

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 Srivastava
Neelanjali Srivastav**

Packt

Solutions Architect's Handbook

Second Edition

Chapter 1

The Meaning of Solution Architecture

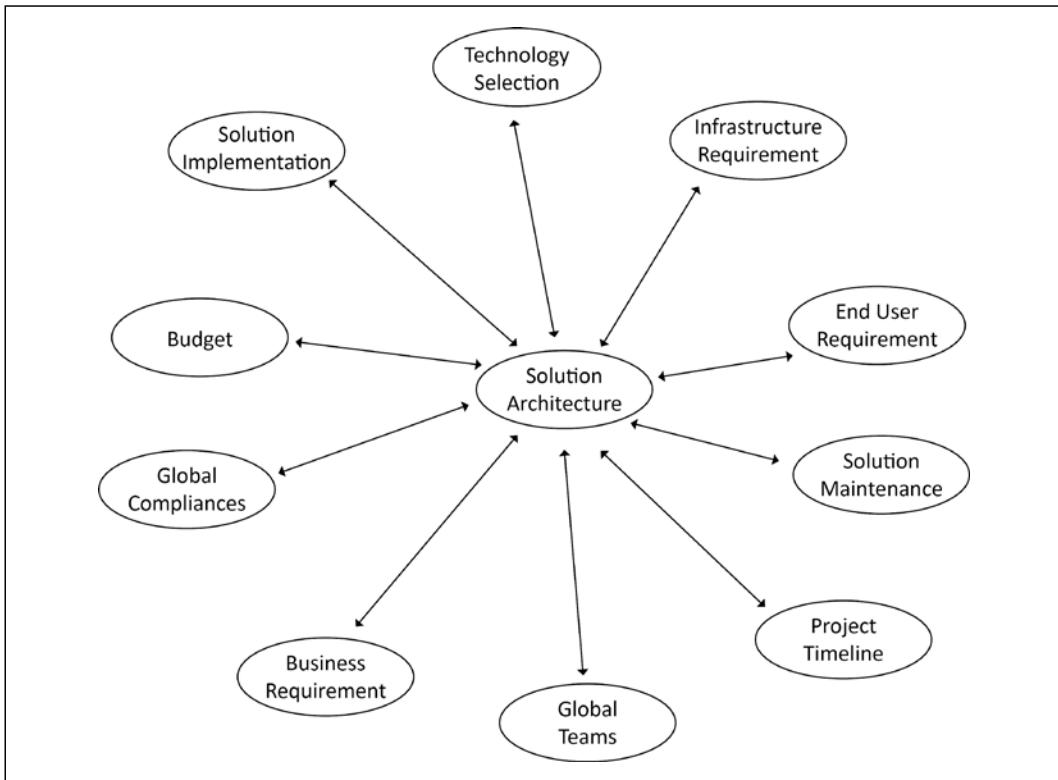


Figure 1.1: Circle of solution architecture

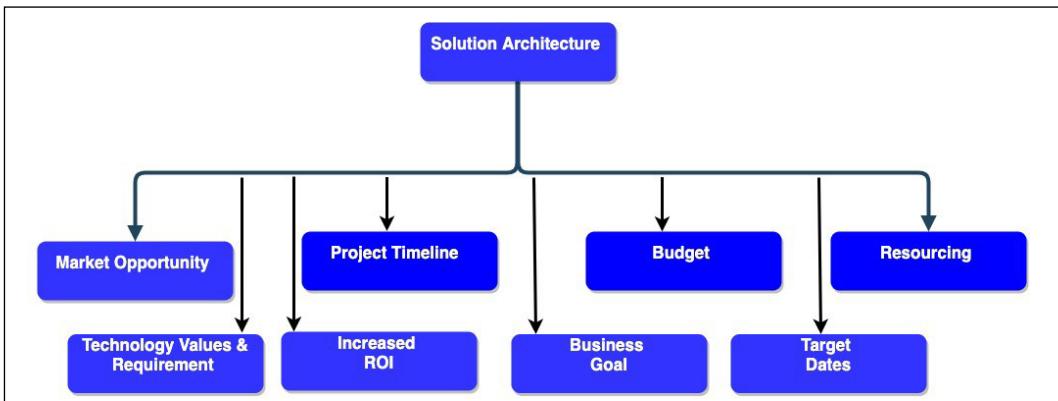


Figure 1.2: A solution architecture's beneficial attributes

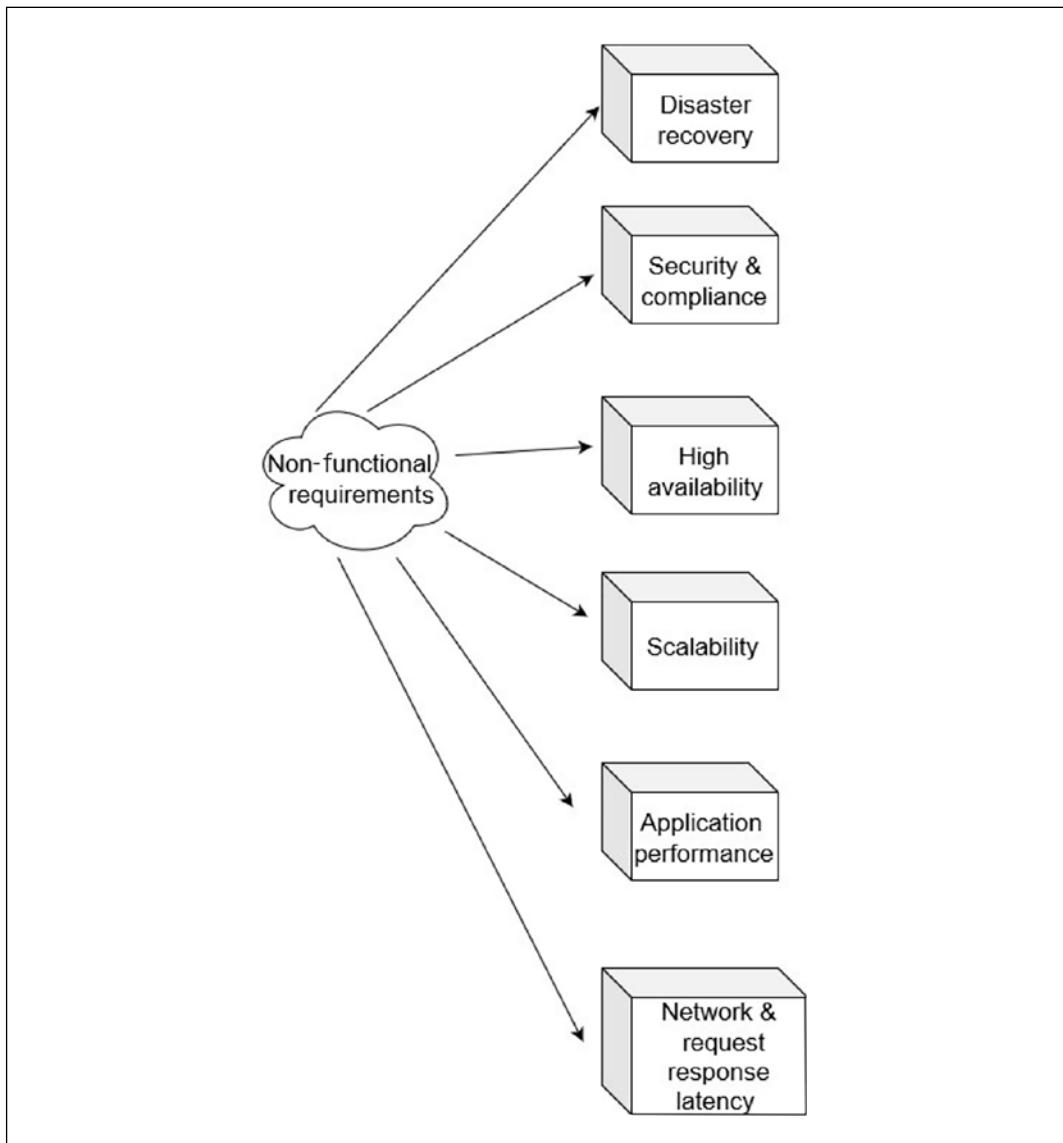


Figure 1.3: Non-functional attributes of solution architecture

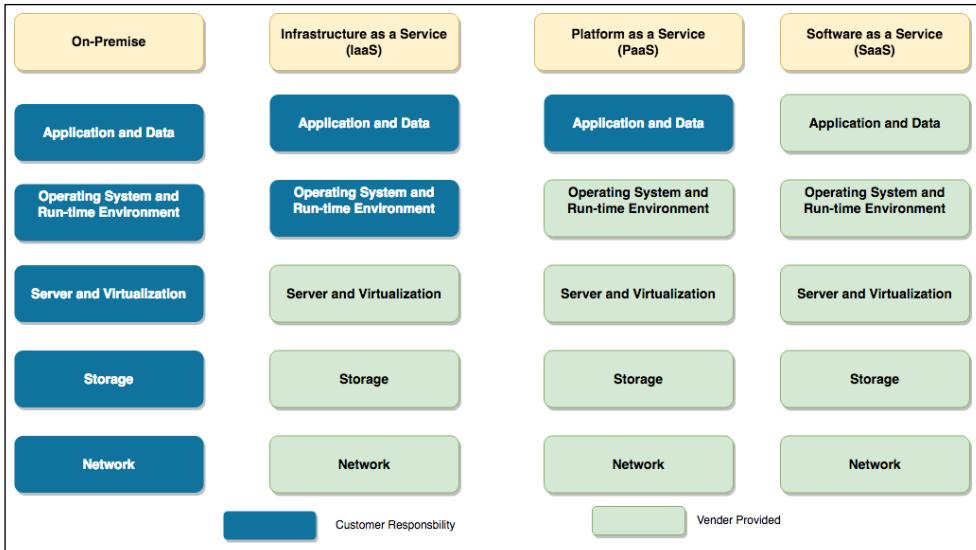


Figure 1.4: Types of cloud computing models

The screenshot shows the AWS Management Console interface. At the top, there's a search bar and a 'Customer Support' link. Below that is the main title 'AWS Management Console'. The left sidebar contains a navigation menu with several sections:

- AWS services**
- Recently visited services**
- All services**
 - Compute** (with EC2 highlighted by a red box)
 - Lightsail
 - Lambda
 - Batch
 - Elastic Beanstalk
 - Serverless Application Repository
 - AWS Outposts
 - EC2 Image Builder
 - Containers**
 - Elastic Container Registry
 - Elastic Container Service
 - Elastic Kubernetes Service
 - Red Hat OpenShift Service on AWS
 - Storage**
 - S3
 - EFS
 - FSx
 - S3 Glacier
 - Storage Gateway
 - AWS Backup
 - Developer Tools**
 - CodeStar
 - CodeCommit
 - CodeArtifact
 - CodeBuild
 - CodeDeploy
 - CodePipeline
 - Cloud9
 - CloudShell
 - X-Ray
 - AWS FIS
 - Customer Enablement**
 - AWS IQ
 - Support
 - Managed Services
 - Activate for Startups
 - Robotics**
 - AWS RoboMaker
 - Blockchain**
 - Amazon Managed Blockchain
 - Satellite**
- Machine Learning**
 - Amazon SageMaker
 - Amazon Augmented AI
 - Amazon CodeGuru
 - Amazon DevOps Guru
 - Amazon Comprehend
 - Amazon Forecast
 - Amazon Fraud Detector
 - Amazon Kendra
 - Amazon Lex
 - Amazon Personalize
 - Amazon Polly
 - Amazon Rekognition
 - Amazon Textract
 - Amazon Transcribe
 - Amazon Translate
 - AWS DeepComposer
 - AWS DeepLens
 - AWS DeepRacer
 - AWS Panorama
 - Amazon Monitron
 - Amazon HealthLake
 - Amazon Lookout for Vision
 - Amazon Lookout for Equipment
 - Amazon Lookout for Metrics
- AWS Cost Management**
 - AWS Cost Explorer
 - AWS Budgets
 - AWS Marketplace Subscriptions
- Front-end Web & Mobile**
 - AWS Amplify
 - Mobile Hub
 - AWS AppSync
 - Device Farm
 - Amazon Location Service
- AR & VR**
 - Amazon Sumerian
- Application Integration**
 - Step Functions
 - AWS AppFlow
 - Amazon EventBridge
 - Amazon MQ
 - Simple Notification Service
 - Simple Queue Service
 - SWF
 - Managed Apache Airflow

Figure 1.5: AWS console and service offerings

Chapter 2

Solution Architects in an Organization

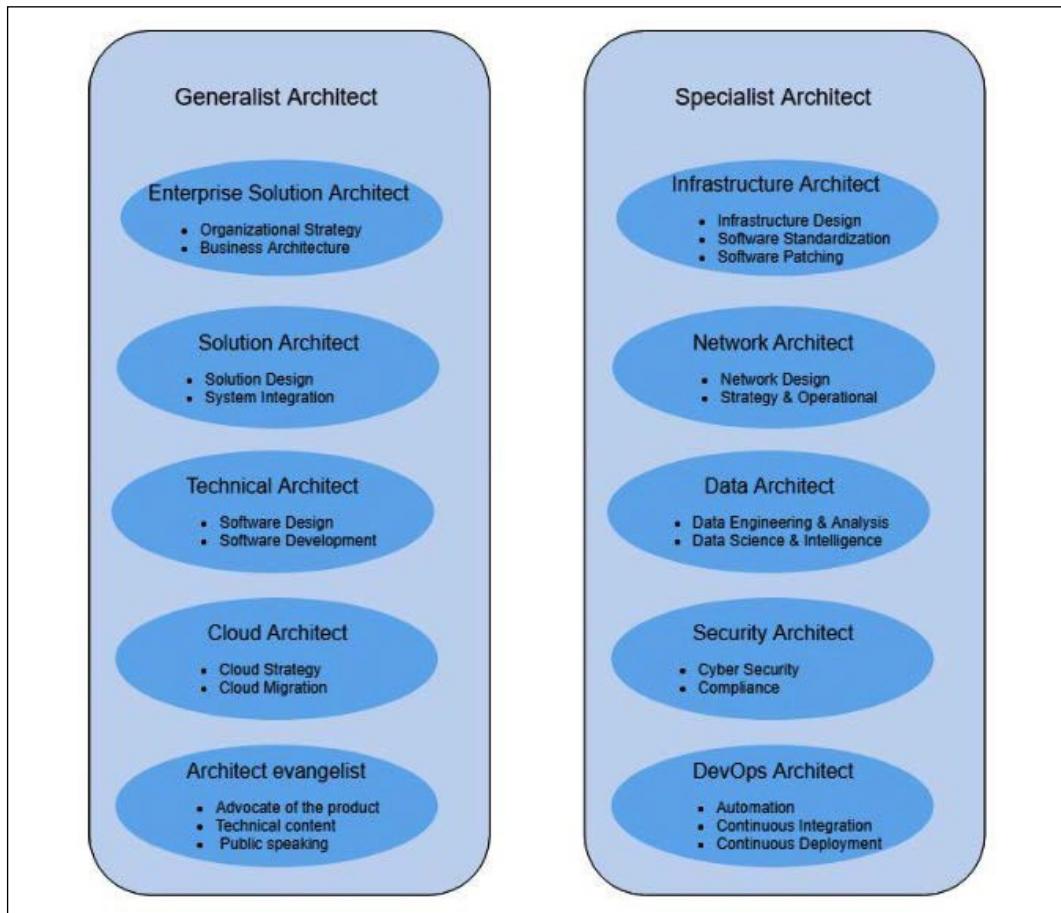


Figure 2.1: Types of solution architect

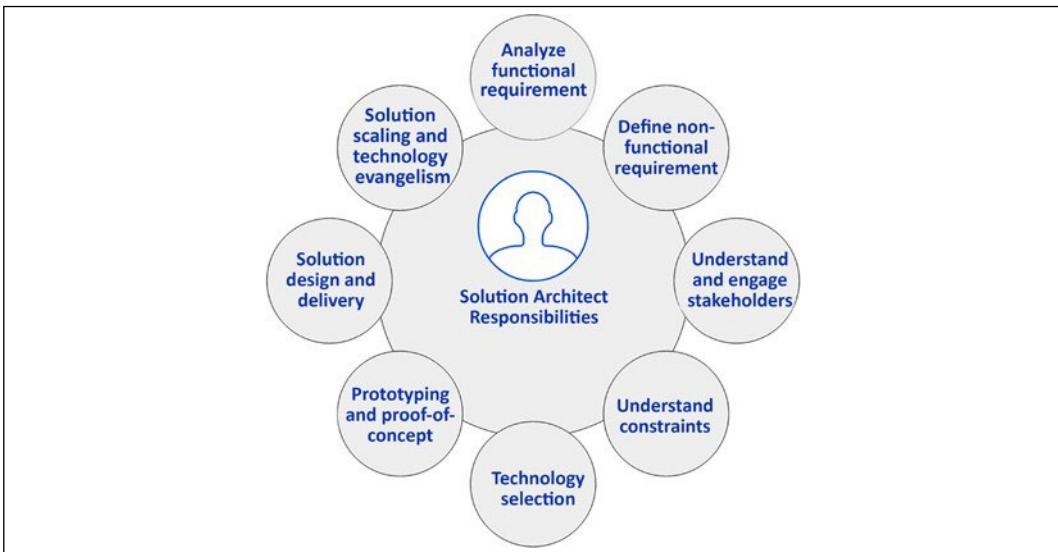


Figure 2.2: Solution architect's responsibility model

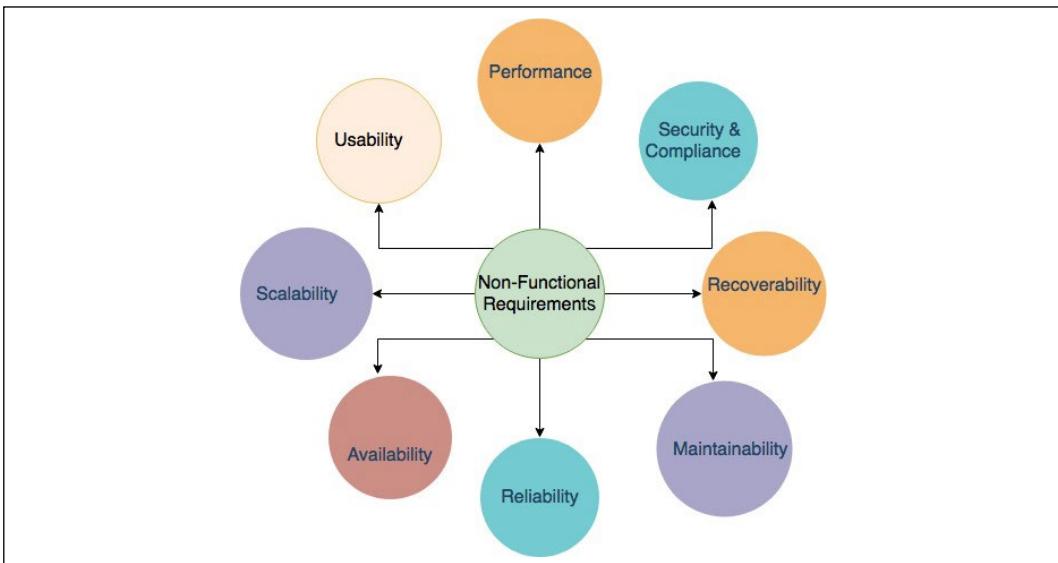


Figure 2.3: NFRs in a solution design

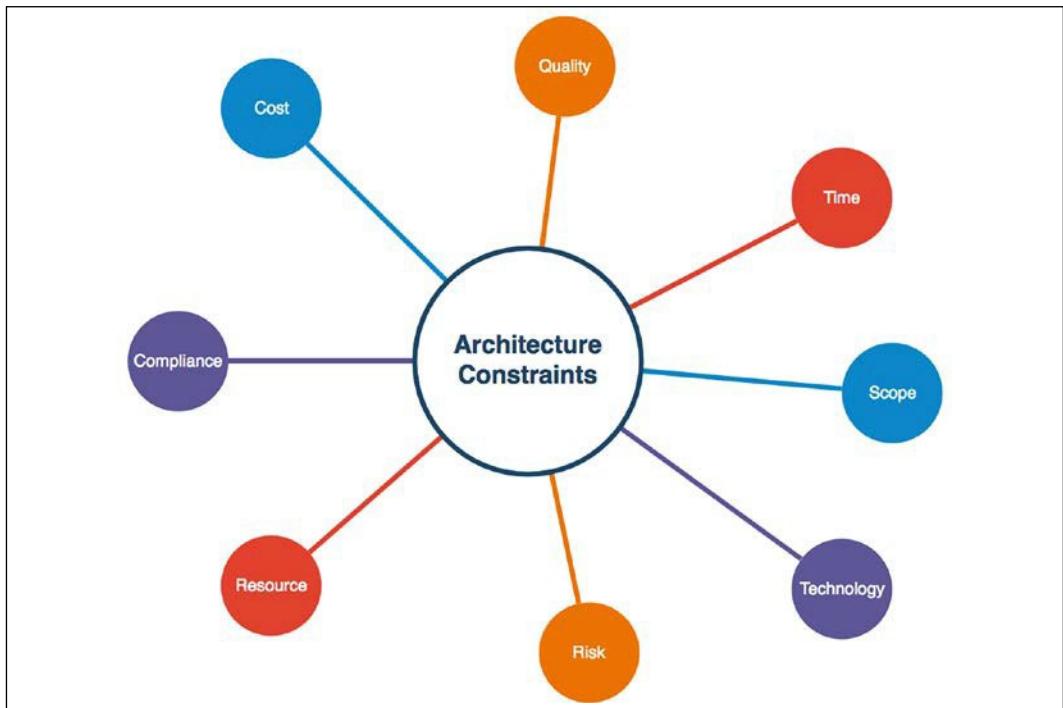


Figure 2.4: Architectural constraints in a solution design

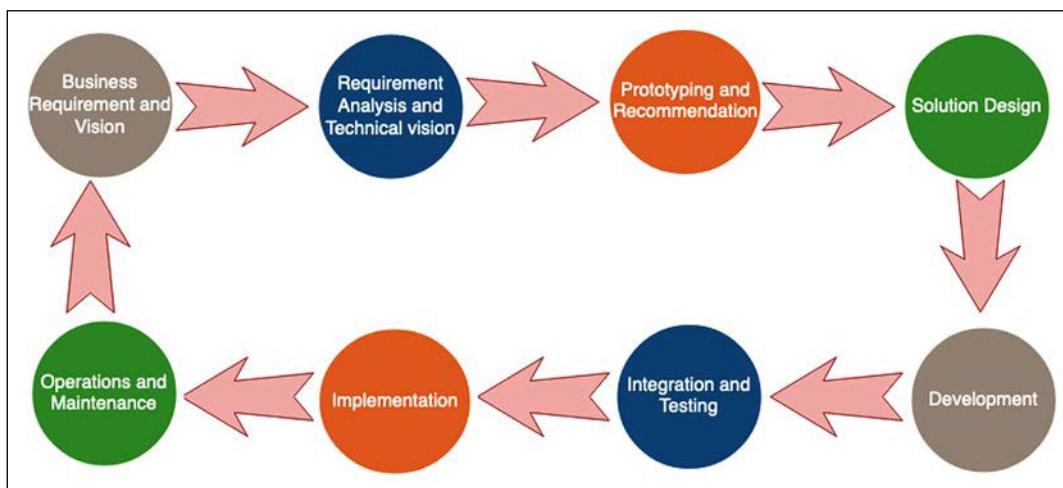


Figure 2.5: Solution delivery life cycle

Chapter 3

Attributes of the Solution Architecture

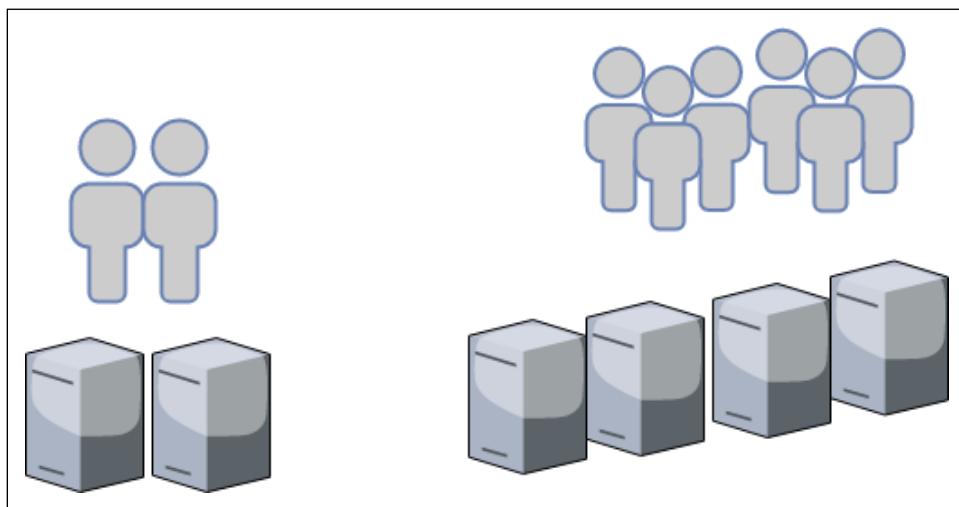


Figure 3.1: Horizontal scaling

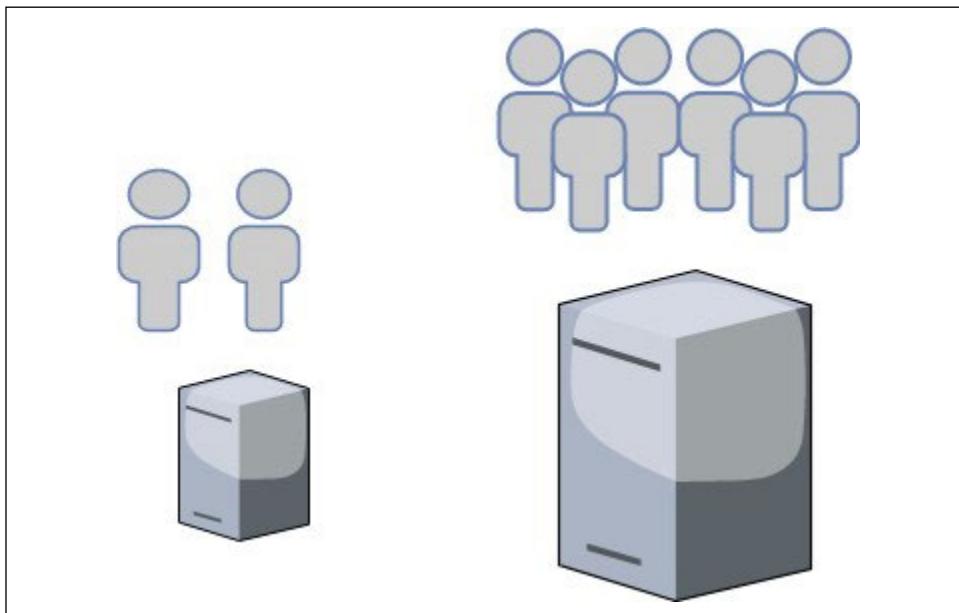


Figure 3.2: Vertical scaling

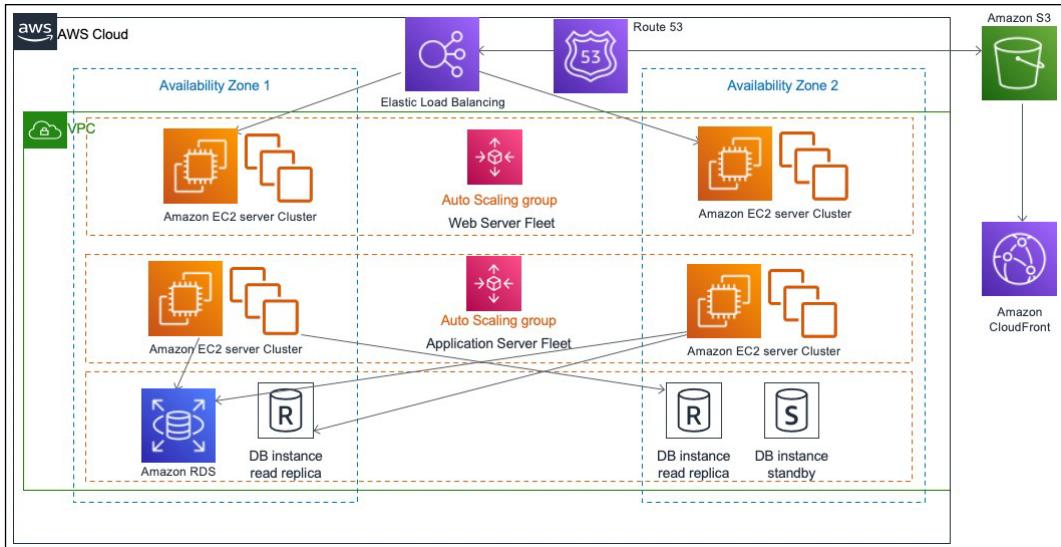


Figure 3.3: Scaling three-tier architecture

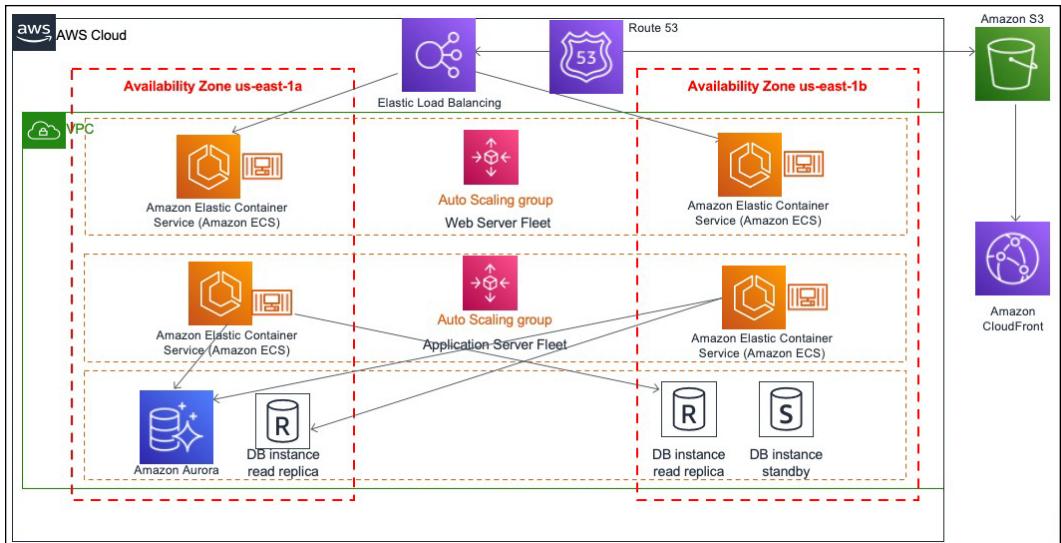
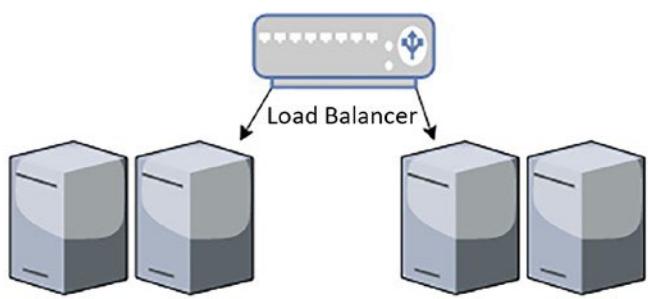
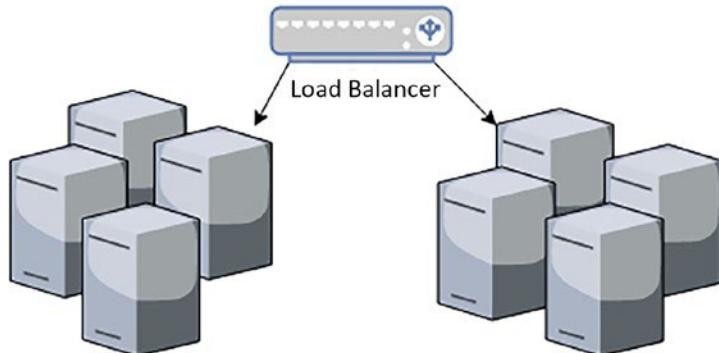


