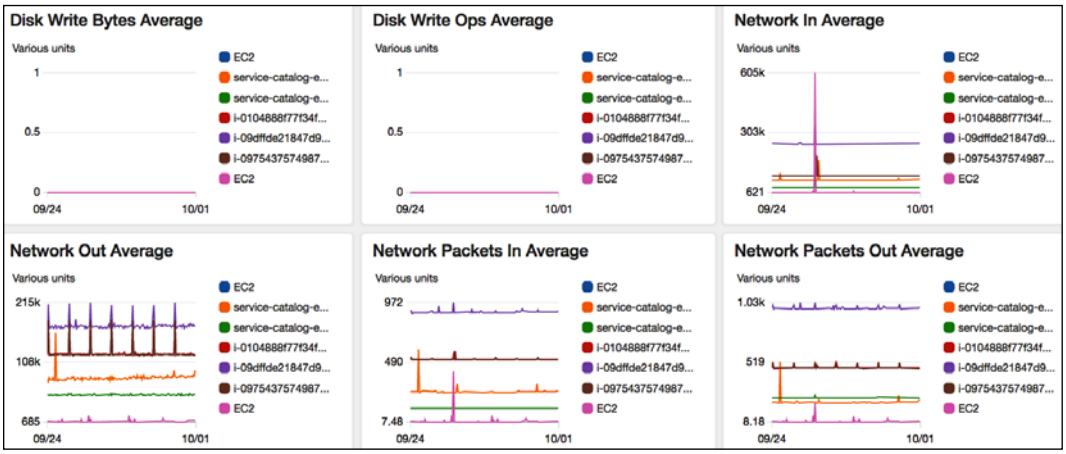


Figure 10.4: Infrastructure monitoring dashboard

Figure 10.5: Application monitoring dashboard

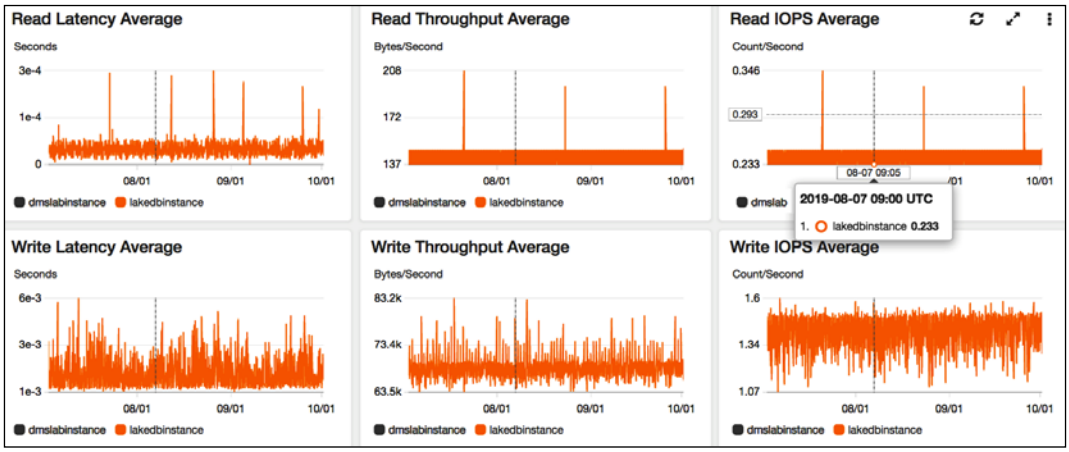


Figure 10.6: Platform monitoring dashboard for a Relational Database Management System (RDBMS)

Message
2019-02-05 20:08:14
2 789211807855 eni-0c7812c55522bd887 172.31.0.23 172.31.0.252 49232 1433 6 40 1860 1549397294 1549397893 ACCEPT OK
2 789211807855 eni-0c6918ddd57f2978f 104.248.247.78 172.31.0.202 33794 8088 6 1 40 1549397503 1549397563 REJECT OK
2 789211807855 eni-0c6918ddd57f2978f 78.128.112.98 172.31.0.202 58594 3393 6 1 40 1549397503 1549397563 REJECT OK
2 789211807855 eni-0c6918ddd57f2978f 172.104.121.206 172.31.0.202 38620 465 6 1 40 1549397503 1549397563 REJECT OK
2 789211807855 eni-0c6918ddd57f2978f 193.32.160.35 172.31.0.202 48479 40004 6 1 40 1549397503 1549397563 REJECT OK
2 789211807855 eni-0c6918ddd57f2978f 172.31.0.202 172.31.0.23 46346 1433 6 20 1280 1549397503 1549398103 ACCEPT OK
2 789211807855 eni-0c6918ddd57f2978f 172.31.0.23 172.31.0.202 1433 46346 6 20 820 1549397503 1549398103 ACCEPT OK
2 789211807855 eni-0c6918ddd57f2978f 172.31.0.202 172.31.0.23 44622 1433 6 20 1280 1549397503 1549398103 ACCEPT OK

Figure 10.7: Raw network log streamed in a centralized datastore

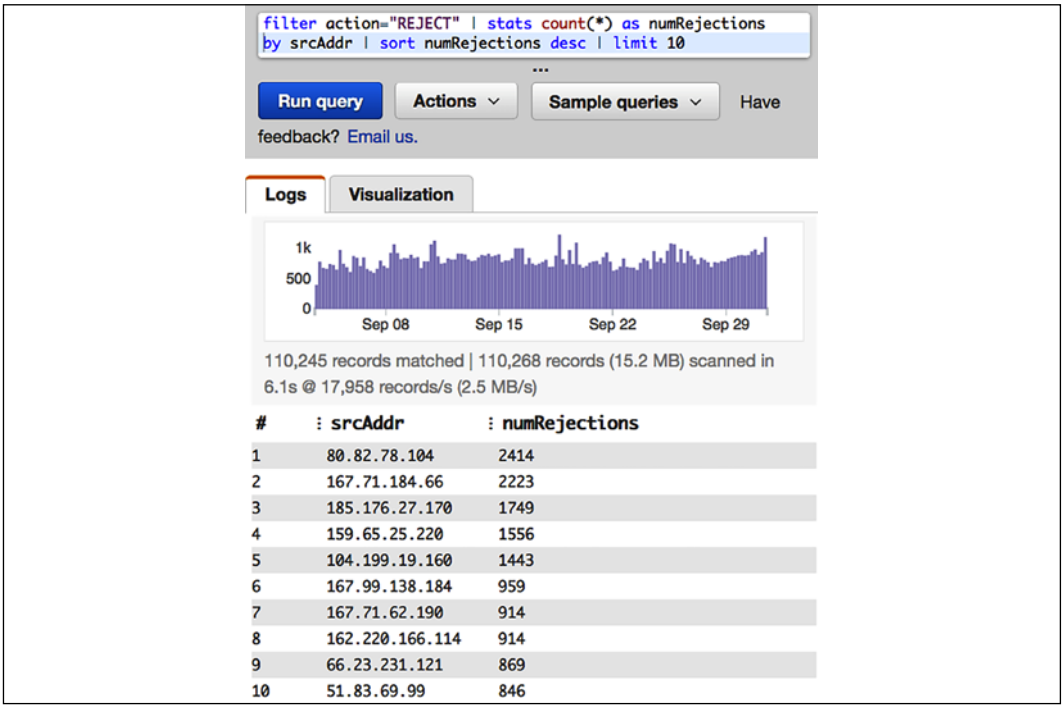


Figure 10.8: Insight from raw network log by running query

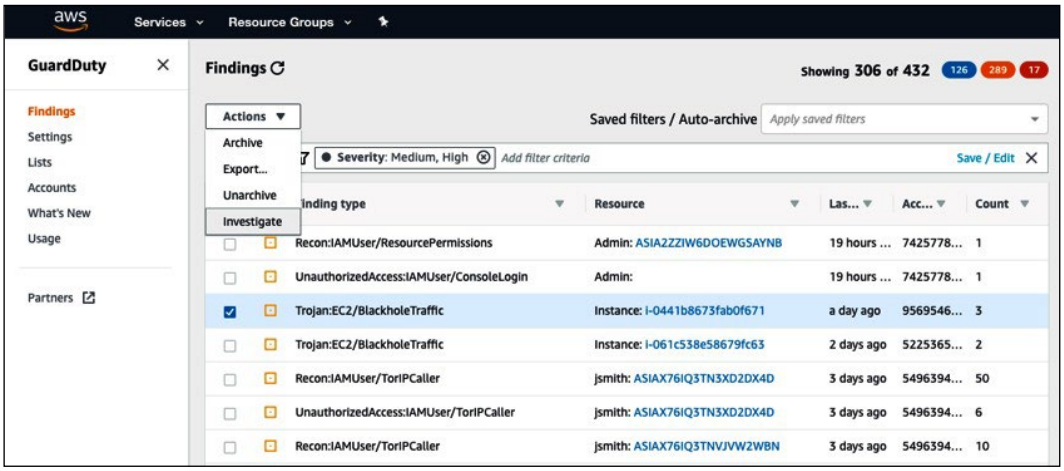


Figure 10.9: Security monitoring using Amazon GuardDuty

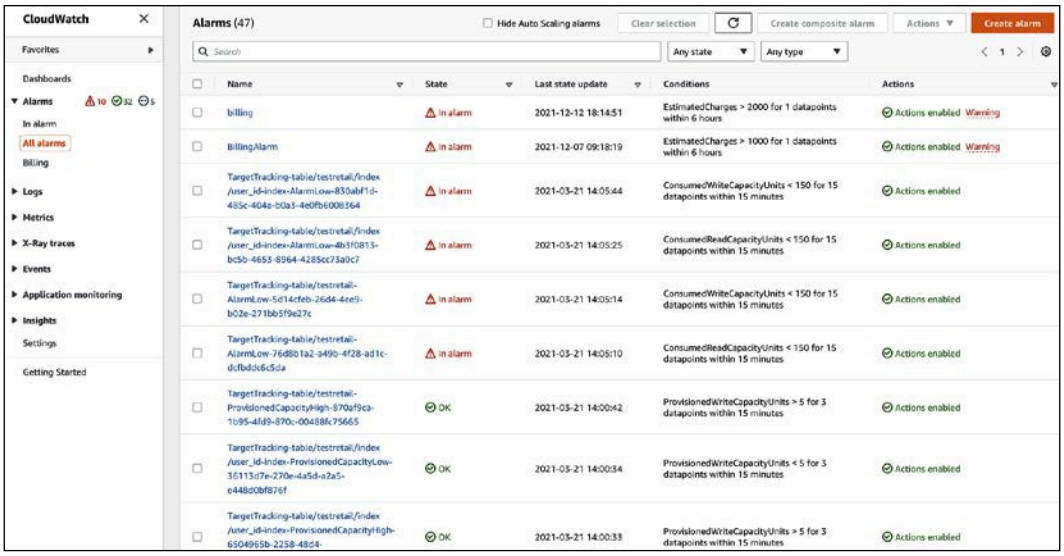


Figure 10.10: Alarm dashboard

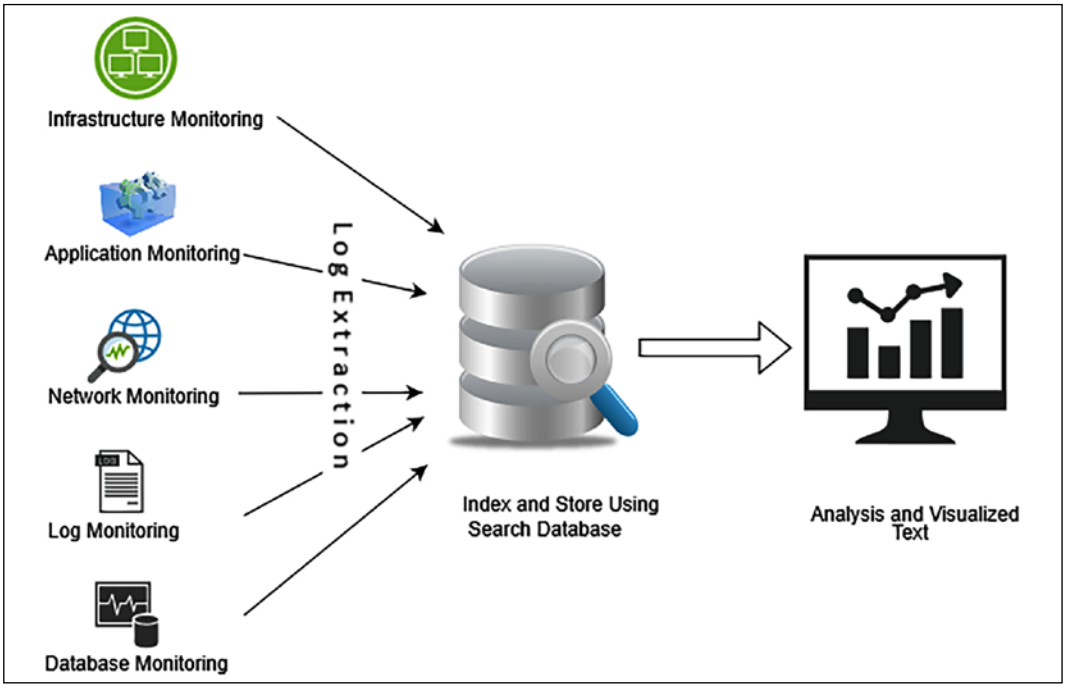


Figure 10.11: Big data approach for ITOA

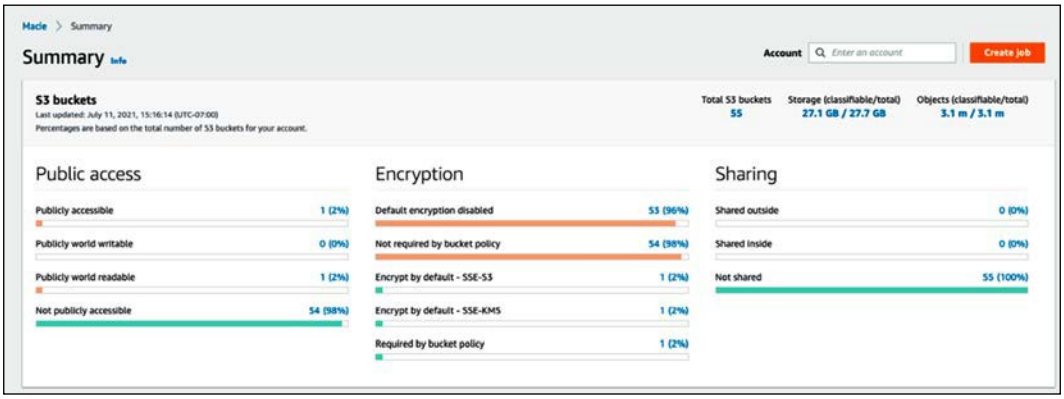


Figure 10.12: Data audit report summary from Amazon Macie

Chapter 11
Cost Considerations

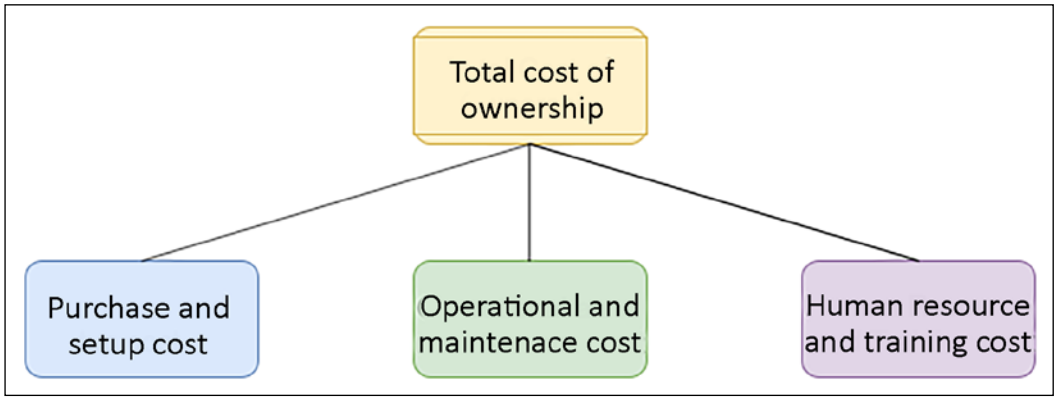


Figure 11.1: TCO for software

Budget	Forecast
Represents future results and cash flow for business objectives that you want to achieve	Represents revenue and the current situation of the business
Plans for the long term, for example, 1-5 years	Plans month to month or quarterly
Is adjusted infrequently, maybe once a year, based on business drivers	Is updated regularly based on actual business progress
Helps to decide business directions such as organization restructuring based on actual cost versus budgeted cost	Helps to adjust short-term operational costs such as additional staffing
Helps to determine performance by comparing planned cost versus actual cost	Isn't used for performance variation but for streamlining progress

Table 11.1: Differences between budget and forec

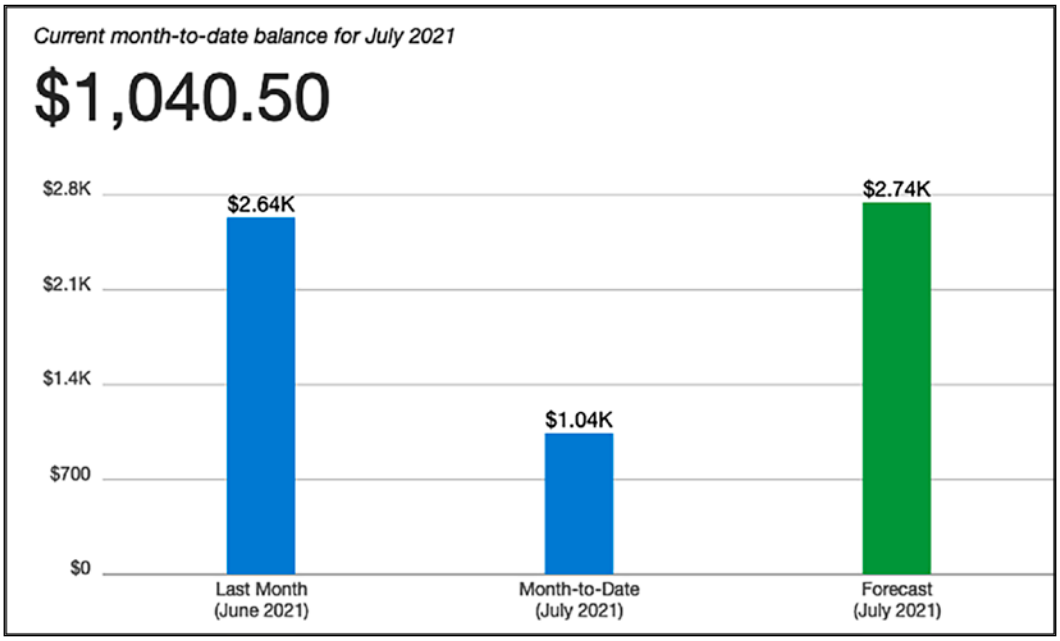


Figure 11.2: Billing and forecast report

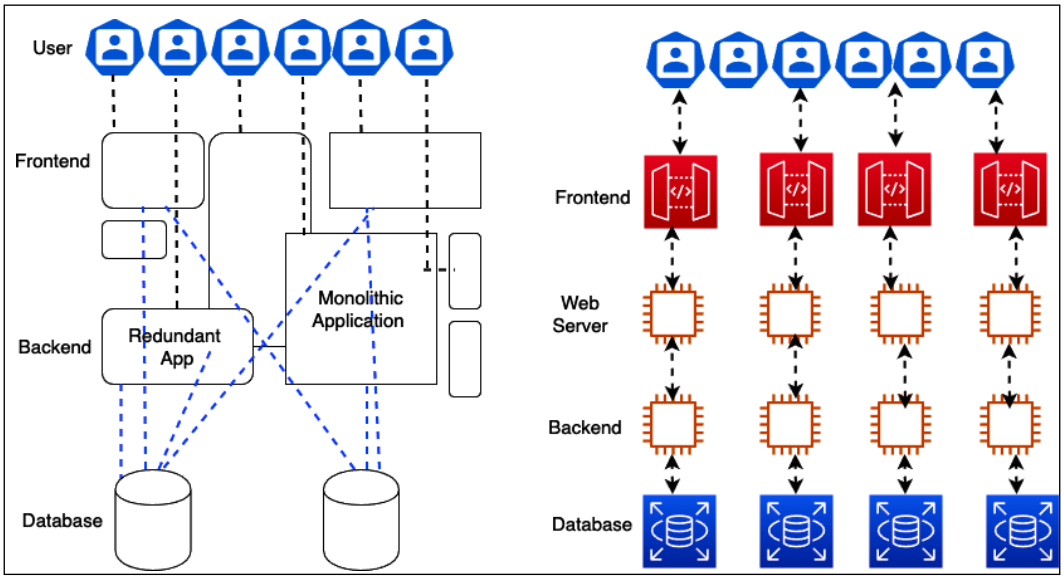


Figure 11.3: Architectural standardization

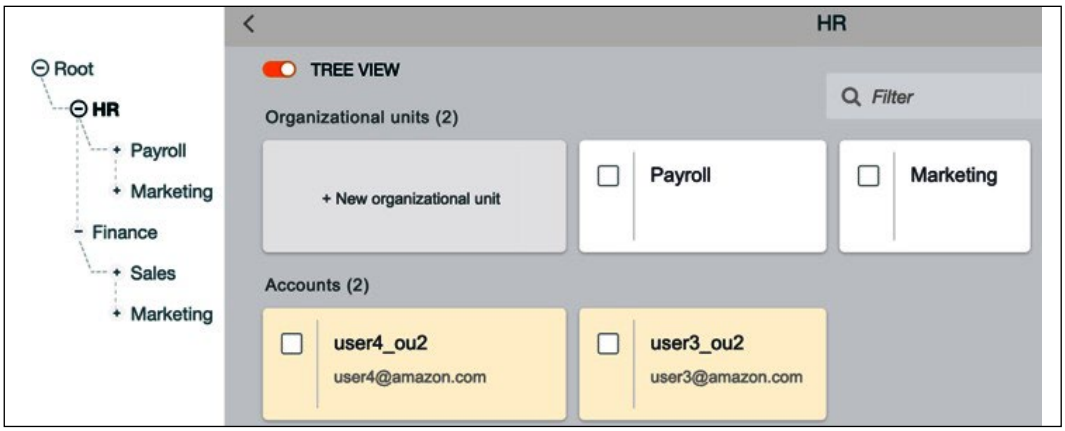


Figure 11.4: Enterprise account structure for organization units (OUs)

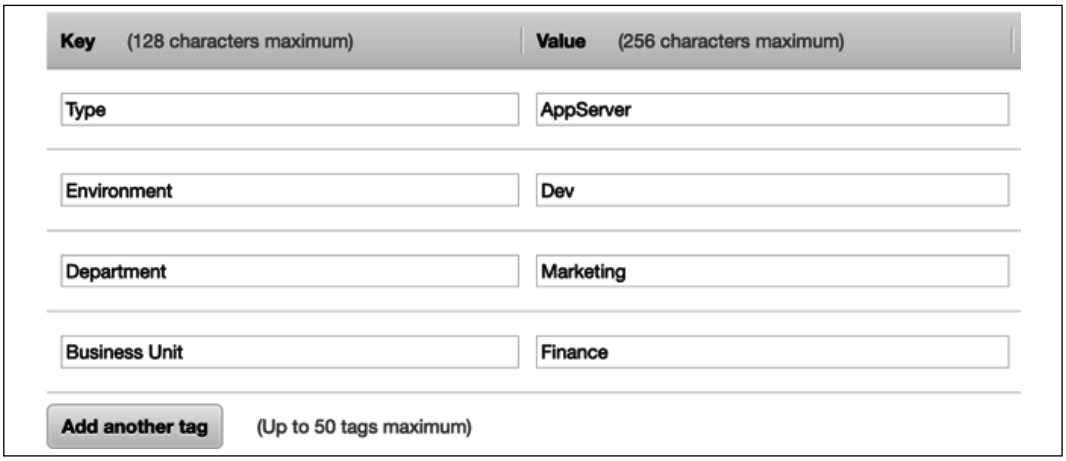
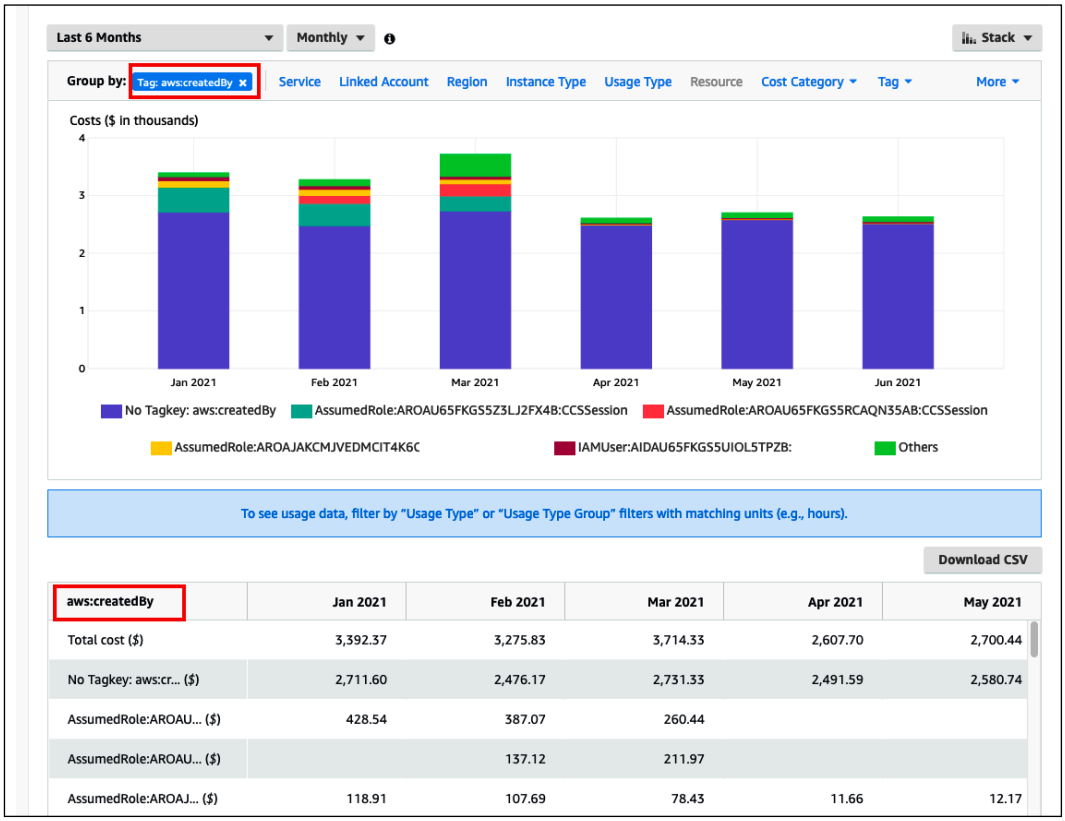


Figure 11.5: Resource tagging for cost visibility



aws:createdBy	Jan 2021	Feb 2021	Mar 2021	Apr 2021	May 2021
Total cost ($)	3,392.37	3,275.83	3,714.33	2,607.70	2,700.44
No Tagkey: aws:cr... ($)	2,711.60	2,476.17	2,731.33	2,491.59	2,580.74
AssumedRole:AROAU... ($)	428.54	387.07	260.44		
AssumedRole:AROAU... ($)		137.12	211.97		
AssumedRole:AROAJ... ($)	118.91	107.69	78.43	11.66	12.17

Figure 11.6: Resources amount expenditures dashboard for a cost tag

Service	Jan 2021	Feb 2021	Mar 2021	Apr 2021	May 2021
Total cost ($)	2,273.89	2,393.40	3,642.59	2,428.20	2,704.70
SageMaker ($)	859.36	776.71	859.36	857.13	825.09
Neptune ($)			375.51	771.09	1,038.20
Kinesis Analytics ($)	327.36	295.68	327.36	316.80	327.36
Relational Databa... ($)	538.86	487.38	788.93	1.75	1.13
DMS ($)	162.94	147.17	162.94	105.12	108.62

Figure 11.7: Resource cost and usage report

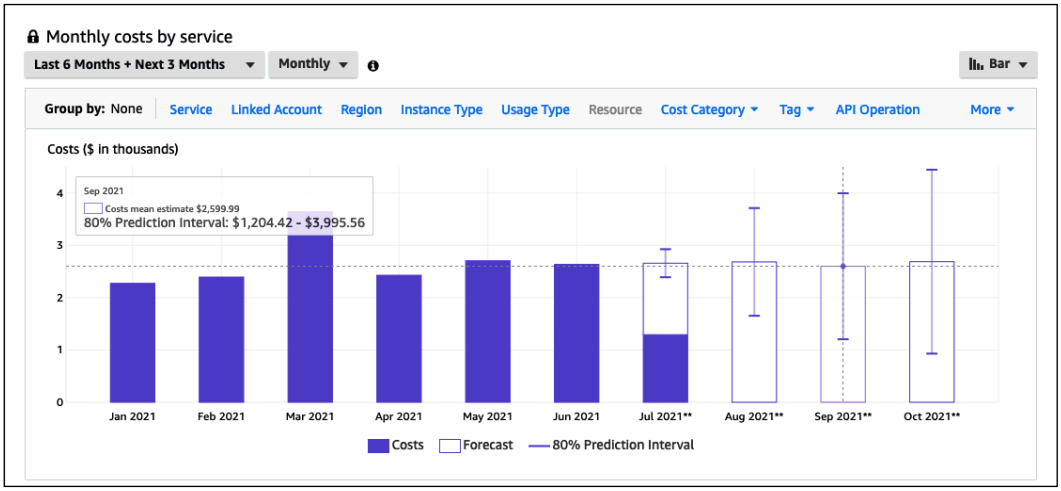


Figure 11.8: Cost trend and cost forecast report

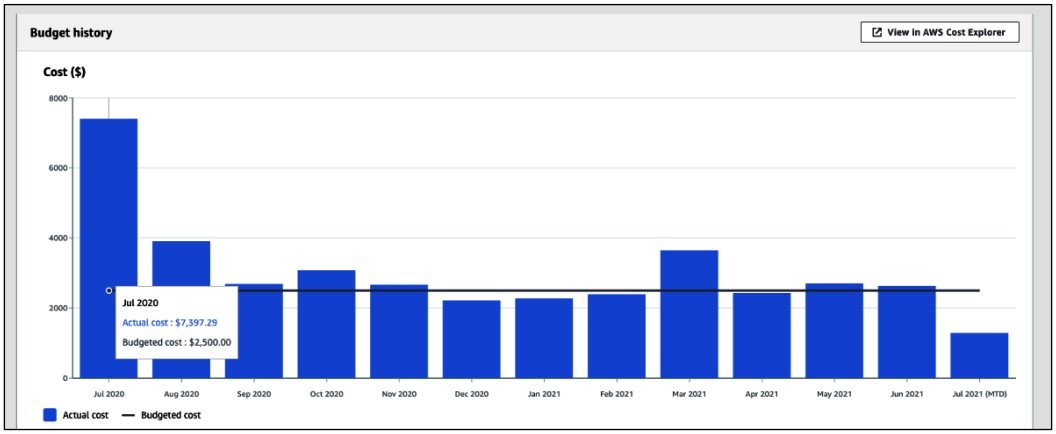


Figure 11.9: Cost and budget report

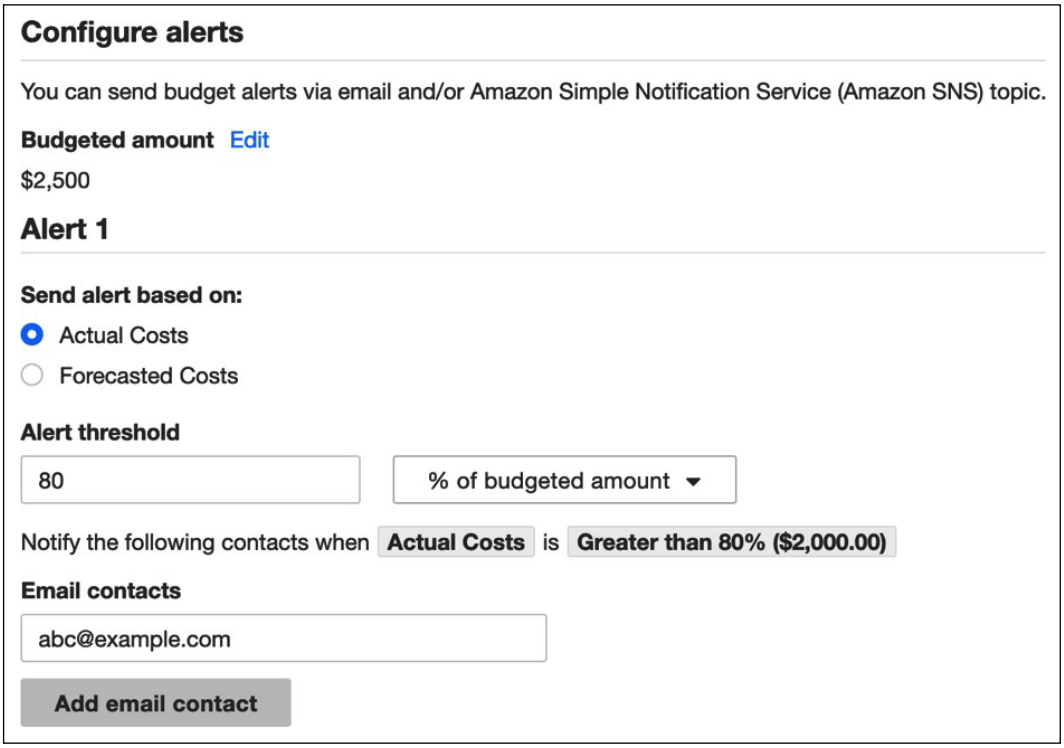


Figure 11.10: Alert based on actual cost

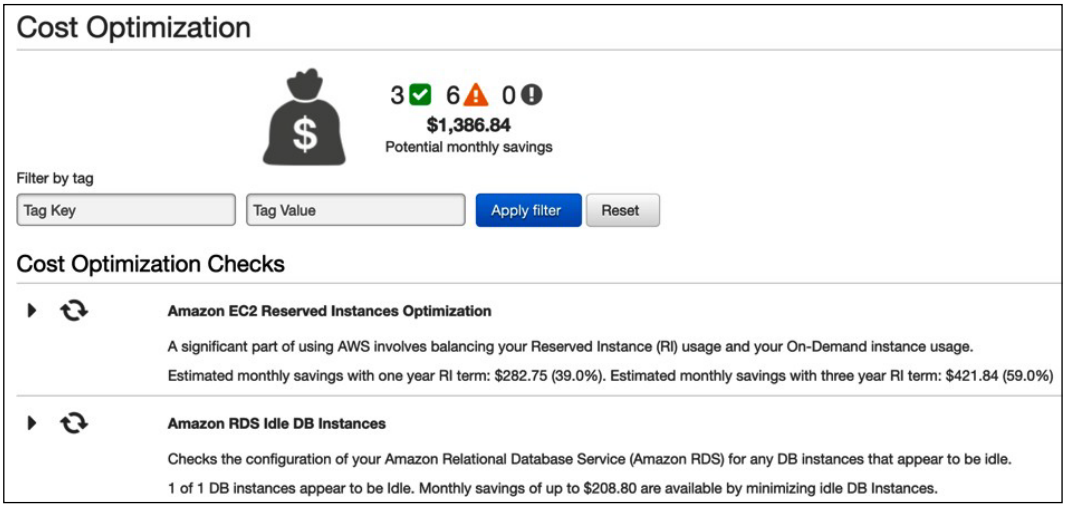


Figure 11.11: Cost-saving suggestions from AWS Trusted Advisor

Chapter 12
DevOps and Solution Architecture Framework

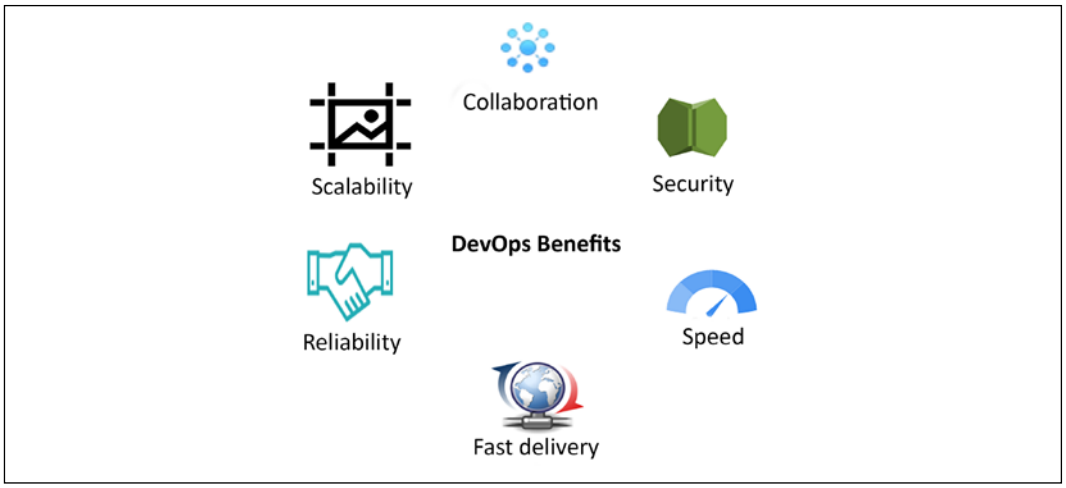


Figure 12.1: Benefits of DevOps

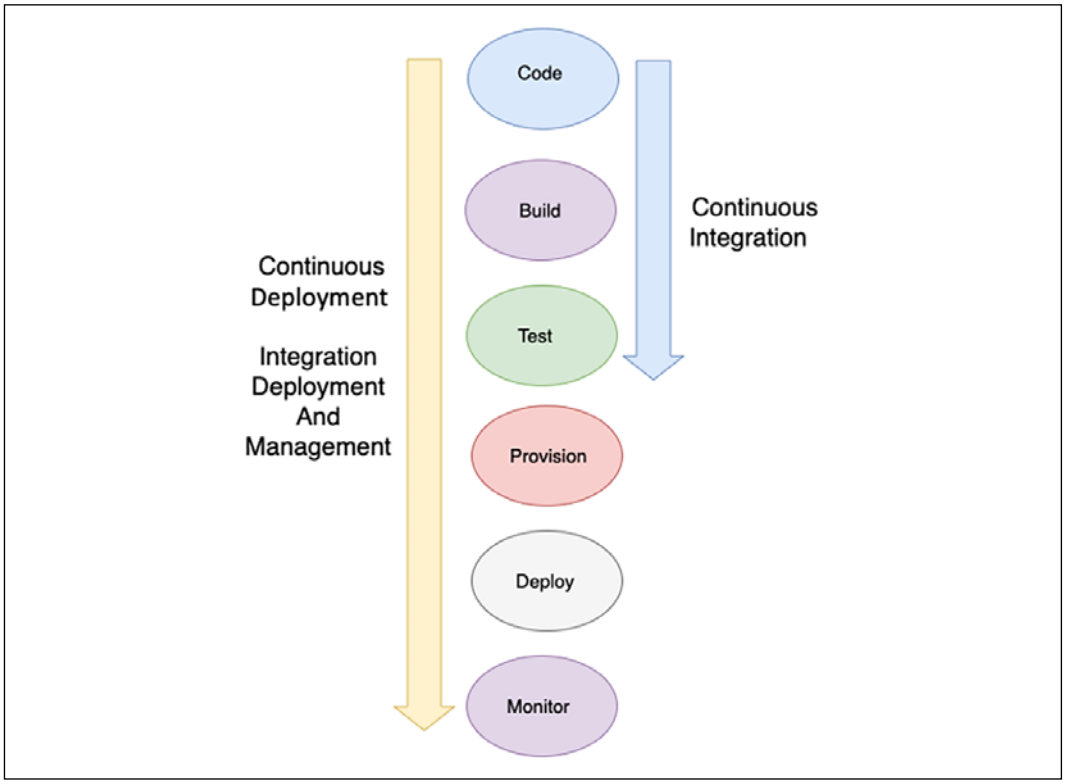


Figure 12.2: CI/CD

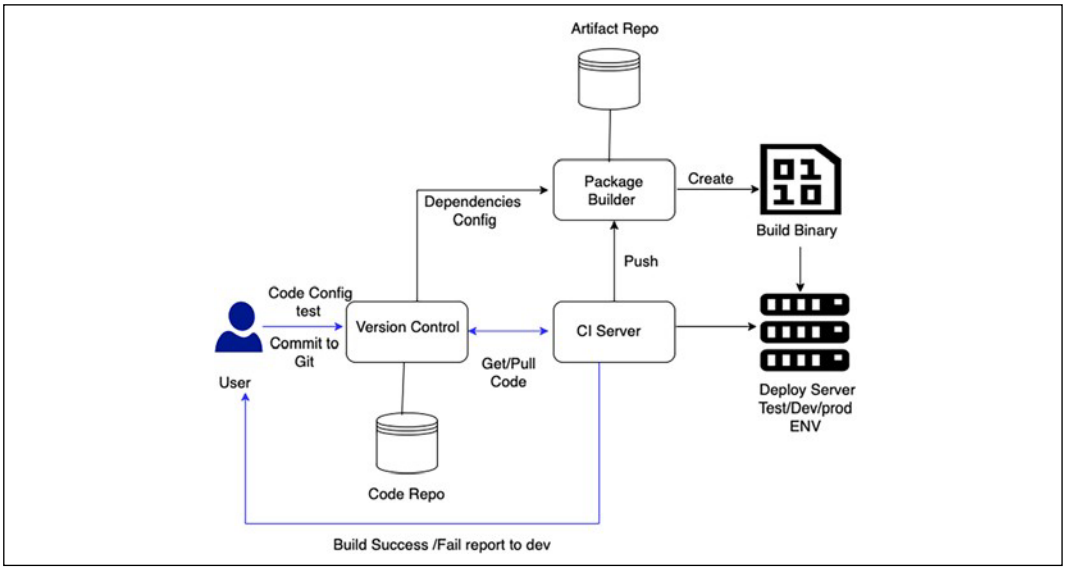


Figure 12.3: CI/CD for DevOps

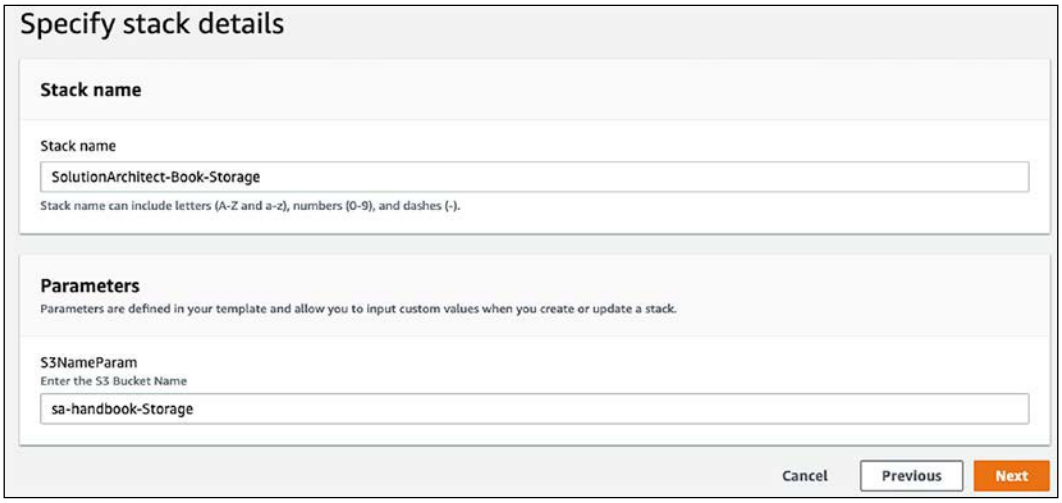


Figure 12.4: Infrastructure as code using AWS CloudFormation

```
{

    "AWSTemplateFormatVersion" : "2010-09-09",
    "Description" : "Create a S3 Storage with parameter to choose own
bucket name",
    "Parameters": {
    "S3NameParam" : {
            "Type": "String",
                "Default": "architect-book-storage",
                "Description": "Enter the S3 Bucket Name",
                "MinLength": "5",
                "MaxLength": "30"
        }
    },

    "Resources" : {
        "Bucket" : {
            "Type" : "AWS::S3::Bucket",
            "DeletionPolicy" : "Retain",
            "Properties" : {
                "AccessControl" : "PublicRead",
                "BucketName" : {
                    "Ref": "S3NameParam"
                },
                "Tags" : [ {
                    "Key": "Name", "Value": "MyBucket"
                }
                ]
            }
        }
    },

    "Outputs" : {
        "BucketName" : {
            "Description" : "BucketName",
            "Value" : {
                "Ref": "S3NameParam"
            }
        }
    }

}
```