Figure 3.4: High availability and resilience architecture



100% high availability, 50% fault-tolerance



100% high availability, 100% fault-tolerance

Figure 3.5: Fault-tolerance architecture

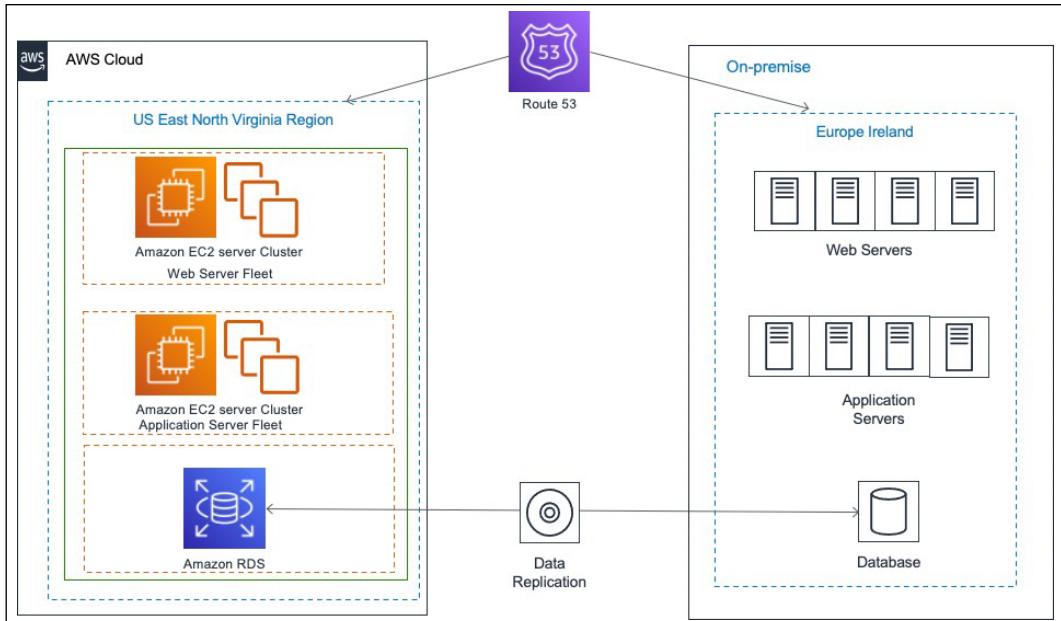


Figure 3.6: Hybrid multi-site disaster recovery architecture

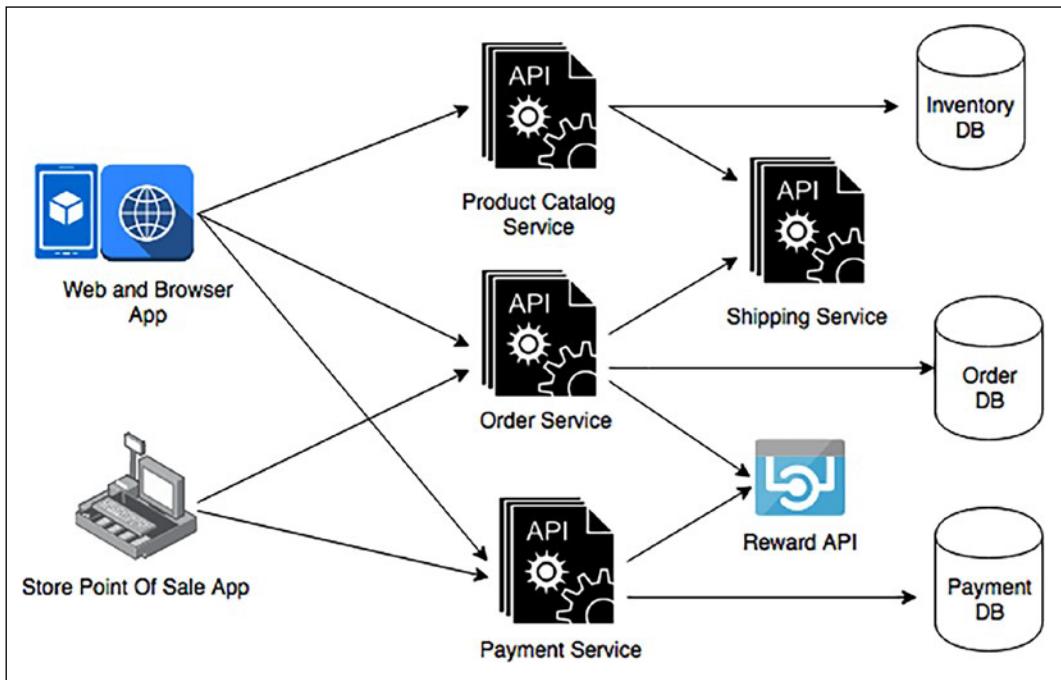


Figure 3.7: Extensible API-based architecture

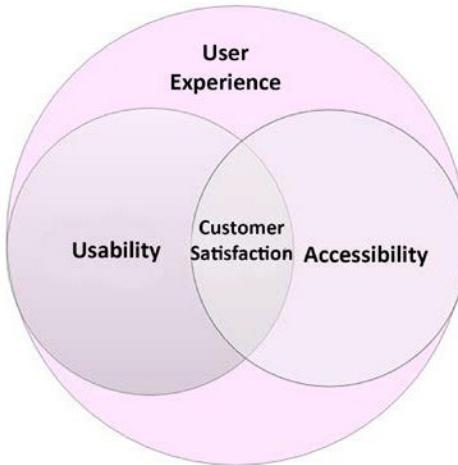


Figure 3.8: Customer satisfaction with usability and accessibility

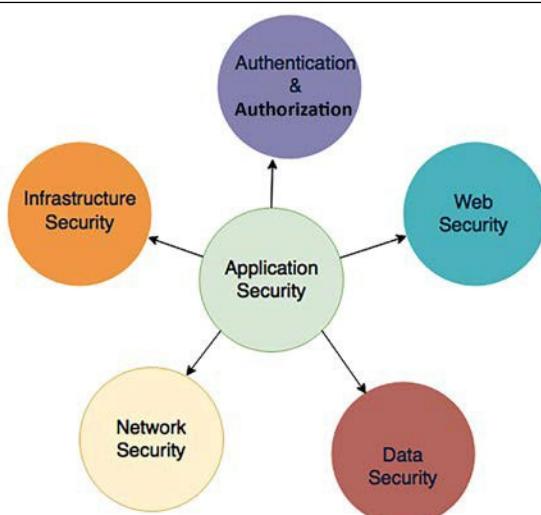


Figure 3.9: Security aspects in solution design

Chapter 4

Principles of Solution Architecture Design

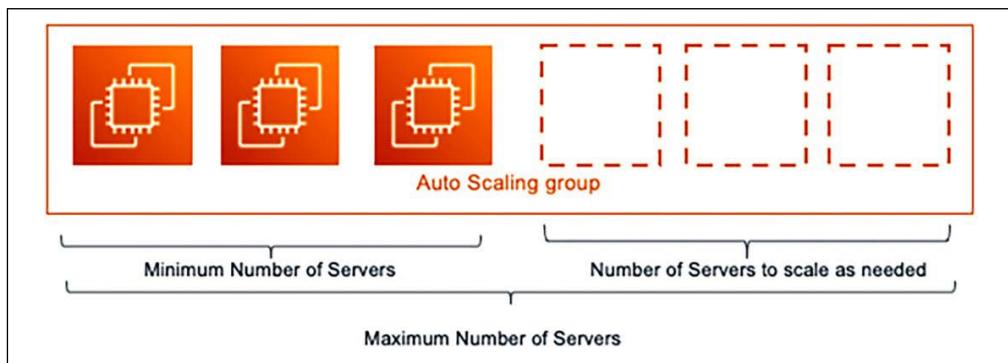


Figure 4.1: Server Auto Scaling

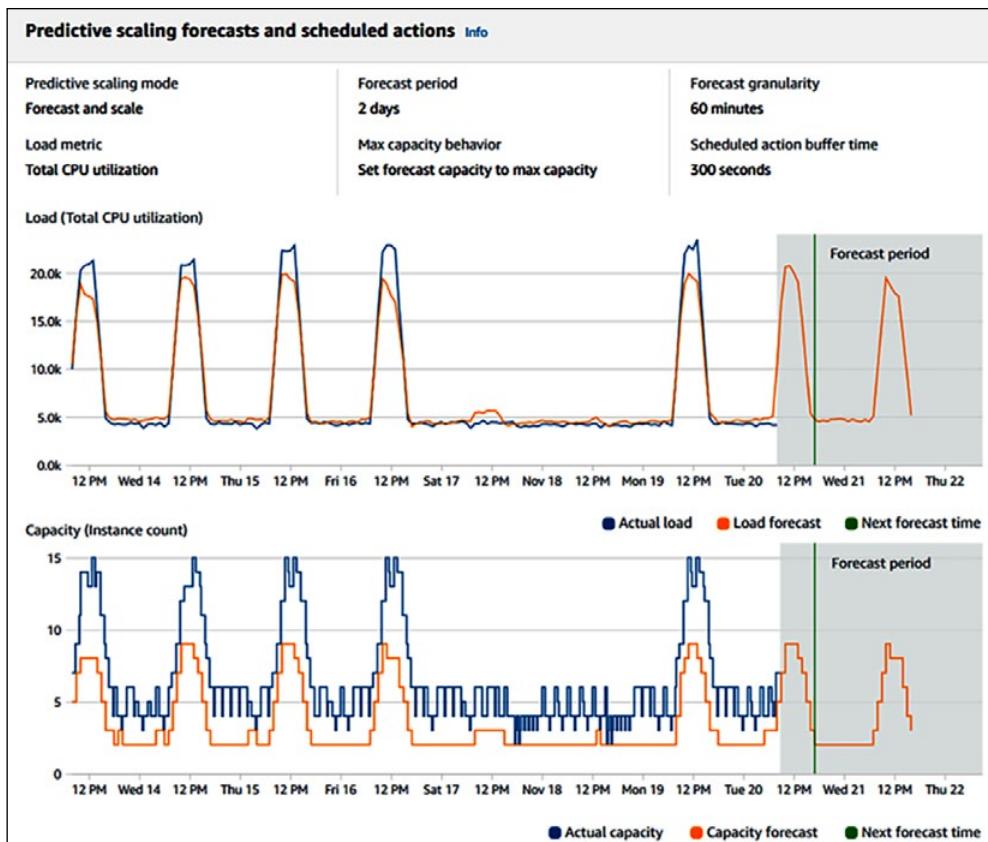


Figure 4.2: Predictive scaling forecast

Scheduled scaling actions (32)		
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

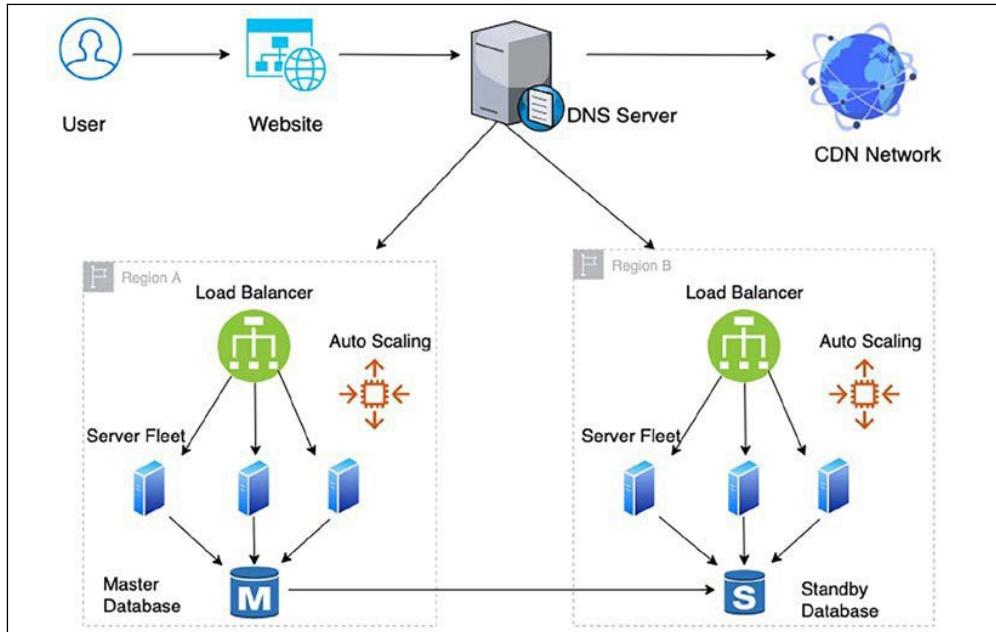


Figure 4.4: Application architecture resiliency

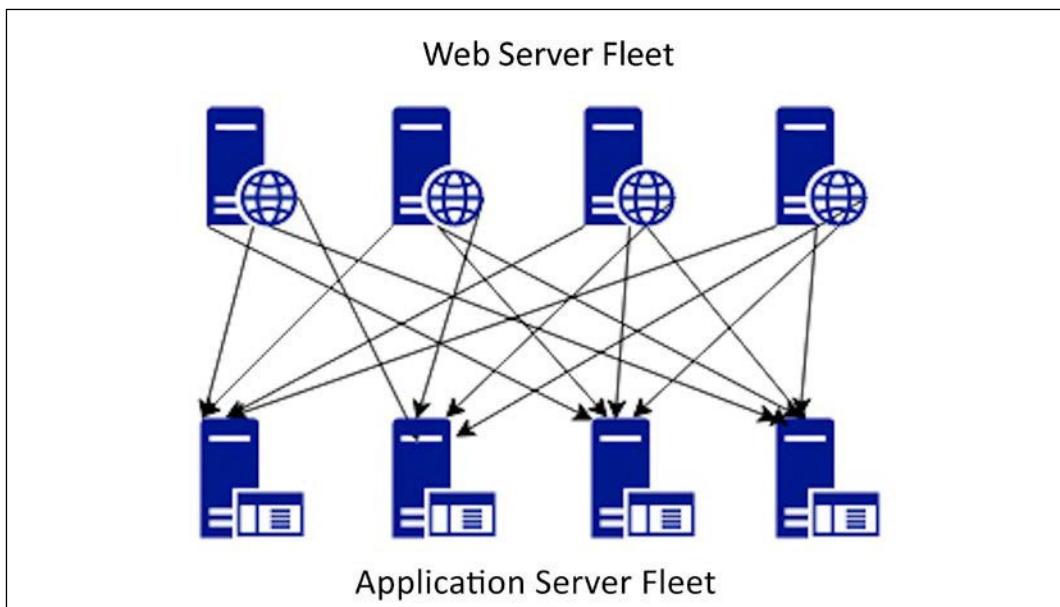


Figure 4.5: Tightly coupled architecture

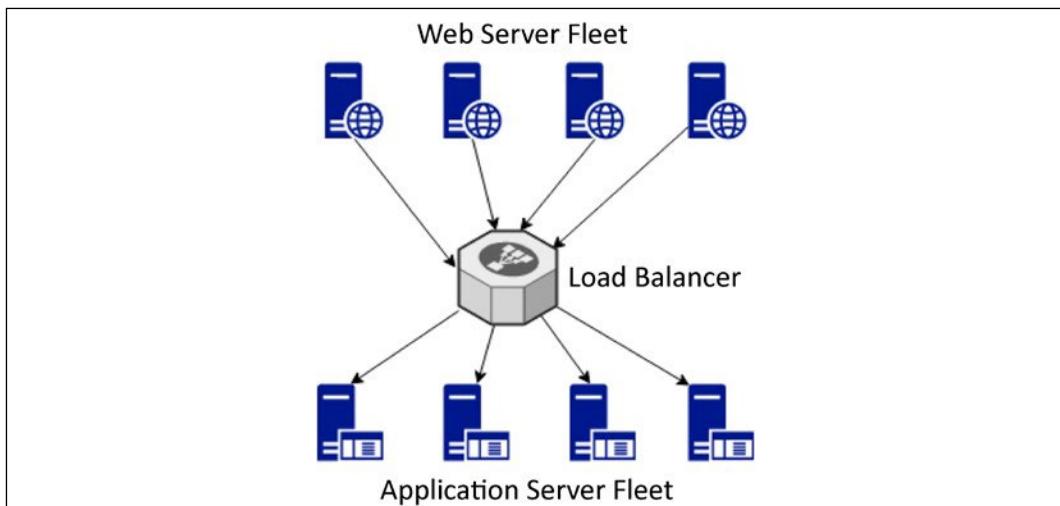


Figure 4.6: Load balancer-based, loosely coupled architecture

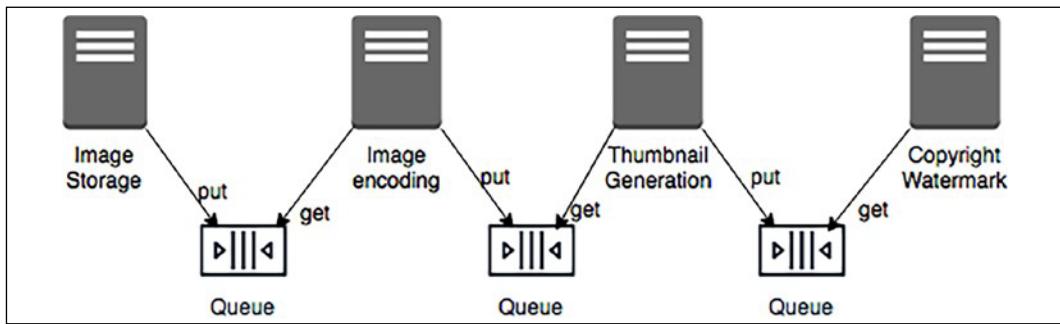


Figure 4.7: Queue-based, loosely coupled architecture

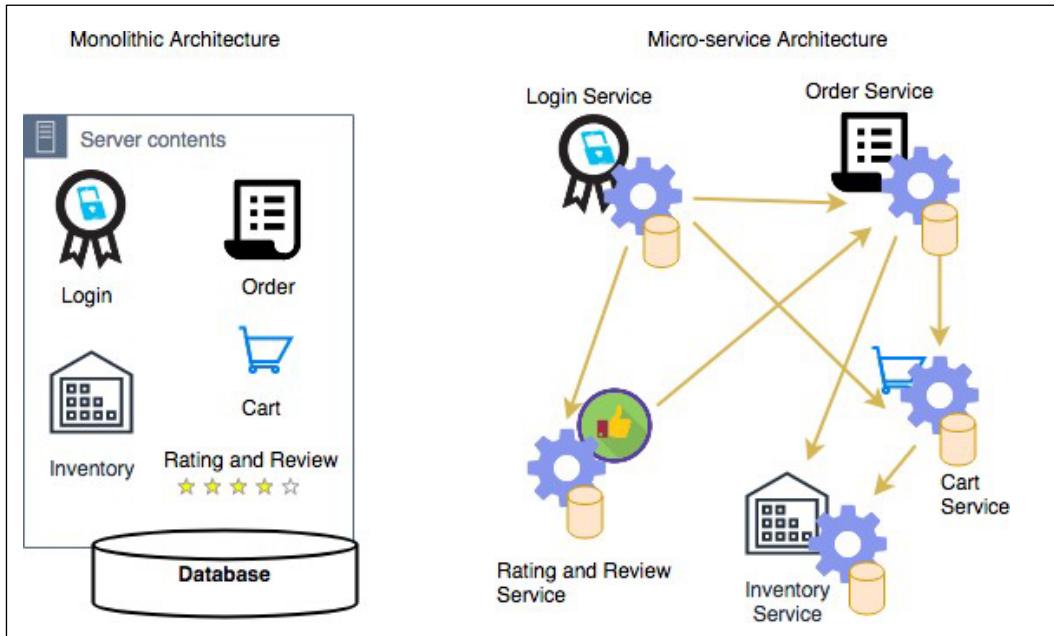


Figure 4.8: Monolithic and microservice architectures

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

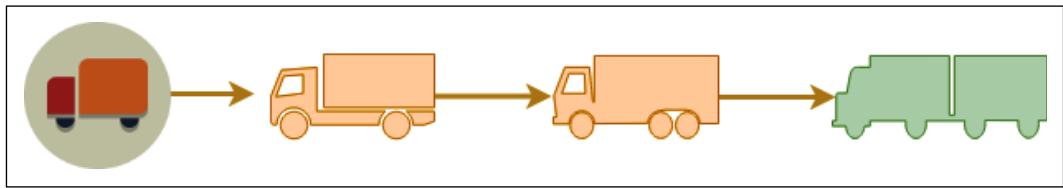


Figure 4.9: MVP approach to building the solution

Chapter 5

Cloud Migration and Hybrid Cloud Architecture Design

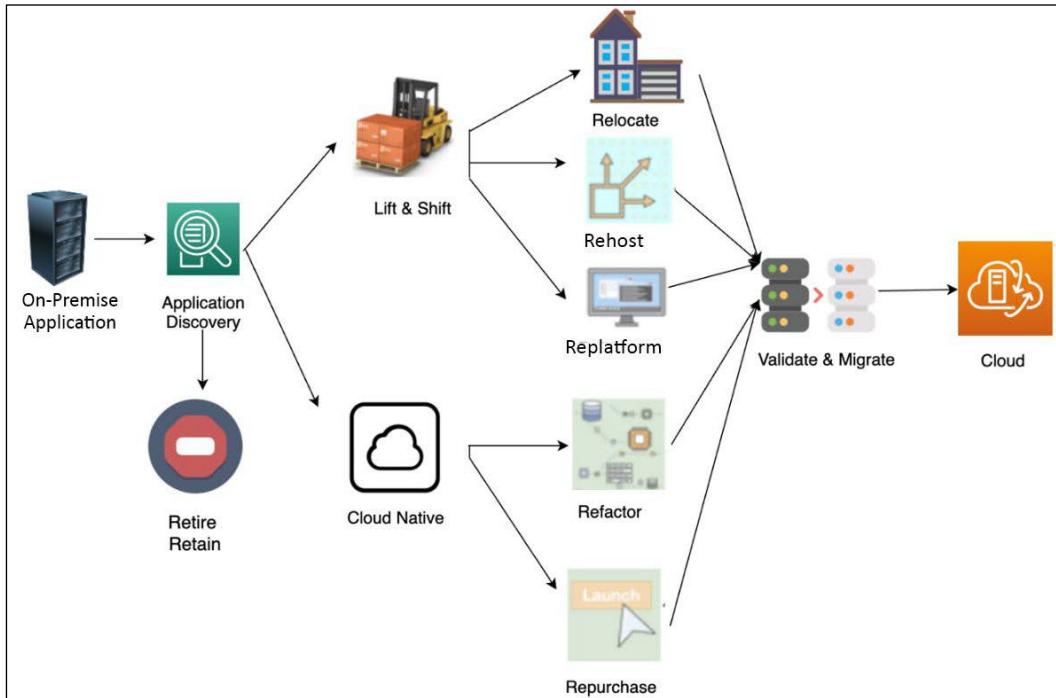


Figure 5.1: Cloud migration strategy

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 environment 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 assets that are no longer useful and remove them entirely	NA	NA
--------	--	----	----

Table 5.1: Migration strategies based on effort and opportunity

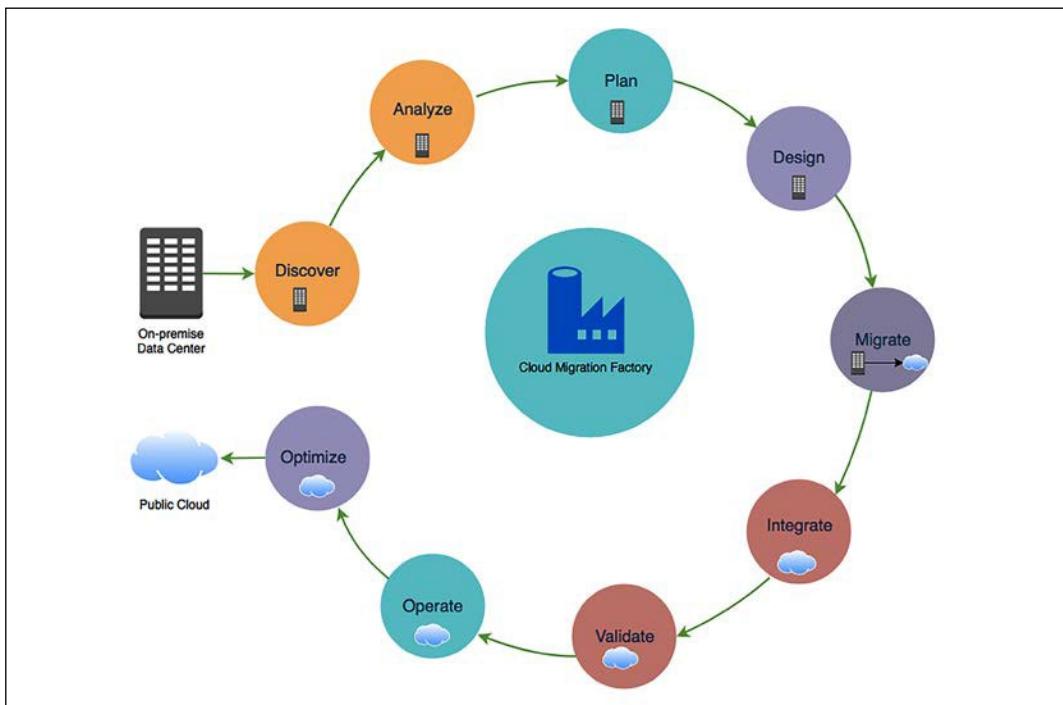


Figure 5.2: Cloud migration steps

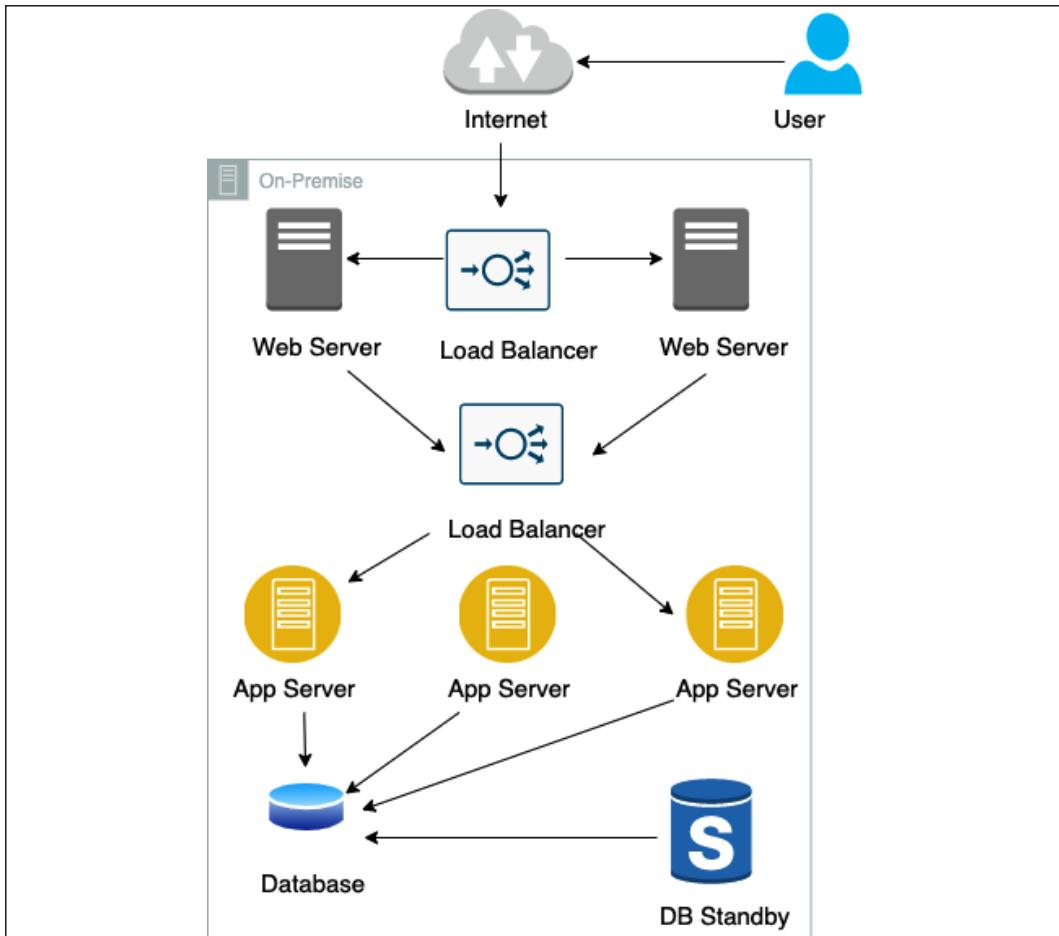


Figure 5.3: On-premise architecture mapping

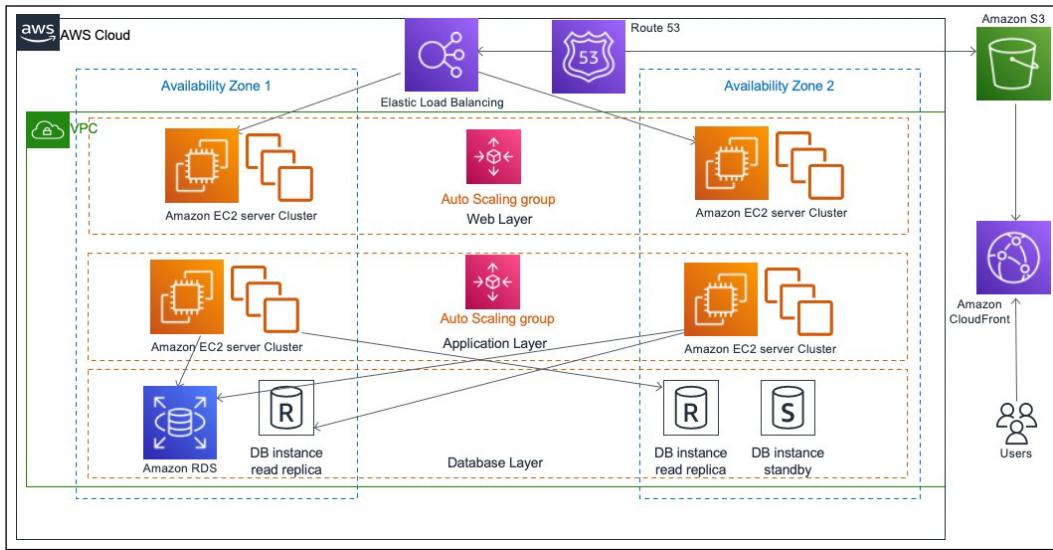


Figure 5.4: On-premise to AWS cloud architecture mapping

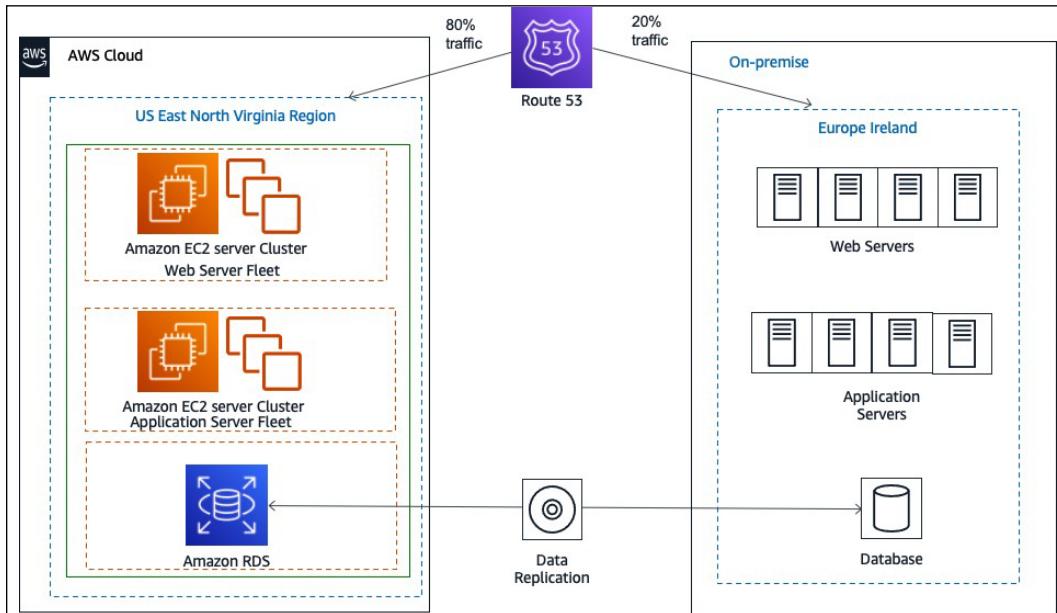


Figure 5.5: Live migration cutover using blue-green deployment

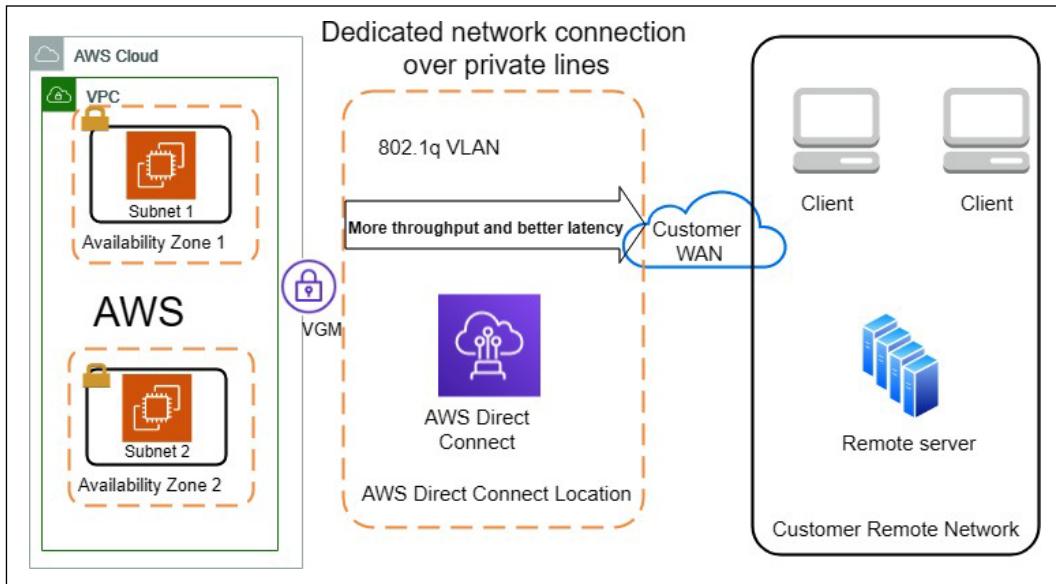


Figure 5.6: Hybrid cloud architecture (on-premises to cloud connectivity)

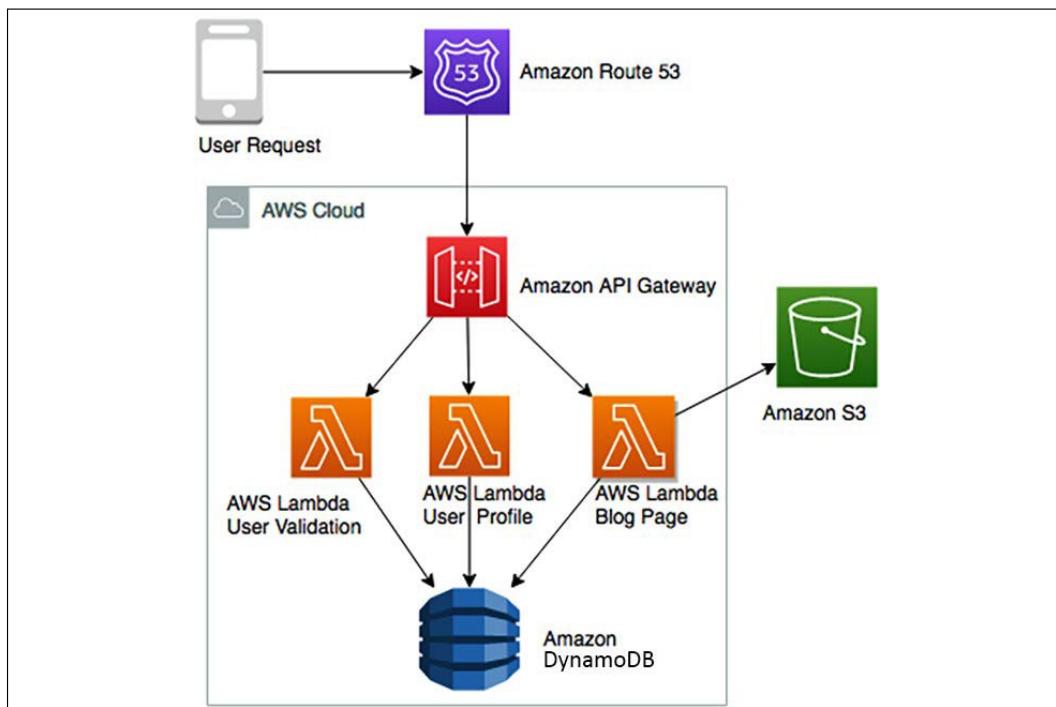


Figure 5.7: Cloud native micro-blogging application architecture

Chapter 6

Solution Architecture Design Patterns

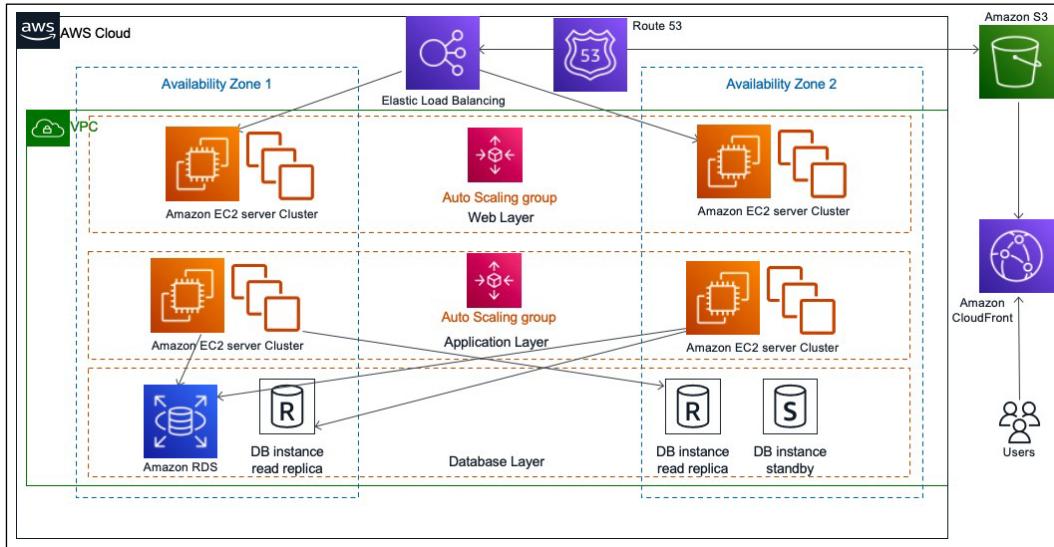


Figure 6.1: Three-tier website architecture

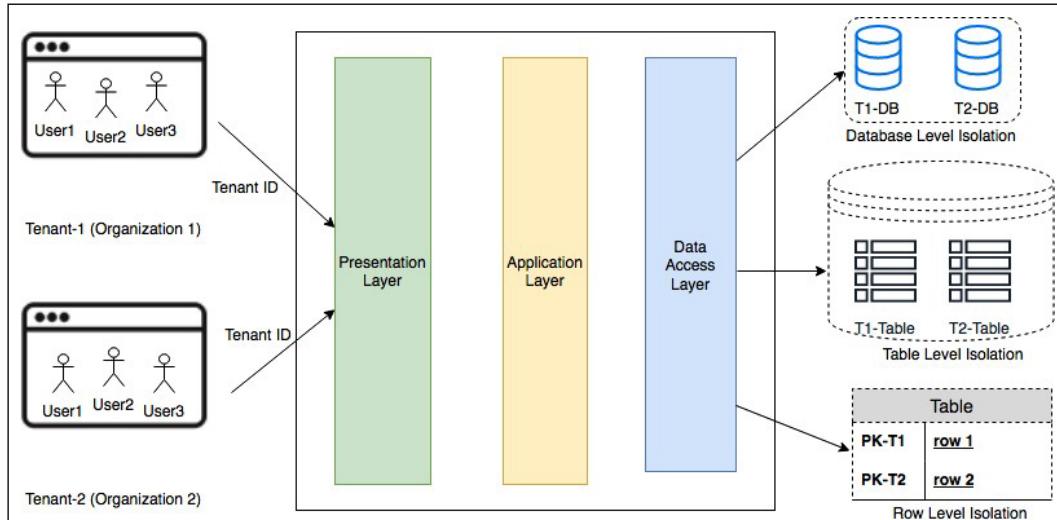


Figure 6.2: Multi-tenant SaaS architecture

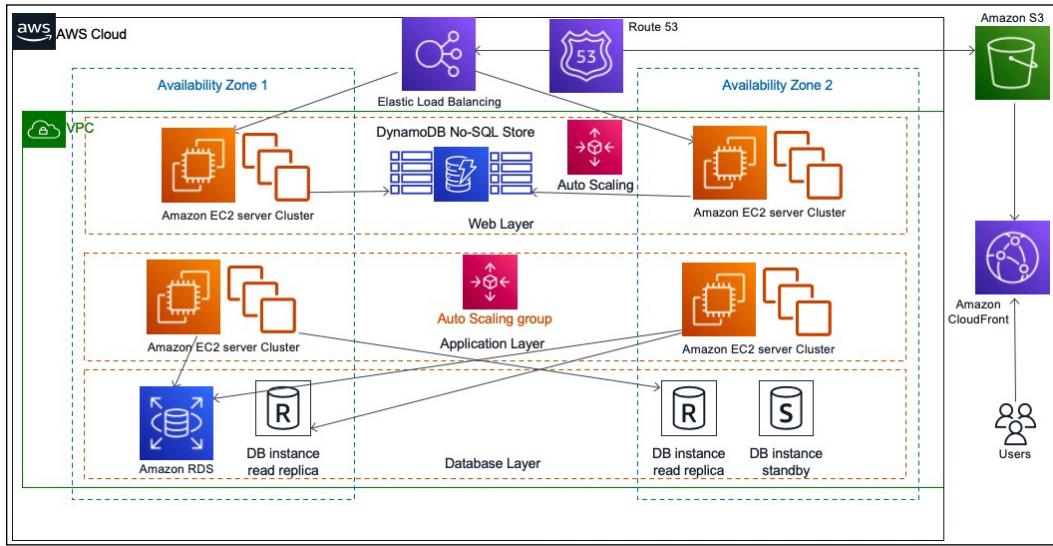


Figure 6.3: A stateless application architecture

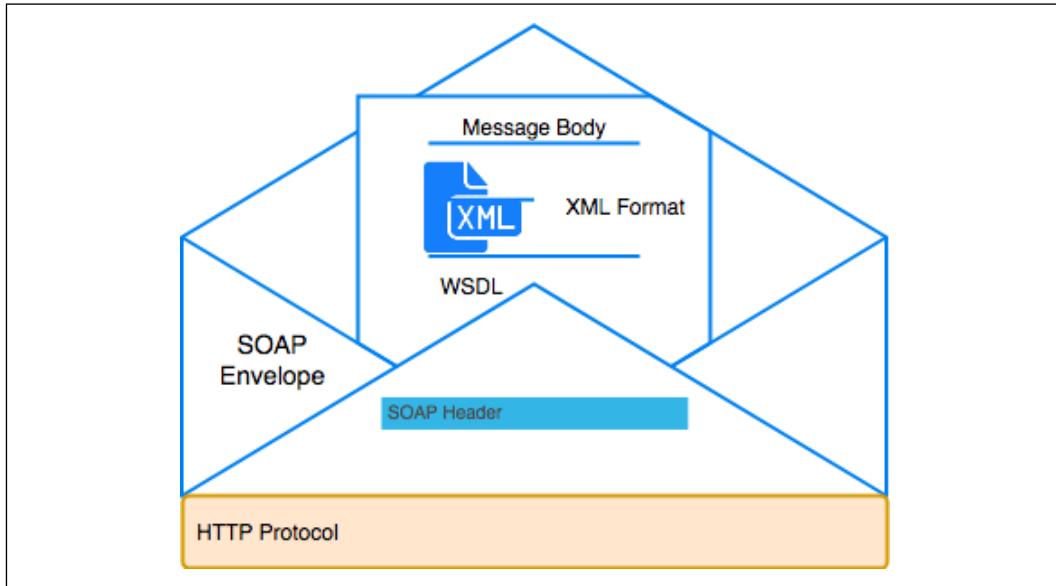


Figure 6.4: SOAP envelope for web service data exchange

```

<?xml 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:orderno>
            </m:getorderinfo>
        </m:order>
    </env:Body>

```

Code listing 6.1: Example SOAP envelope 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