Code listing 12.1: Providing IaC capability to automated infrastructures on the AWS cloud platform

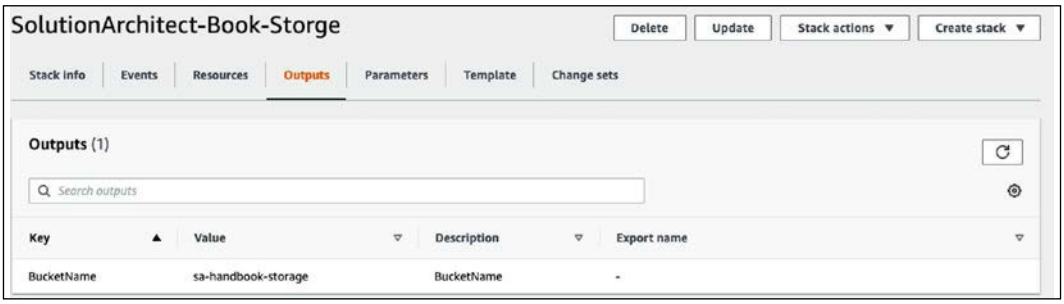


Figure 12.5: Automated AWS S3 object storage creation using AWS CloudFormation

	Ansible	**Puppet**	**Chef**
Mechanism	Controller machine applies changes to servers using **Secure Shell** (**SSH**)	Master synchronizes changes to Puppet node	Chef workstation looks for changes in Chef servers and pushes them to the Chef node
Architecture	Any server can be the controller	Centralized control by Puppet master	Centralized control by Chef server
Script Language	YAML	Domain-specific on Ruby	Ruby
Scripting Terminology	Playbook and roles	Manifests and modules	Recipes and cookbooks
Test Execution	Sequential order	Non-sequential order	Sequential order

sTable 12.2: Differences between the Ansible, Puppet, and Chef configuration management tools