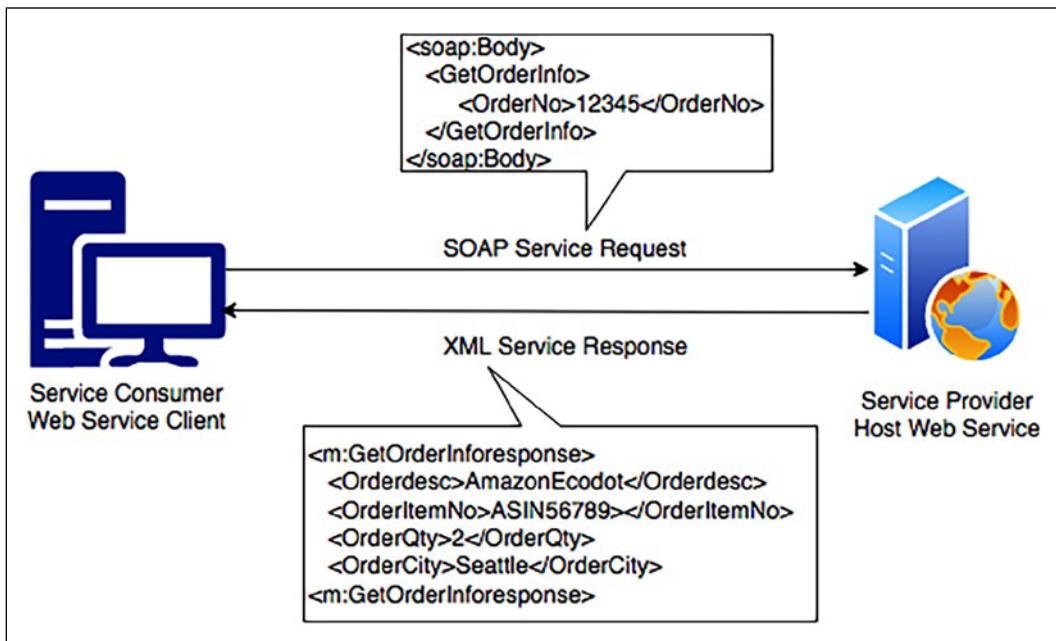


Figure 6.5: SOAP-based web service

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

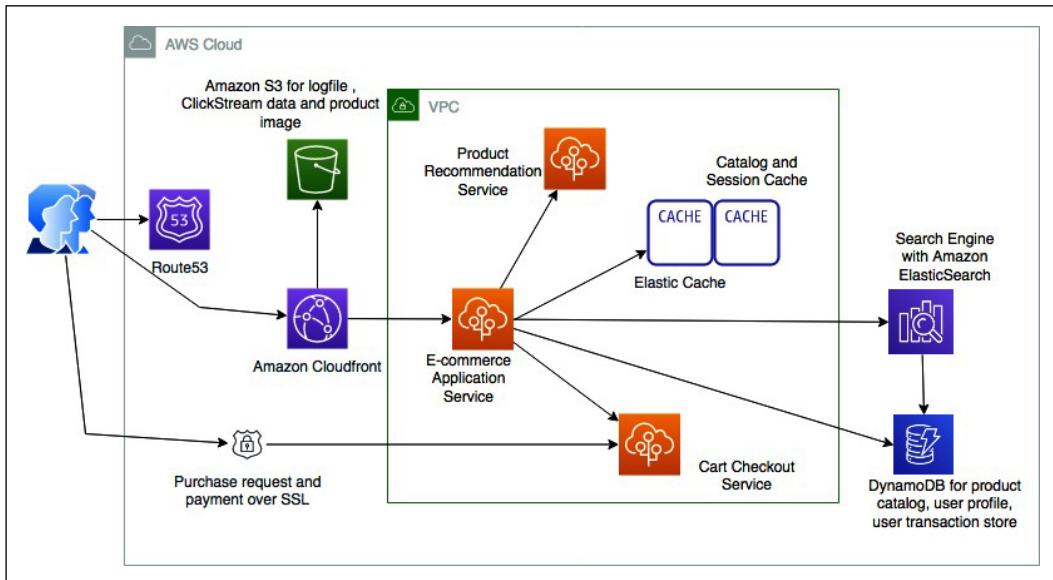


Figure 6.6: E-commerce website SOA

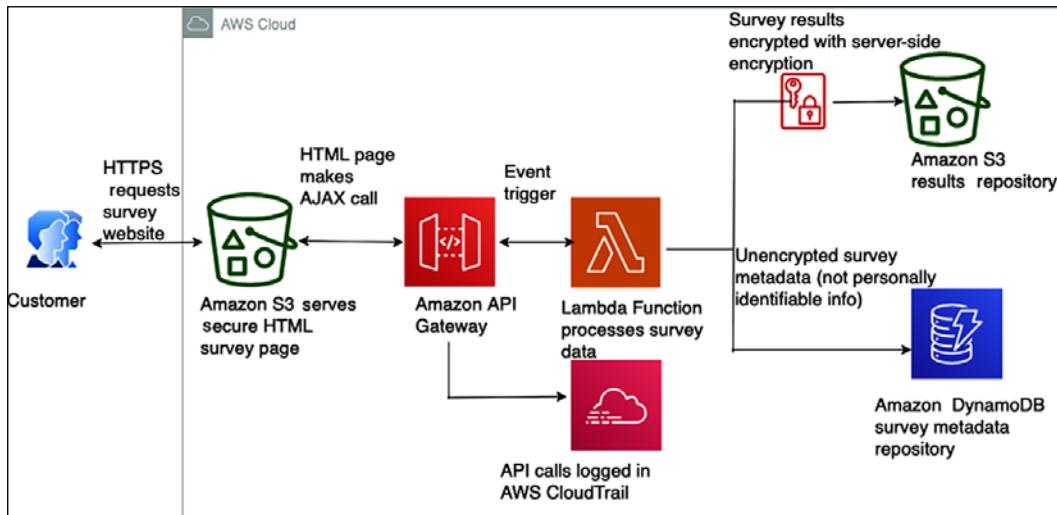


Figure 6.7: Serverless architecture for a secure survey delivery

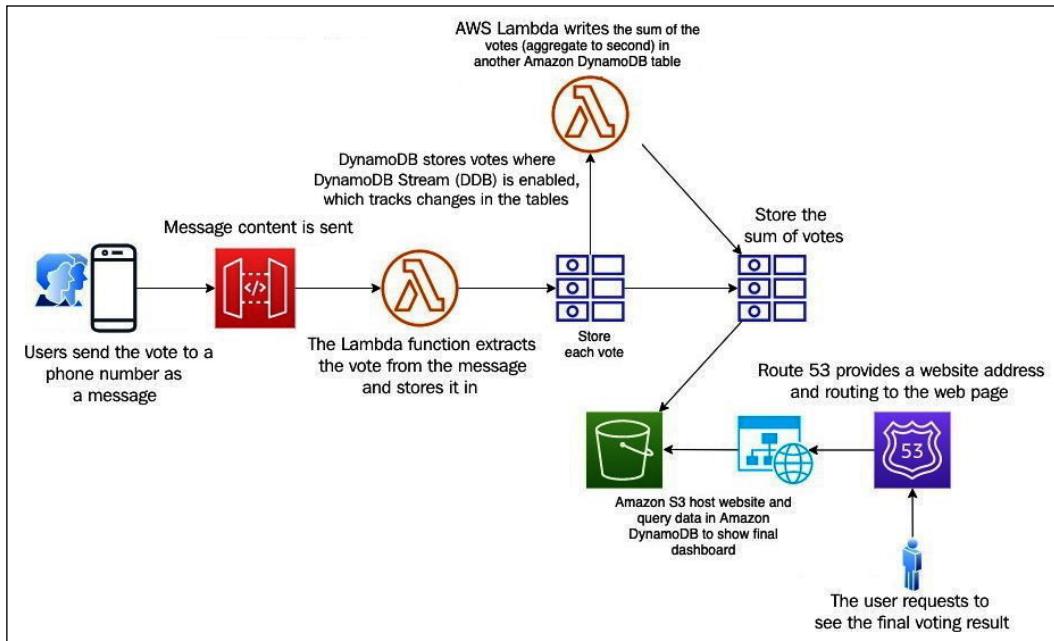


Figure 6.8: Microservice-based real-time voting application architecture

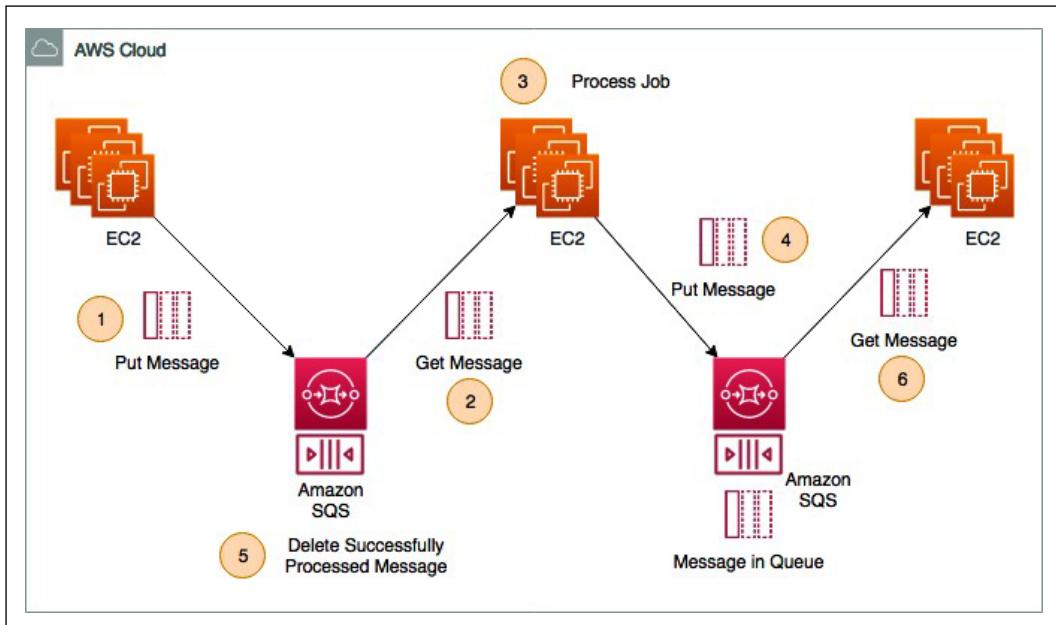


Figure 6.9: Queuing chain pattern architecture

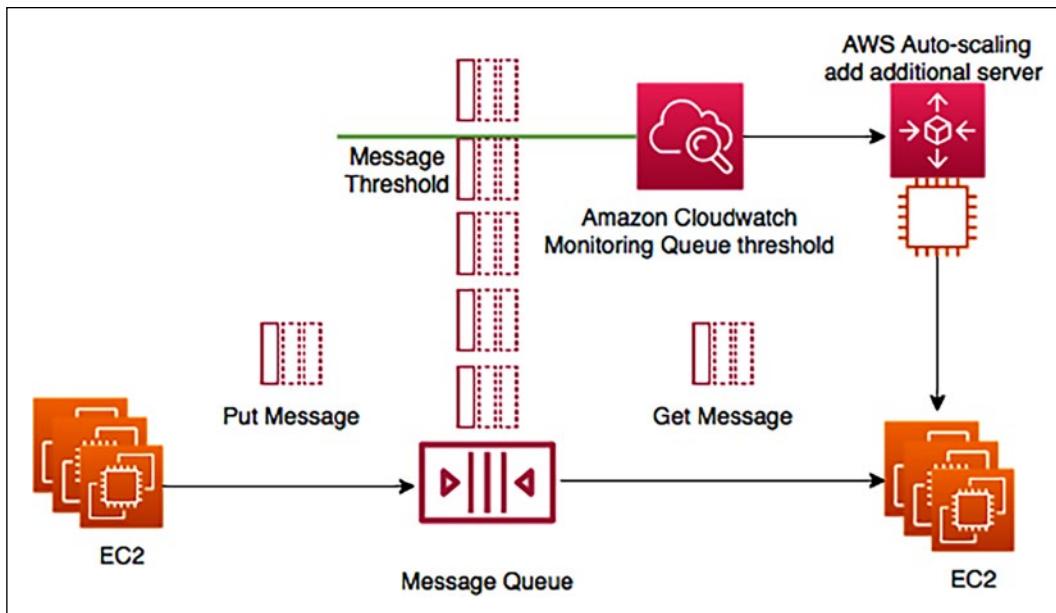


Figure 6.10: Job observer pattern architecture

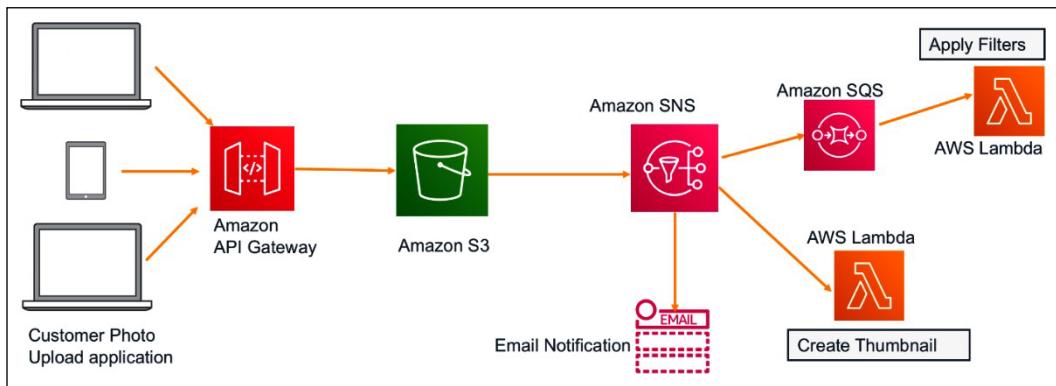


Figure 6.11: Photo studio application pub/sub event-driven architecture

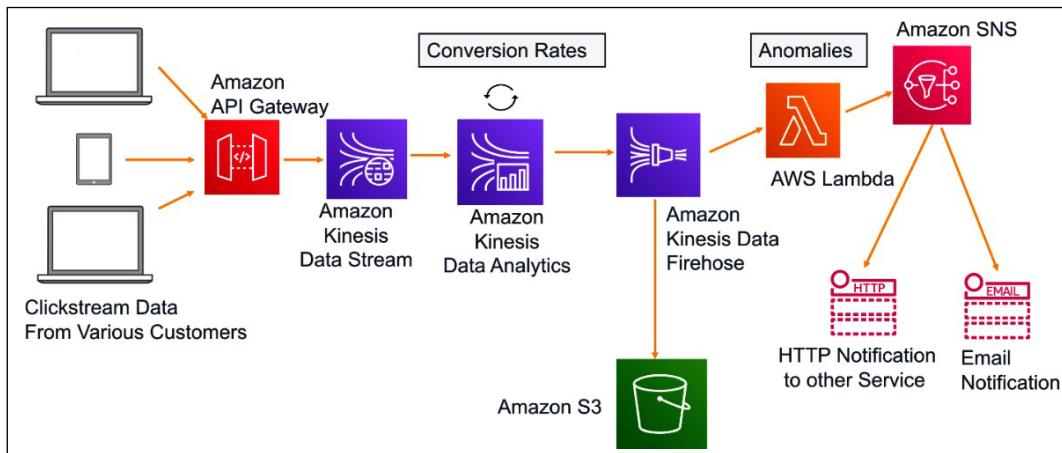


Figure 6.12: Clickstream analysis event stream architecture

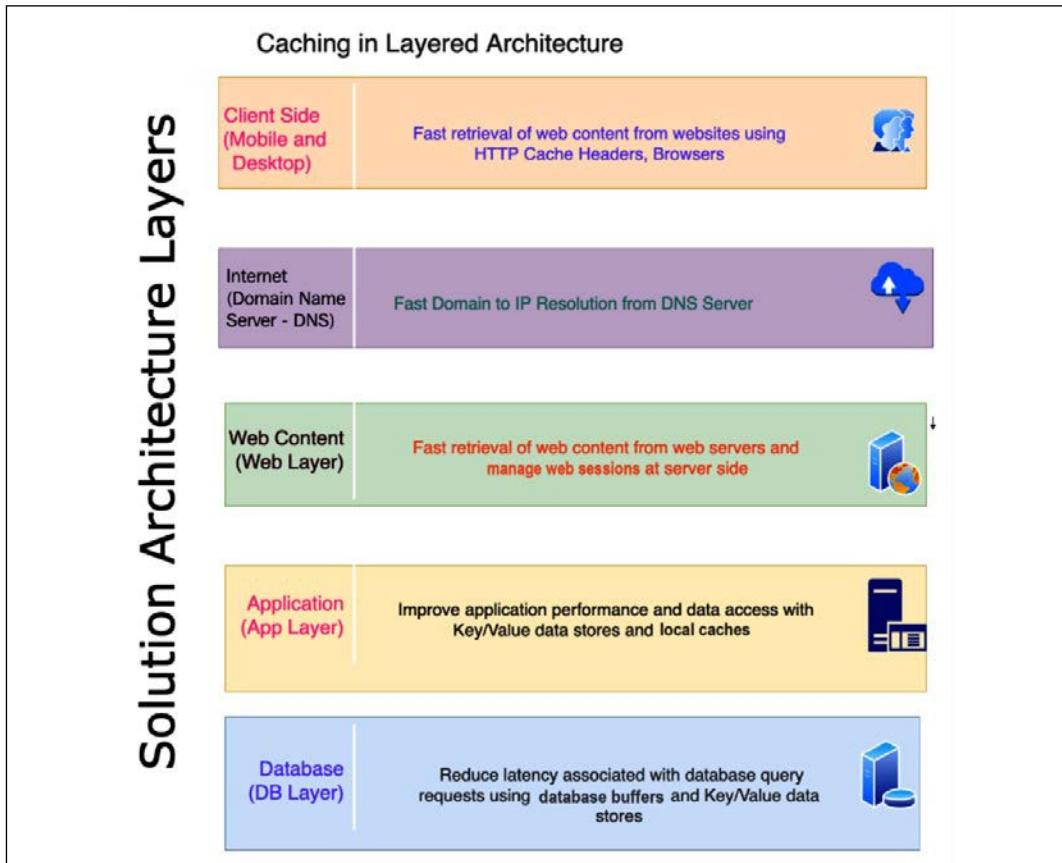


Figure 6.13: Caching at the architecture layers

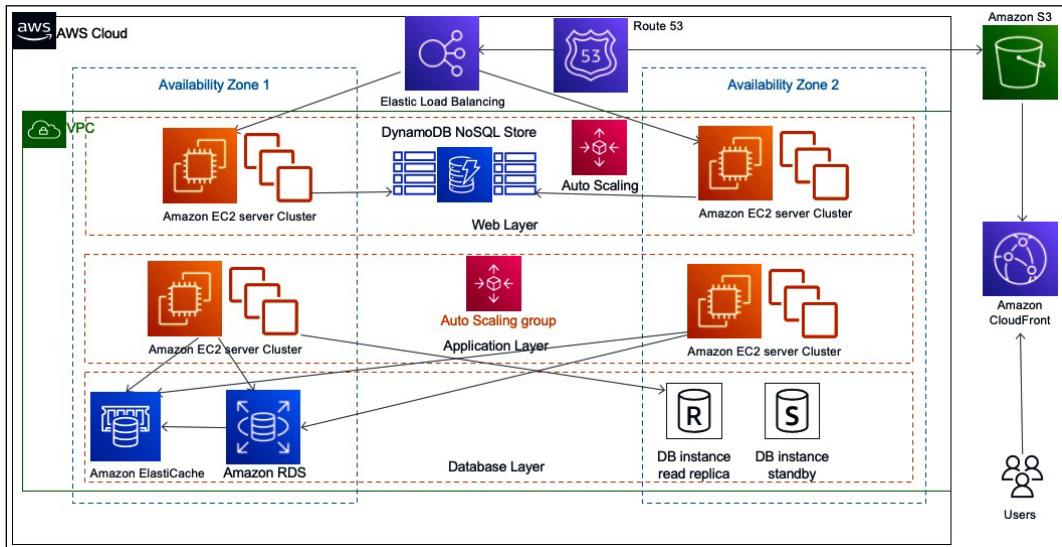


Figure 6.14: Cache distribution pattern architecture

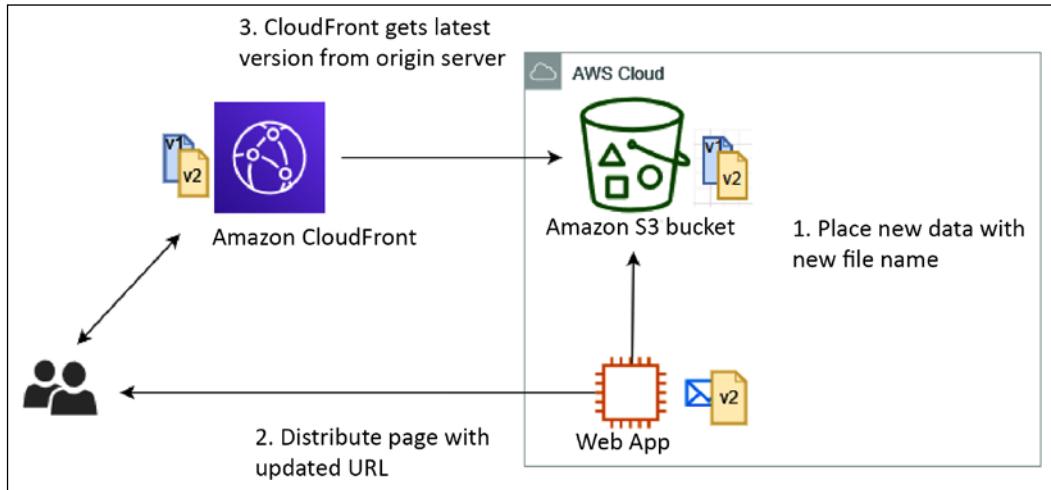


Figure 6.15: Rename distribution pattern architecture

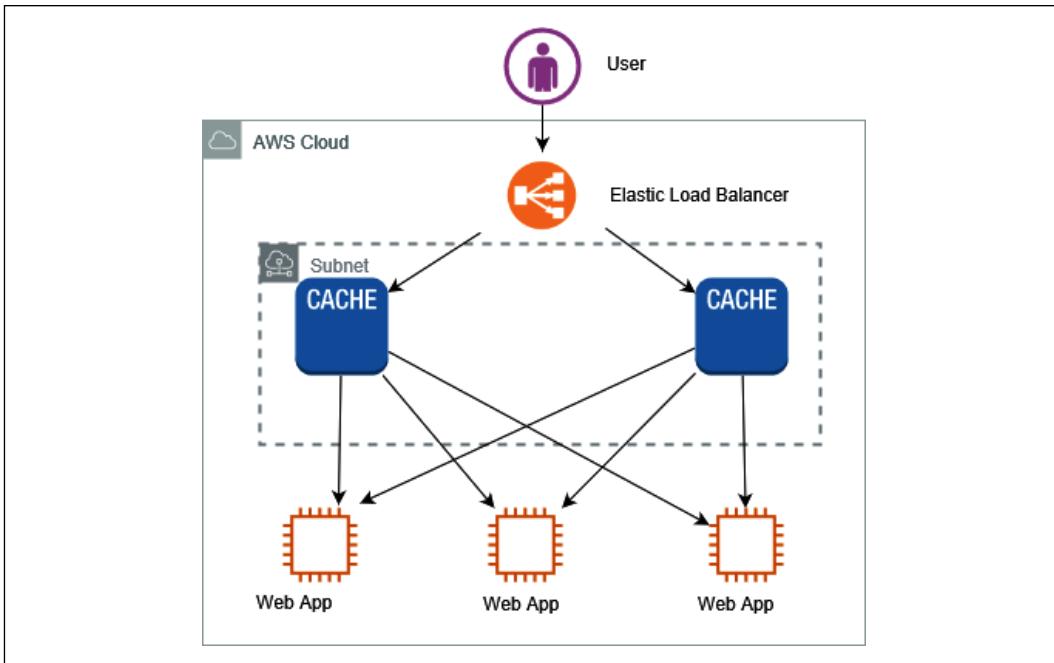


Figure 6.16: Cache proxy pattern architecture

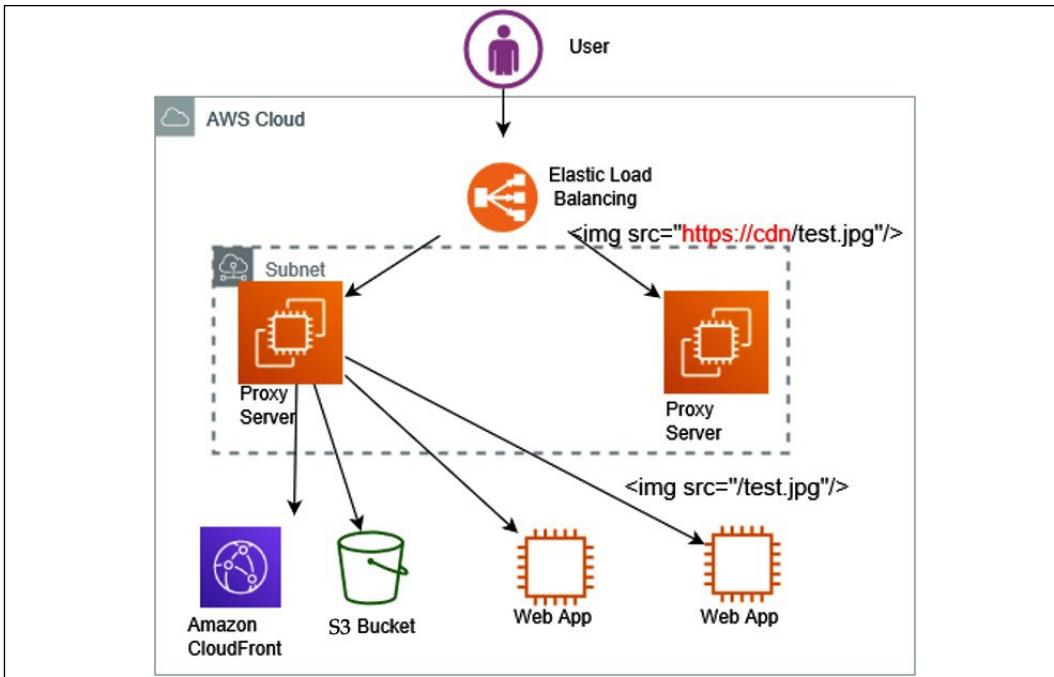


Figure 6.17: Rewrite proxy pattern architecture

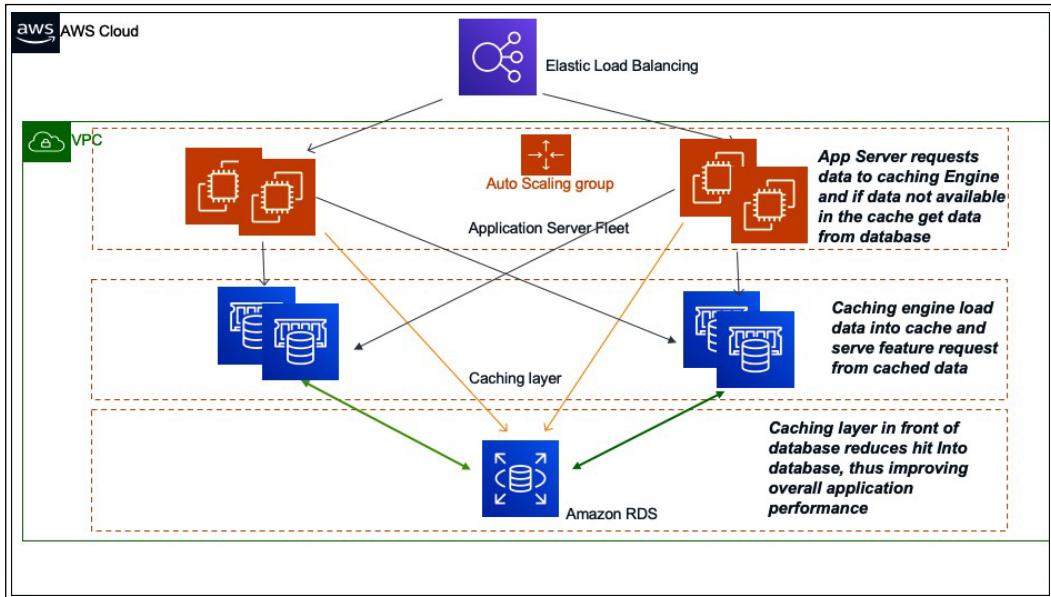


Figure 6.18: Application caching pattern architecture

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

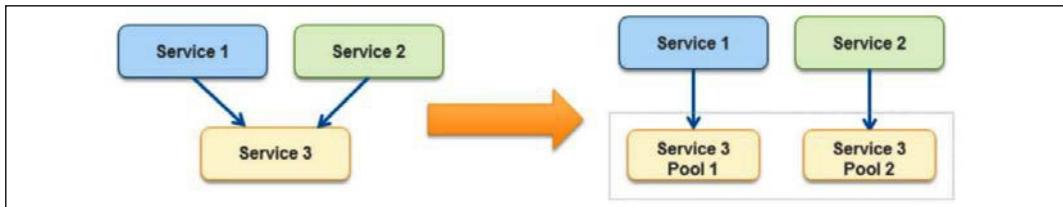


Figure 6.19: Bulkhead pattern

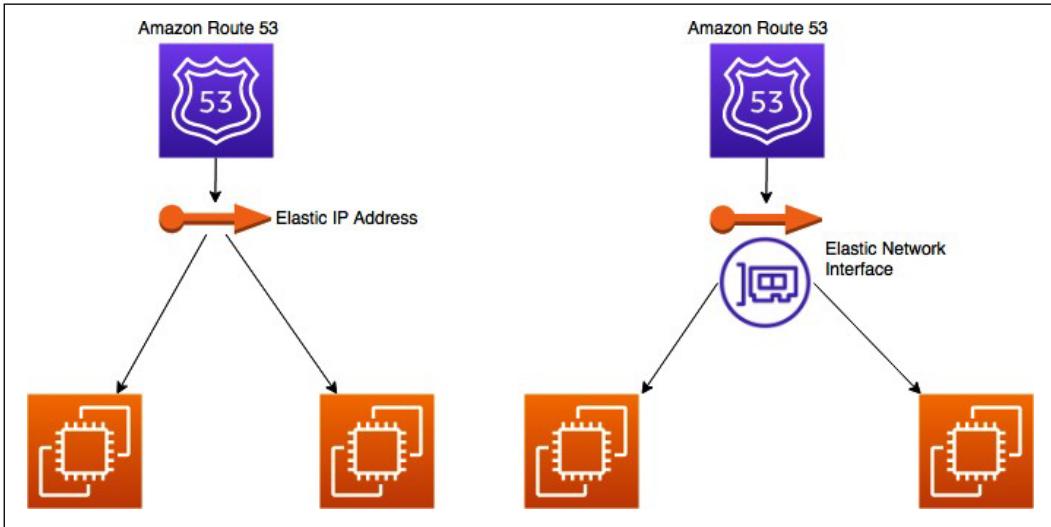


Figure 6.20: Floating IP and interface pattern

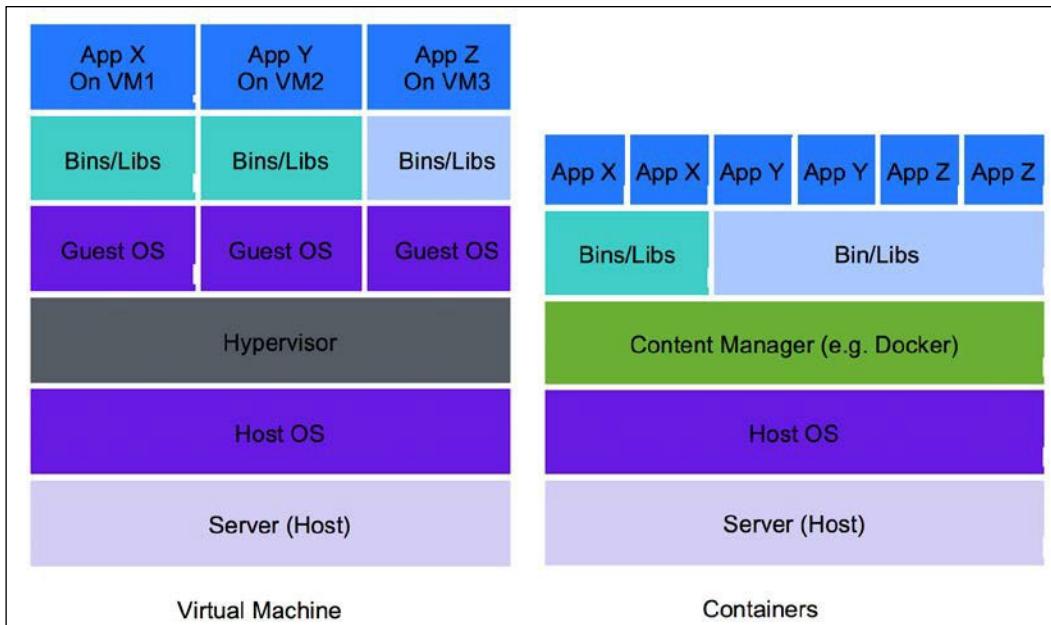


Figure 6.21: Virtual machines and containers for application deployment

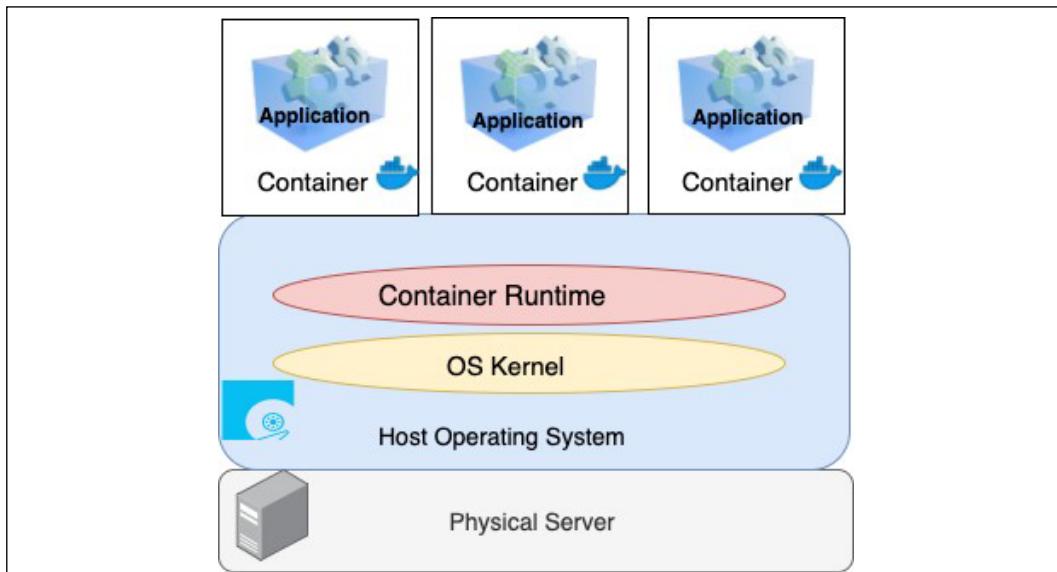


Figure 6.22: Container layer in application infrastructure

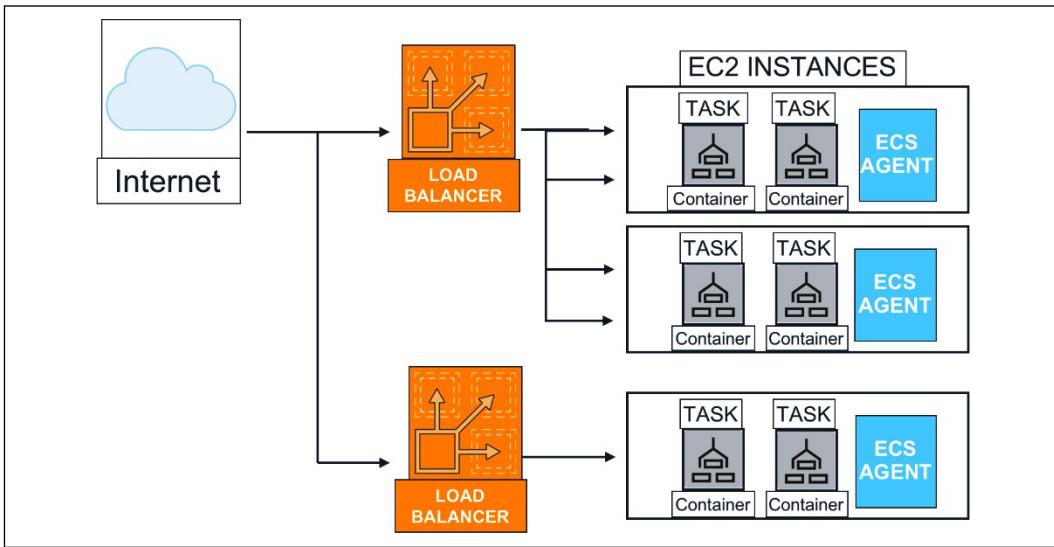


Figure 6.23: Container deployment architecture

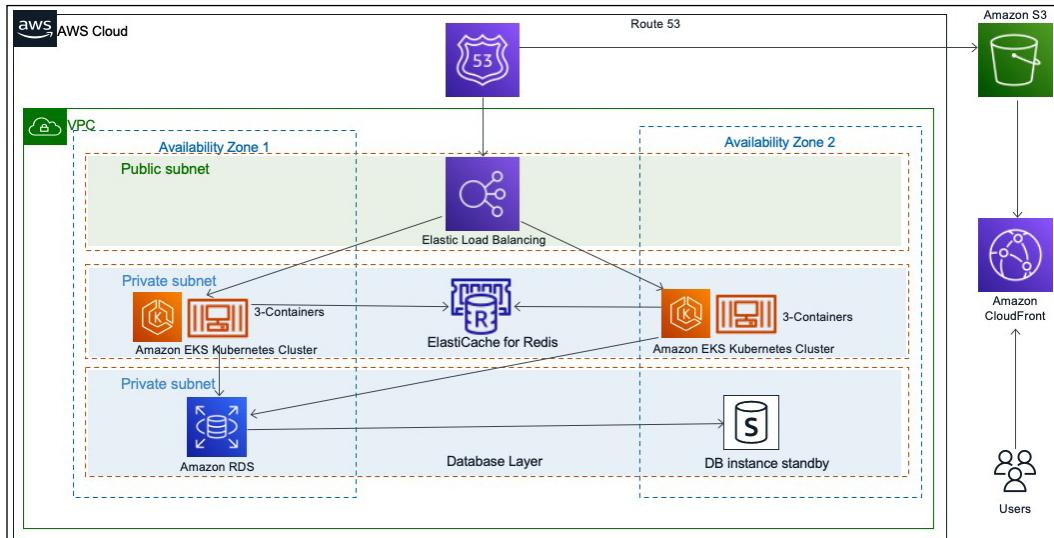


Figure 6.24: Deploying a stateful application on a container

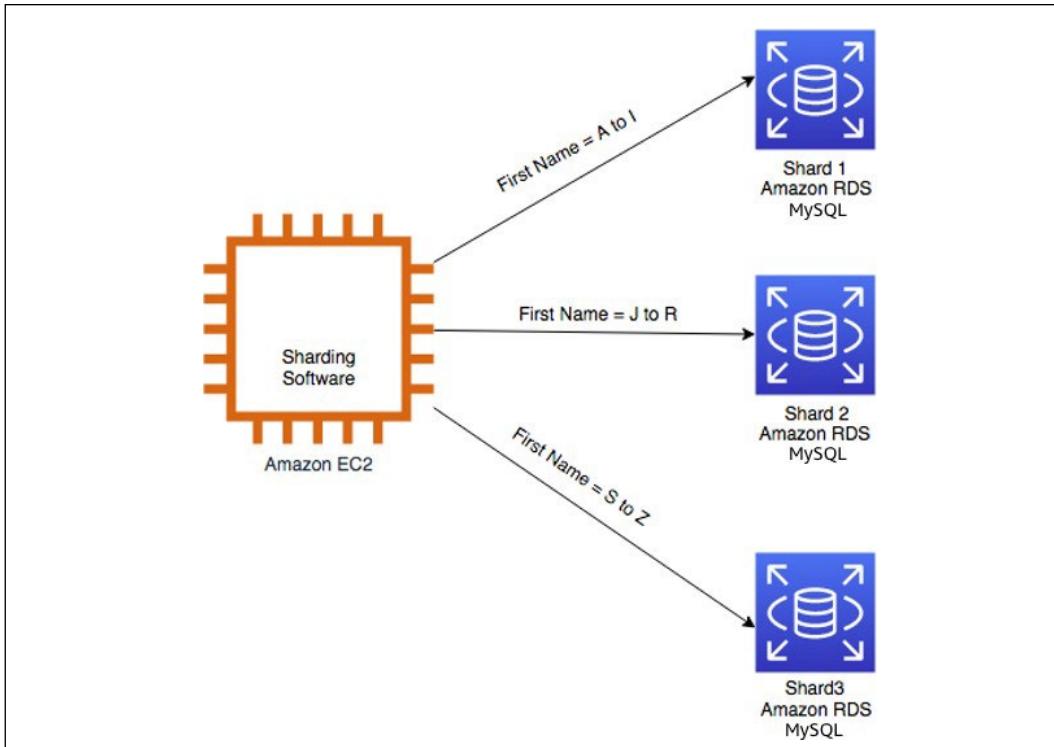


Figure 6.25: Relational database sharding

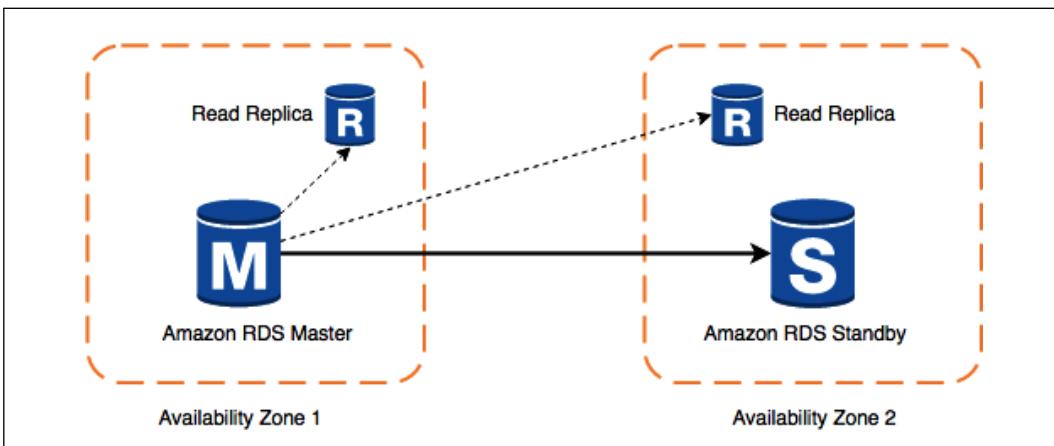


Figure 6.26: High-availability database pattern

Chapter 7

Performance Considerations

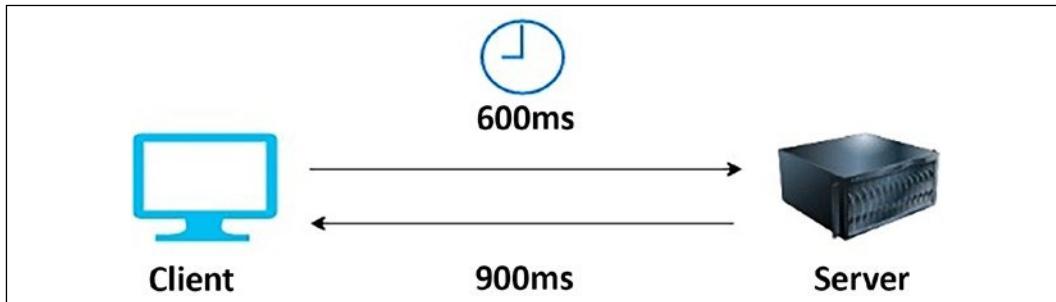


Figure 7.1: Request-response latency in a client-server model

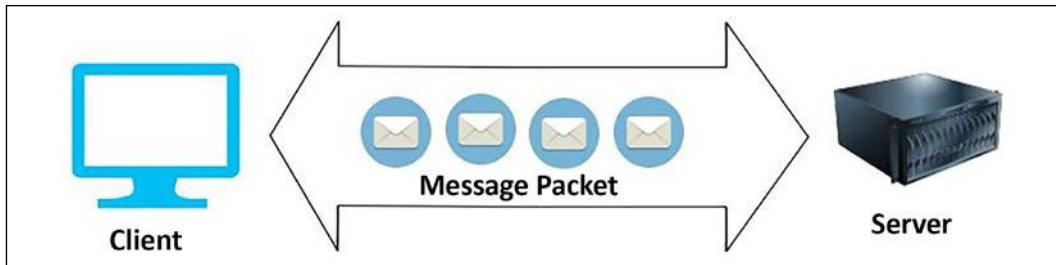


Figure 7.2: Throughput in a network

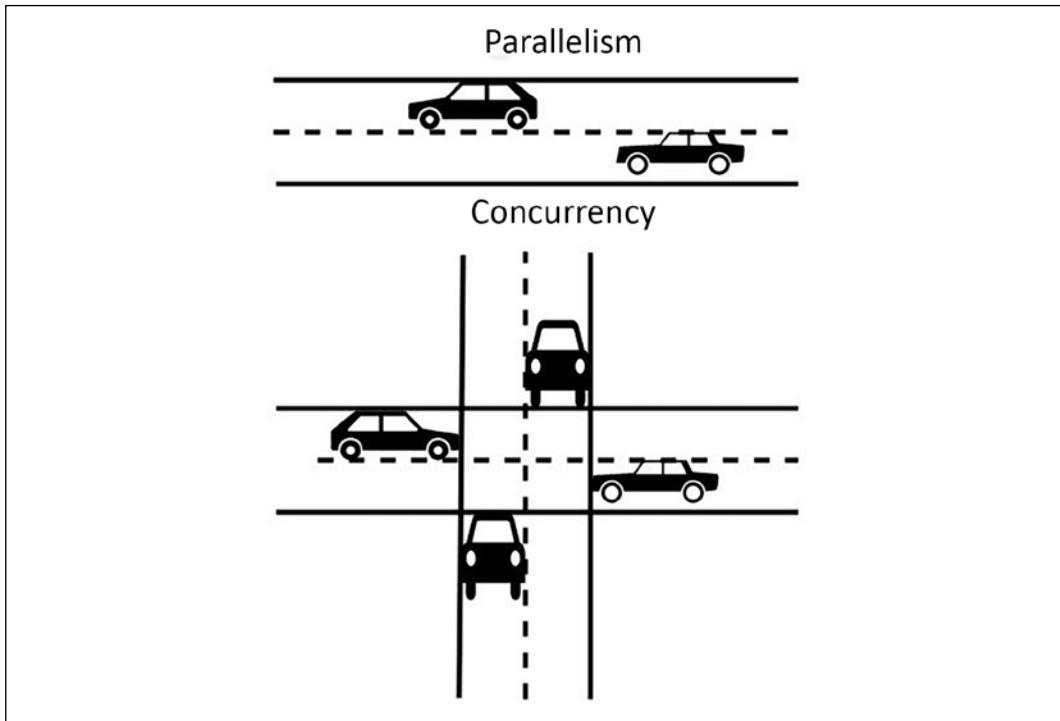


Figure 7.3: Concurrency versus parallelism

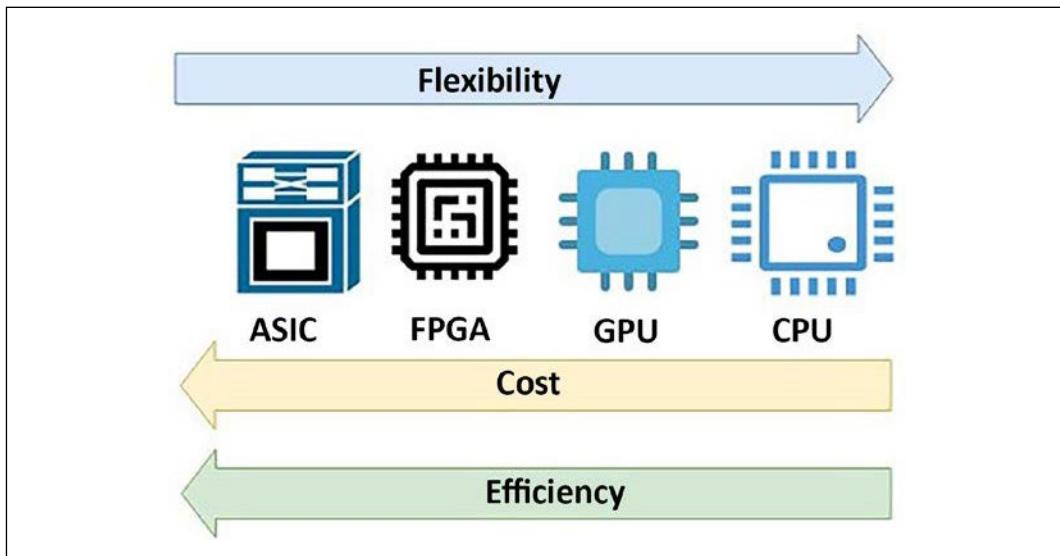


Figure 7.4: Comparison between CPUs, GPUs, FPGAs, and ASICs

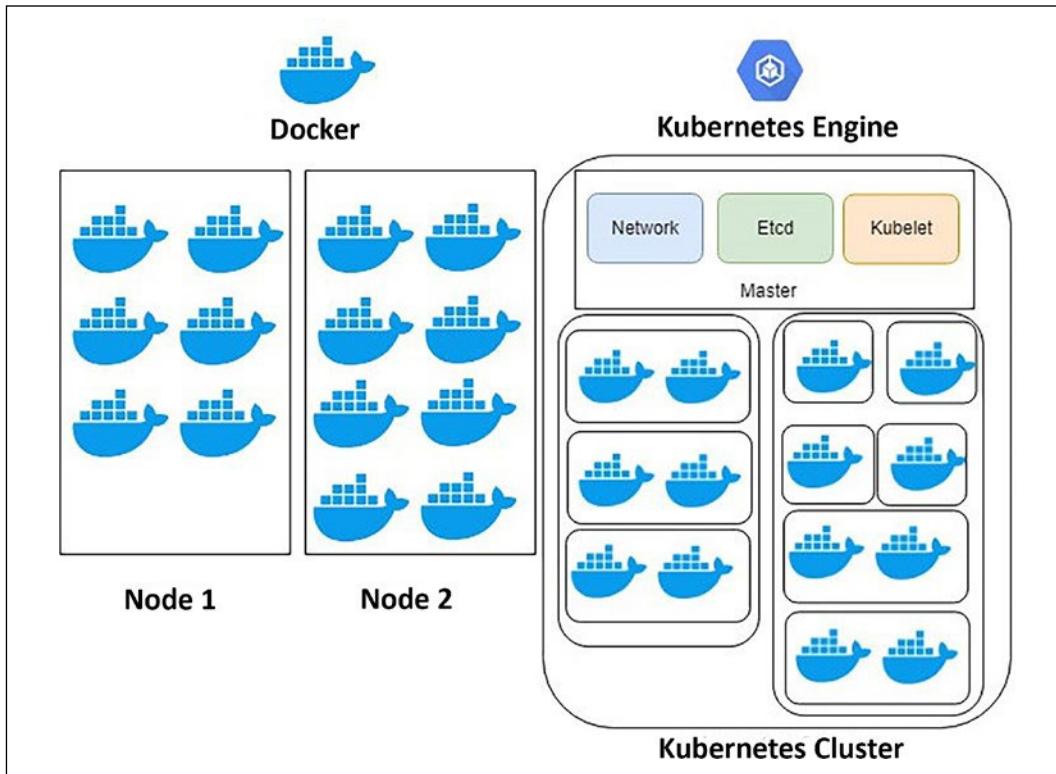


Figure 7.5: Docker and Kubernetes

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

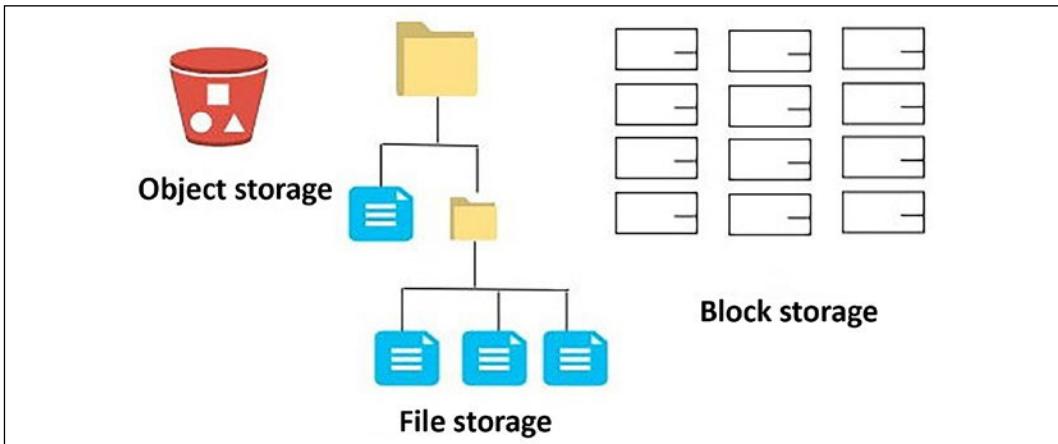


Figure 7.6: Data storage systems

Edit details - ASG-SA

X

Launch Instances Using Launch Template Launch Configuration

Launch Configuration webserverCopy

Desired Capacity 3

Min 2

Max 5

Availability Zone(s) eu-west-1a eu-west-1b

Subnet(s) subnet-0be5c2b238624205e(10.0.0.0/24) | PublicSubnetA-saurabh | eu-west-1a subnet-0a499ab52ff71bacd(10.0.1.0/24) | PublicSubnetB-saurabh | eu-west-1b

Classic Load Balancers testec2-SAelb-1OE7V07XZAQVB

Target Groups

Health Check Type EC2

Health Check Grace Period 300

Instance Protection

Figure 7.7: Auto-scaling configuration

Chapter 8

Security Considerations

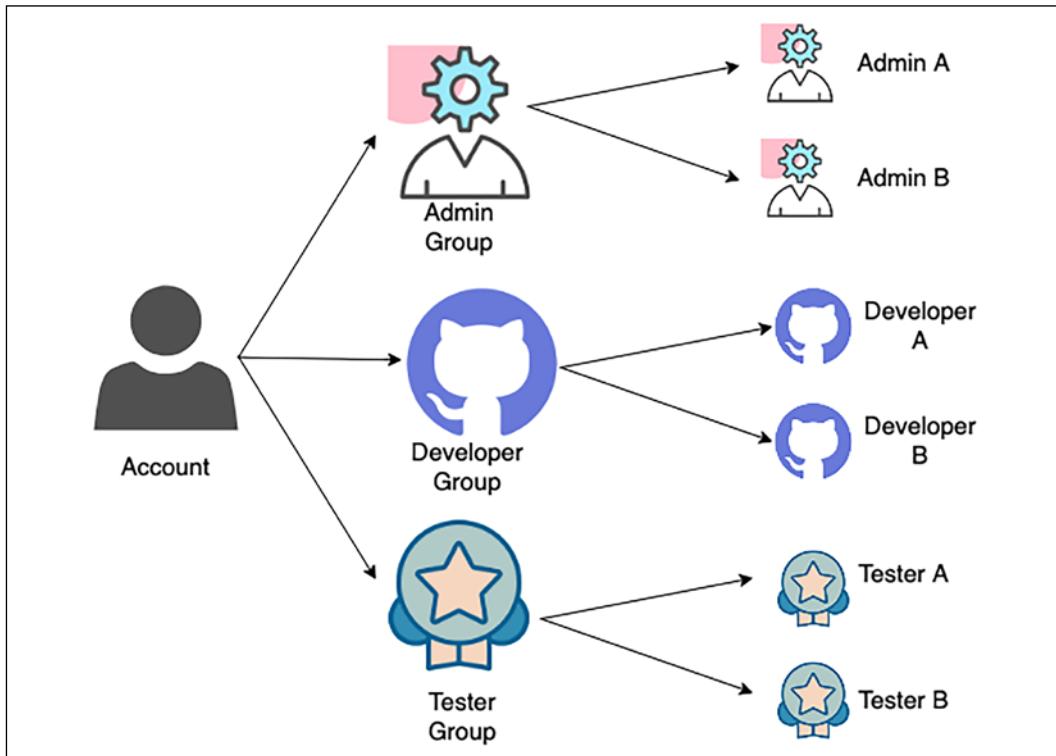


Figure 8.1: User group organization

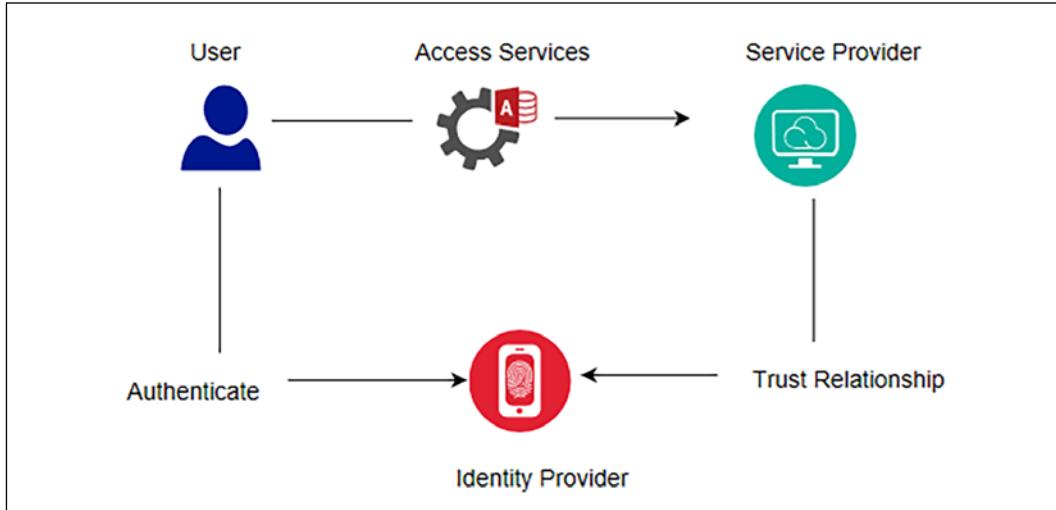


Figure 8.2: FIM authentication flow

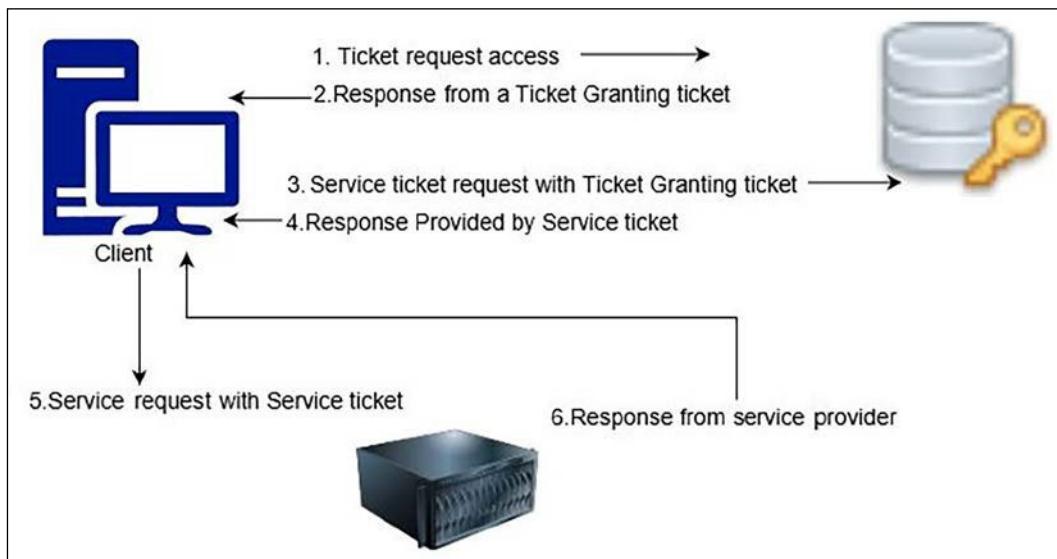


Figure 8.3: Kerberos authentication

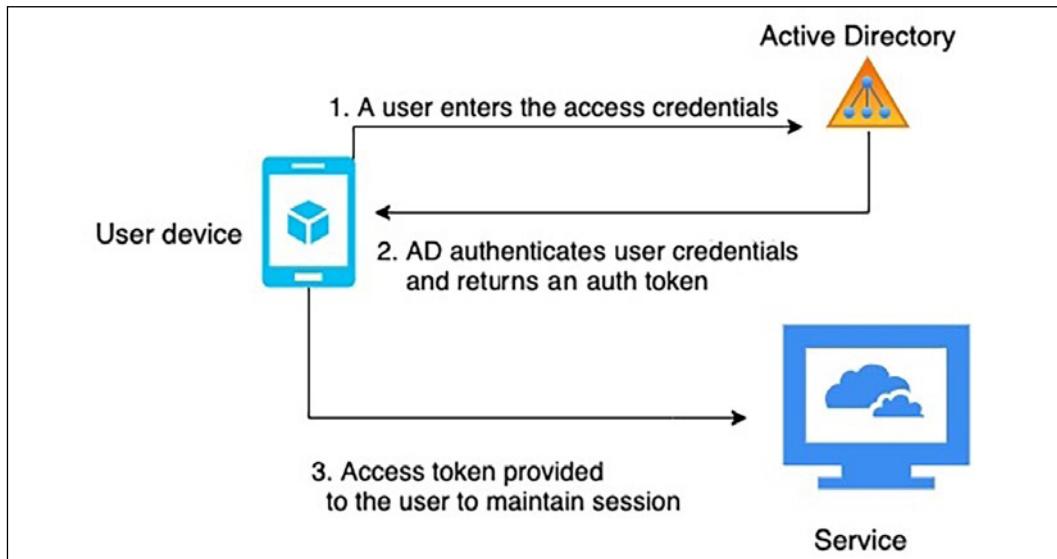


Figure 8.4: AD authentication flow

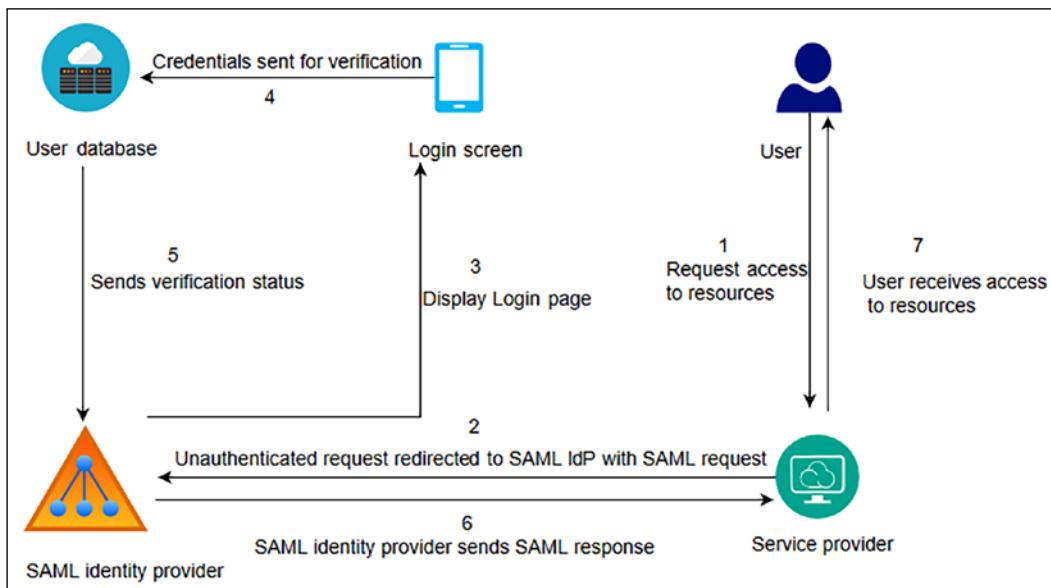


Figure 8.5: User authentication using SAML

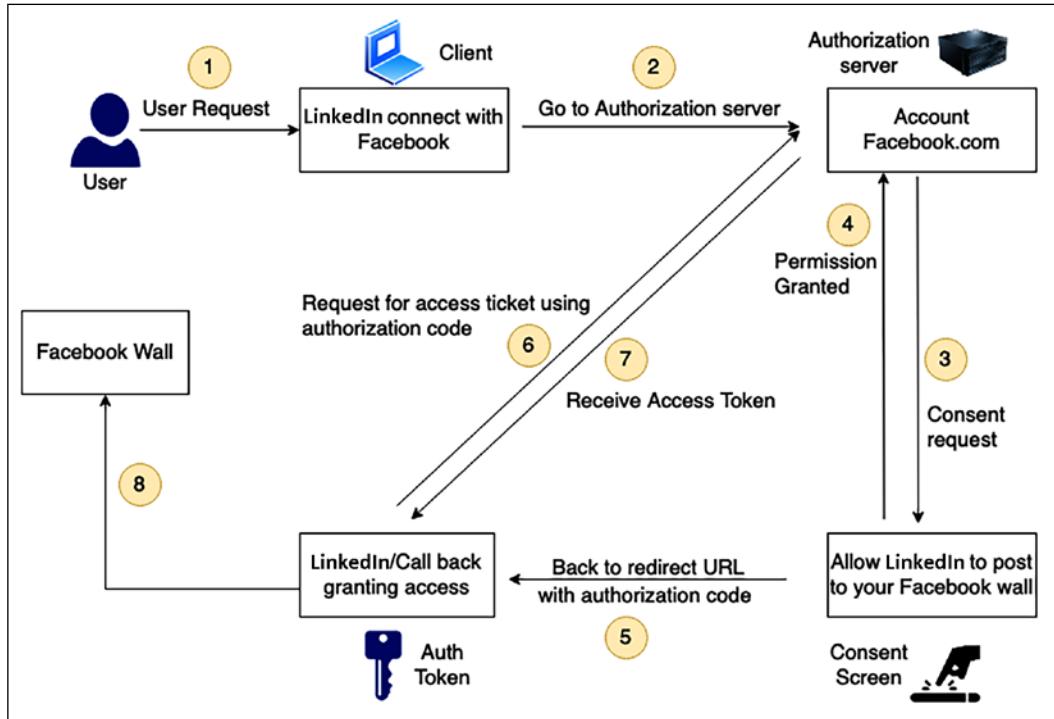


Figure 8.6: User access delegation with OAuth 2.0

Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIiOixOTE5MjAyMCIsIm5hbWUiOiJTb2x1dGlvbiBBcmNoaXR1Y3QgSGFuZGJvb2siLCJpYXQiOjIxMjExMDExMX0.kjV743Dko6XciP3Kp3YrrRoVvSJvBXzK3JkHWqGKhIE
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE	
{ "alg": "HS256", "typ": "JWT" }	
PAYLOAD: DATA	
{ "sub": "19192020", "name": "Solution Architect Handbook", "iat": 212110101 }	
VERIFY SIGNATURE	
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input type="checkbox"/> secret base64 encoded	

Figure 8.7: Sample JWT

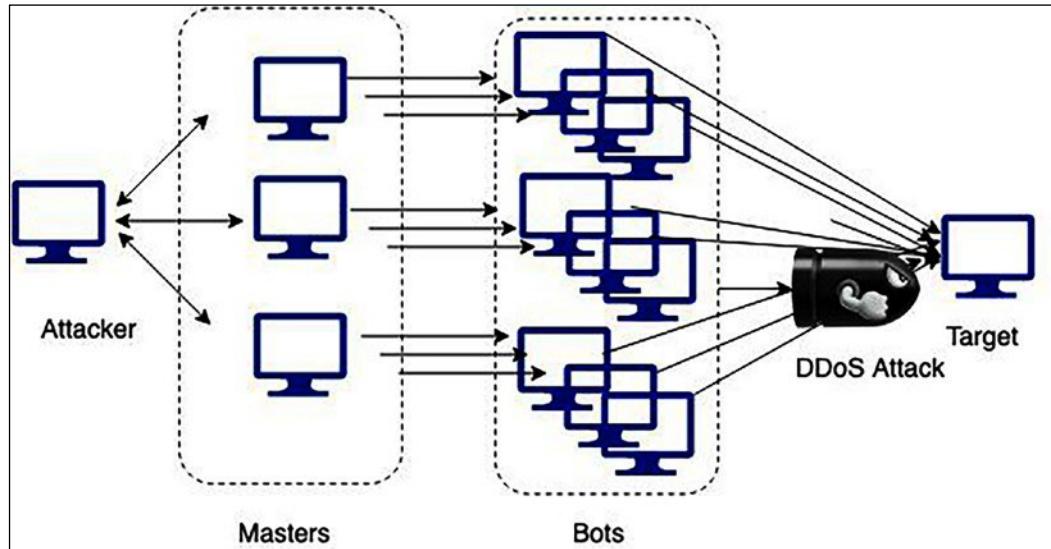


Figure 8.8: DDoS attack

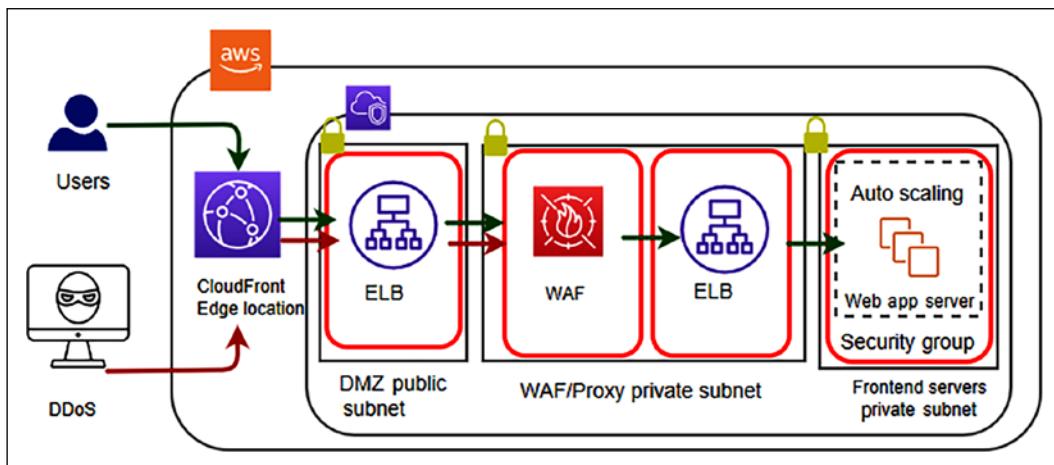


Figure 8.9: DDoS WAF sandwich mitigation strategy

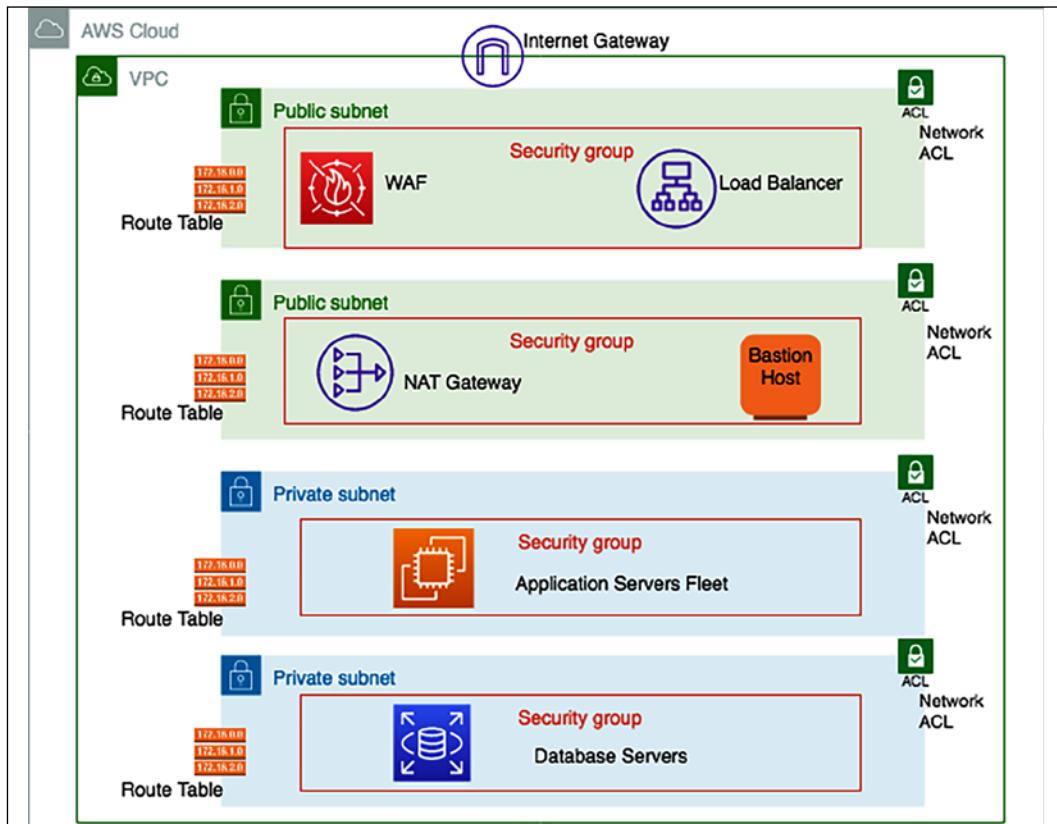


Figure 8.10: Network configuration for infrastructure security

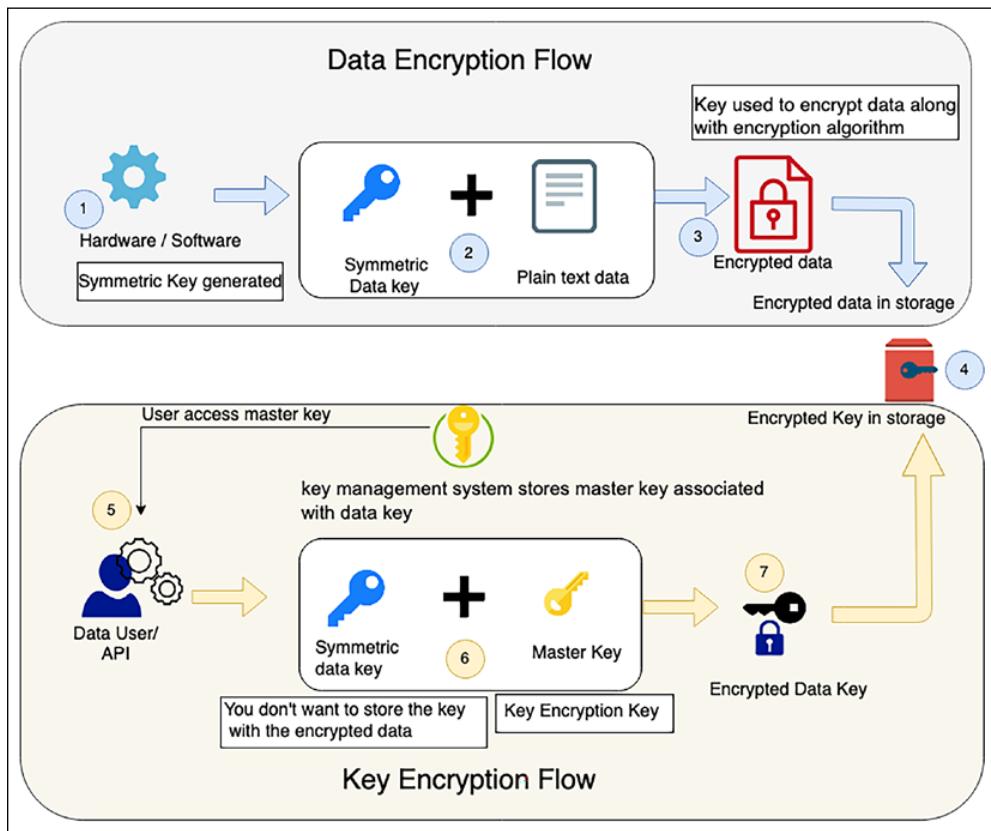


Figure 8.11: Envelope encryption

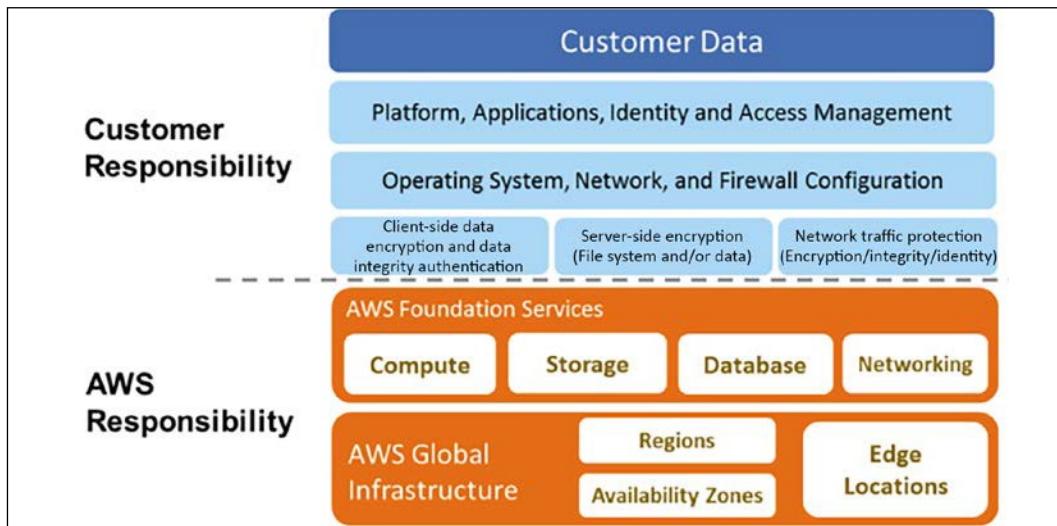


Figure 8.12: AWS cloud shared security responsibility model

Chapter 9

Architectural Reliability Considerations

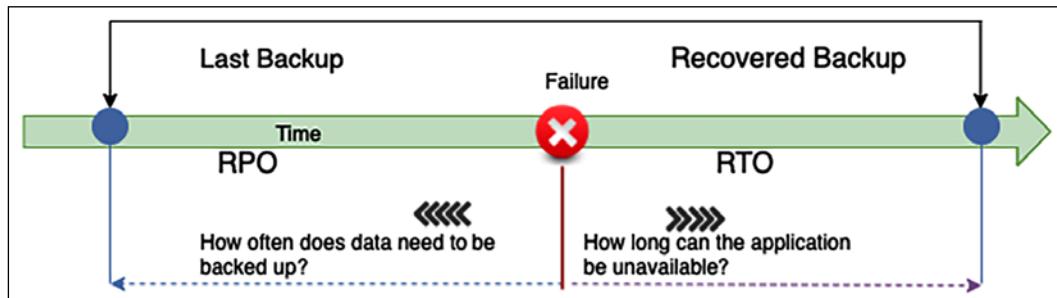


Figure 9.1: RTO and RPO

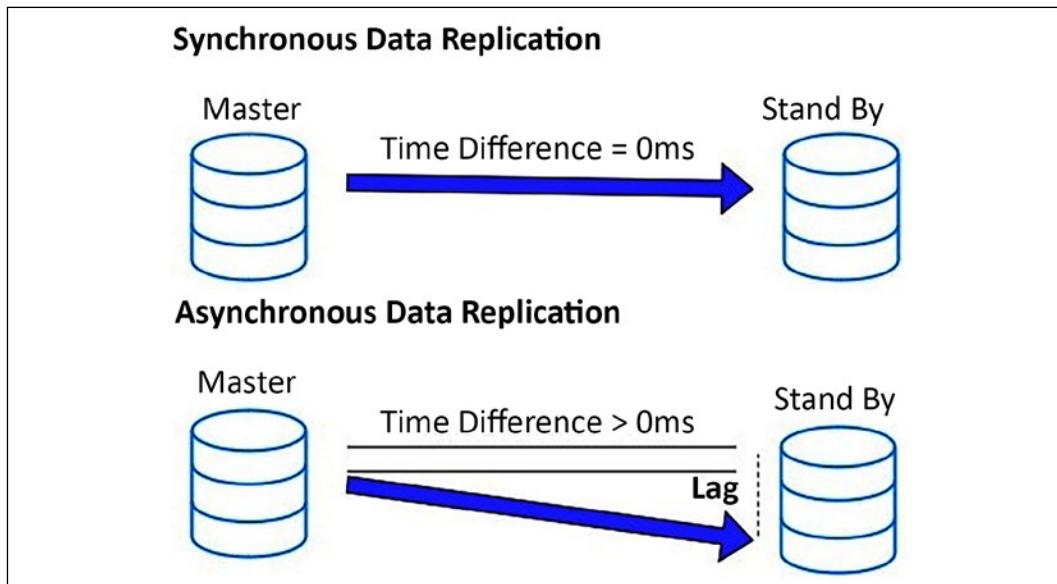


Figure 9.2: Synchronous and asynchronous data replication

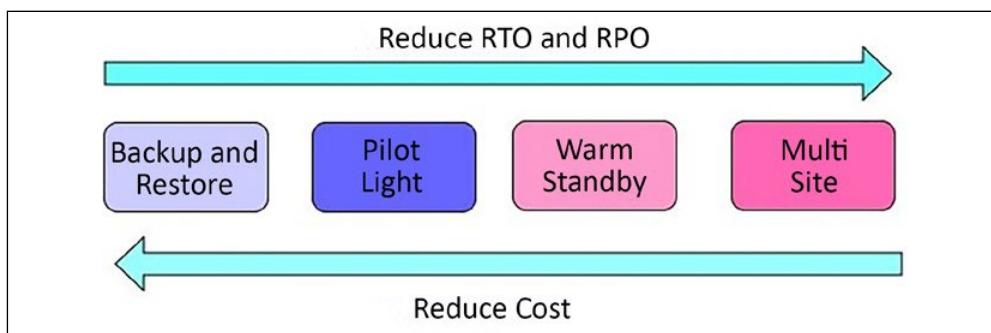


Figure 9.3: The spectrum of DR options

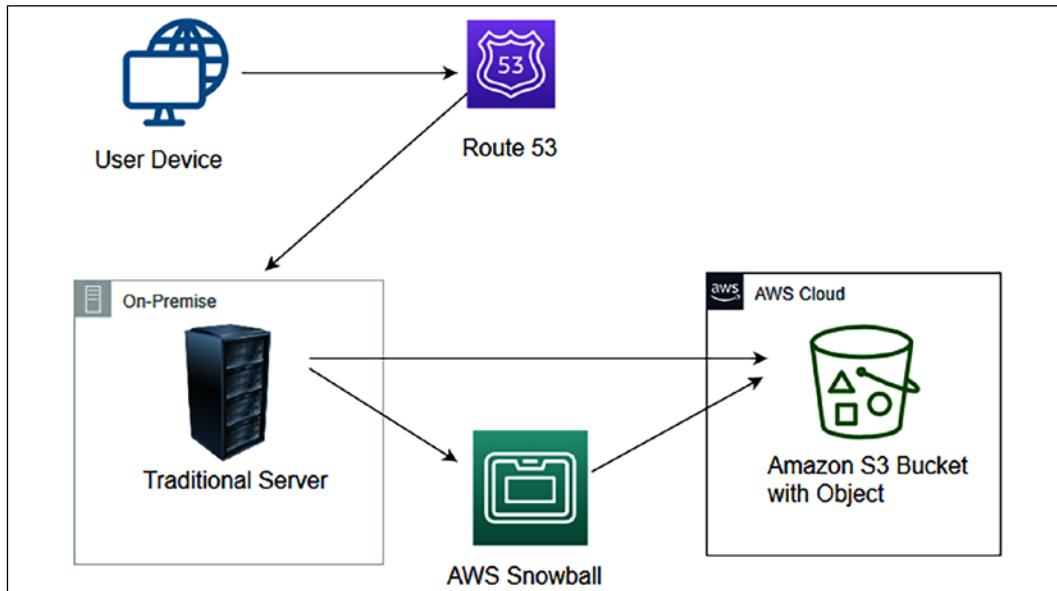


Figure 9.4: Data backup to Amazon S3 from on-premises infrastructure

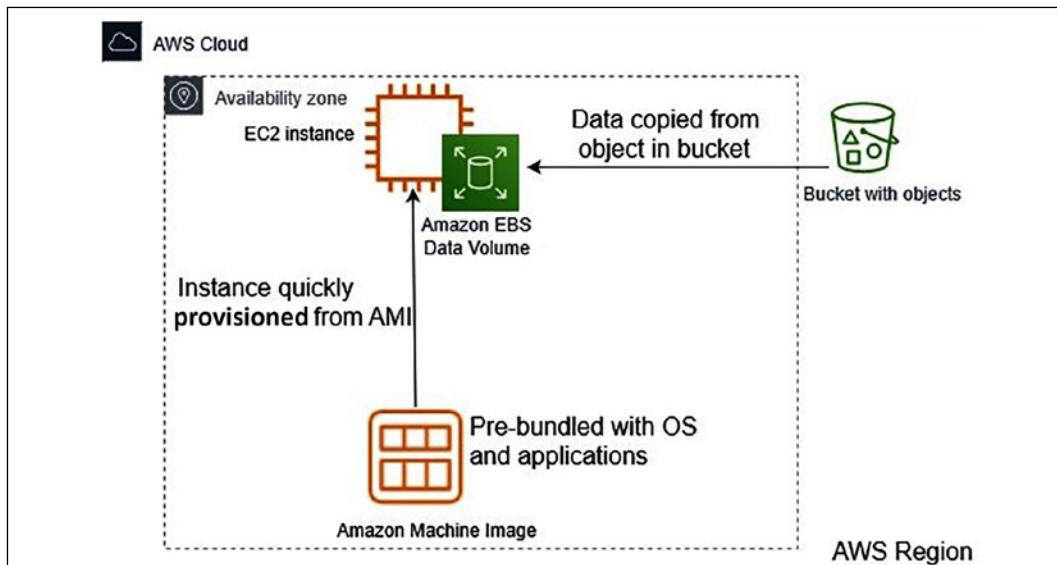


Figure 9.5: Restoring systems from Amazon S3 backups in the cloud

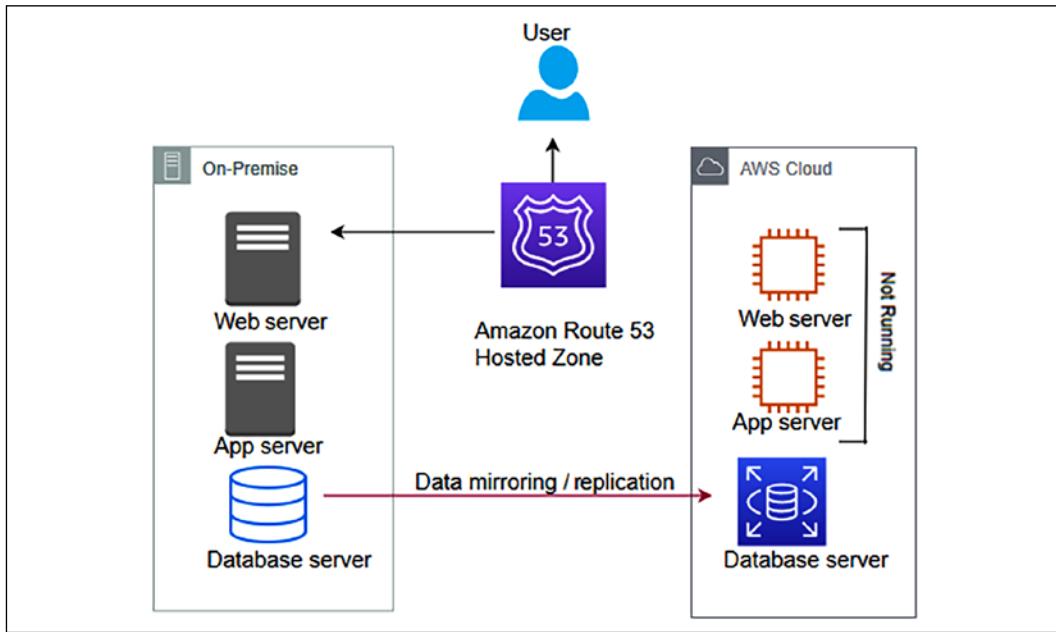


Figure 9.6: The pilot light data replication to DR site scenario

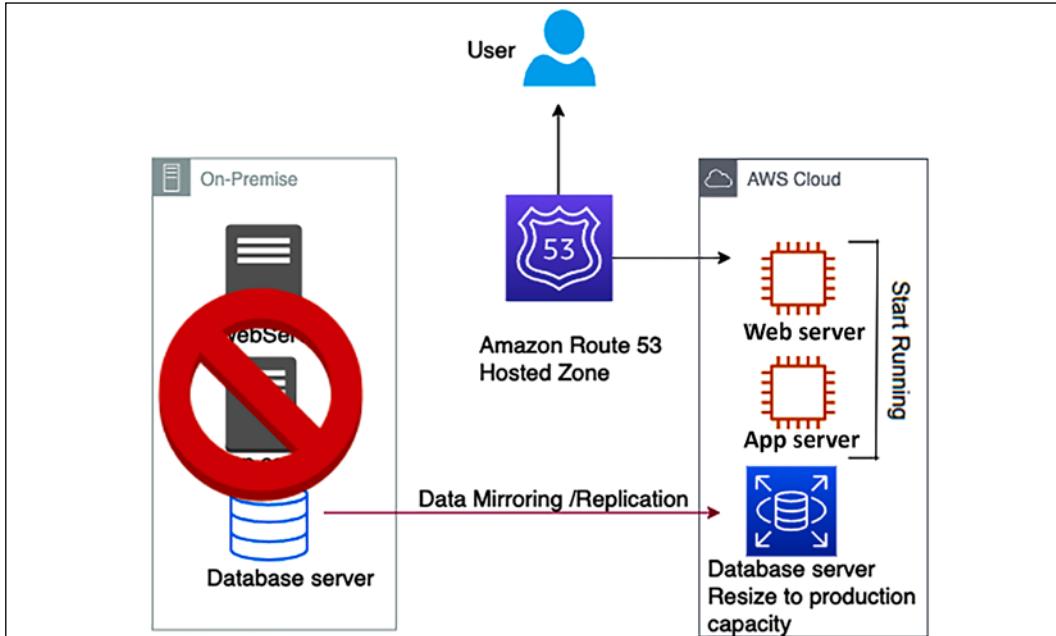


Figure 9.7: Recovery in the pilot light method

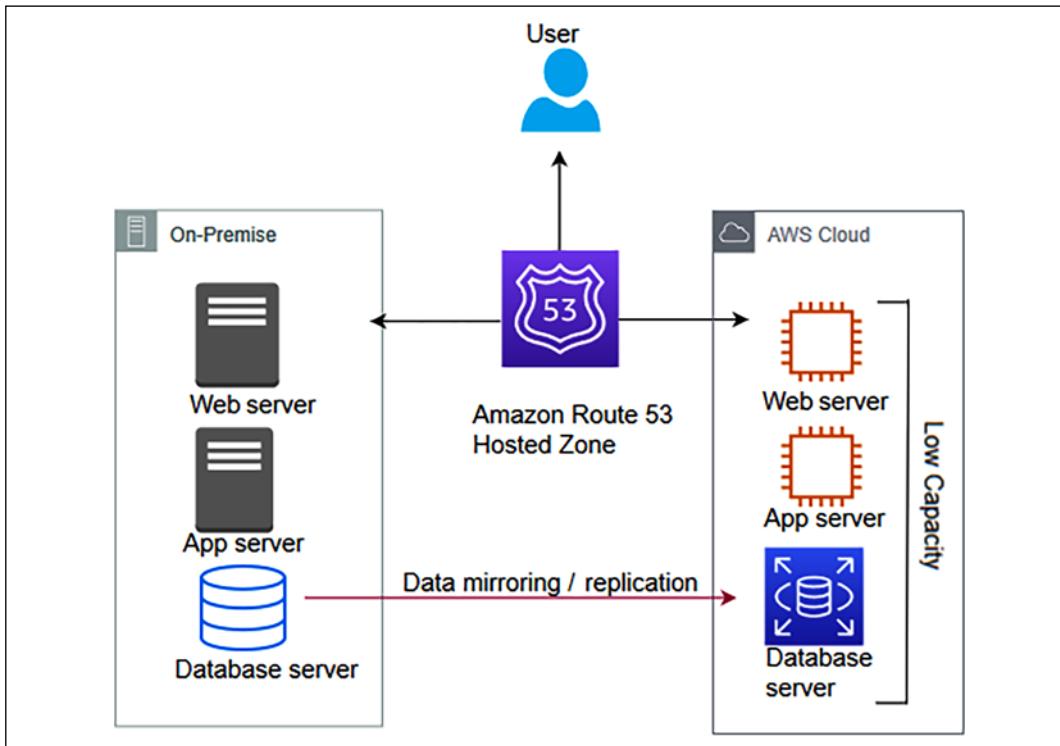


Figure 9.8: Warm standby scenario running an active-active workload with a low capacity

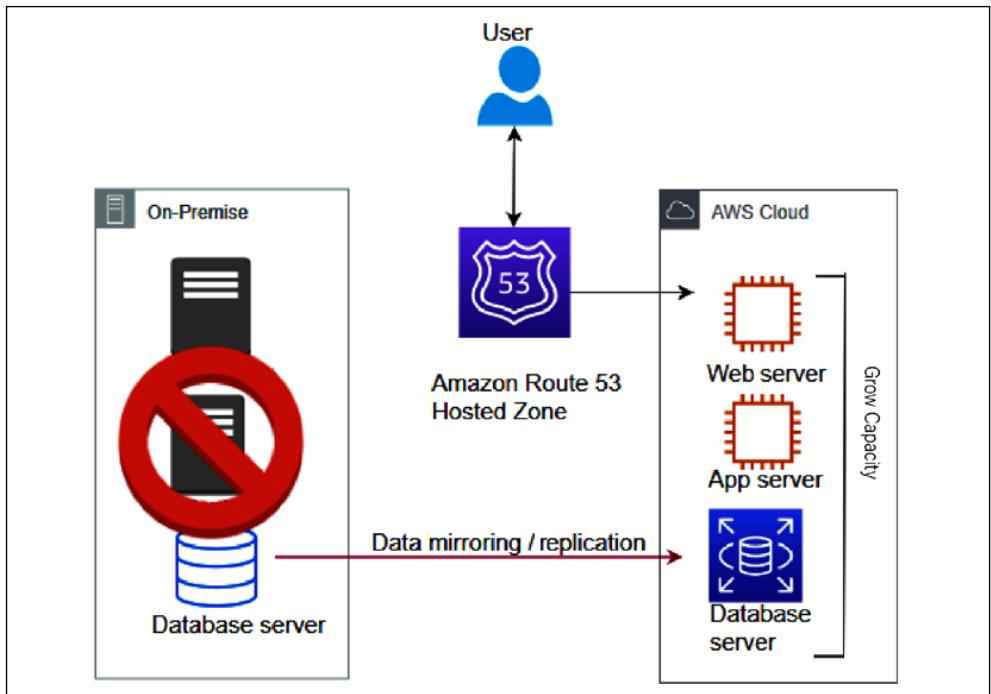


Figure 9.9: Recovery phase in the warm standby scenario

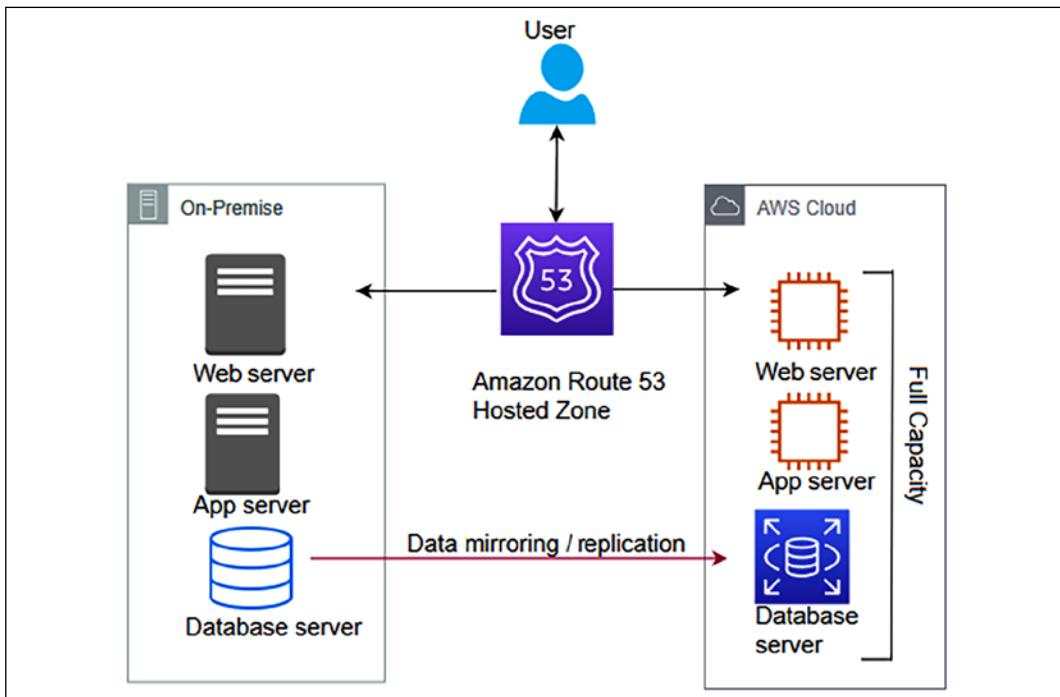


Figure 9.10: Multi-site scenario running an active-active workload with full capacity