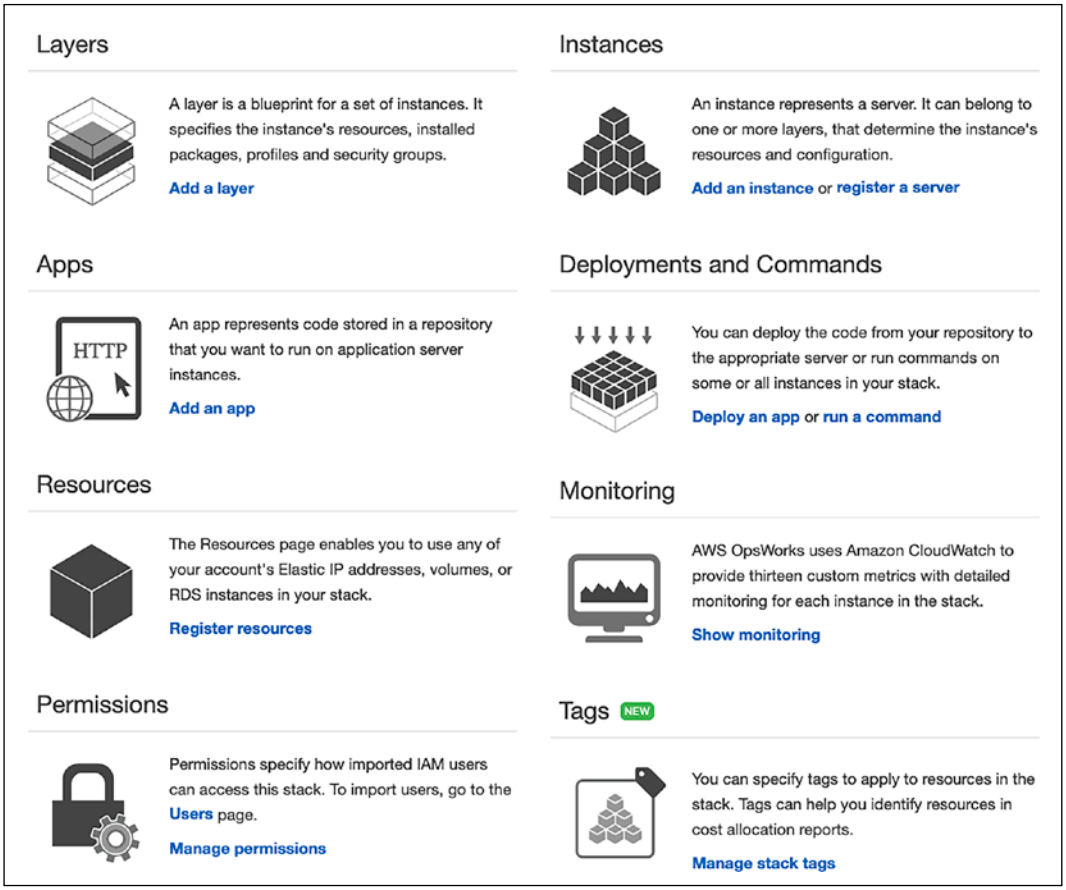


Figure 12.6: AWS OpsWorks service capabilities for managed Chef and Puppet

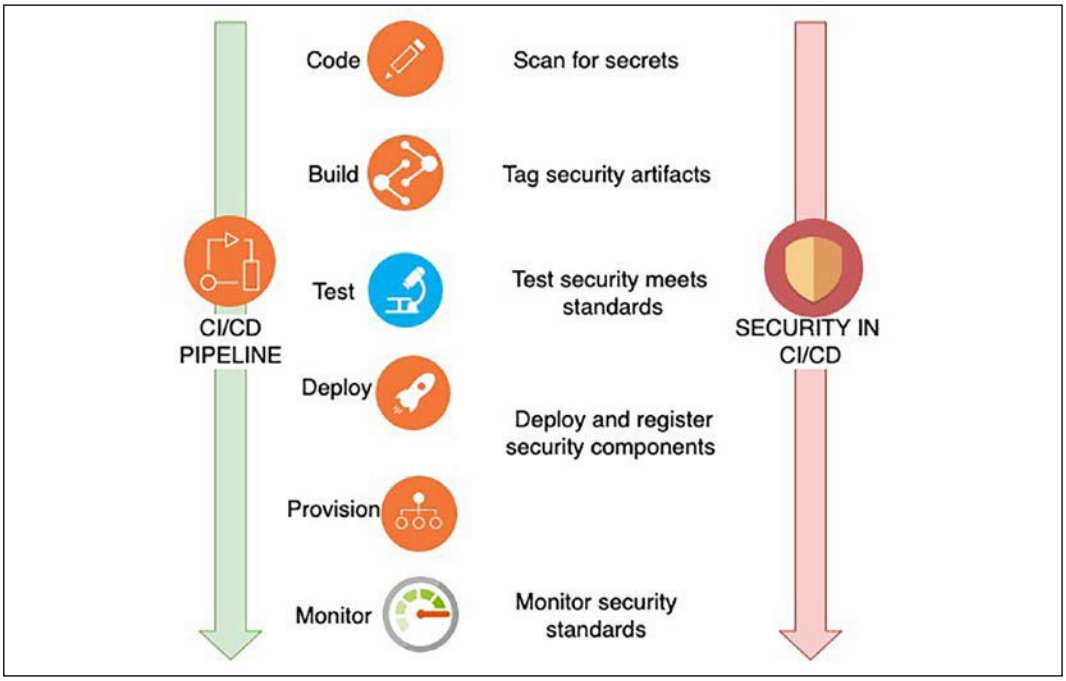


Figure 12.7: DevSecOps and CI/CD

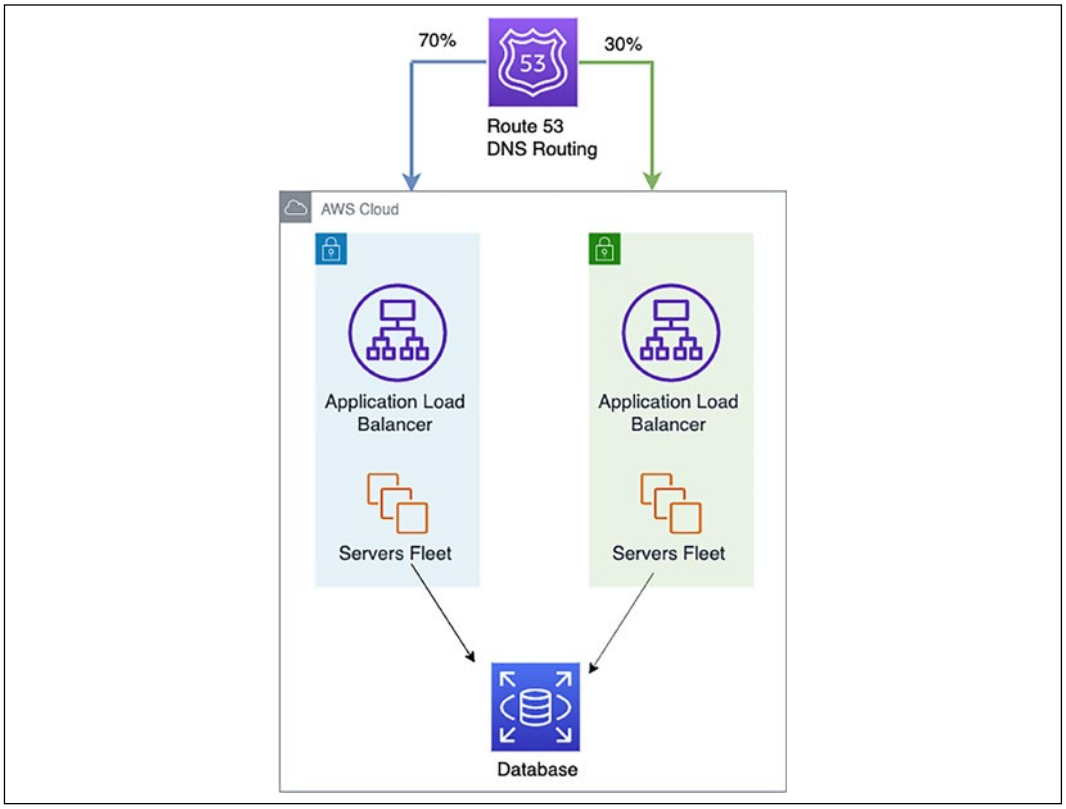


Figure 12.8: Blue-green deployment DNS gradual cutover

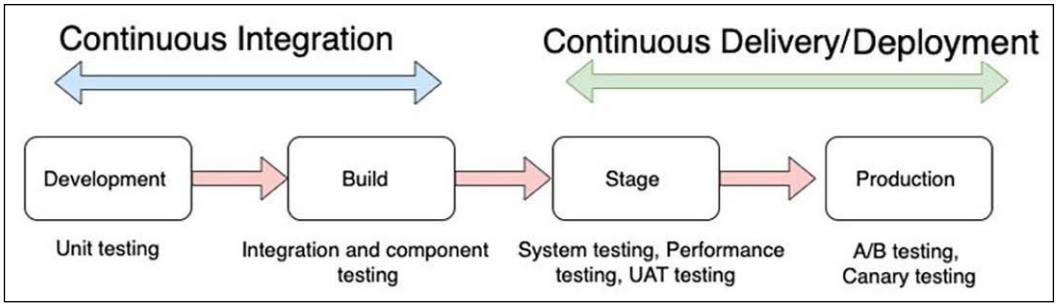


Figure 12.9: Continuous testing in CI/CD

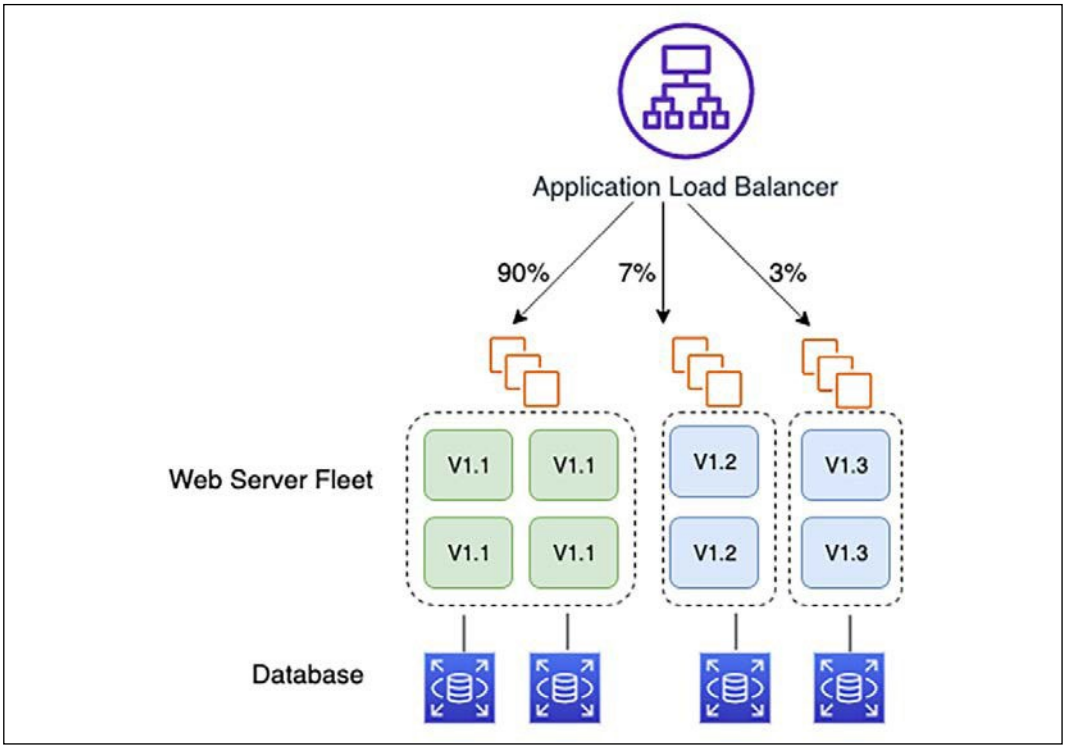


Figure 12.10: Split users by feature experiment using A/B testing

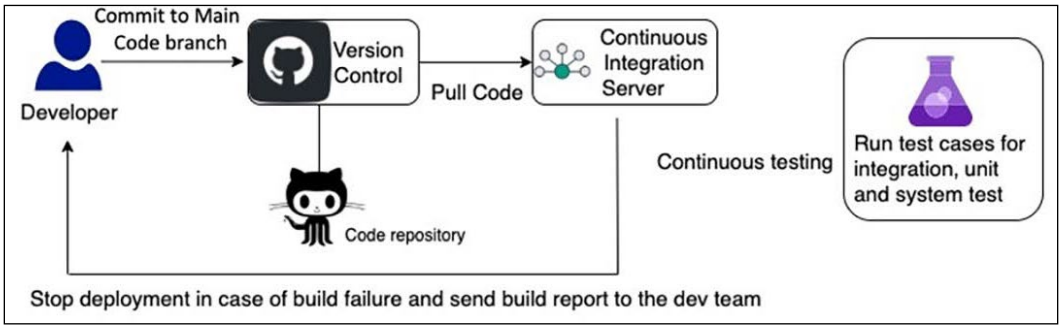


Figure 12.11: Automation of CI

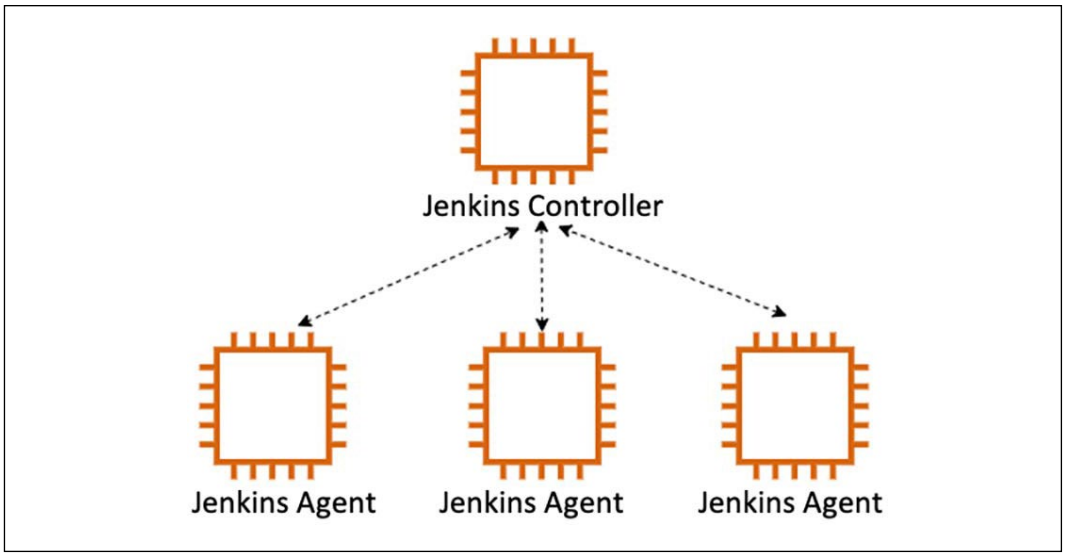


Figure 12.12: Auto-scaling of Jenkins CI servers

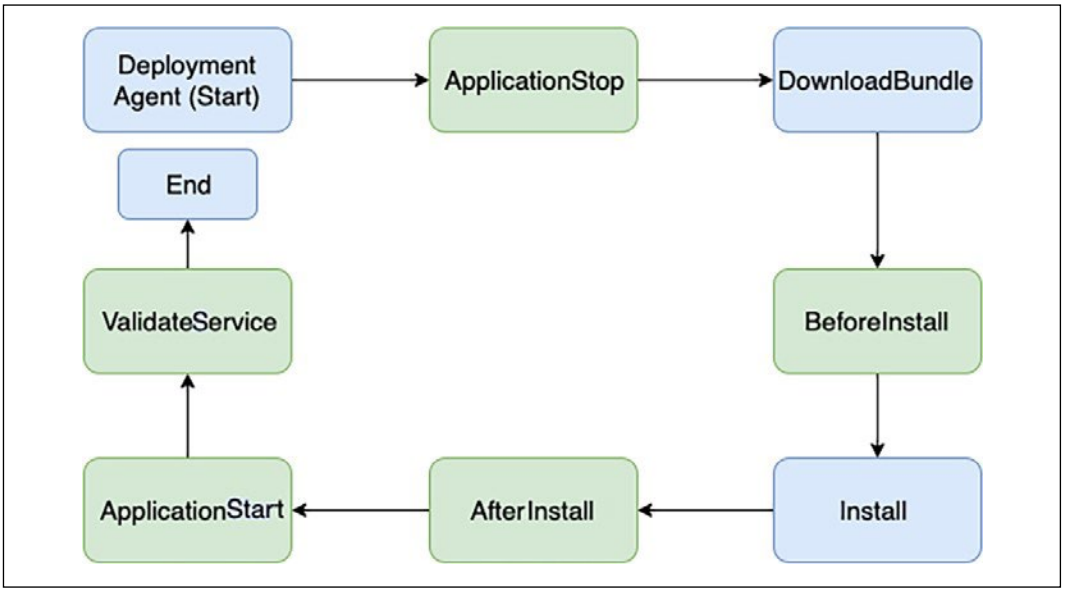


Figure 12.13: Deployment life cycle event

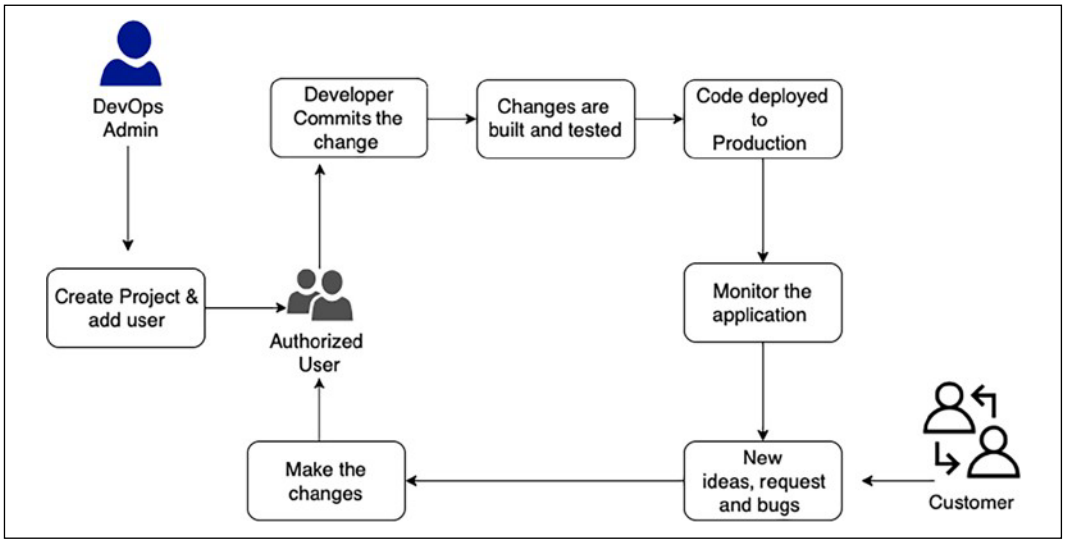


Figure 12.14: CI/CD workflow best practice

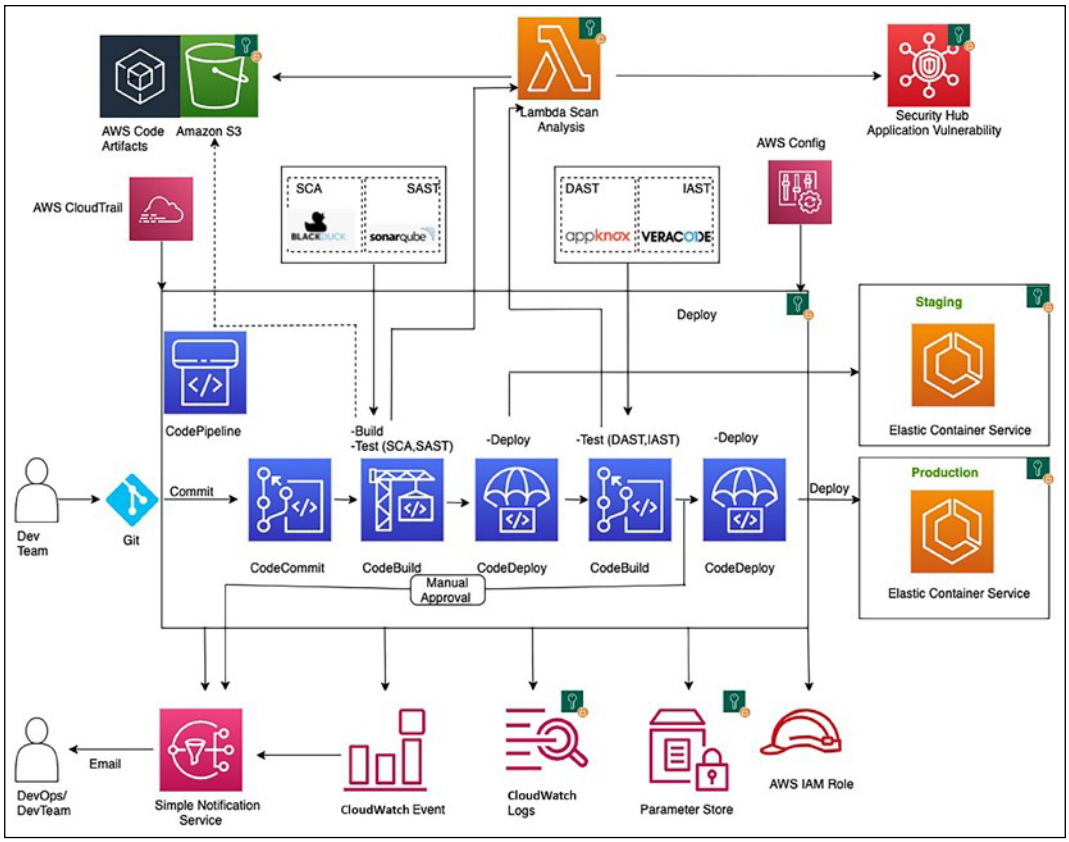


Figure 12.15: DevSecOps CI/CD pipeline architecture in the AWS cloud

Chapter 13
Data Engineering for Solution Architecture

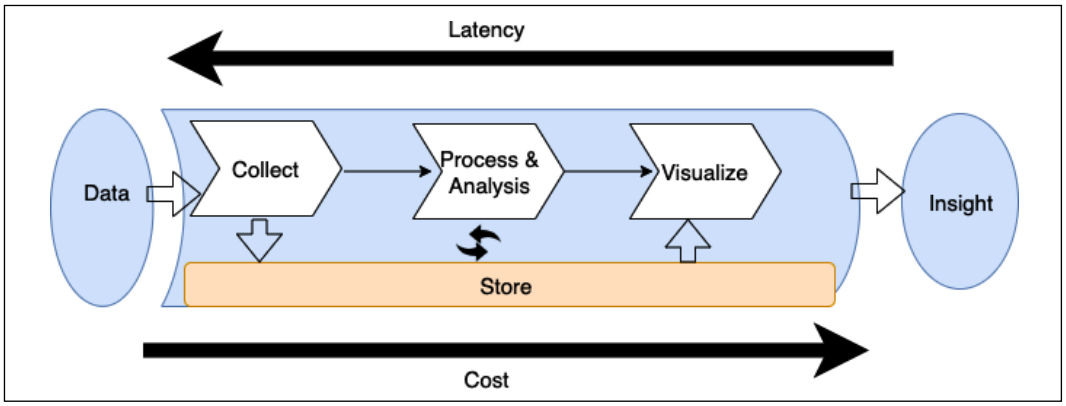


Figure 13.1: Big data pipeline for data architecture design

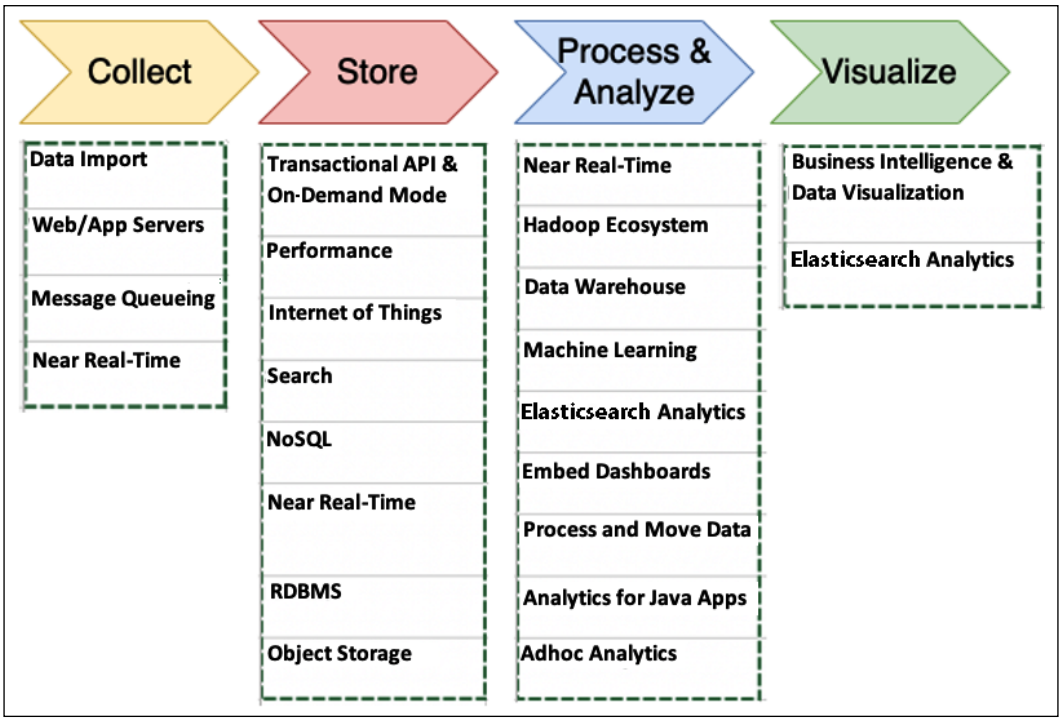


Figure 13.2: Tools and processes for big data architecture design

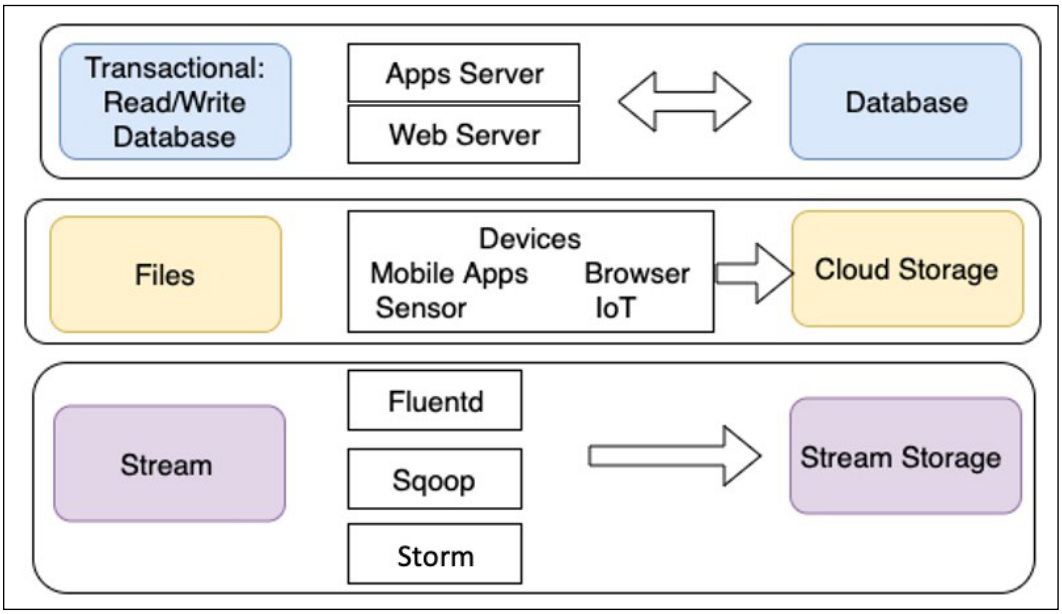


Figure 13.3: Type of data ingestion

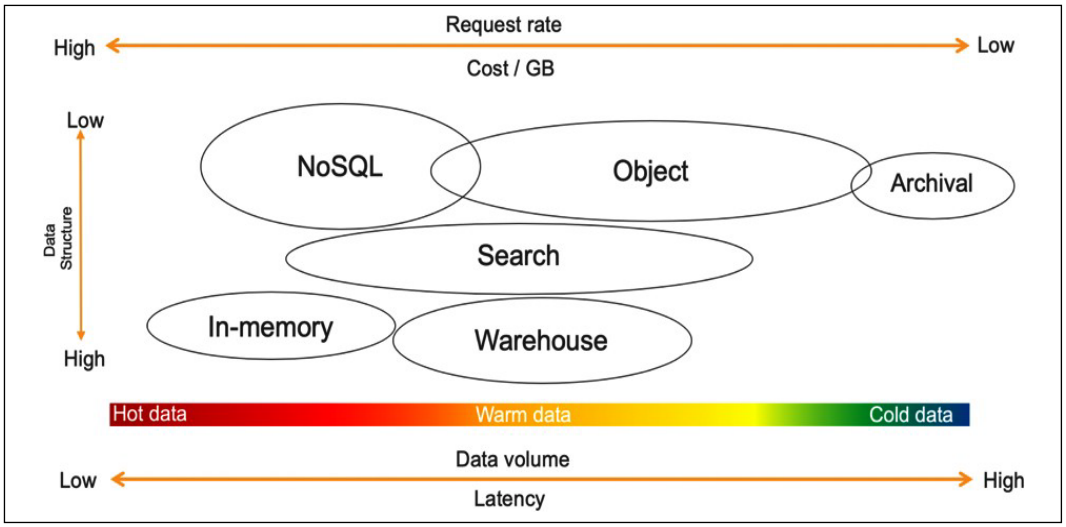


Figure 13.4: Understanding data storage

Properties	SQL Databases	NoSQL Databases
Data model	In SQL databases, the relational model normalizes data into tables containing rows and columns. A schema includes tables, columns, relationships between tables, indexes, and other database elements.	NoSQL databases do not enforce a schema. A partition key is commonly used to retrieve values from column sets. It stores semi-structured data such as JSON, XML, or other documents such as data catalogs and file indexes.