Chapter 10

Operational Excellence Considerations

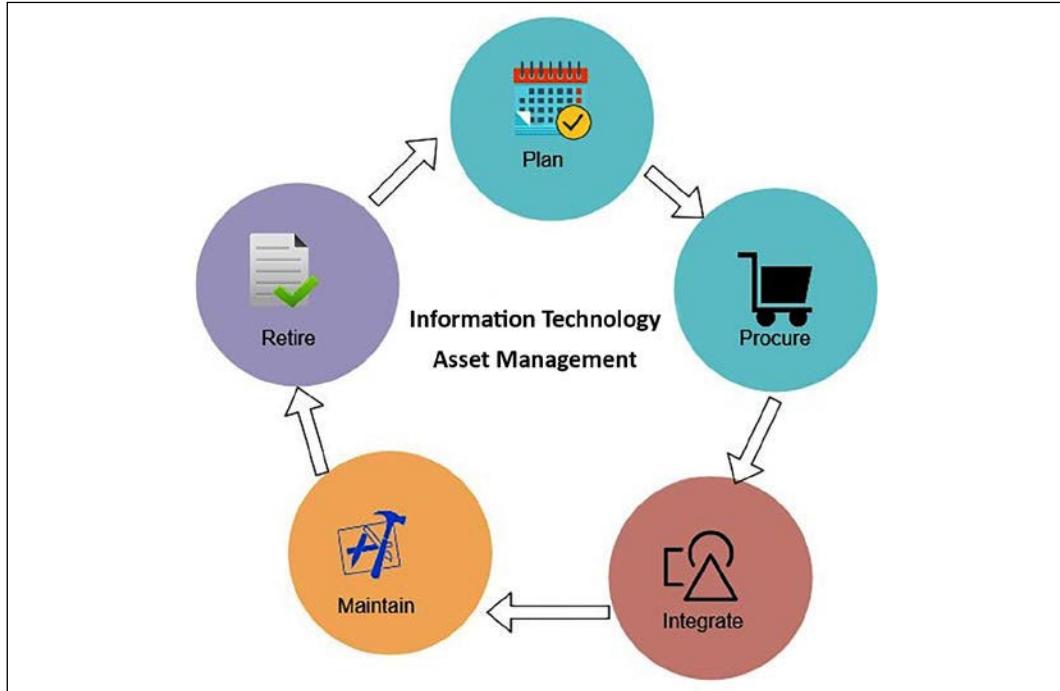


Figure 10.1: ITAM process

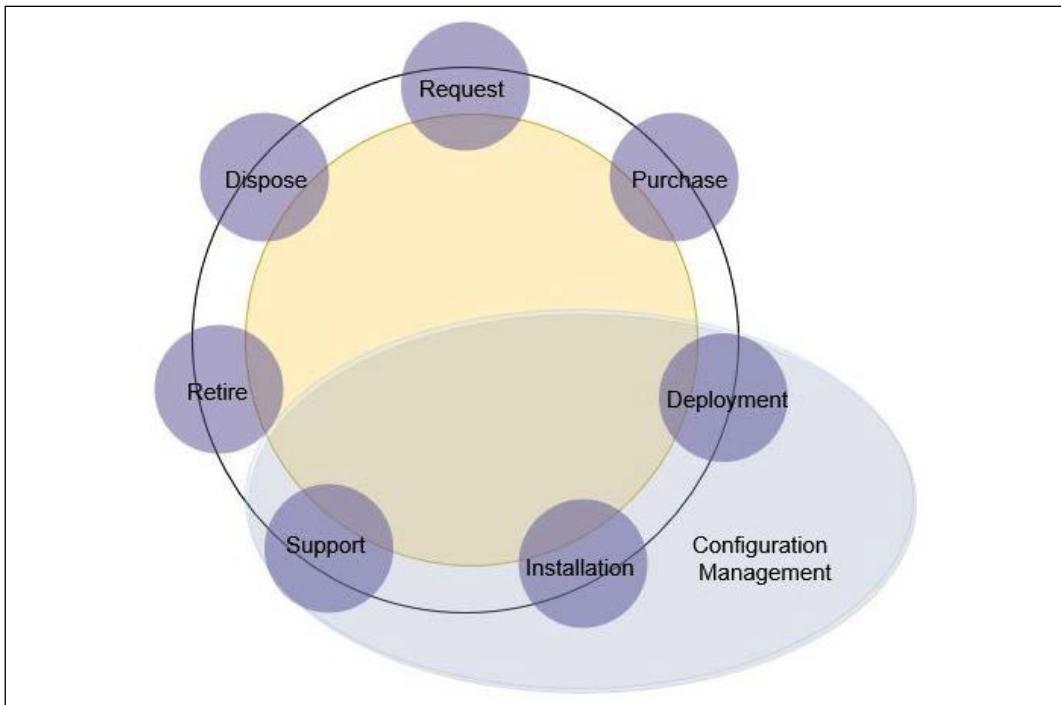


Figure 10.2: IT asset life cycle versus configuration management

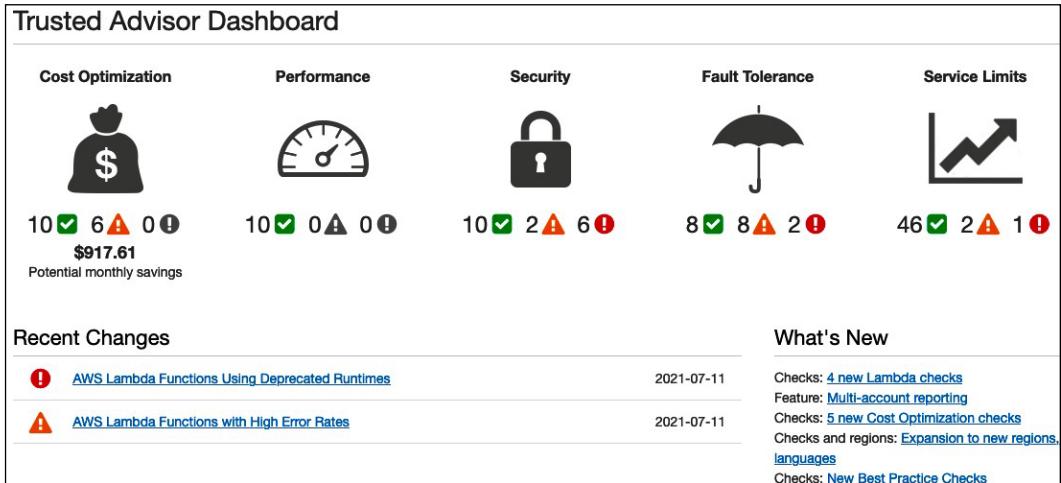


Figure 10.3: AWS Trusted Advisor dashboard



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

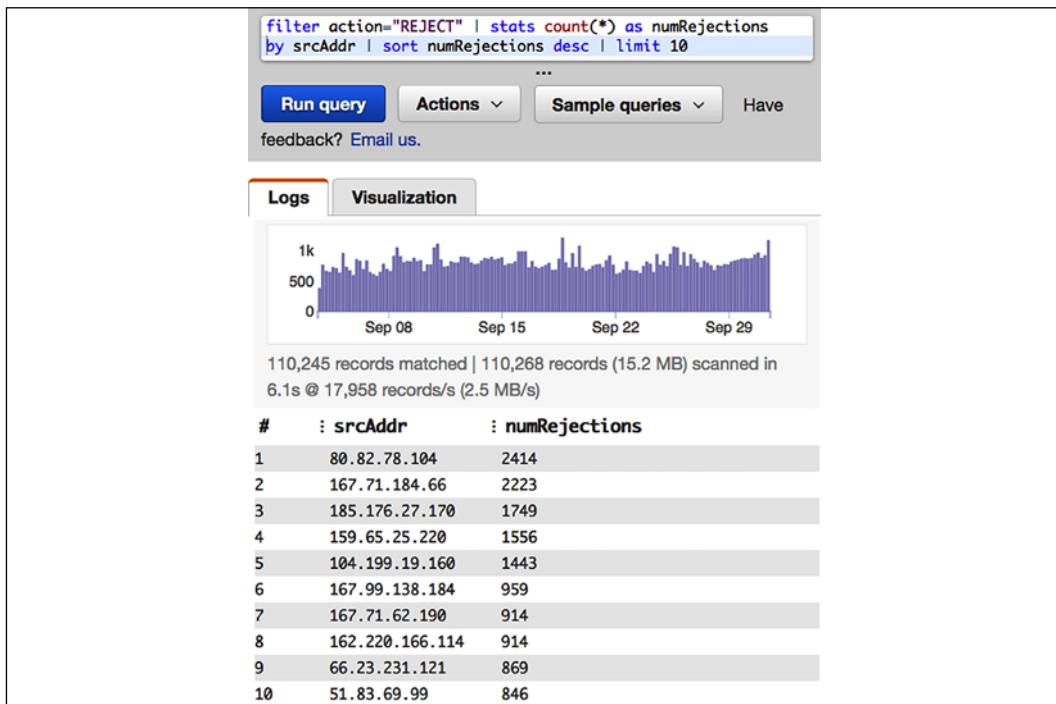


Figure 10.8: Insight from raw network log by running query

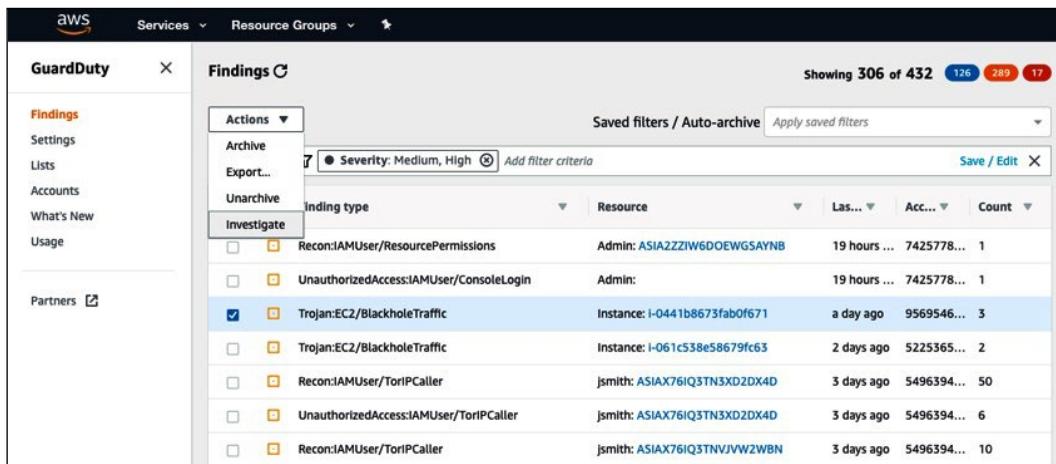


Figure 10.9: Security monitoring using Amazon GuardDuty

CloudWatch						
Favorites		Alarms (47)				
Dashboards		Actions				
Alarms 10 32 5						
In alarm	All alarms	Name	State	Last state update	Conditions	Actions
Billing		billing	⚠️ in alarm	2021-12-12 18:14:51	EstimatedCharges > 2000 for 1 datapoints within 6 hours	Actions enabled Warning
Logs		BillingAlarm	⚠️ in alarm	2021-12-07 09:18:19	EstimatedCharges > 1000 for 1 datapoints within 6 hours	Actions enabled Warning
Metrics		TargetTracking-table/testretail-/index/_User_id-index-alarm:low-835abff1d-4f5c-404e-b0a1-4e0fb1e008364	⚠️ in alarm	2021-03-21 14:05:44	ConsumedWriteCapacityUnits < 150 for 15 datapoints within 15 minutes	Actions enabled
X-Ray traces		TargetTracking-table/testretail/_User_id-index-alarm:low-108135-bc50-4653-8964-4285cc73a0c7	⚠️ in alarm	2021-03-21 14:05:25	ConsumedReadCapacityUnits < 150 for 15 datapoints within 15 minutes	Actions enabled
Events		TargetTracking-table/testretail-/Alarm_low-5c14cf6-26d4-4e9- b02e-271bb5f9c27c	⚠️ in alarm	2021-03-21 14:05:14	ConsumedWriteCapacityUnits < 150 for 15 datapoints within 15 minutes	Actions enabled
Application monitoring		TargetTracking-table/testretail-/Alarm_low-76801a2-3490-4f28-ad1c-dfb0dc6c50a	⚠️ in alarm	2021-03-21 14:05:10	ConsumedReadCapacityUnits < 150 for 15 datapoints within 15 minutes	Actions enabled
Insights		TargetTracking-table/testretail-/ProvisionedCapacityHigh-870af9ca-1095-4f09-870-00488fc75665	🟡 OK	2021-03-21 14:00:42	ProvisionedWriteCapacityUnits > 5 for 3 datapoints within 15 minutes	Actions enabled
Settings		TargetTracking-table/testretail/_User_id-index-ProvisionedCapacityLow-3f115cf7-e70a-4e5d-e245-e448d0bf87ef	🟡 OK	2021-03-21 14:00:34	ProvisionedWriteCapacityUnits < 5 for 3 datapoints within 15 minutes	Actions enabled
Getting Started		TargetTracking-table/testretail/_User_id-index-ProvisionedCapacityHigh-6504965b-2358-48d4-	🟡 OK	2021-03-21 14:00:33	ProvisionedWriteCapacityUnits > 5 for 3 datapoints within 15 minutes	Actions enabled

Figure 10.10: Alarm dashboard

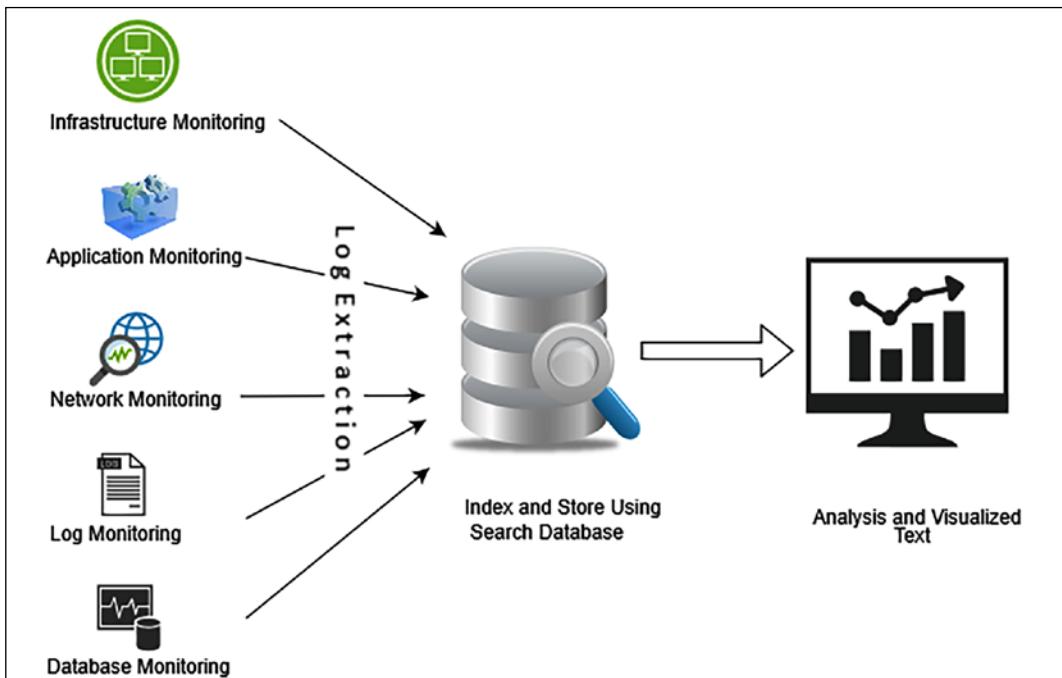


Figure 10.11: Big data approach for ITOA



Figure 10.12: Data audit report summary from Amazon Macie

Chapter 11

Cost Considerations

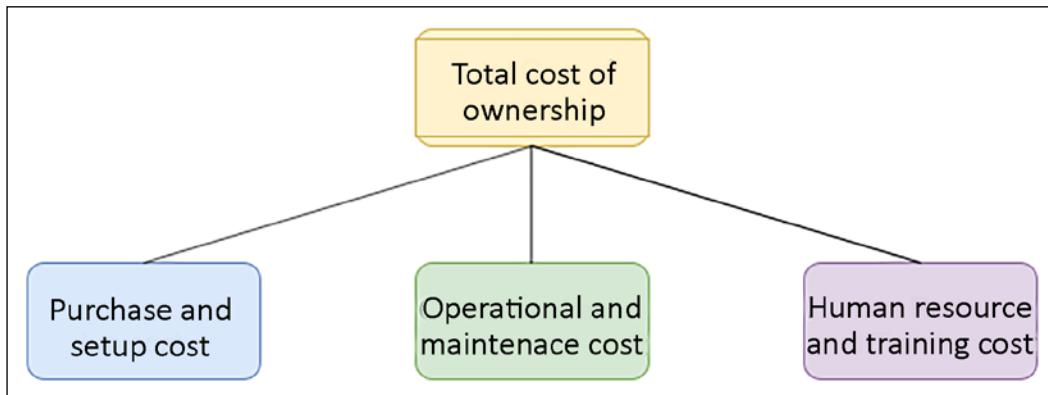


Figure 11.1: TCO for software

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 forecast

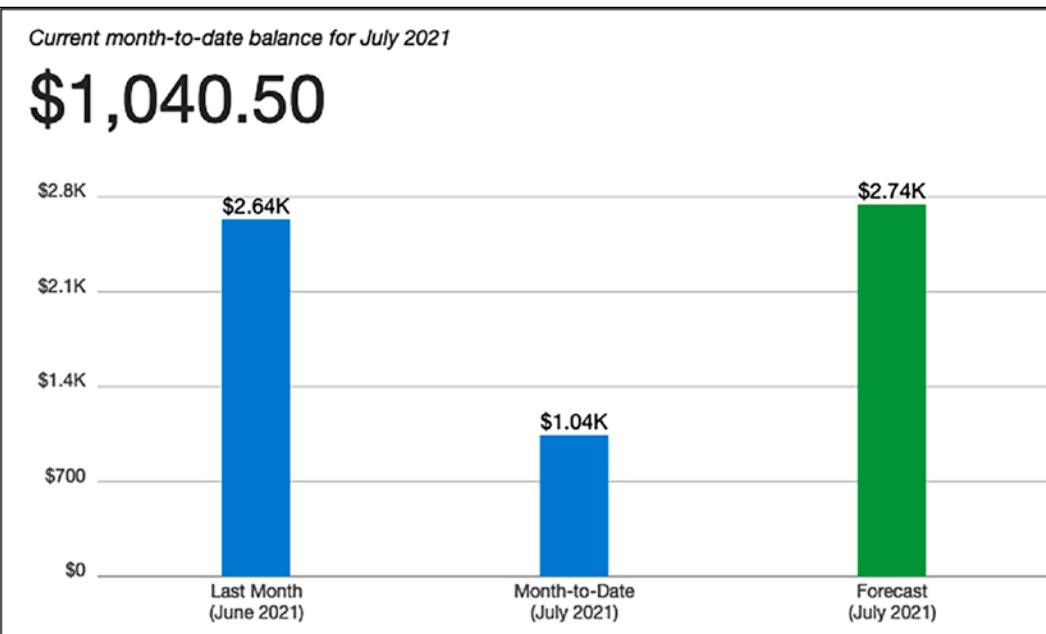


Figure 11.2: Billing and forecast report

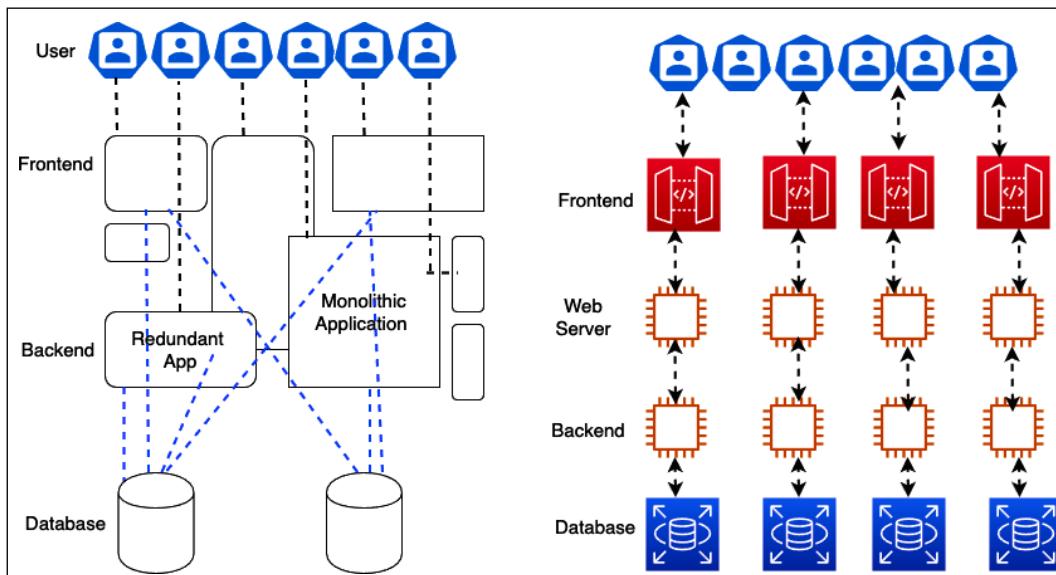


Figure 11.3: Architectural standardization

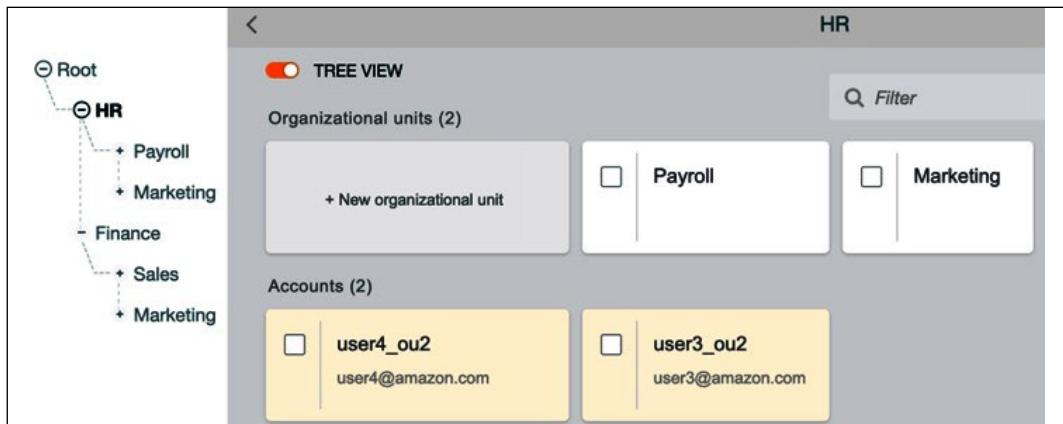


Figure 11.4: Enterprise account structure for organization units (OUs)

Key (128 characters maximum)	Value (256 characters maximum)
Type	AppServer
Environment	Dev
Department	Marketing
Business Unit	Finance
Add another tag (Up to 50 tags maximum)	

Figure 11.5: Resource tagging for cost visibility

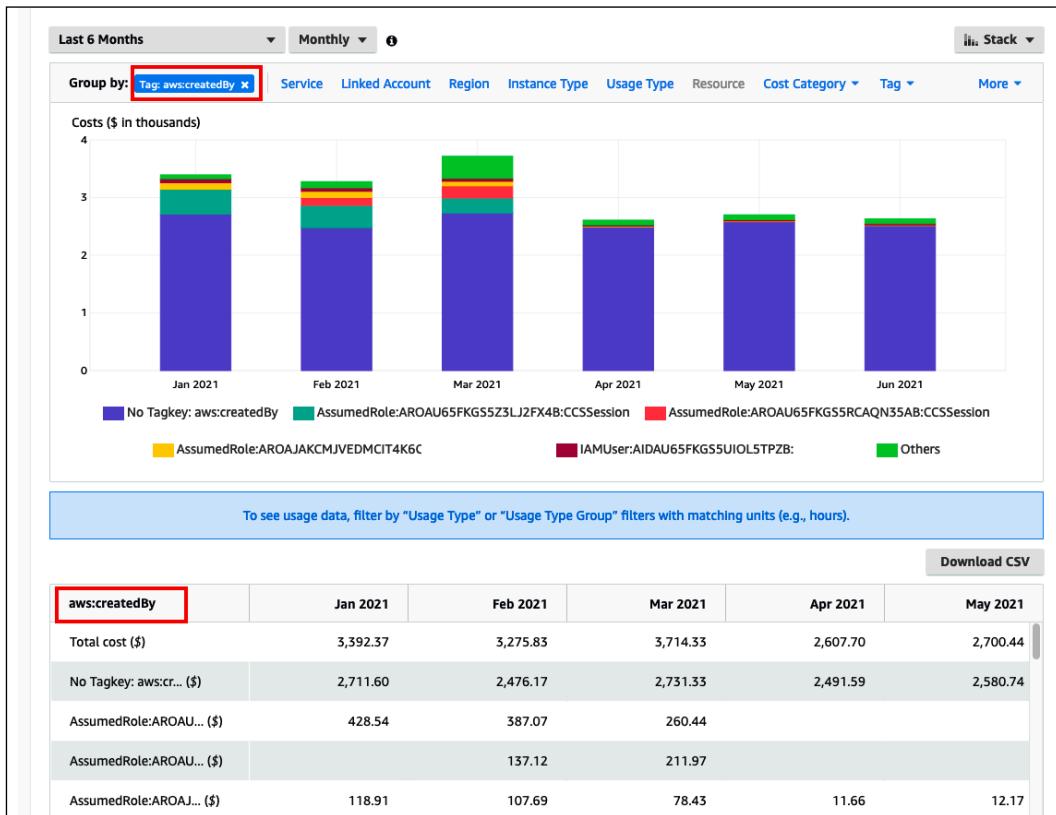


Figure 11.6: Resources amount expenditures dashboard for a cost tag

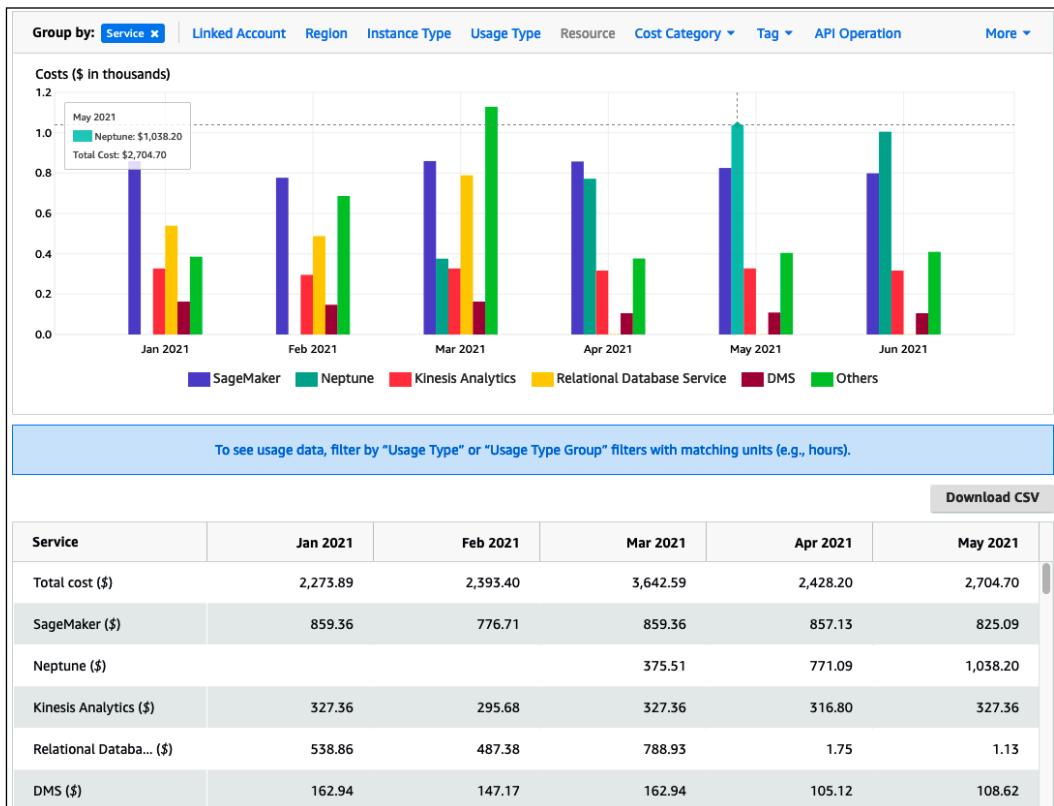


Figure 11.7: Resource cost and usage report

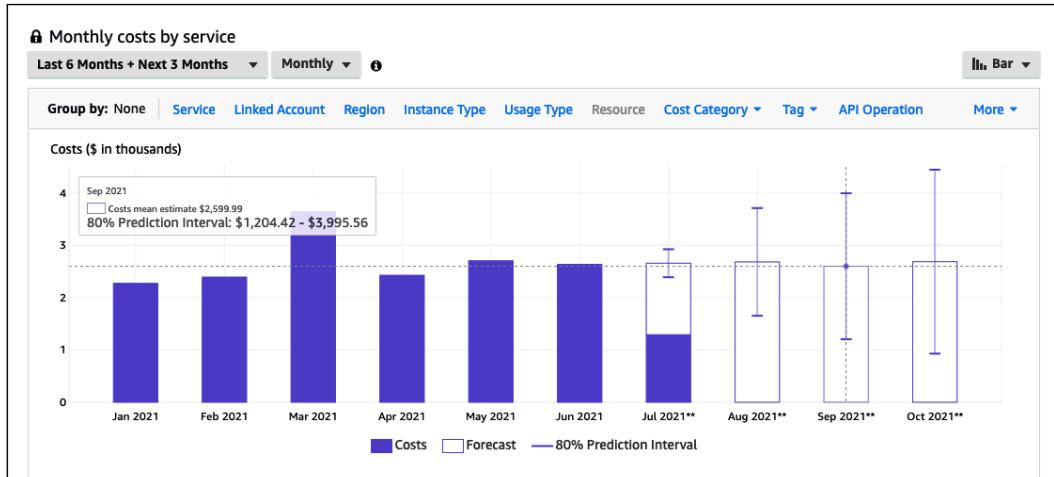


Figure 11.8: Cost trend and cost forecast report

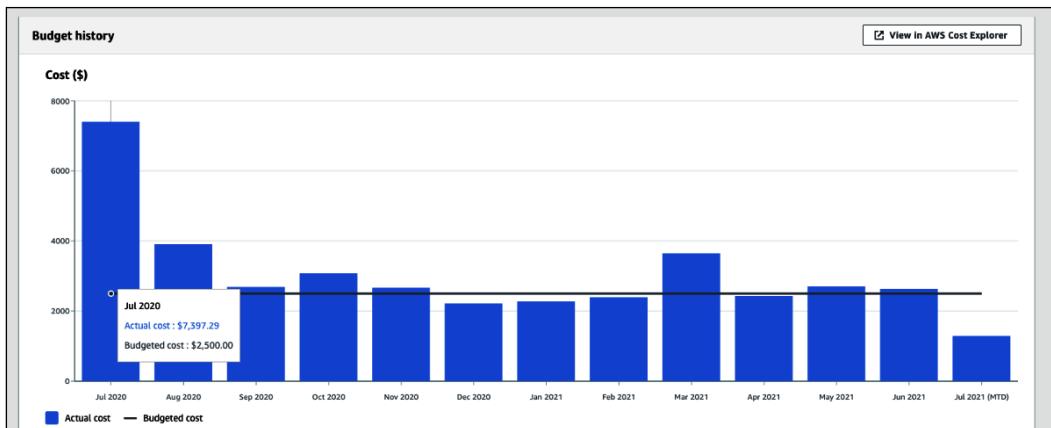


Figure 11.9: Cost and budget report

Configure alerts

You can send budget alerts via email and/or Amazon Simple Notification Service (Amazon SNS) topic.

Budgeted amount [Edit](#)

\$2,500

Alert 1

Send alert based on:

- Actual Costs
- Forecasted Costs

Alert threshold

80

% of budgeted amount ▾

Notify the following contacts when **Actual Costs** is **Greater than 80% (\$2,000.00)**

Email contacts

abc@example.com

Add email contact

Figure 11.10: Alert based on actual cost

Cost Optimization



3 ✓ 6 ▲ 0 ⓘ
\$1,386.84
Potential monthly savings

Filter by tag

Tag Key

Tag Value

Apply filter

Reset

Cost Optimization Checks



Amazon EC2 Reserved Instances Optimization

A significant part of using AWS involves balancing your Reserved Instance (RI) usage and your On-Demand instance usage.

Estimated monthly savings with one year RI term: \$282.75 (39.0%). Estimated monthly savings with three year RI term: \$421.84 (59.0%)



Amazon RDS Idle DB Instances

Checks the configuration of your Amazon Relational Database Service (Amazon RDS) for any DB instances that appear to be idle.

1 of 1 DB instances appear to be Idle. Monthly savings of up to \$208.80 are available by minimizing idle DB Instances.

Figure 11.11: Cost-saving suggestions from AWS Trusted Advisor

Chapter 12

DevOps and Solution Architecture Framework



Figure 12.1: Benefits of DevOps

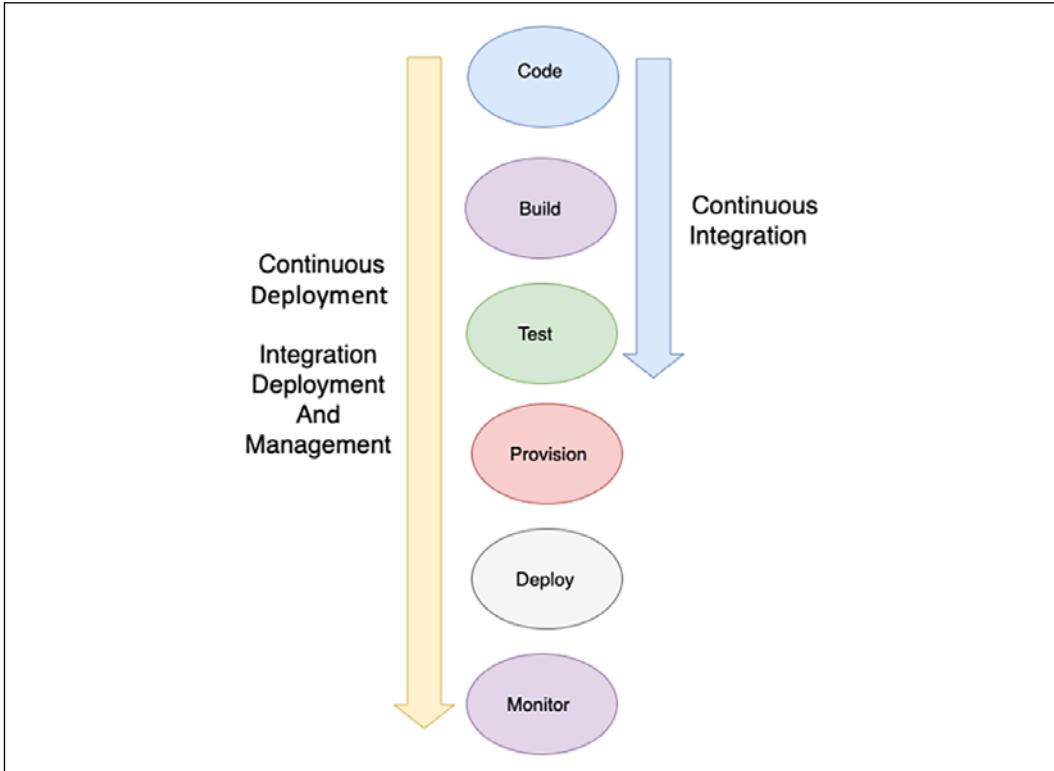


Figure 12.2: CI/CD

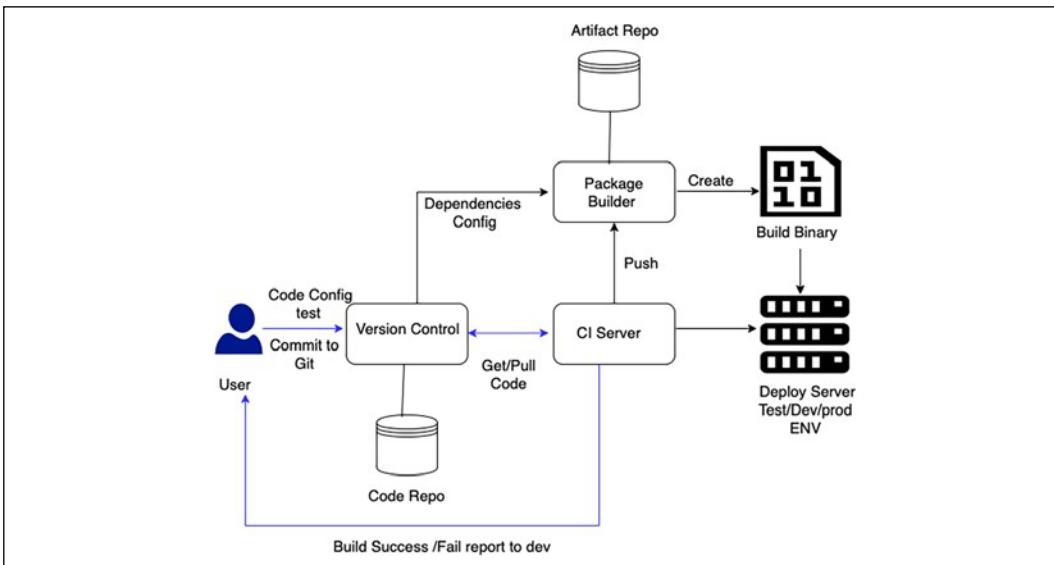


Figure 12.3: CI/CD for DevOps

Specify stack details

Stack name

Stack name: Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

S3NameParam
Enter the S3 Bucket Name:

Cancel **Previous** **Next**

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

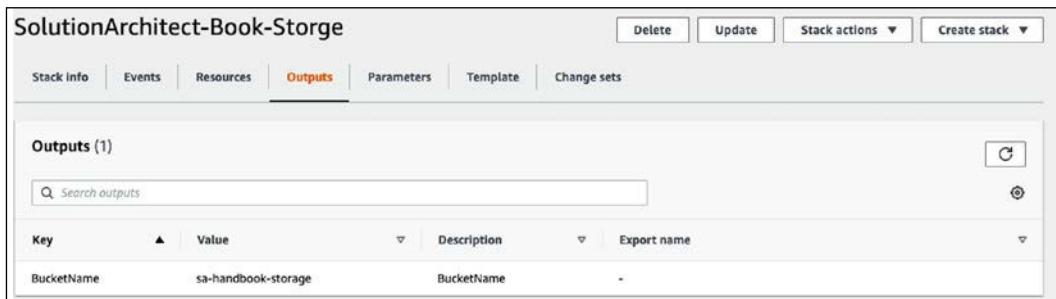


Figure 12.5: Automated AWS S3 object storage creation using AWS CloudFormation

	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

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

<h3>Layers</h3>  <p>A layer is a blueprint for a set of instances. It specifies the instance's resources, installed packages, profiles and security groups.</p> <p>Add a layer</p>	<h3>Instances</h3>  <p>An instance represents a server. It can belong to one or more layers, that determine the instance's resources and configuration.</p> <p>Add an instance or register a server</p>
<h3>Apps</h3>  <p>An app represents code stored in a repository that you want to run on application server instances.</p> <p>Add an app</p>	<h3>Deployments and Commands</h3>  <p>You can deploy the code from your repository to the appropriate server or run commands on some or all instances in your stack.</p> <p>Deploy an app or run a command</p>
<h3>Resources</h3>  <p>The Resources page enables you to use any of your account's Elastic IP addresses, volumes, or RDS instances in your stack.</p> <p>Register resources</p>	<h3>Monitoring</h3>  <p>AWS OpsWorks uses Amazon CloudWatch to provide thirteen custom metrics with detailed monitoring for each instance in the stack.</p> <p>Show monitoring</p>
<h3>Permissions</h3>  <p>Permissions specify how imported IAM users can access this stack. To import users, go to the Users page.</p> <p>Manage permissions</p>	<h3>Tags <small>NEW</small></h3>  <p>You can specify tags to apply to resources in the stack. Tags can help you identify resources in cost allocation reports.</p> <p>Manage stack tags</p>