Transaction	SQL-based traditional RDBMSes support and are compliant with the transactional data properties of ACID.	To achieve horizontal scaling and data model flexibility, NoSQL databases may trade some ACID properties of traditional RDBMSes.
Performance	SQL-based RDBMSes were used to optimize storage when storage was expensive and minimize the disk footprint. For traditional RDBMSes, performance has mostly relied on the disk. To achieve performance query optimizations, index creation and modifications to the table structure are required.	For NoSQL, performance depends upon the underlying hardware cluster size, network latency, and how the application is calling the database.
Scale	SQL-based RDBMS databases are easiest to scale vertically with high configuration hardware. The additional effort requires relational tables to span across distributed systems, such as performing data sharding.	NoSQL databases are designed to scale horizontally using distributed clusters of low-cost hardware to increase throughput without impacting latency.

Table 13.1: Properties of SQL and NoSQL databases

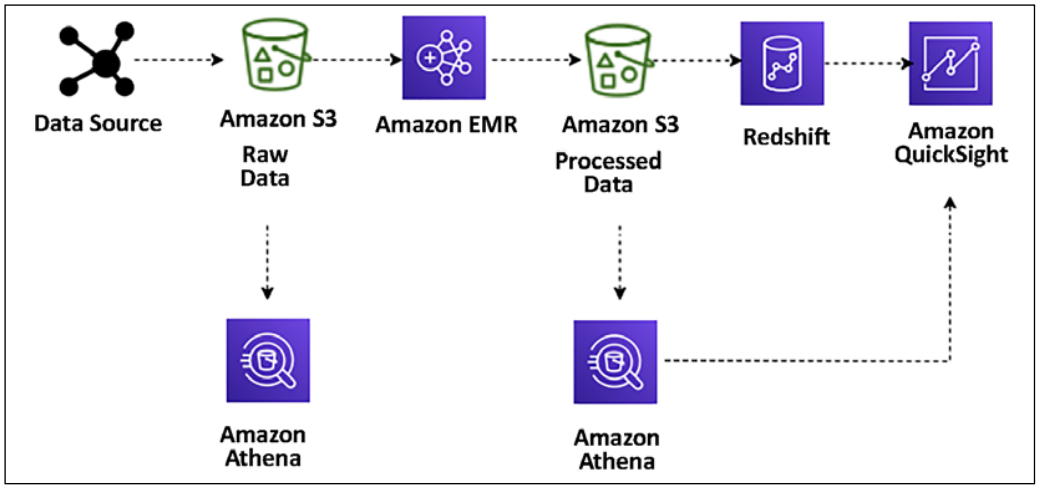


Figure 13.5: Data lake ETL pipeline for big data processing

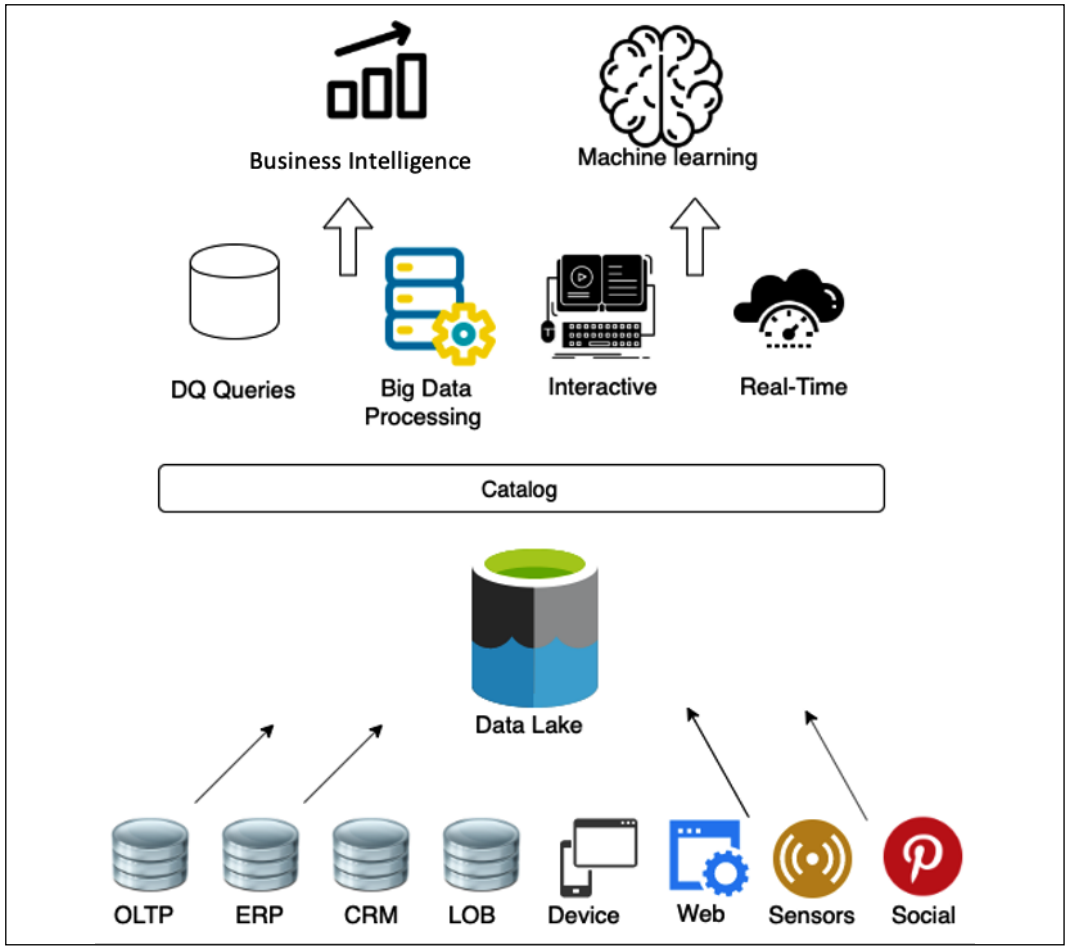


Figure 13.6: Object store for data lake

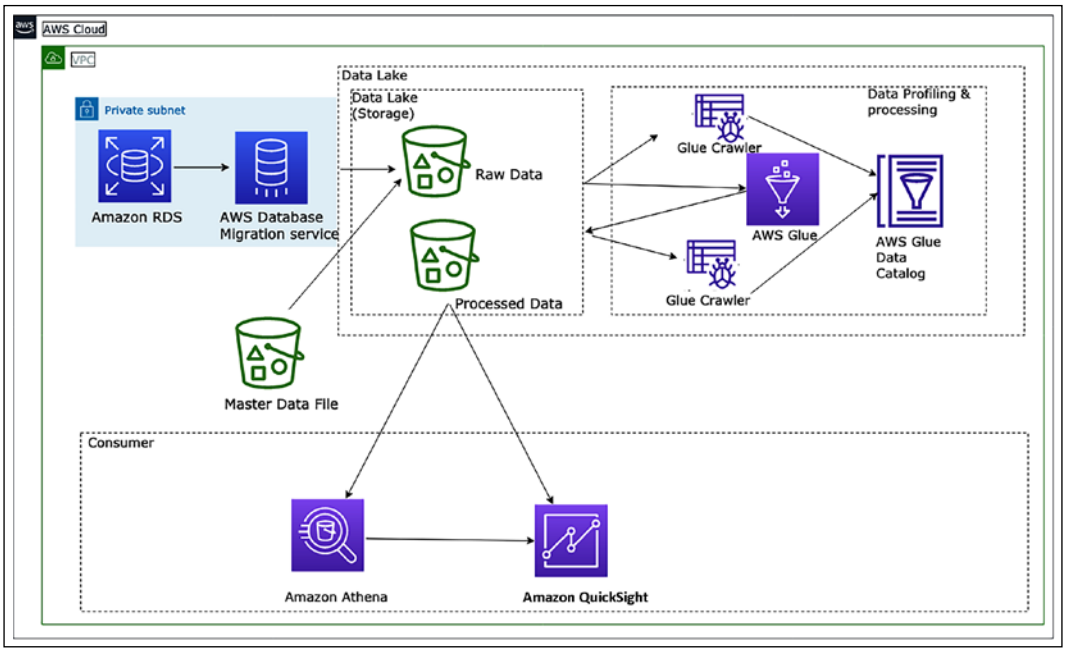


Figure 13.7: Data lake architecture in AWS platform

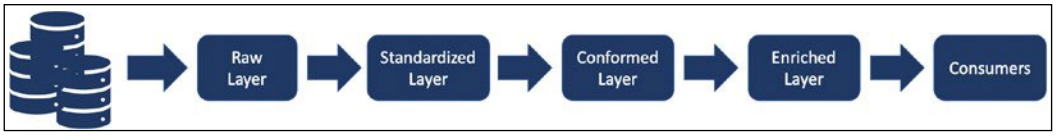


Figure 13.8: Lakehouse architecture layers

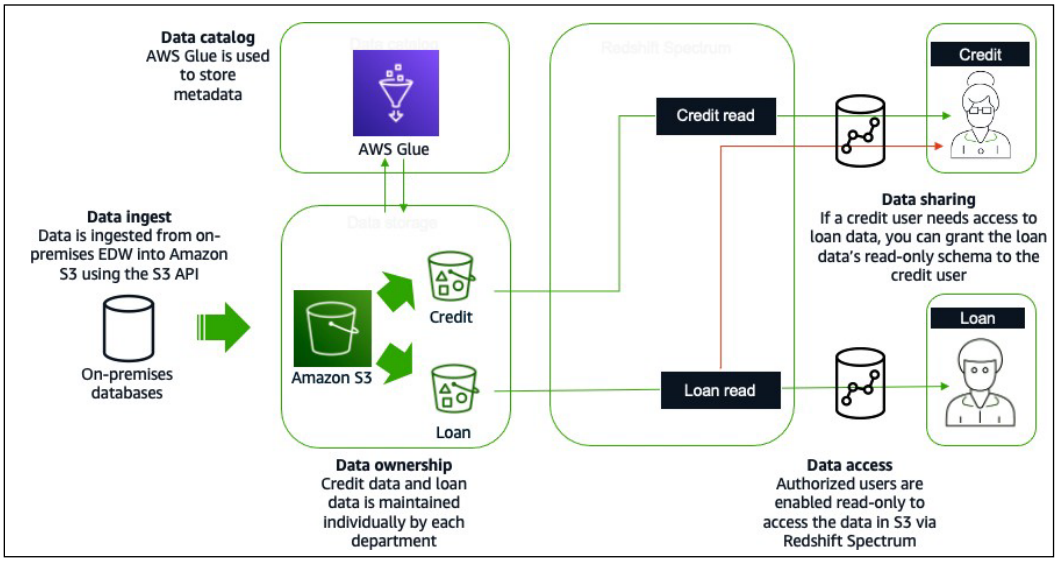


Figure 13.9: Lakehouse architecture in AWS cloud platform using Redshift Spectrum

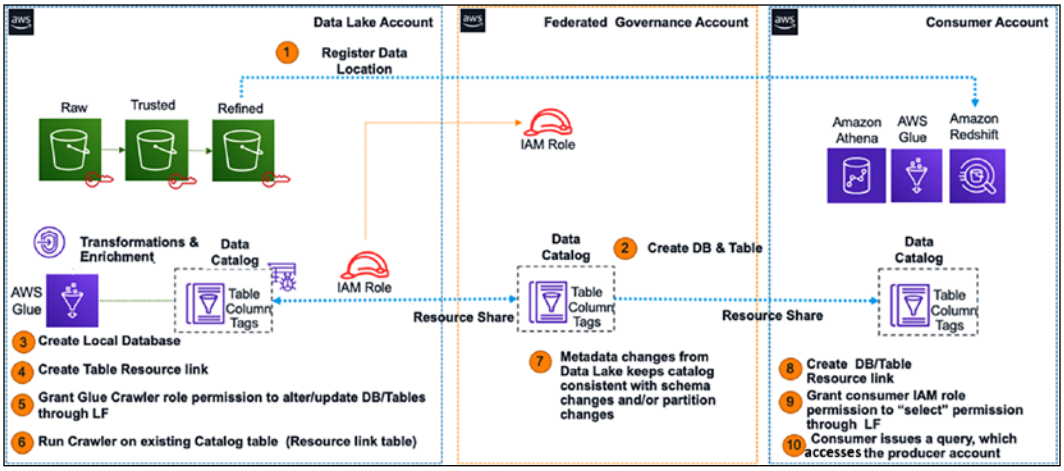


Figure 13.10: Data mesh architecture in AWS cloud platform

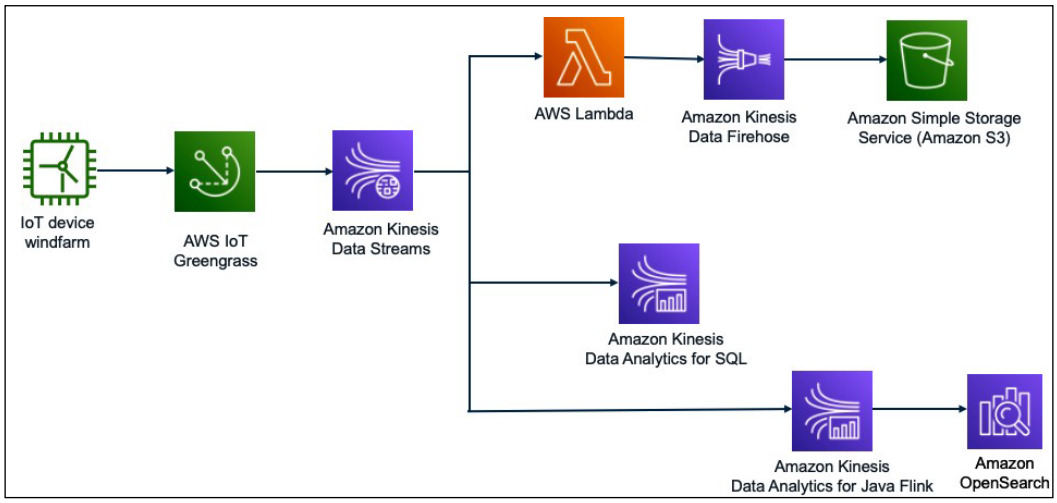


Figure 13.11: Streaming data analytics for IoT data

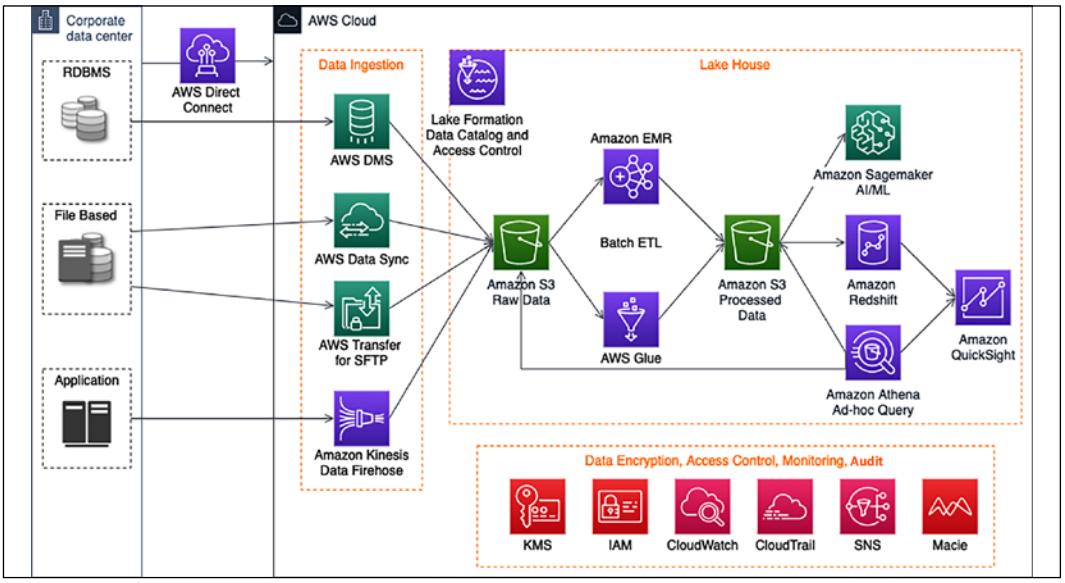


Figure 13.12: Data lake reference architecture

Chapter 14
Machine Learning Architecture

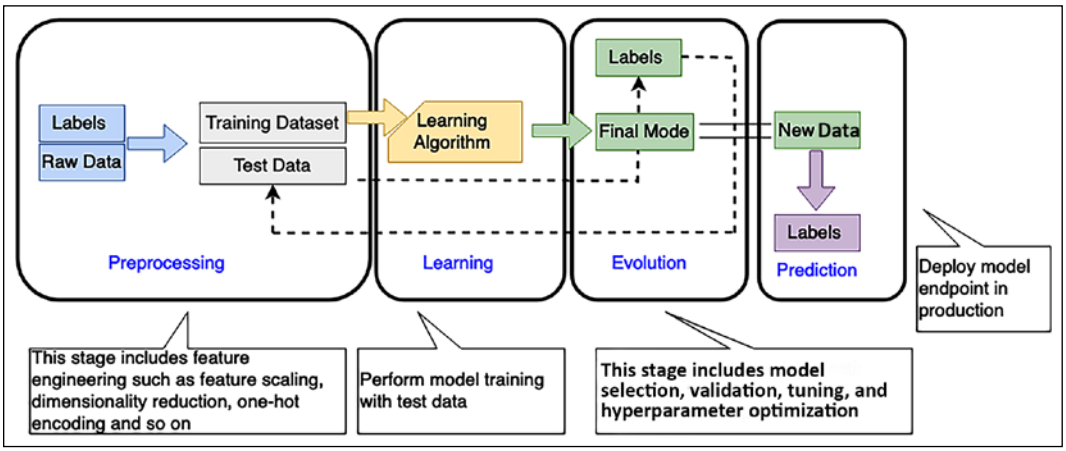


Figure 14.1: ML workflow

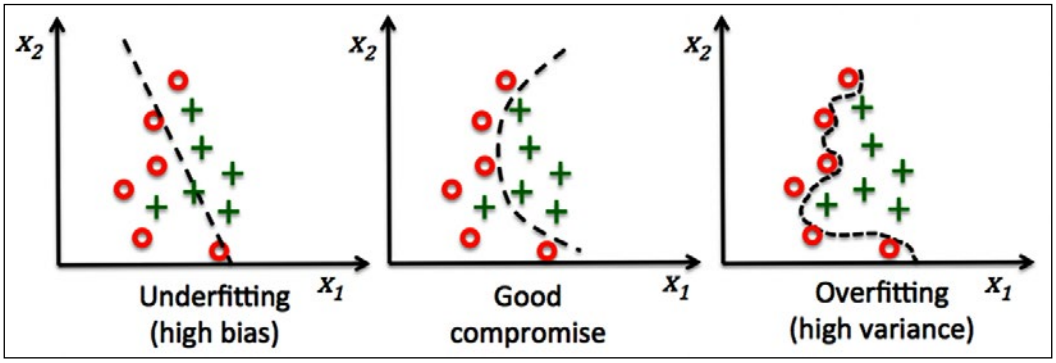


Figure 14.2: ML model overfitting versus underfitting

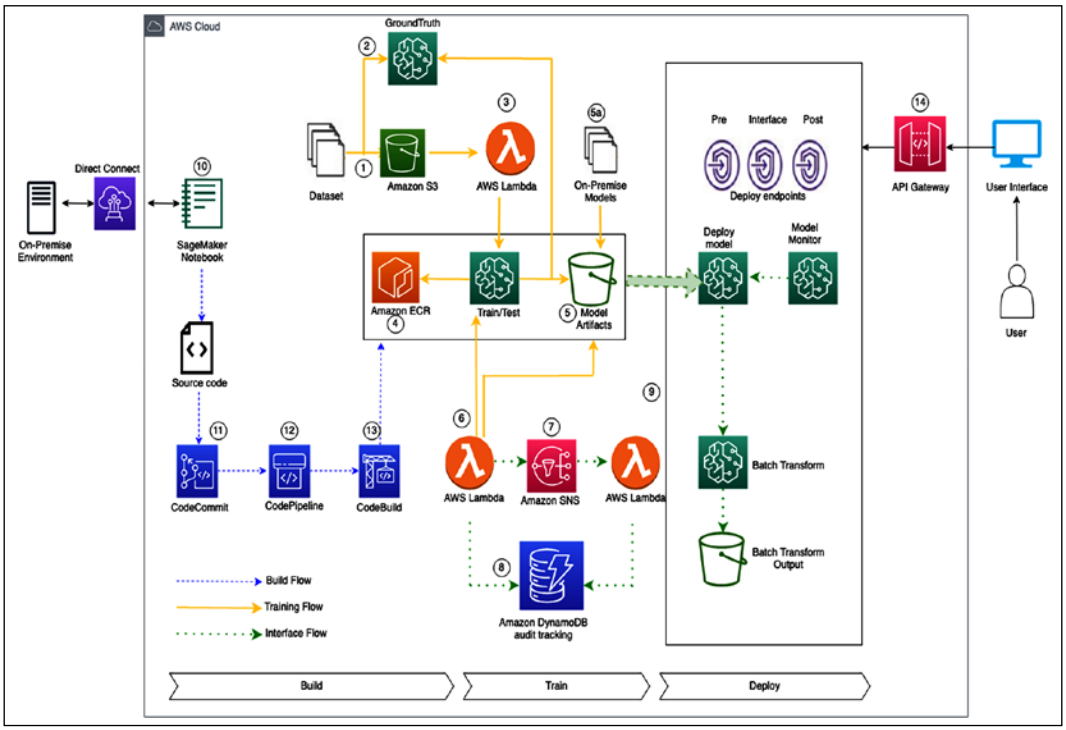


Figure 14.3: ML architecture in the AWS cloud

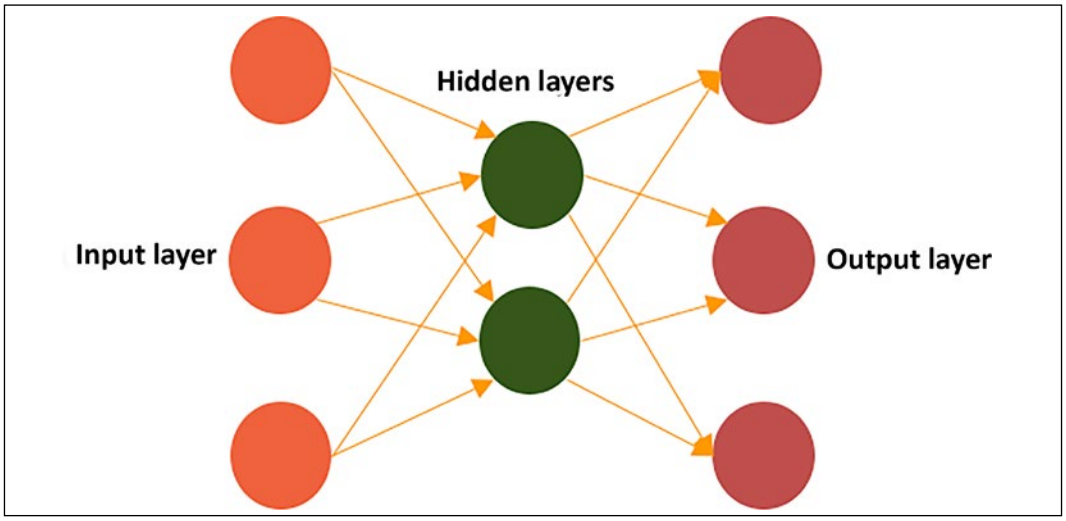


Figure 14.4: A overview of deep learning layers

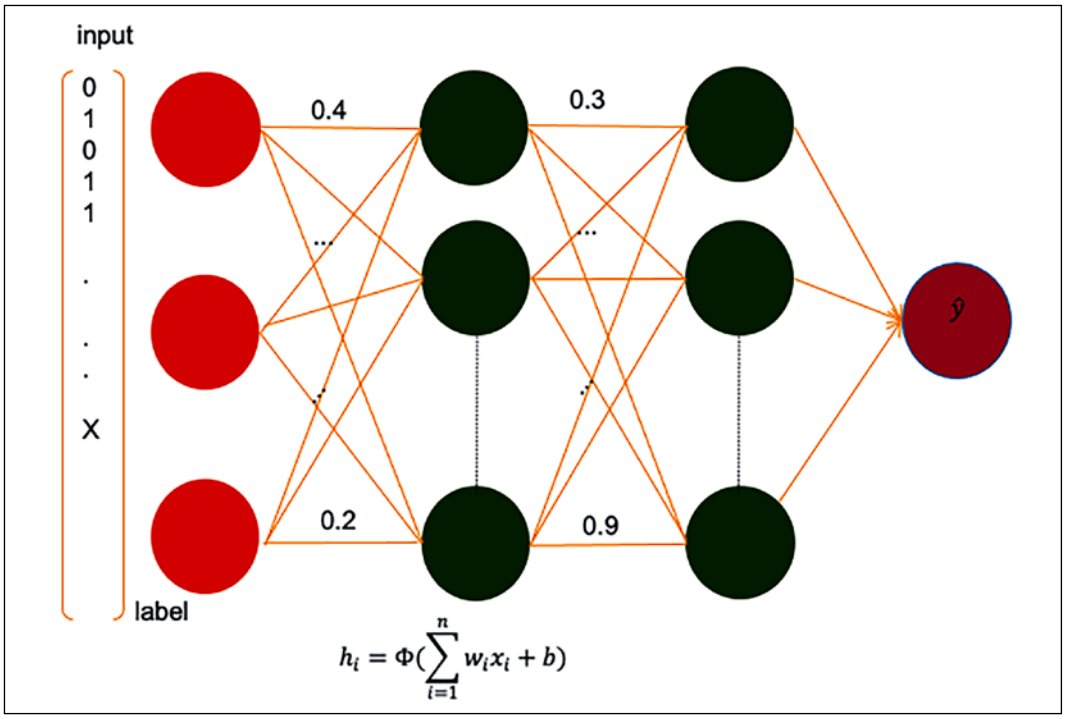


Figure 14.5: Deep learning neural network model

Chapter 15
The Internet of Things Architecture

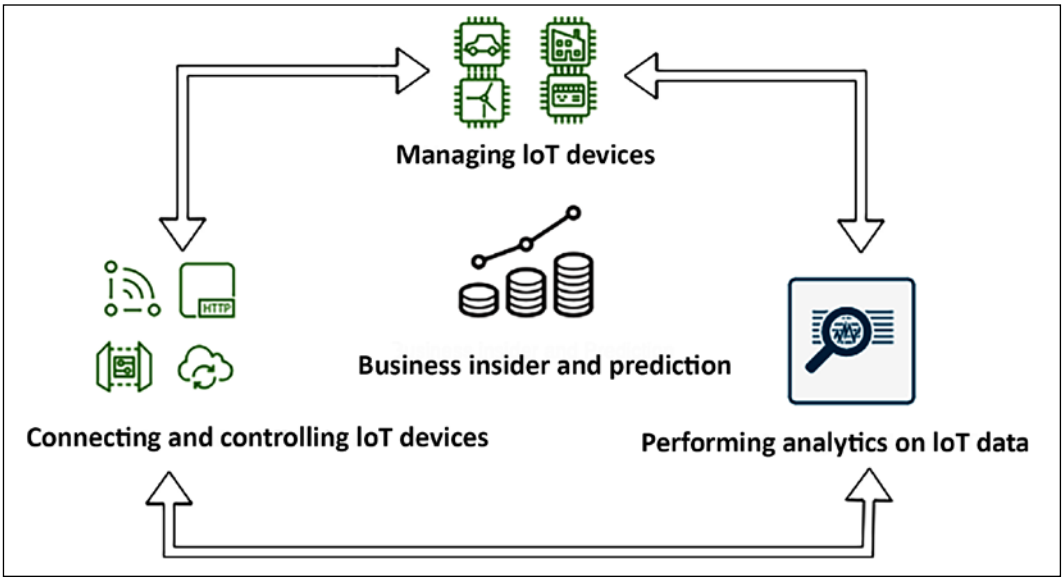


Figure 15.1: IoT architecture cycle

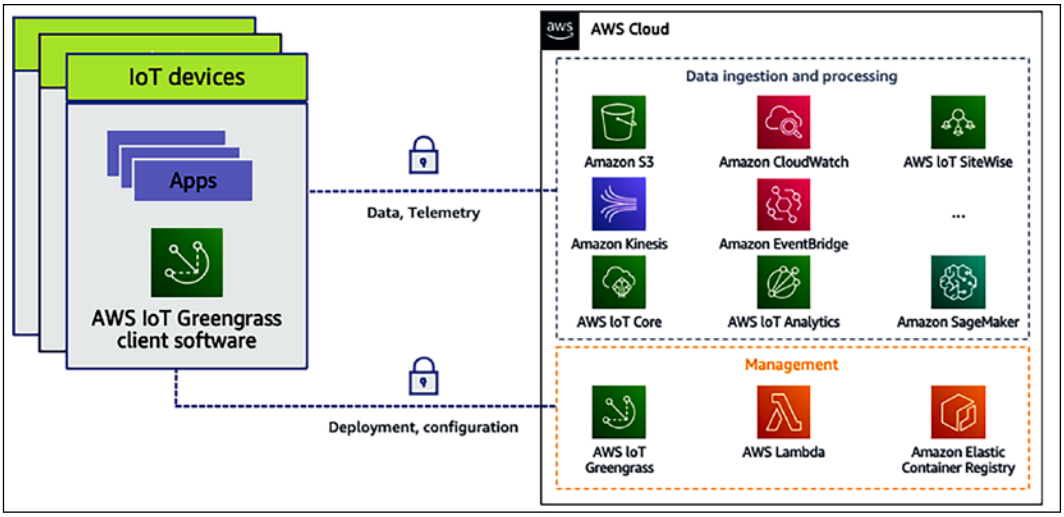


Figure 15.2: AWS IoT Greengrass pre-integrated AWS cloud services

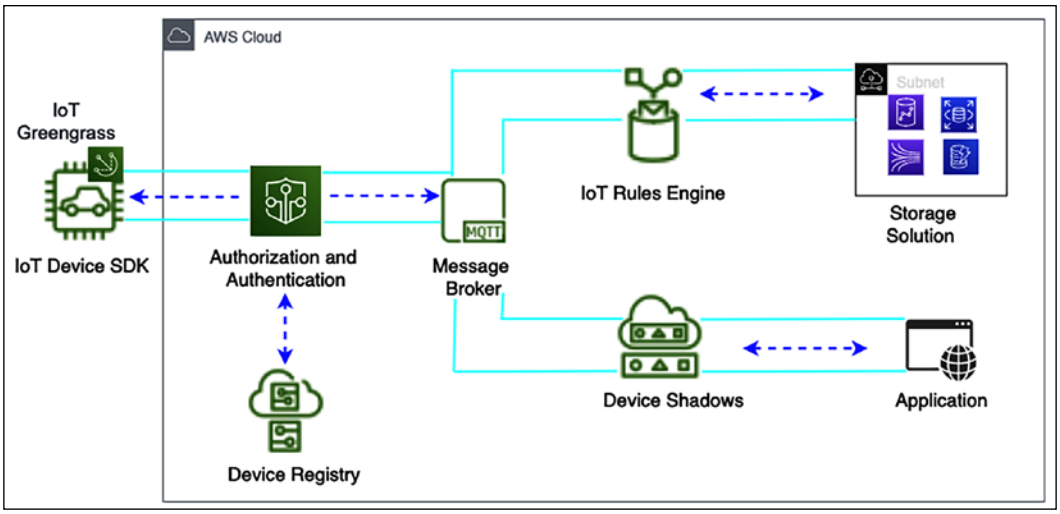


Figure 15.3: IoT architecture on an AWS platform

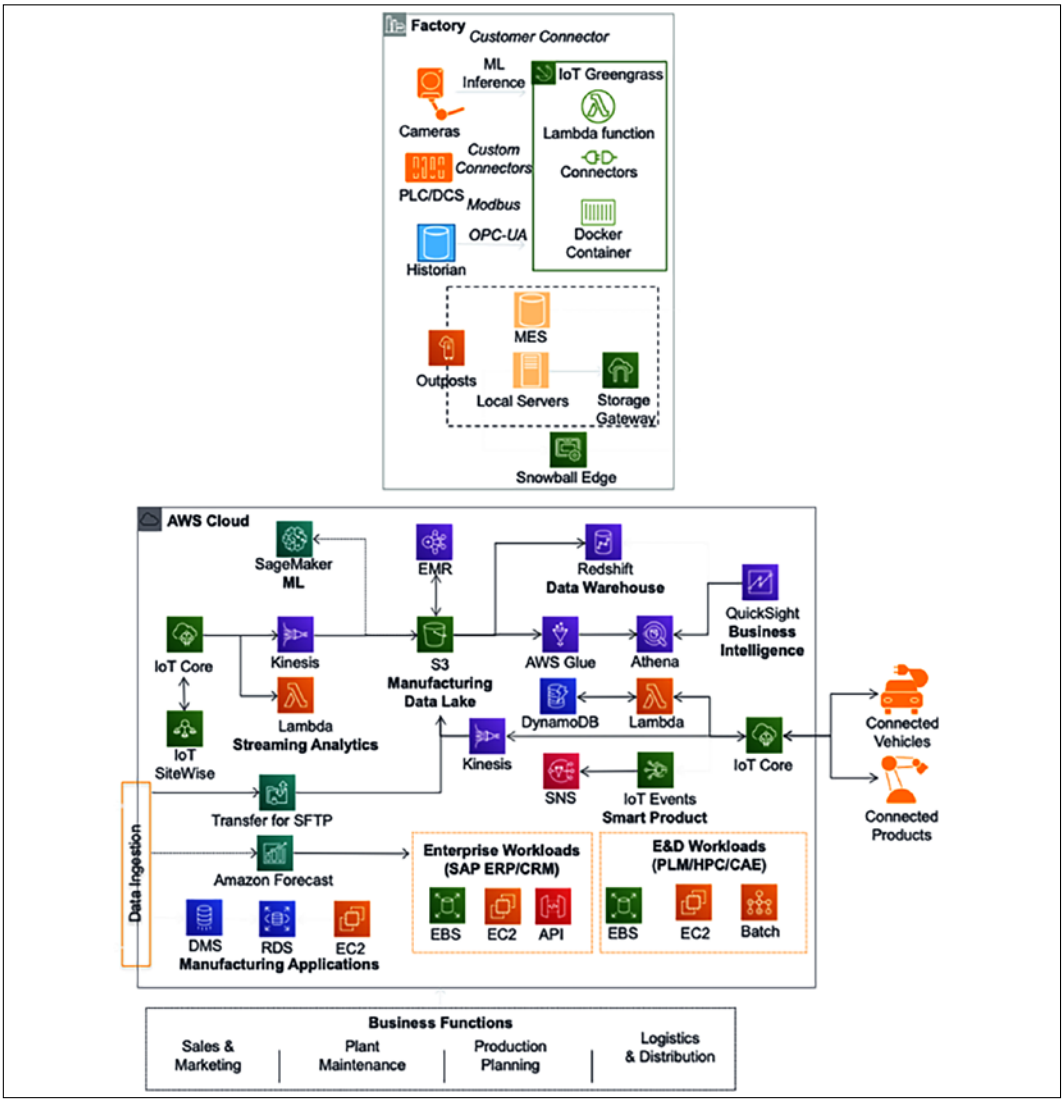


Figure 15.4: Connected Factory architecture in the AWS cloud

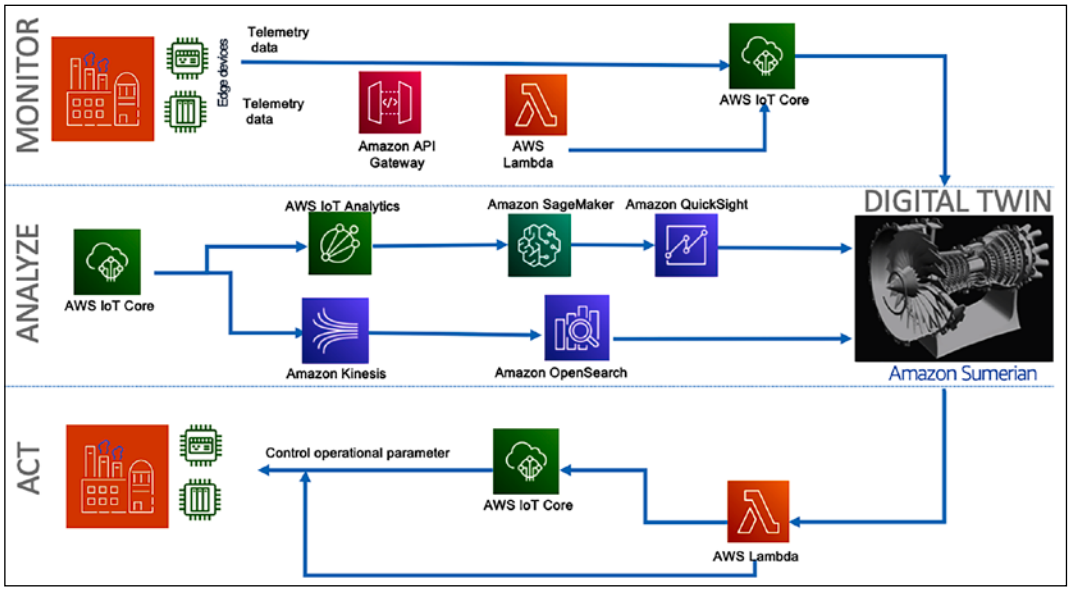


Figure 15.5: Modeling the mind of the machine with a digital twin

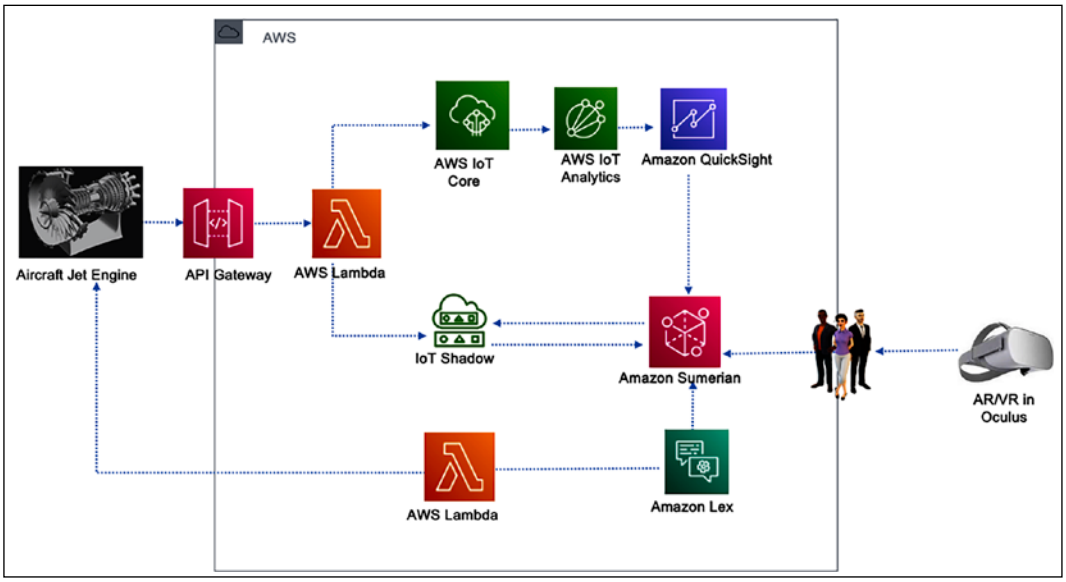


Figure 15.6: Digital twin architecture for an aircraft jet engine

Chapter 16
Quantum Computing

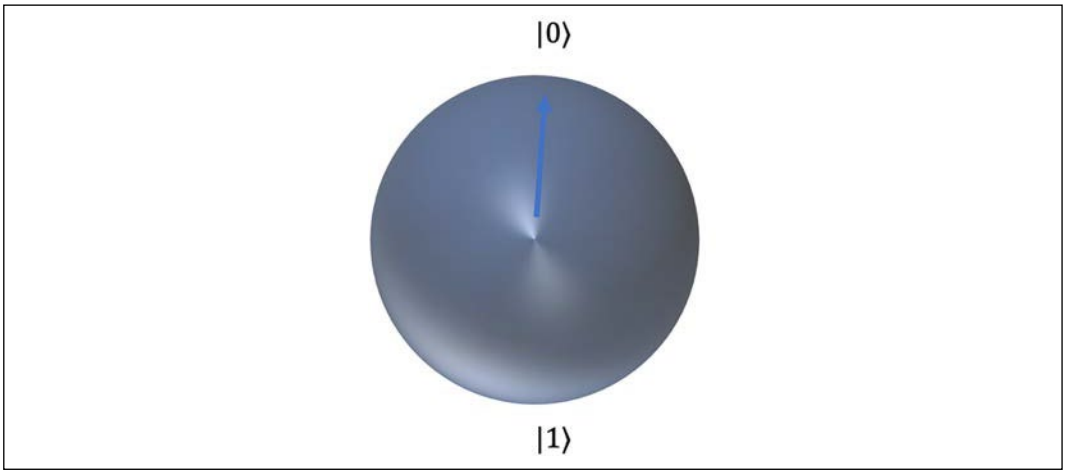


Figure 16.1: Bloch sphere – abstract representation of a qubit

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Equation 16.1: two dimensional representations of |0⟩ and |1⟩

	Classical Computer: Copying X Bit to Y Bit	**Quantum Computer: Entangling X Qubit and Y Qubit**
Correlation	X bit and Y bit are uncorrelated after copying	X qubit and Y qubit correlate; measuring X affects Y instantaneously
Referencing	May assign by reference on same data so X bit and Y bit may point to the same data	Entangled qubits exist individually, but they correlate
Reversibility	It's irreversible—the reverse of an operation like copying back X to Y would destroy Y	It's reversible—the entangled qubits X and Y can be unentangled
Correction	For error correction, bits can be restored from a previous copy	Quantum error correction uses many entangled qubits

Table 16.1: Properties of classical and quantum bits

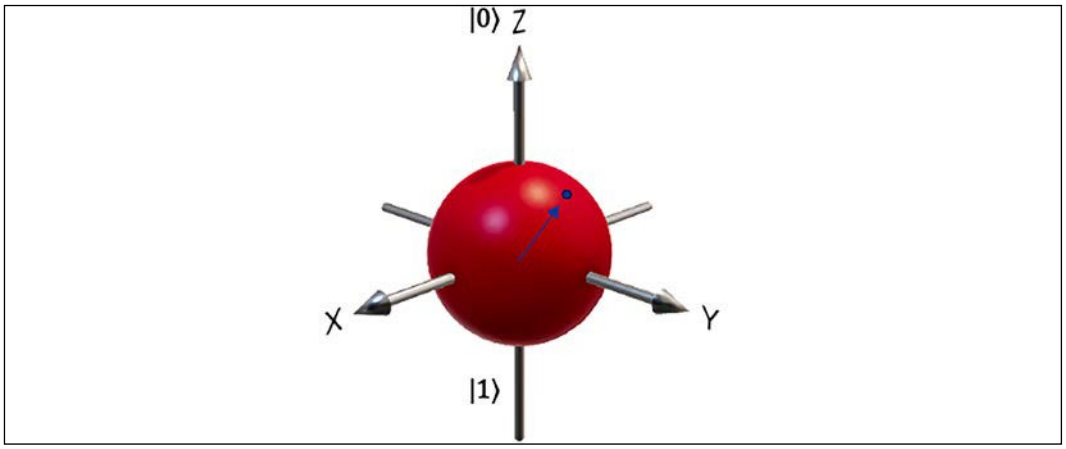


Figure 16.2: 3D representation of an electron's spin along with its axes

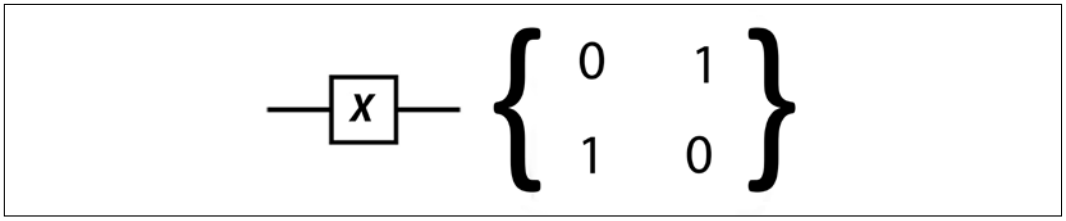


Figure 16.3: Pauli-X gate

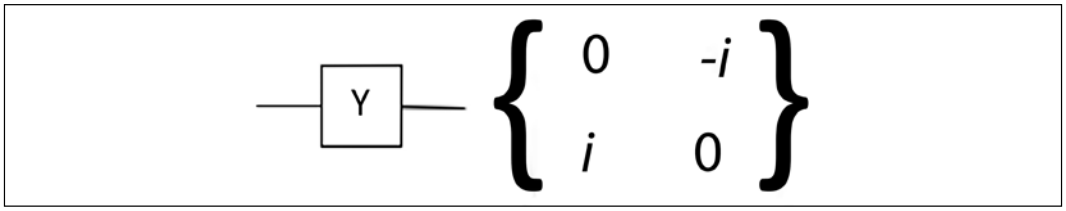


Figure 16.4: Pauli-Y gate

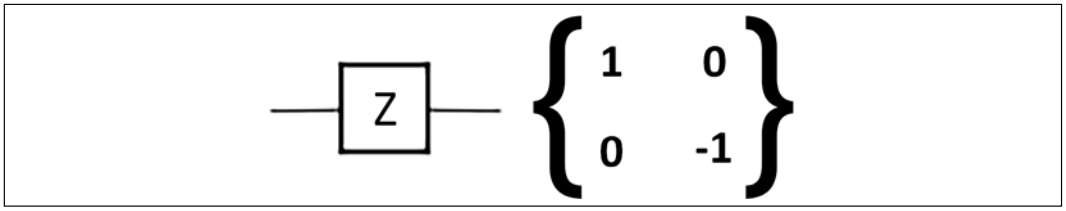


Figure 16.5: Pauli-Z gate

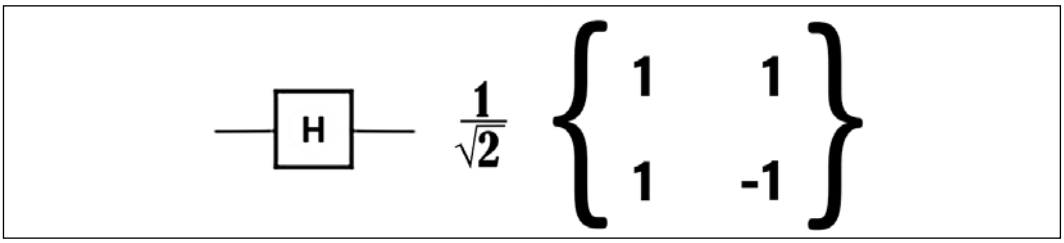


Figure 16.6: Hadamard gate

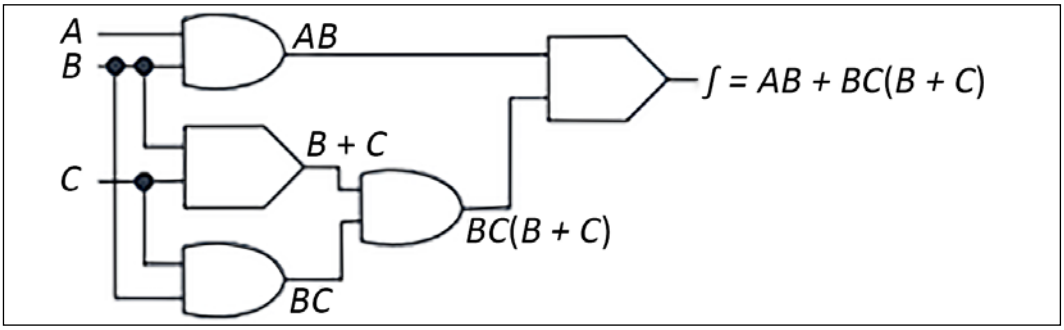


Figure 16.7: Operation in a Boolean circuit

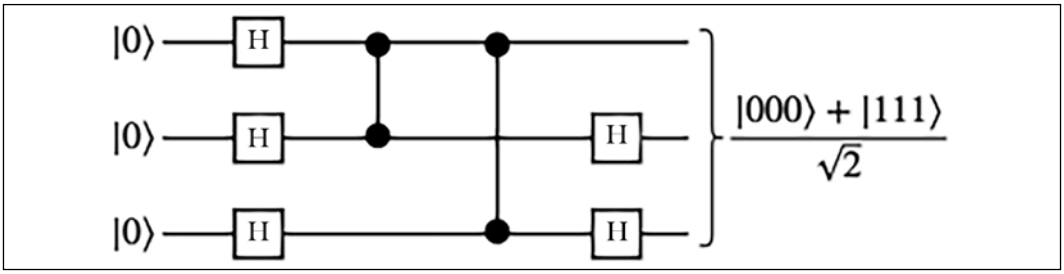


Diagram 16.8: Operation in a quantum circuit

Chapter 17
Rearchitecting Legacy Systems

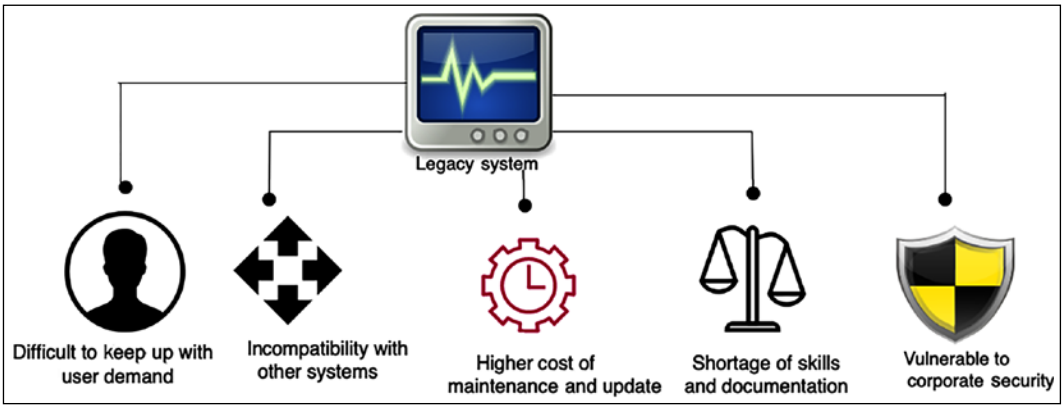


Figure 17.1: Challenges with a legacy system

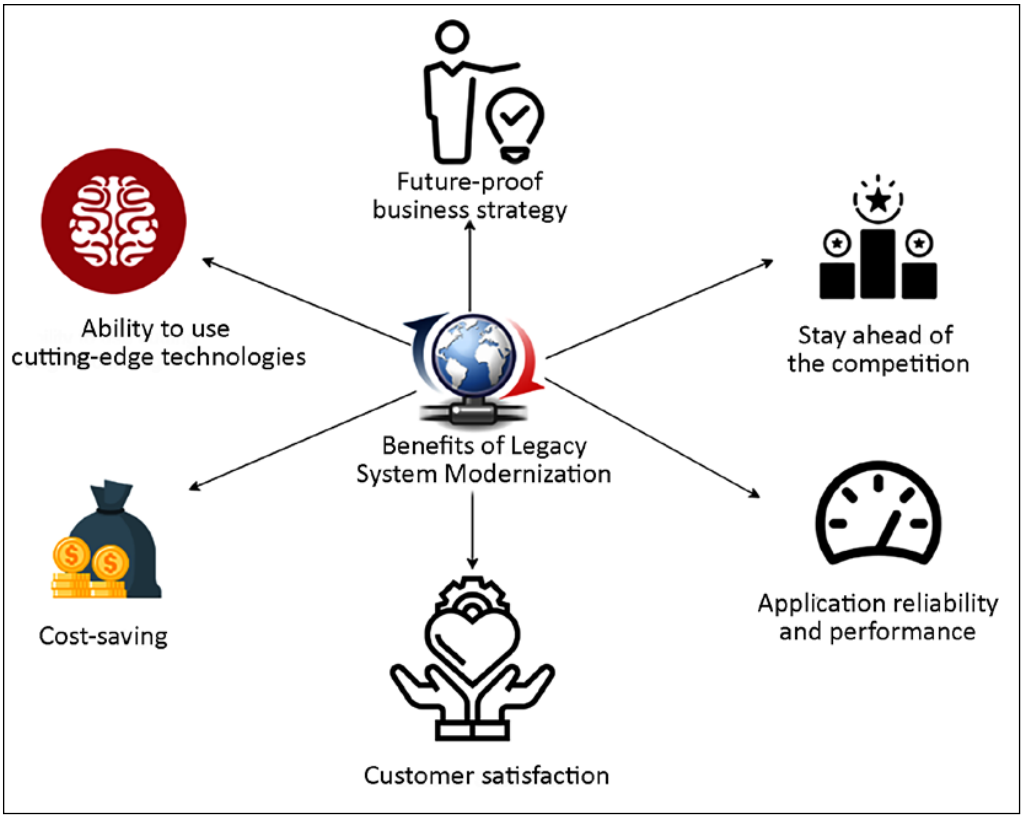


Figure 17.2: Benefits of legacy system modernization

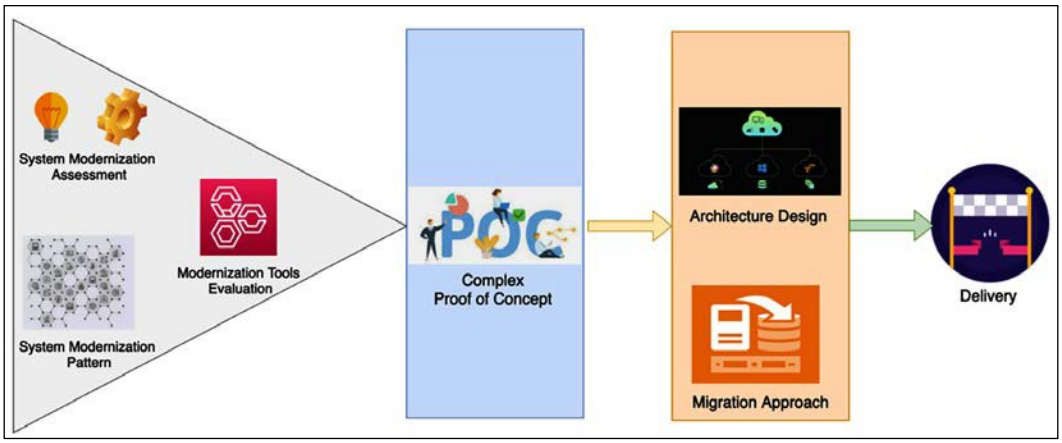


Figure 17.3: The legacy system modernization approach

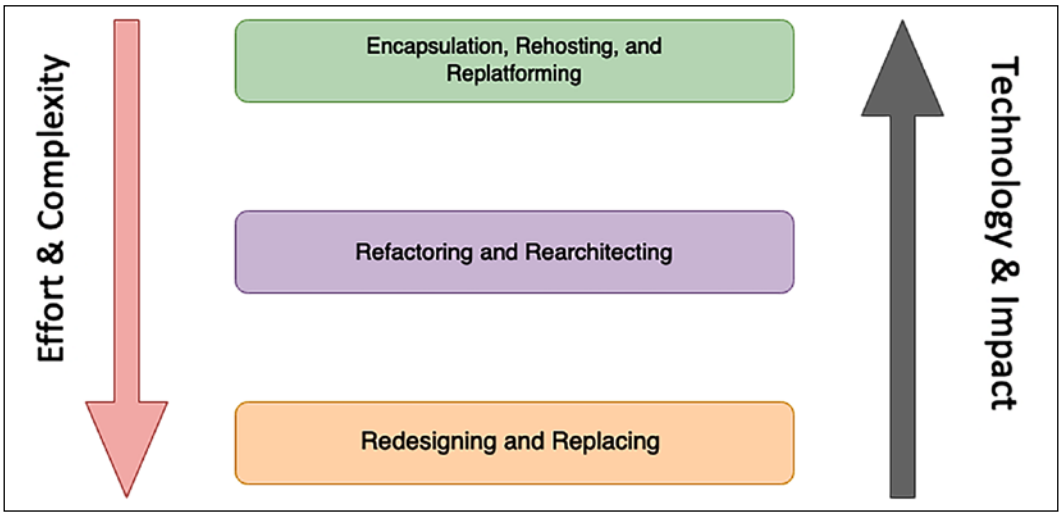


Figure 17.4: Legacy system modernization techniques

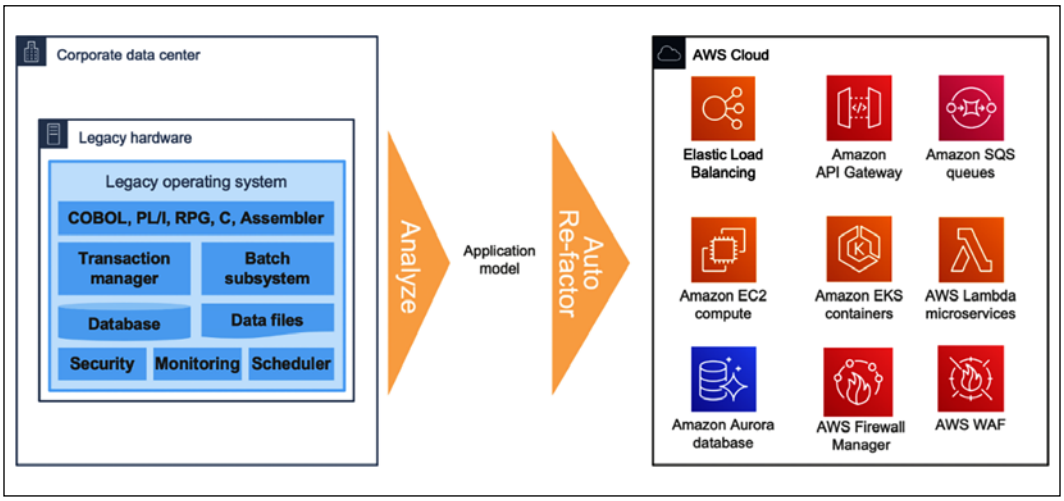


Figure 17.5: Legacy mainframe system modernization to the cloud

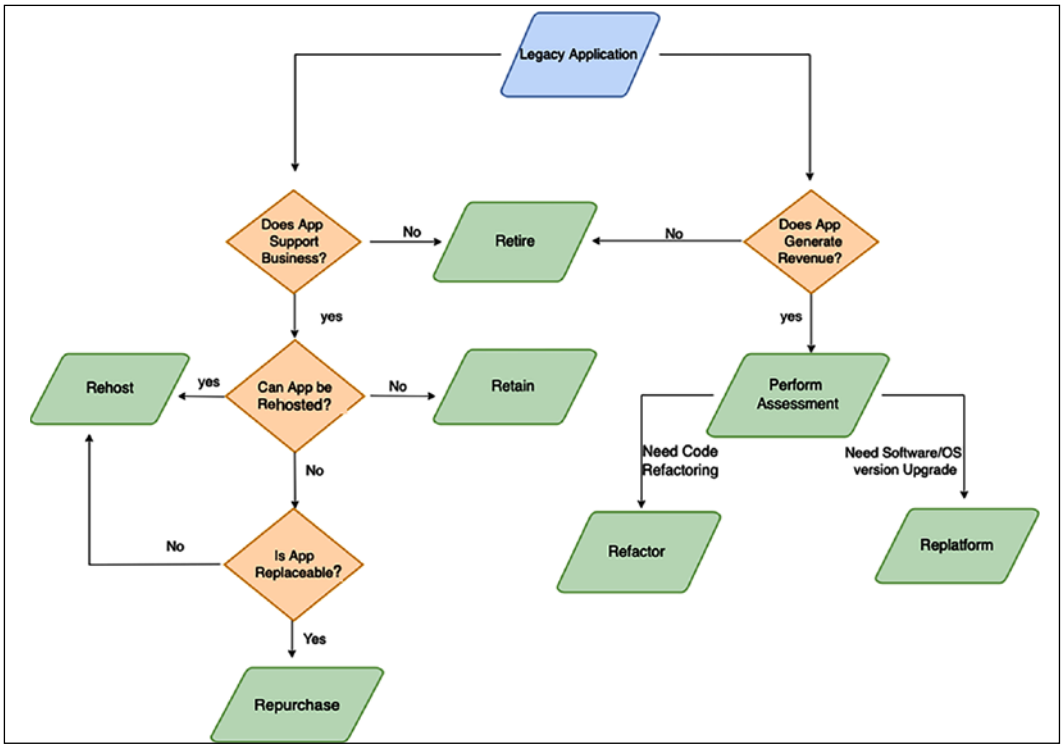


Figure 17.6: Cloud migration path for legacy system modernization

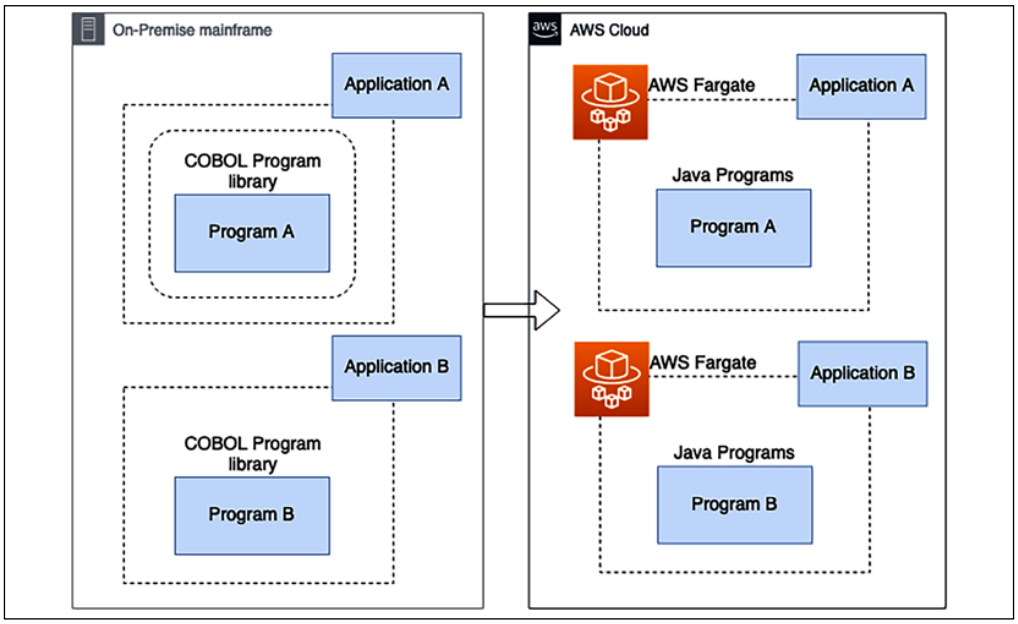


Figure 17.7: Mainframe modernization for a standalone application

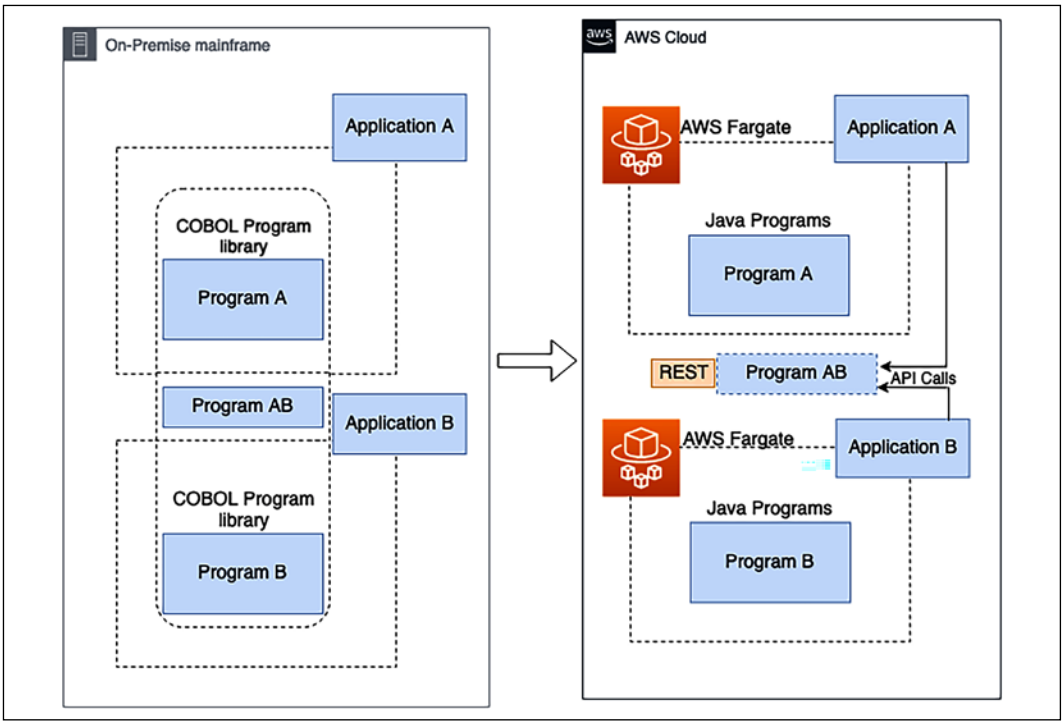


Figure 17.8: Migration of shared program applications using a standalone API

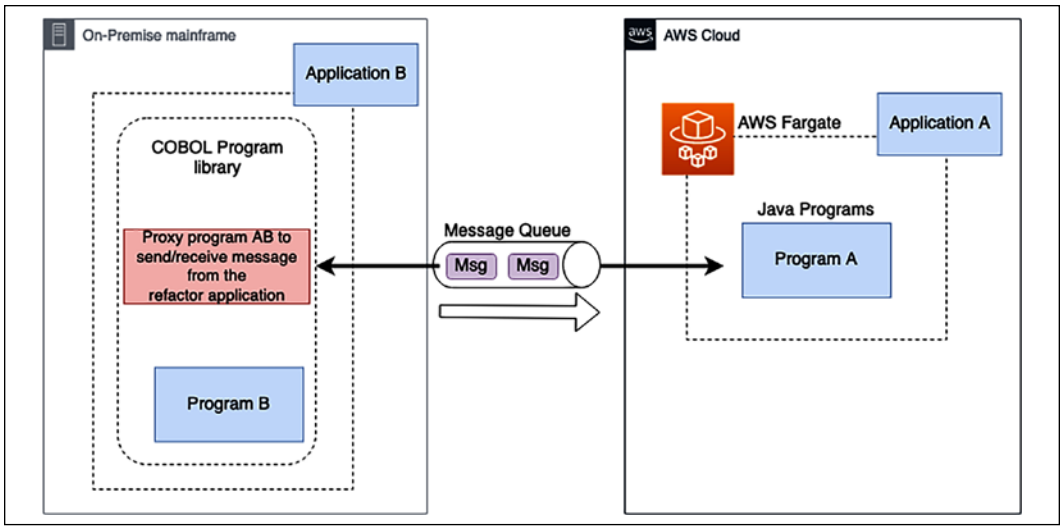


Figure 17.9: Migration of shared program applications using a message queue

Chapter 18
Solution Architecture Document

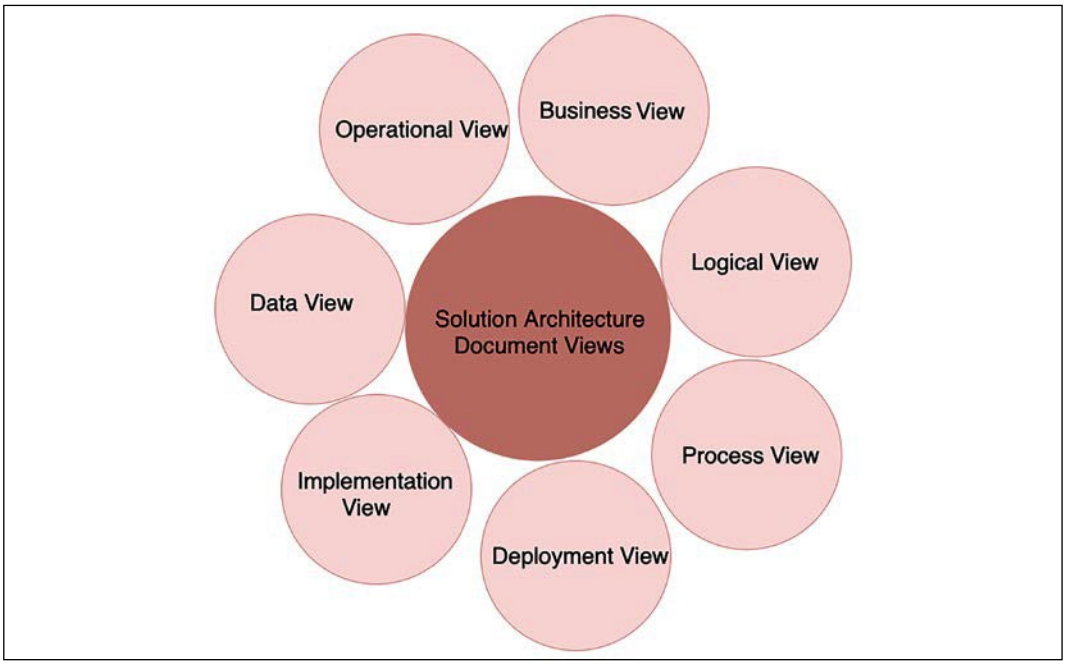


Figure 18.1: SAD views

Contents

Figure 18.2: Structure of a SAD

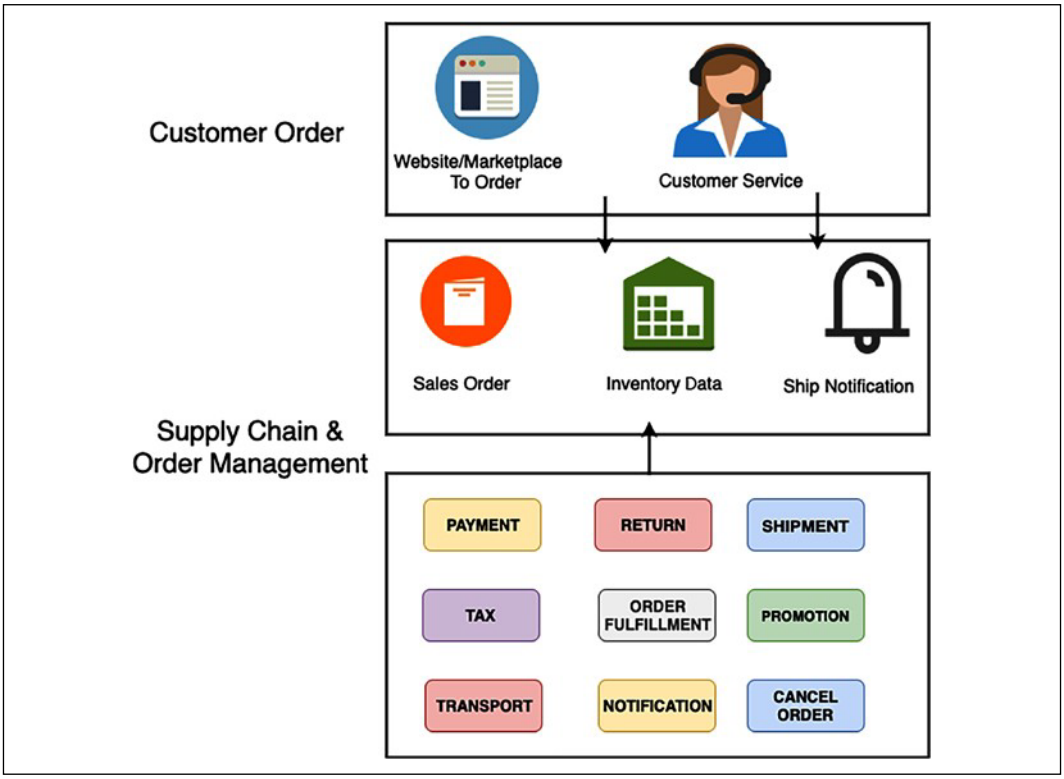


Figure 18.3: Solution overview of an e-commerce platform

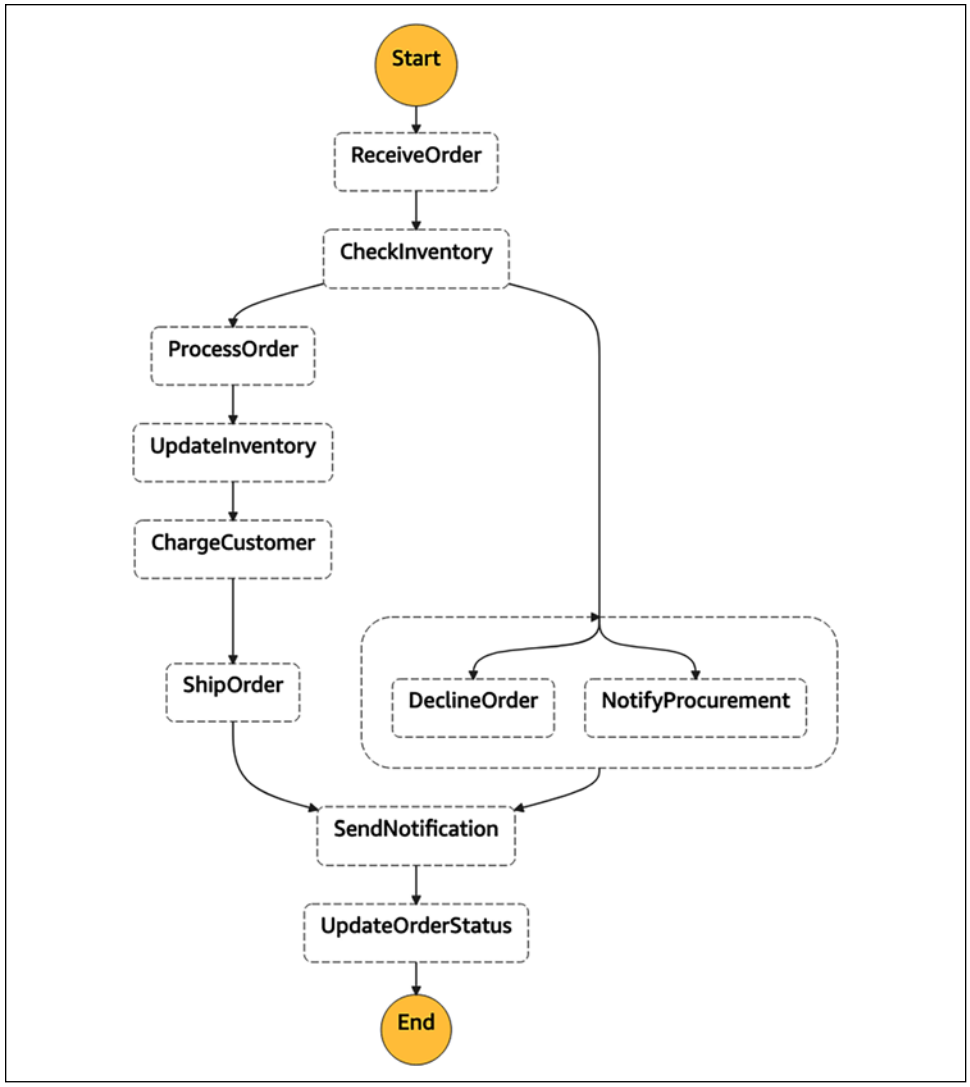


Figure 18.4: Business process diagram of an e-commerce platform

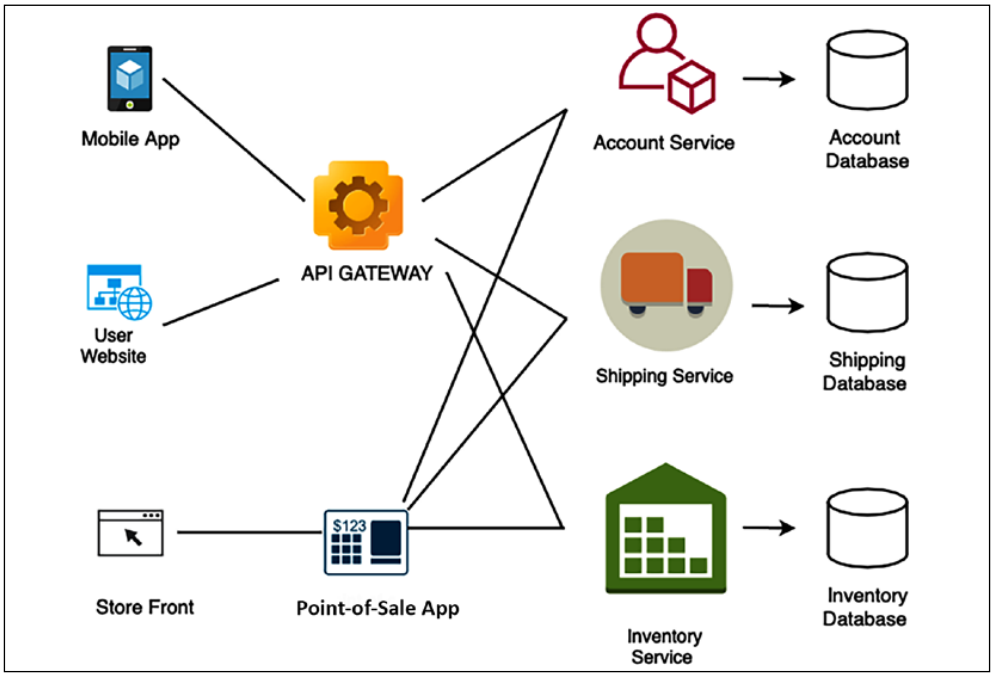


Figure 18.5: Conceptual architecture diagram of an e-commerce platform

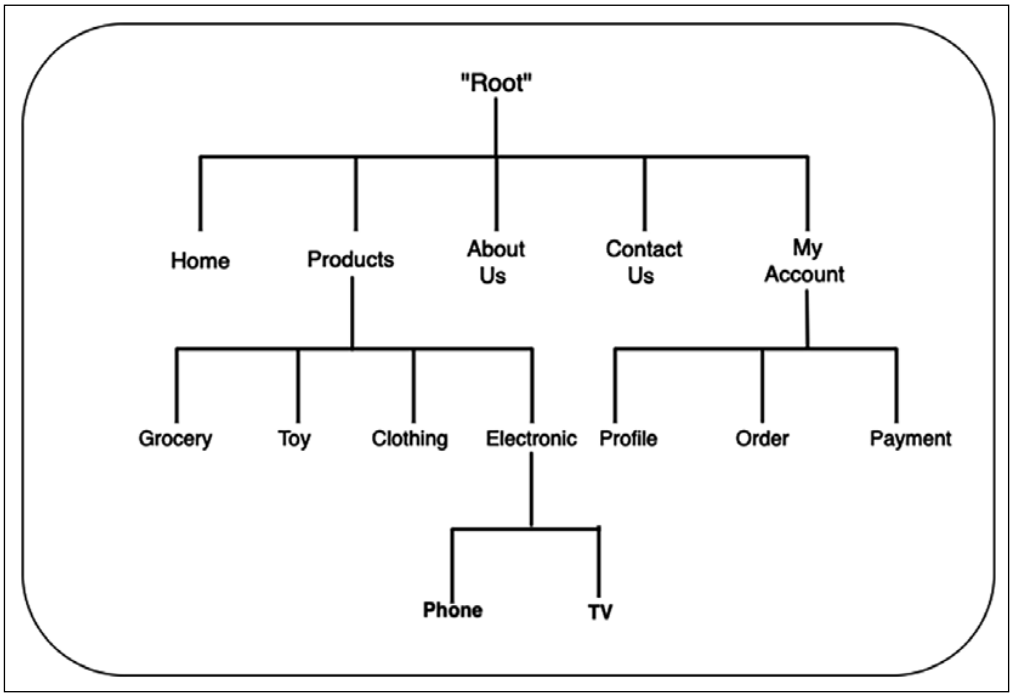


Figure 18.6: Informational architecture diagram of an e-commerce platform

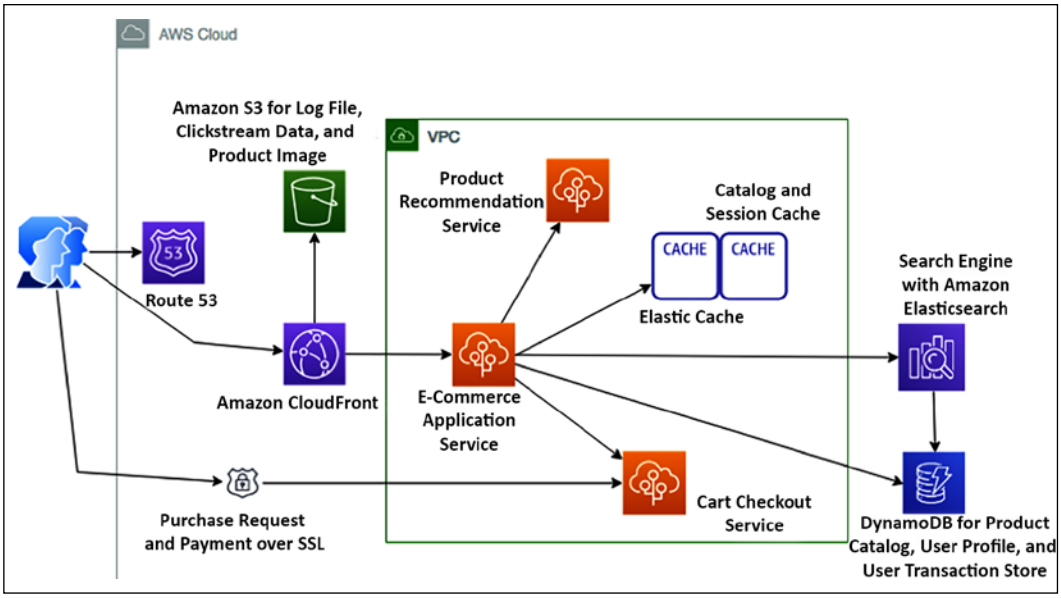


Figure 18.7: Application architecture diagram of an e-commerce platform

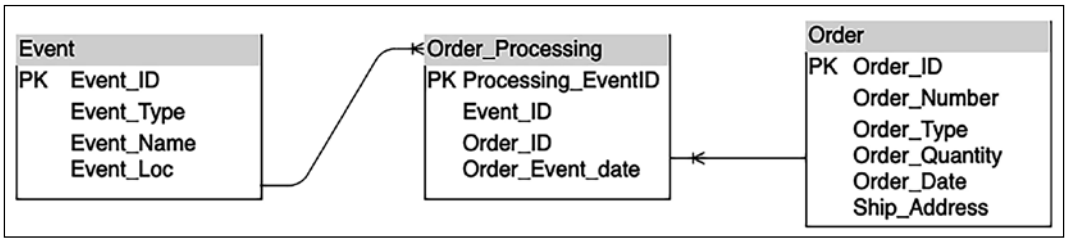


Figure 18.8: ERD of an e-commerce platform

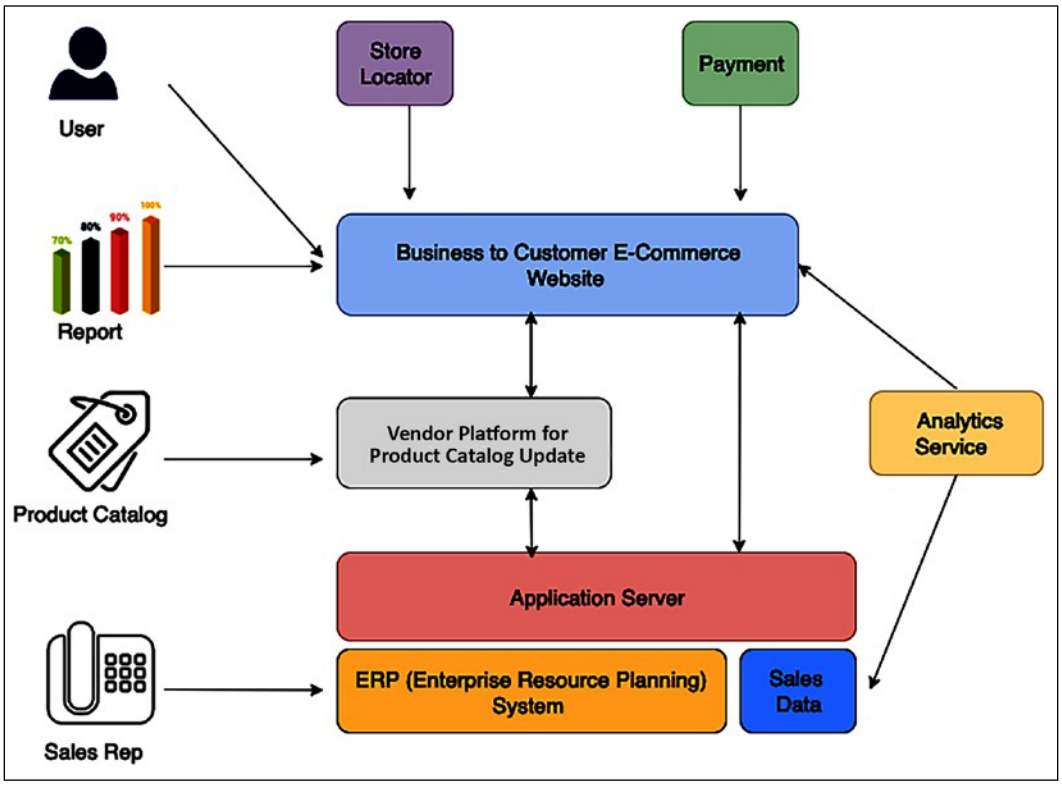


Figure 18.9: Integration architecture diagram of an e-commerce platform

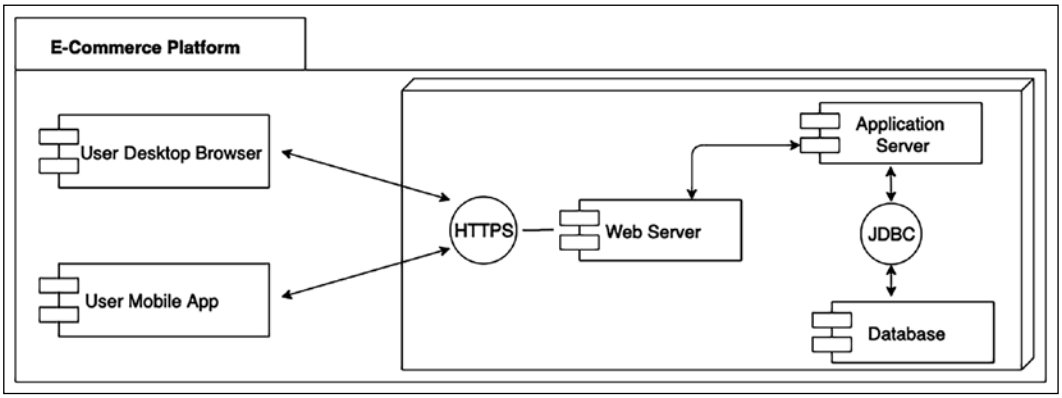


Figure 18.10: Deployment diagram of an e-commerce platform

Chapter 19
Learning Soft Skills to Become a Better Solution Architect

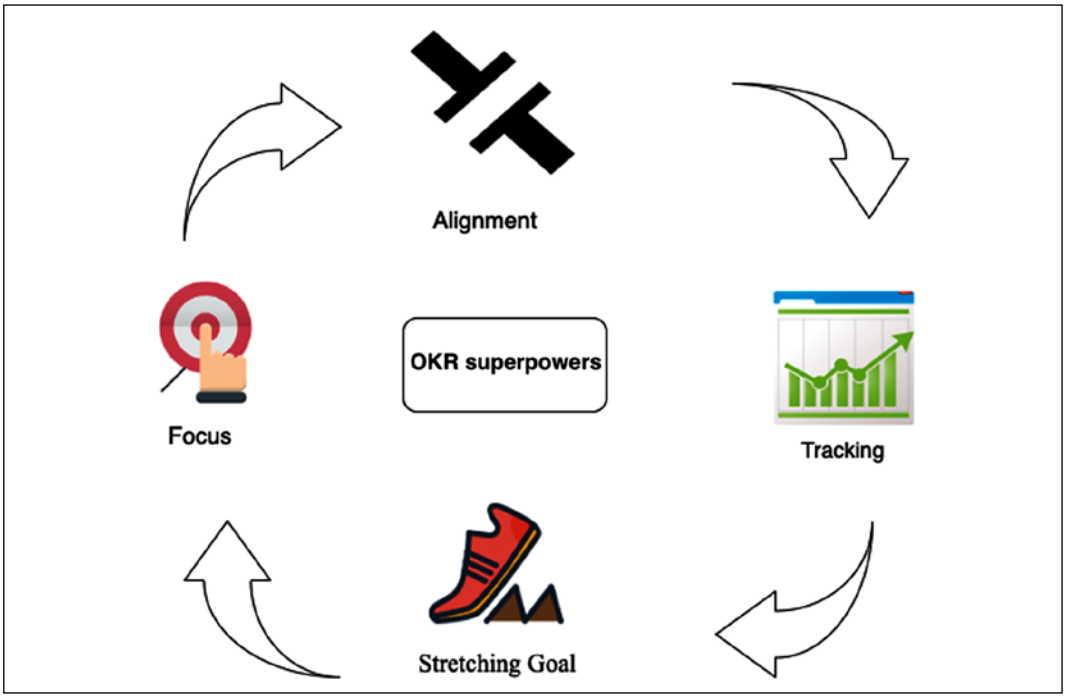


Figure 19.1: Superpowers of OKRs

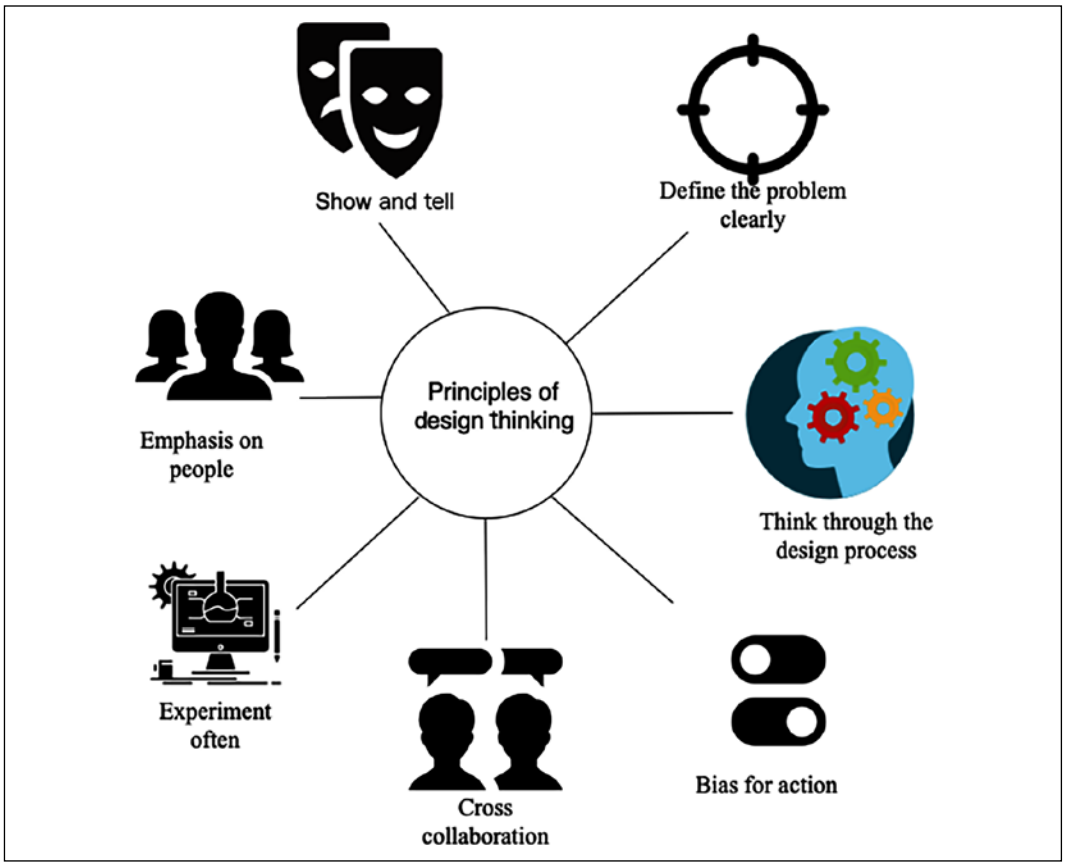


Figure 19.2: Principles of design thinking

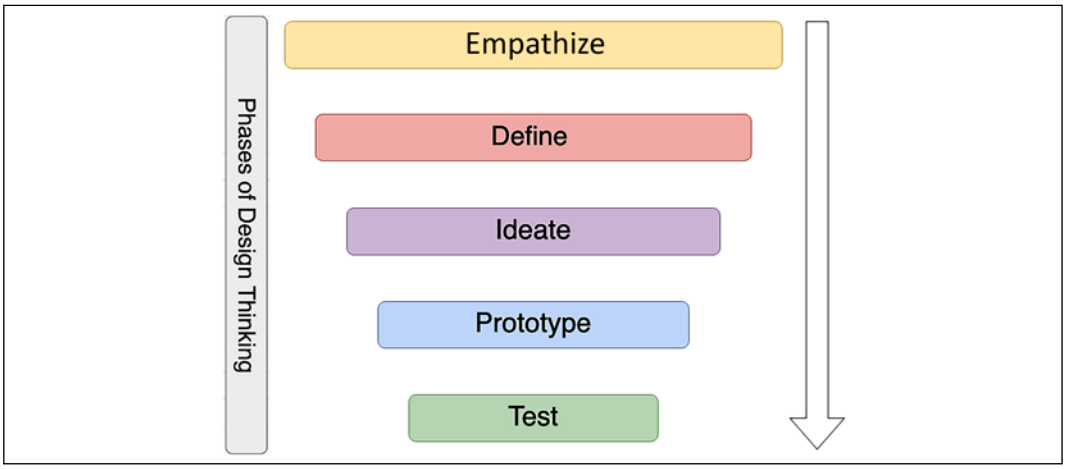


Figure 19.3: Five phases of design thinking