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

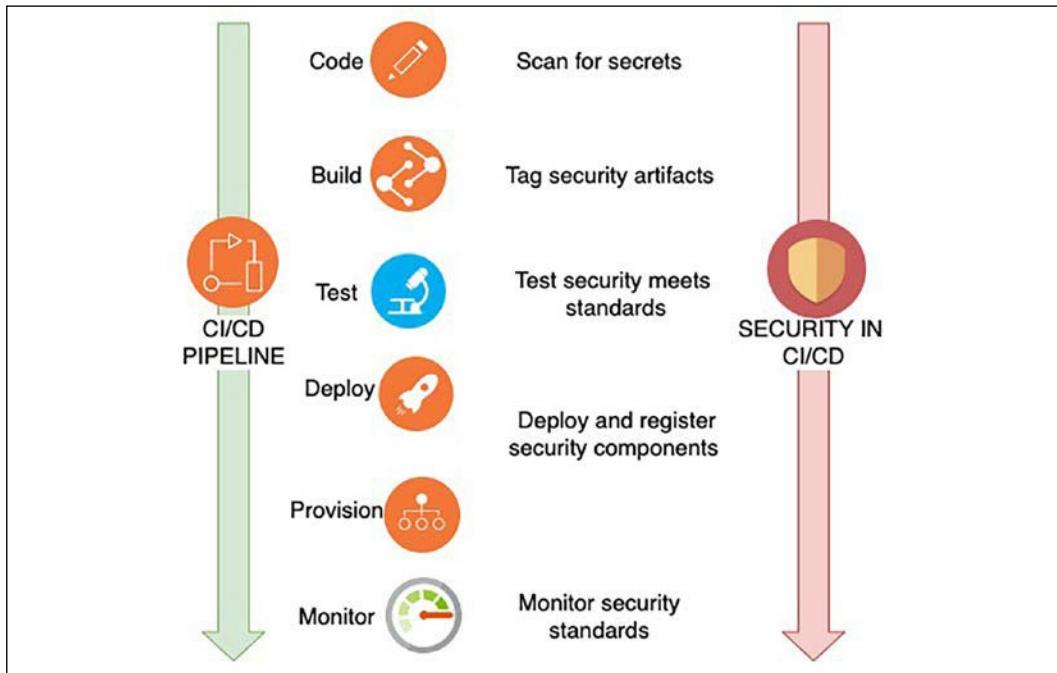


Figure 12.7: DevSecOps and CI/CD

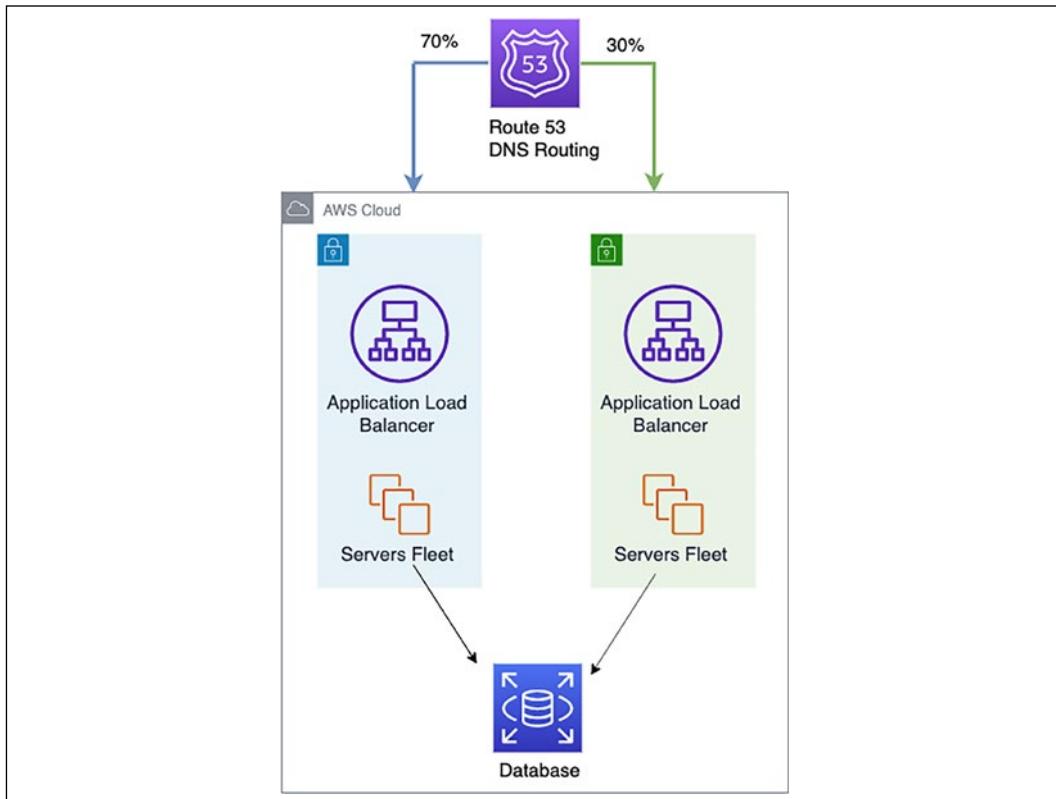


Figure 12.8: Blue-green deployment DNS gradual cutover

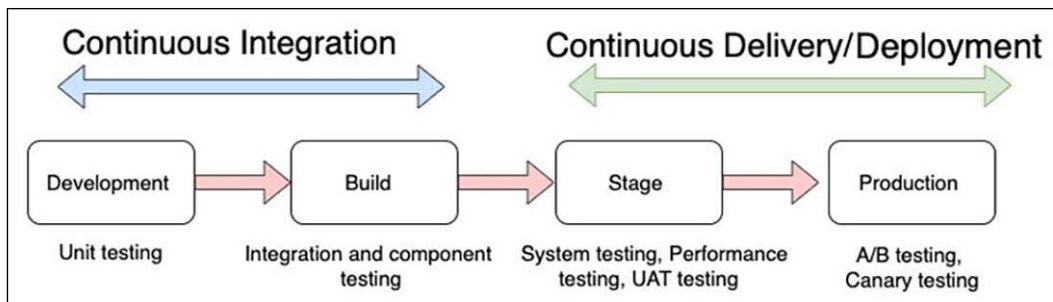


Figure 12.9: Continuous testing in CI/CD

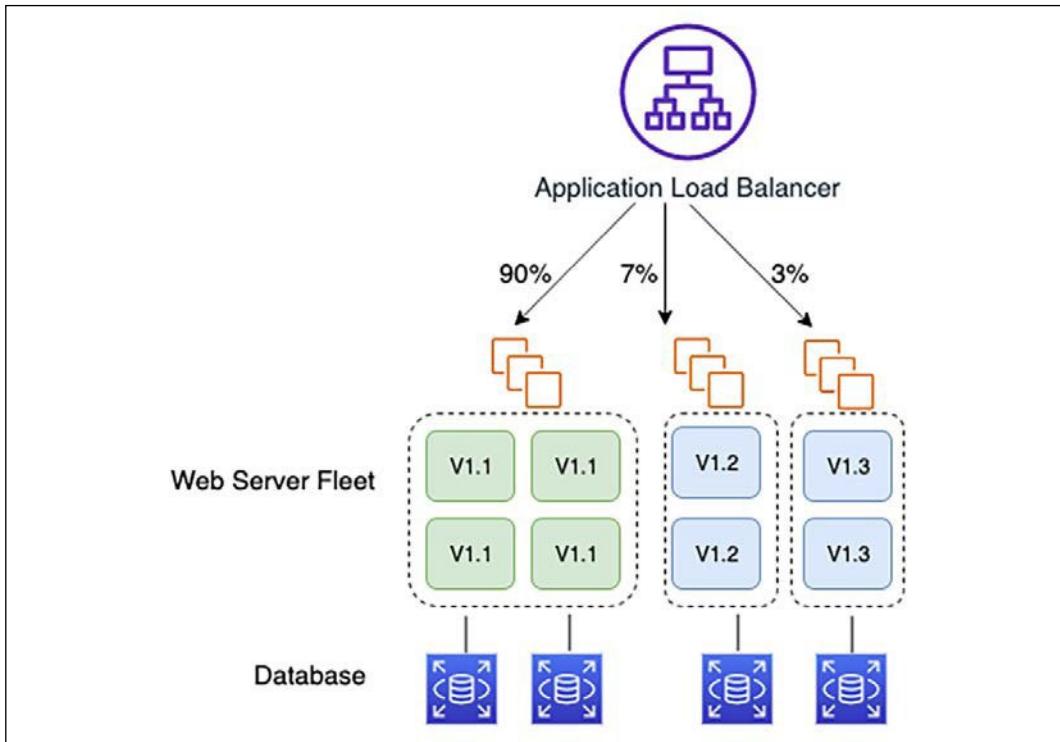


Figure 12.10: Split users by feature experiment using A/B testing

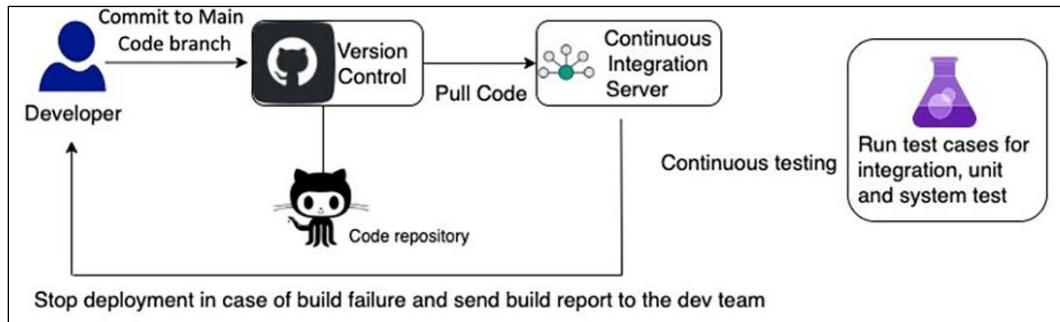


Figure 12.11: Automation of CI

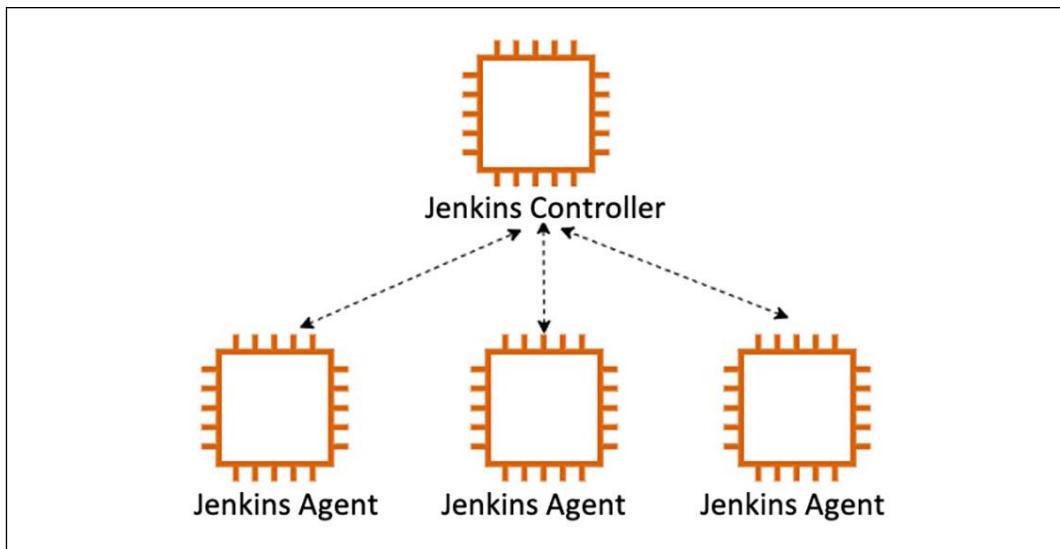


Figure 12.12: Auto-scaling of Jenkins CI servers

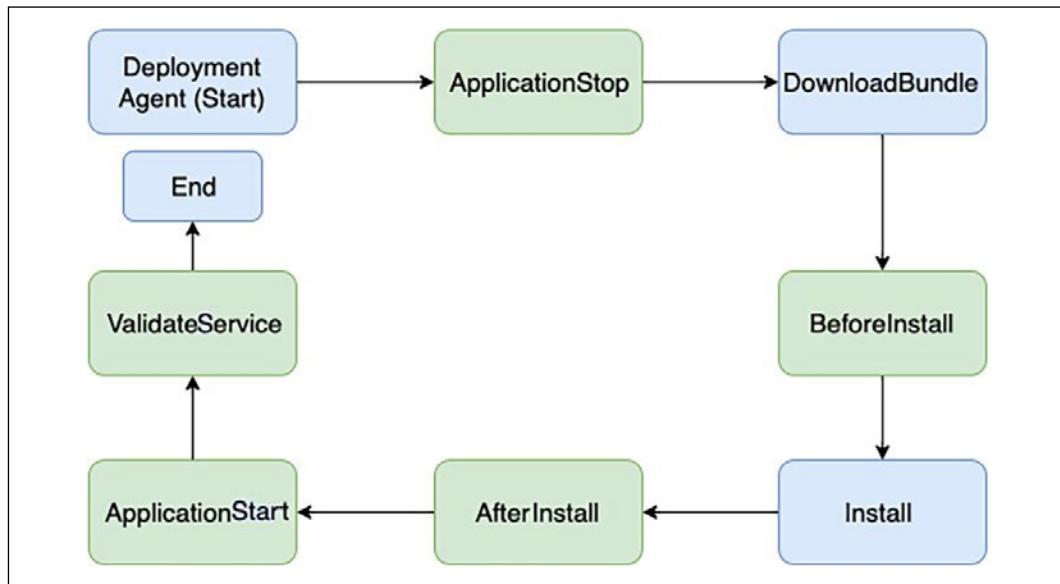


Figure 12.13: Deployment life cycle event

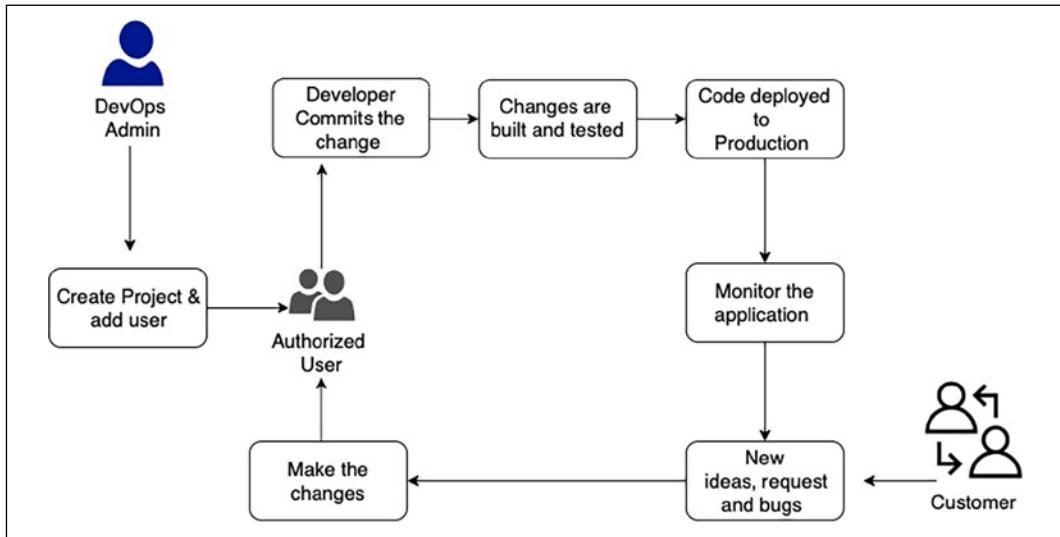


Figure 12.14: CI/CD workflow best practice

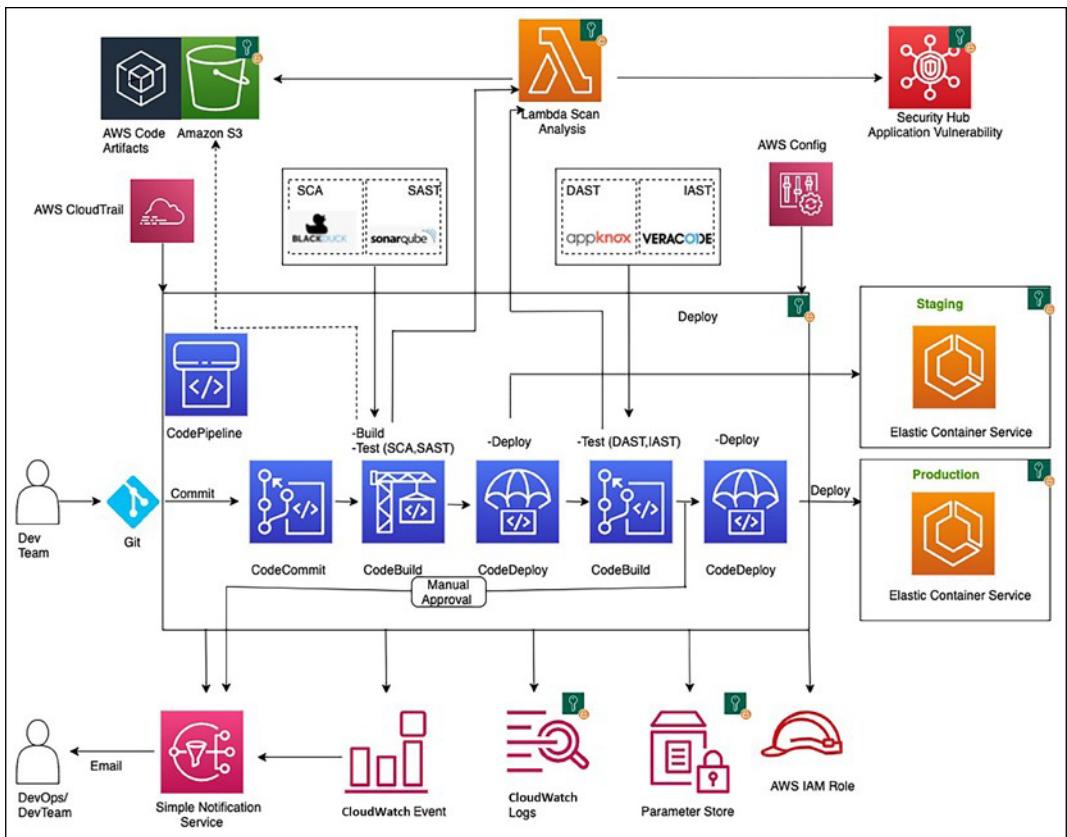


Figure 12.15: DevSecOps CI/CD pipeline architecture in the AWS cloud

Chapter 13

Data Engineering for Solution Architecture

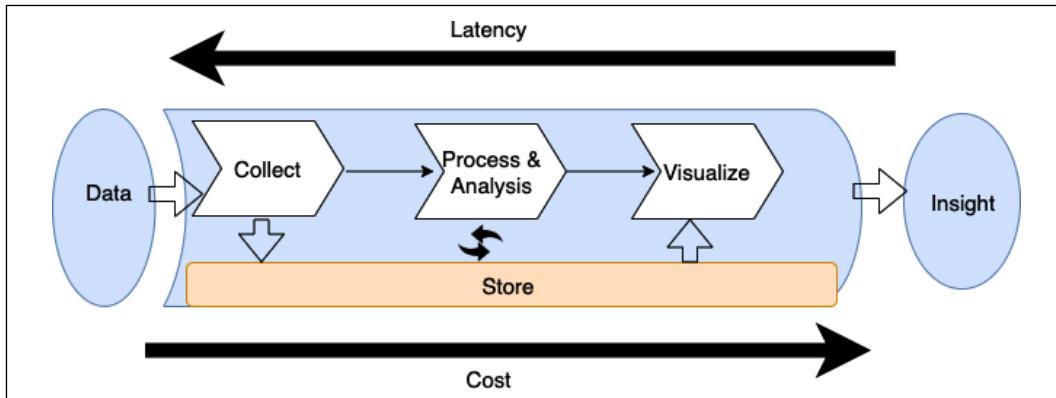


Figure 13.1: Big data pipeline for data architecture design

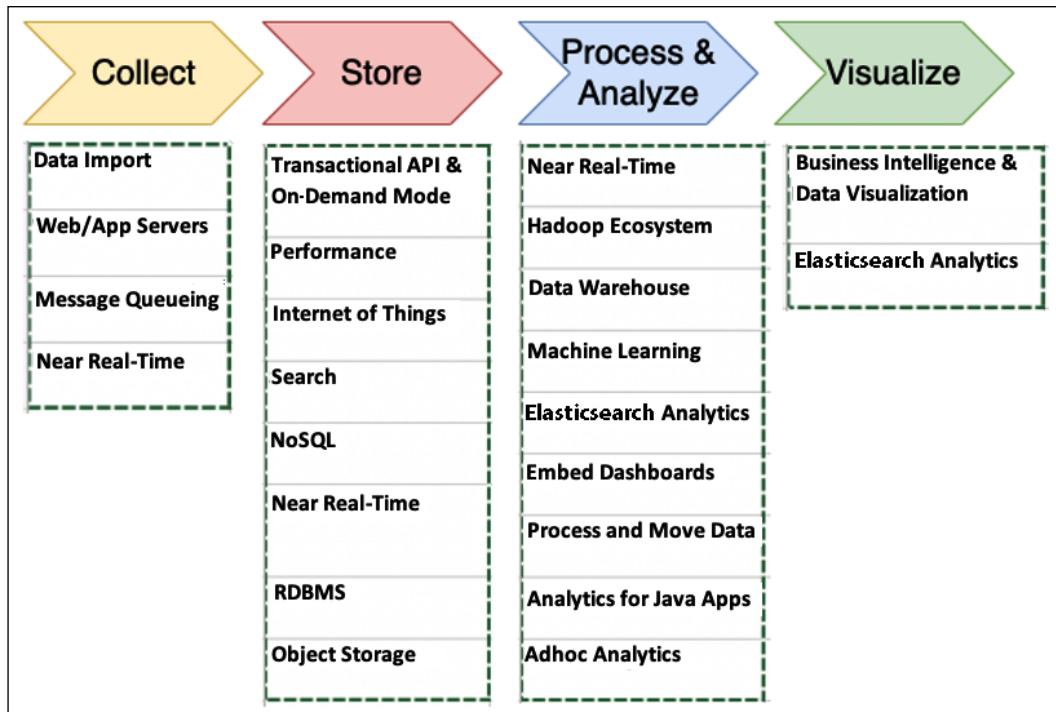


Figure 13.2: Tools and processes for big data architecture design

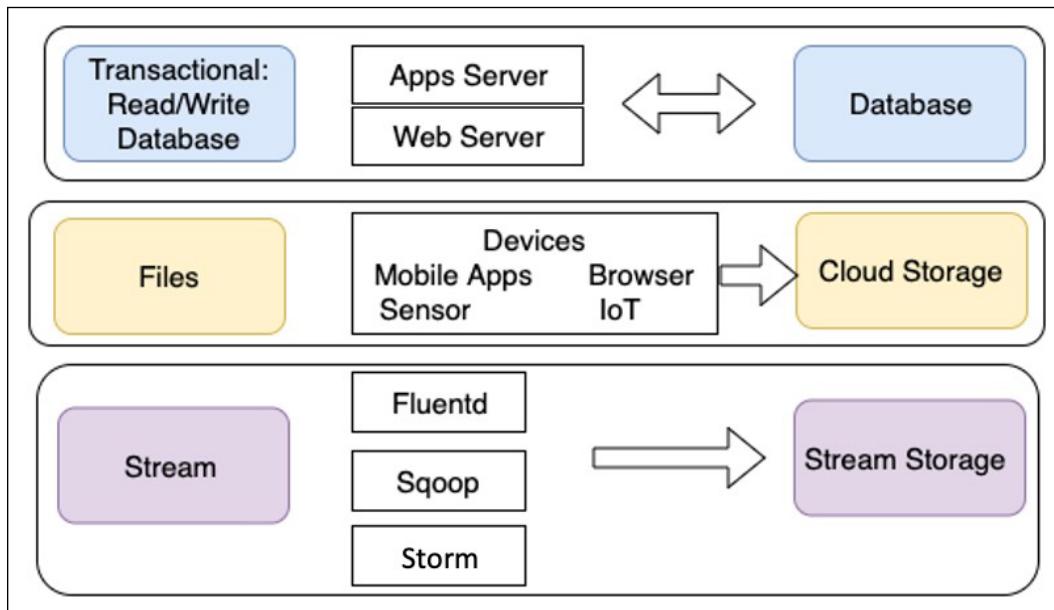


Figure 13.3: Type of data ingestion

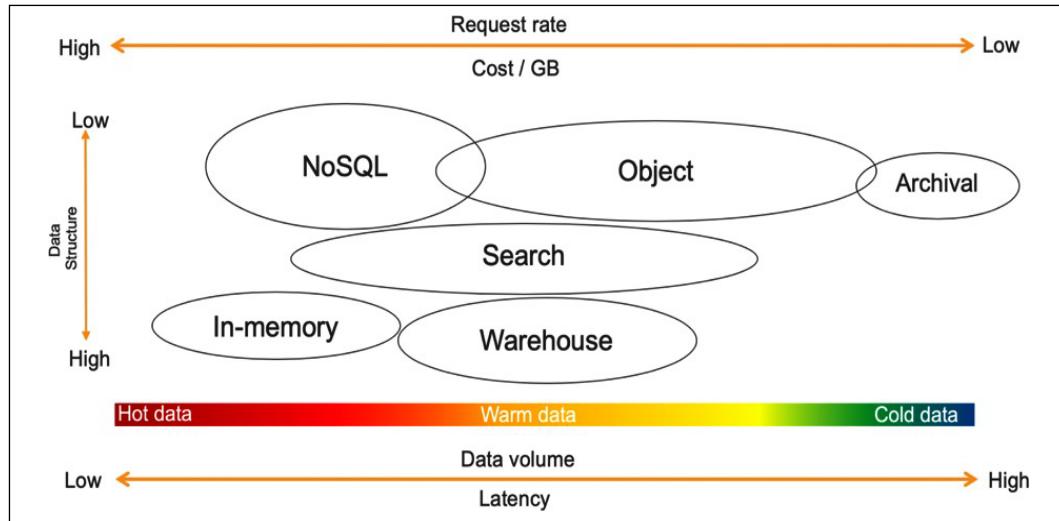


Figure 13.4: Understanding data storage

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

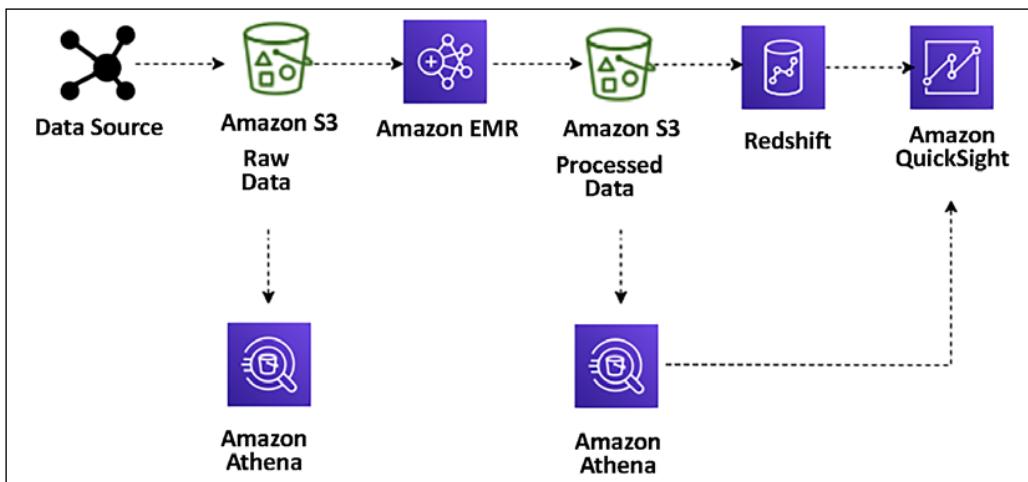


Figure 13.5: Data lake ETL pipeline for big data processing

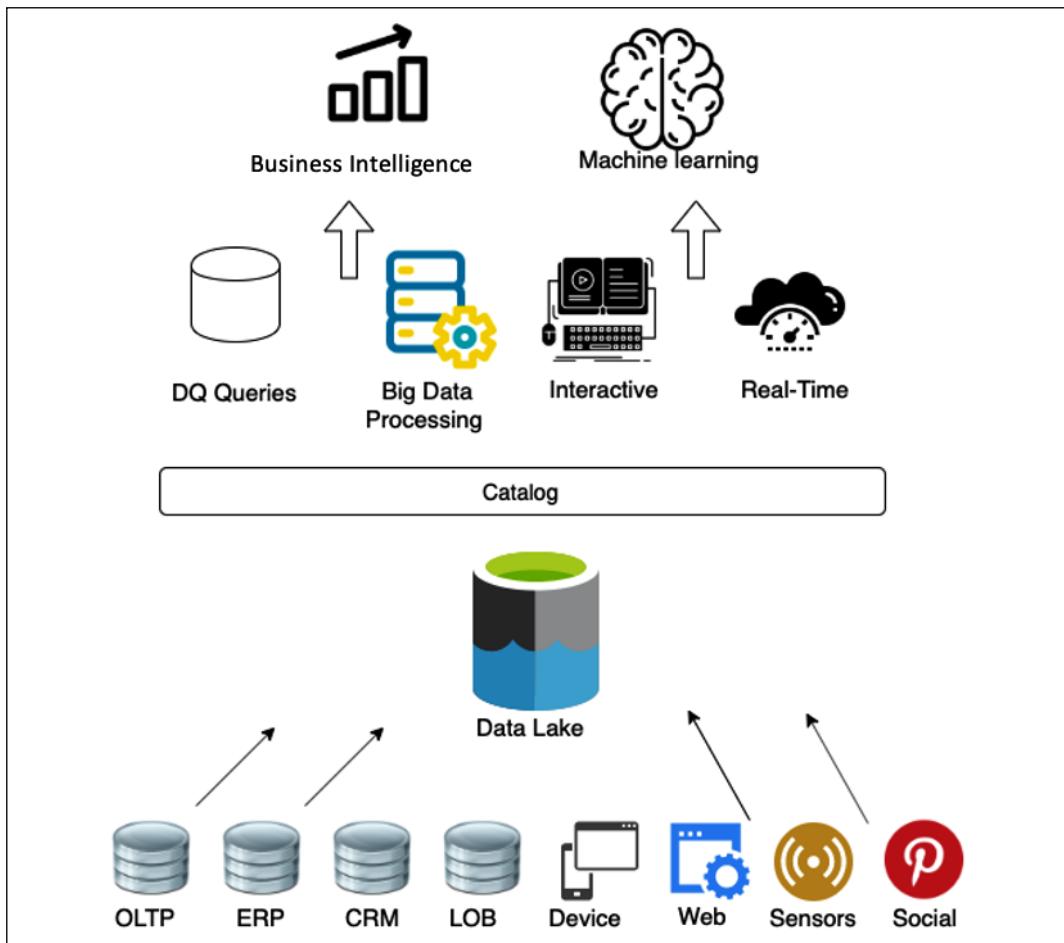


Figure 13.6: Object store for data lake

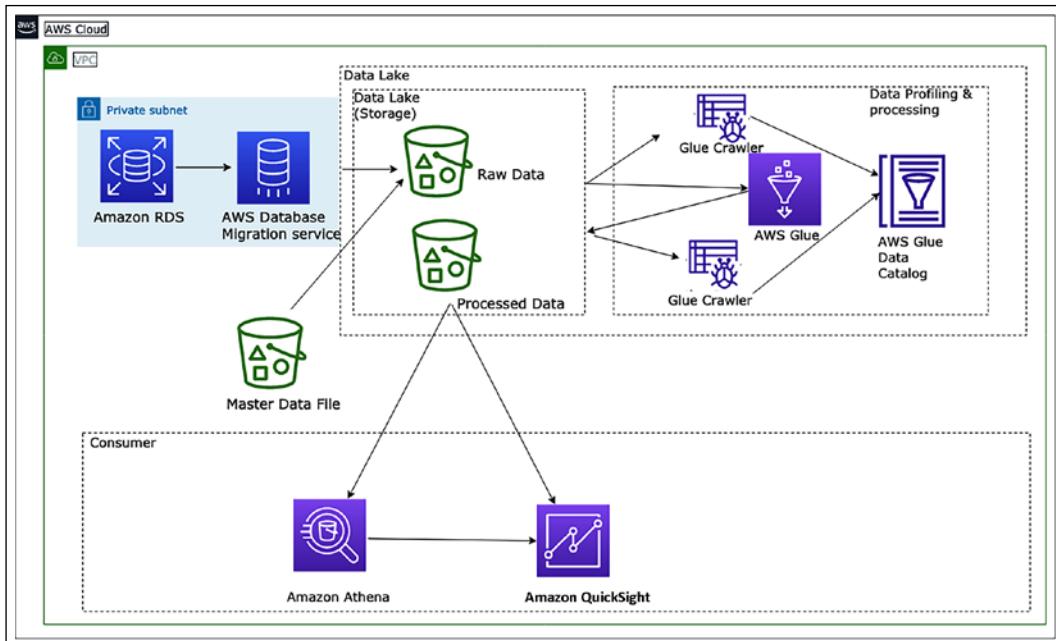


Figure 13.7: Data lake architecture in AWS platform

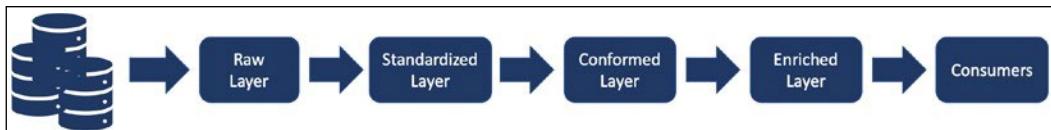


Figure 13.8: Lakehouse architecture layers

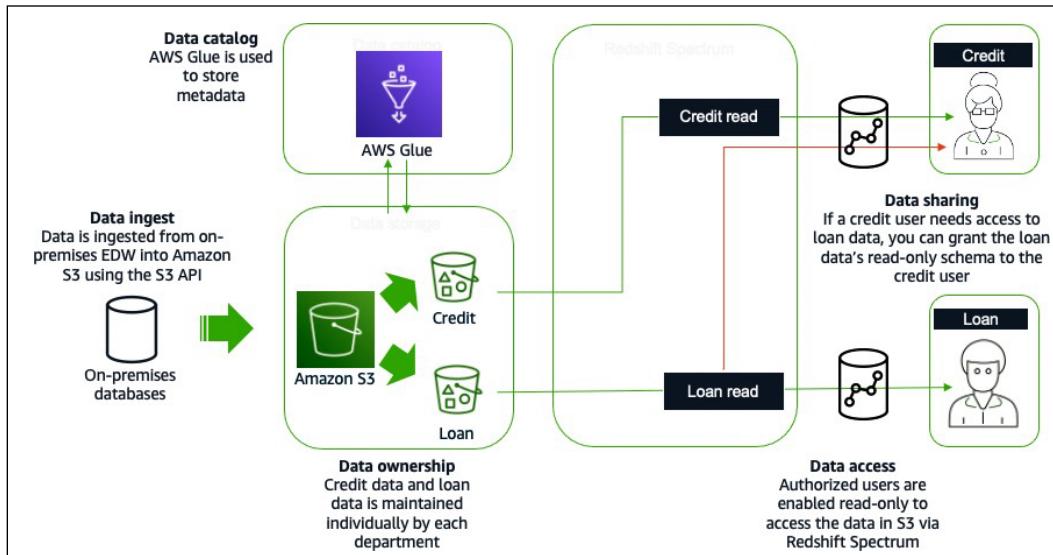


Figure 13.9: Lakehouse architecture in AWS cloud platform using Redshift Spectrum

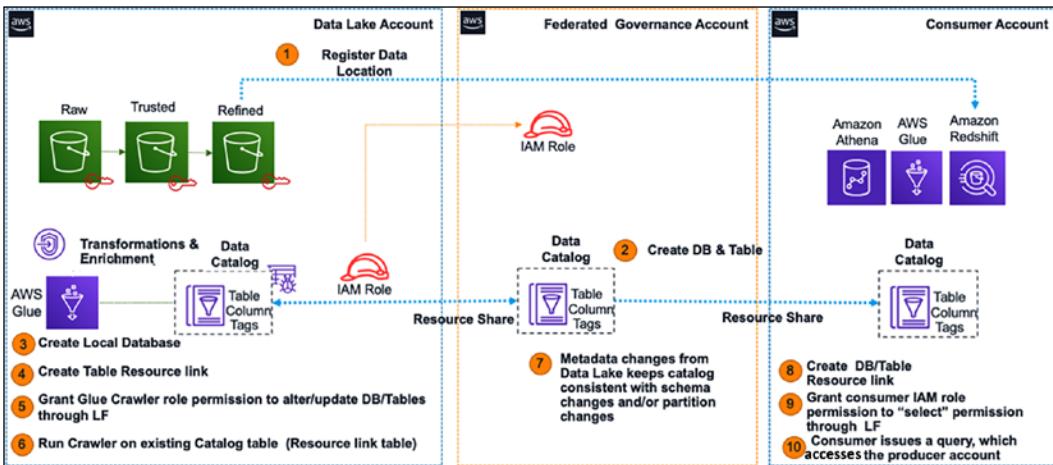


Figure 13.10: Data mesh architecture in AWS cloud platform

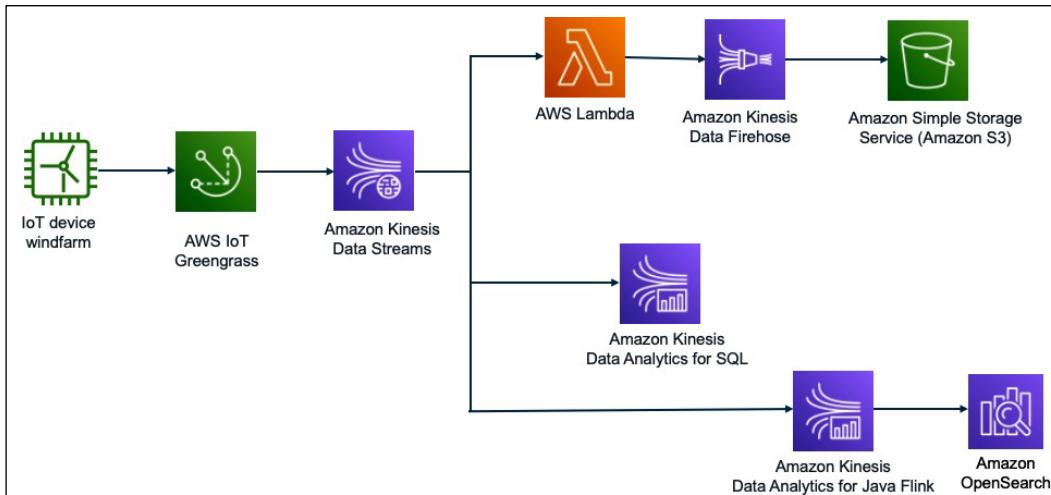


Figure 13.11: Streaming data analytics for IoT data

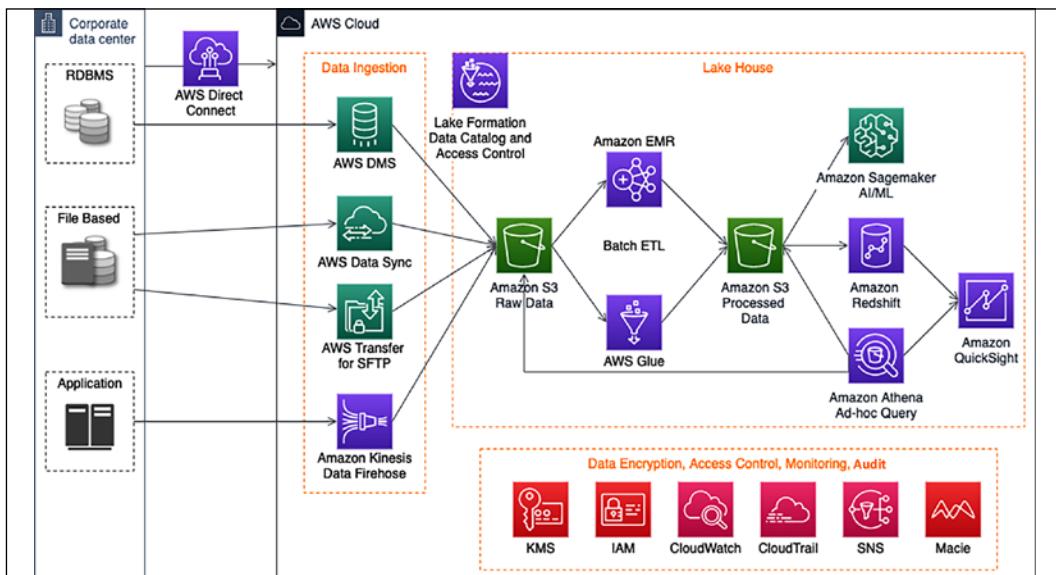


Figure 13.12: Data lake reference architecture

Chapter 14

Machine Learning Architecture

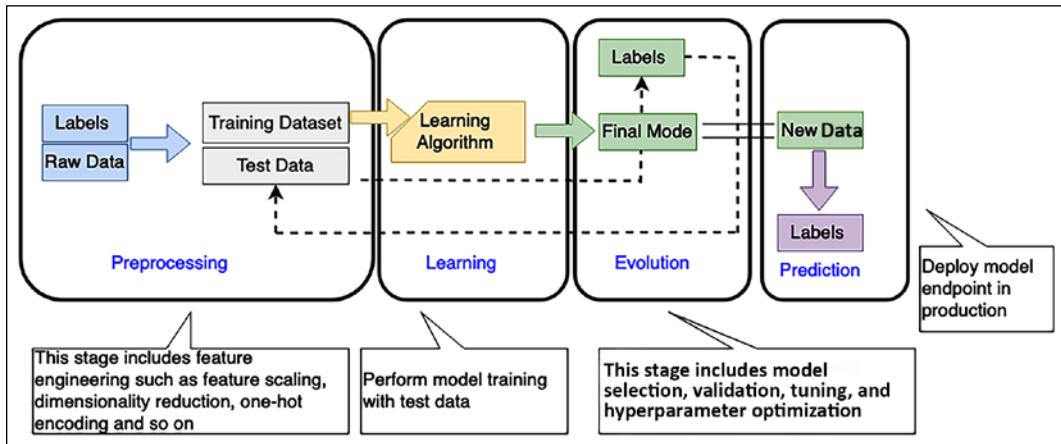


Figure 14.1: ML workflow

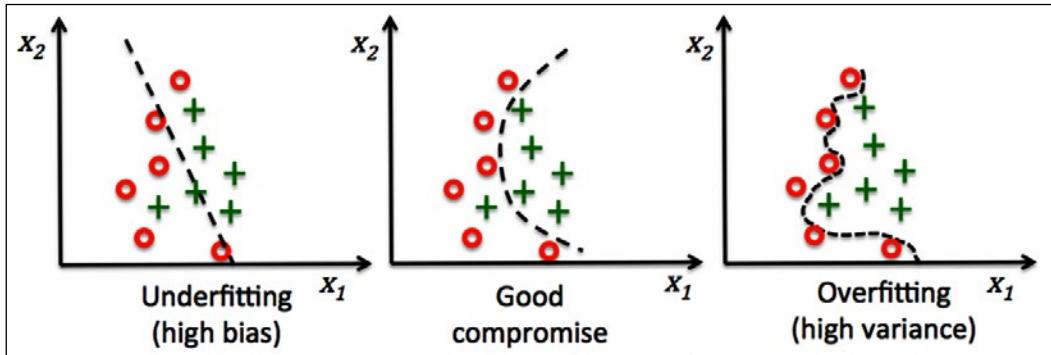


Figure 14.2: ML model overfitting versus underfitting

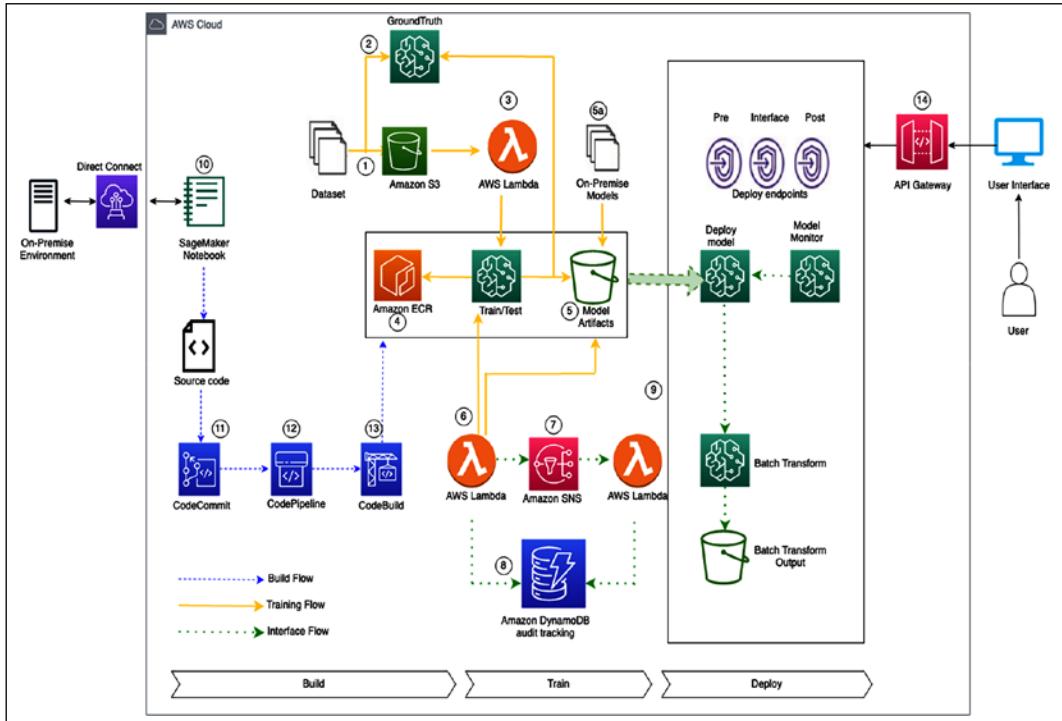


Figure 14.3: ML architecture in the AWS cloud

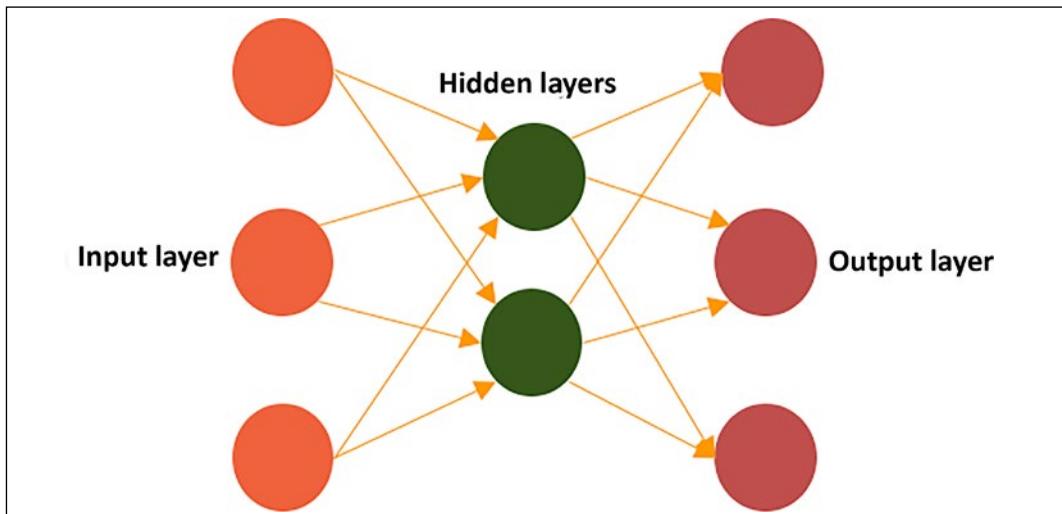


Figure 14.4: A overview of deep learning layers

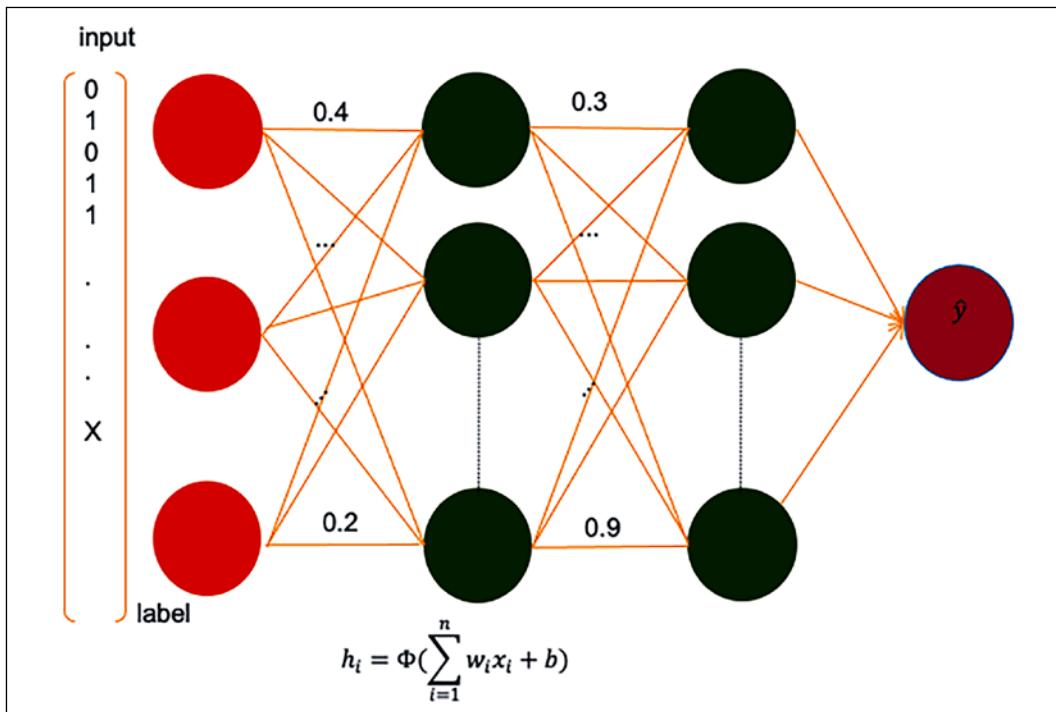


Figure 14.5: Deep learning neural network model

Chapter 15

The Internet of Things Architecture

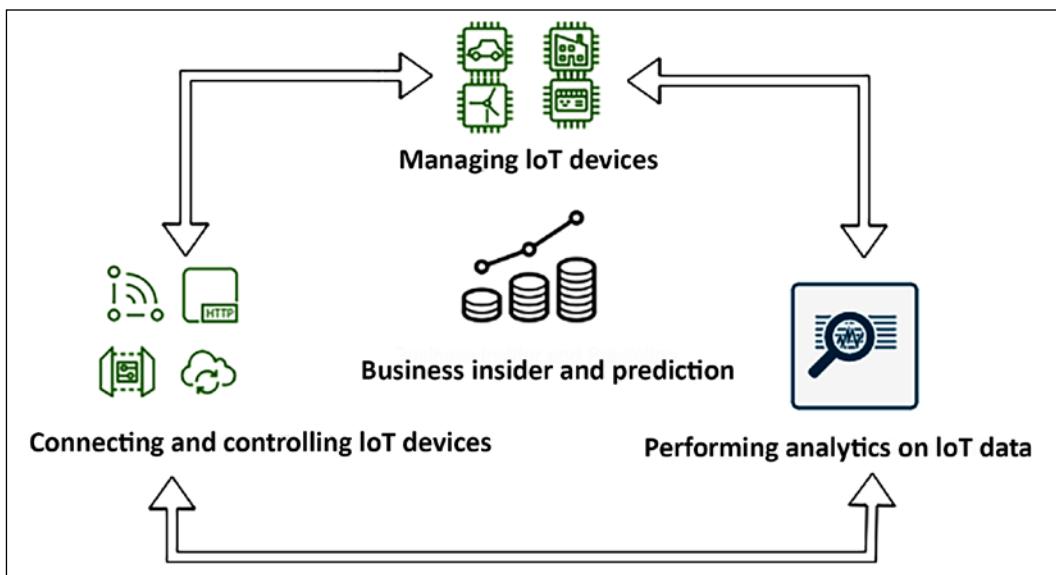


Figure 15.1: IoT architecture cycle

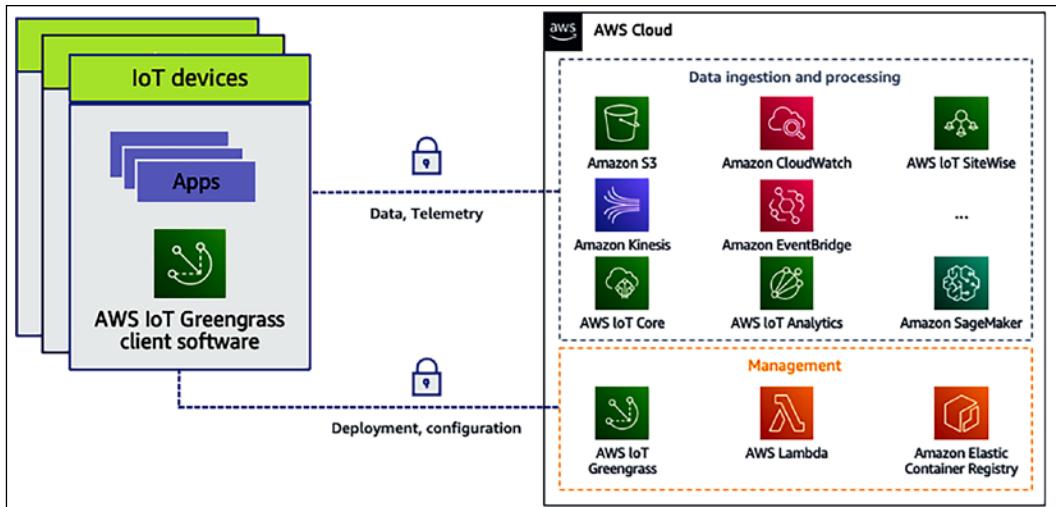


Figure 15.2: AWS IoT Greengrass pre-integrated AWS cloud services

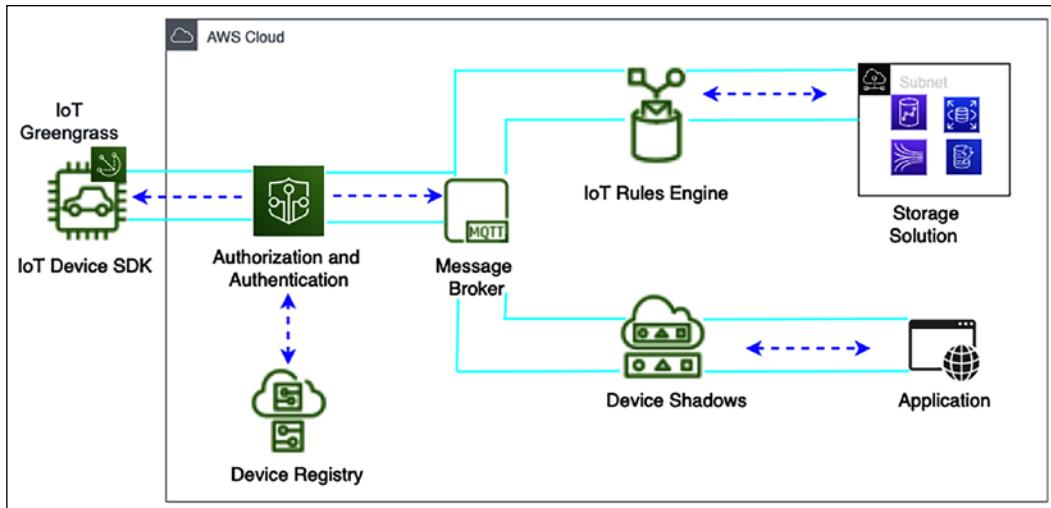


Figure 15.3: IoT architecture on an AWS platform

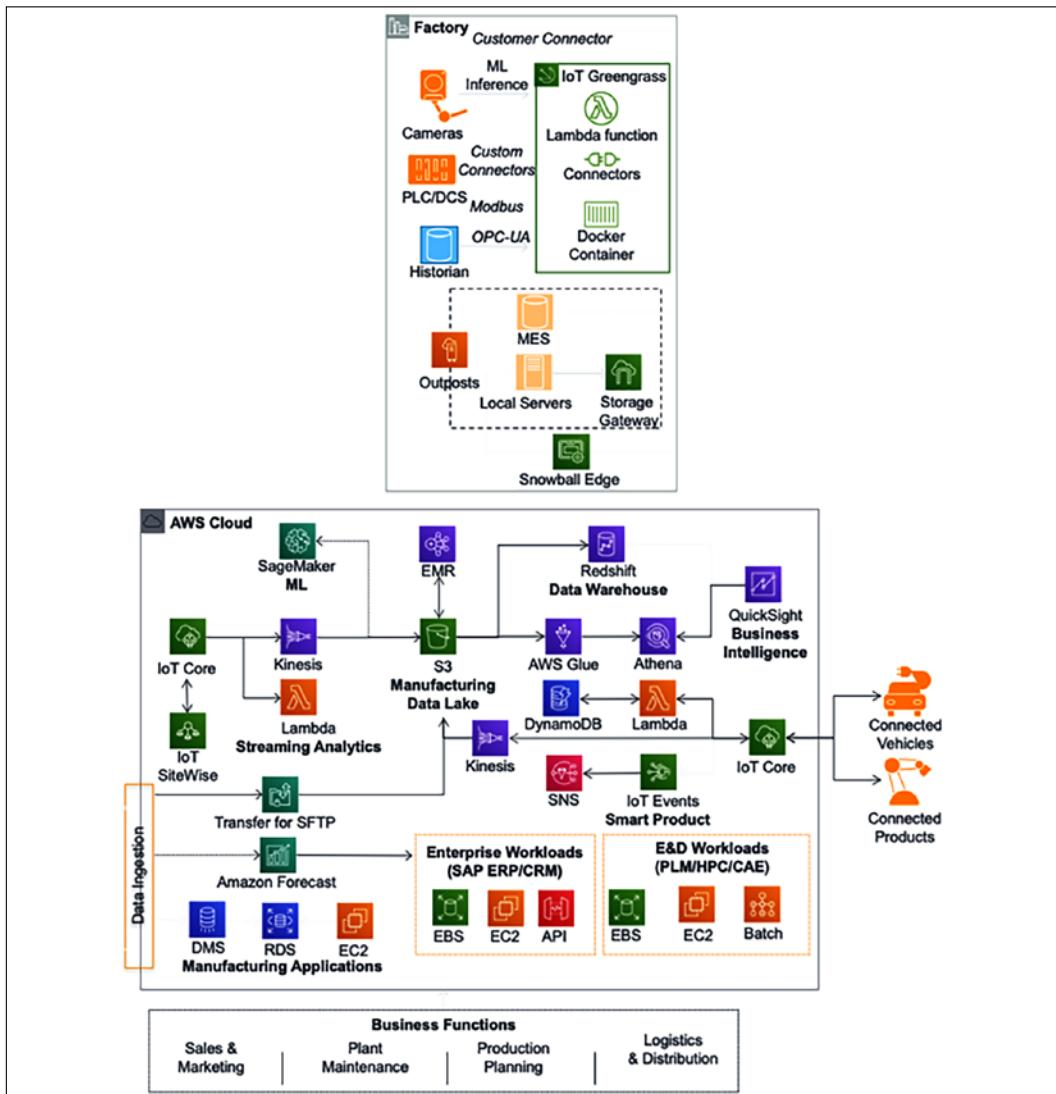


Figure 15.4: Connected Factory architecture in the AWS cloud

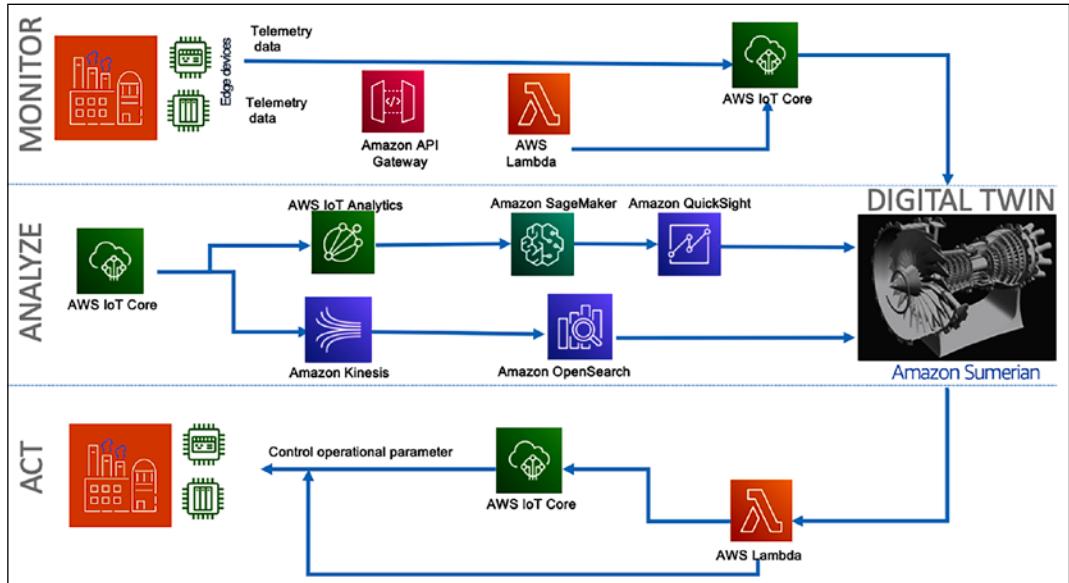


Figure 15.5: Modeling the mind of the machine with a digital twin

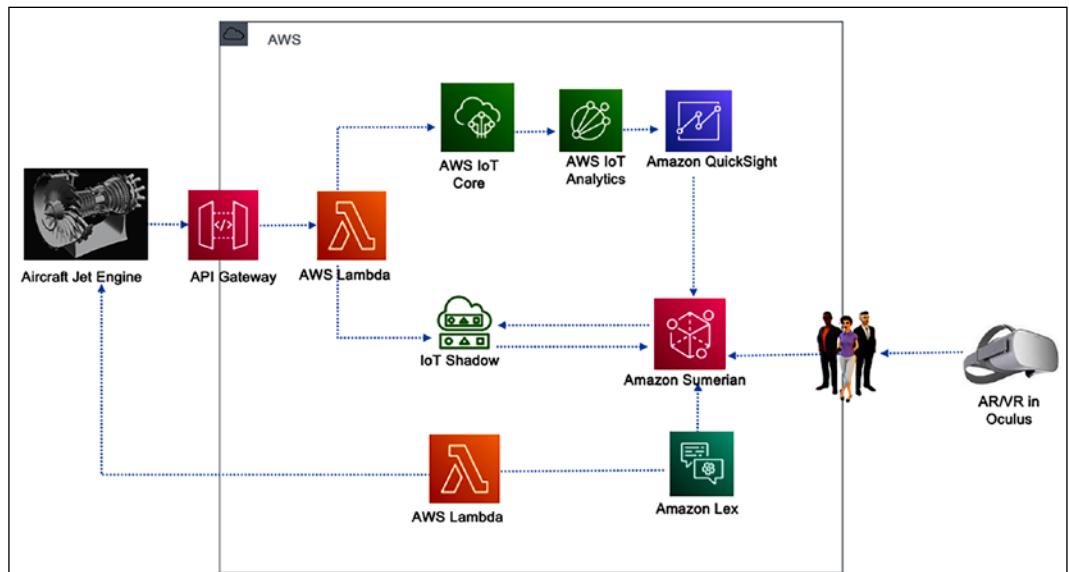


Figure 15.6: Digital twin architecture for an aircraft jet engine

Chapter 16

Quantum Computing

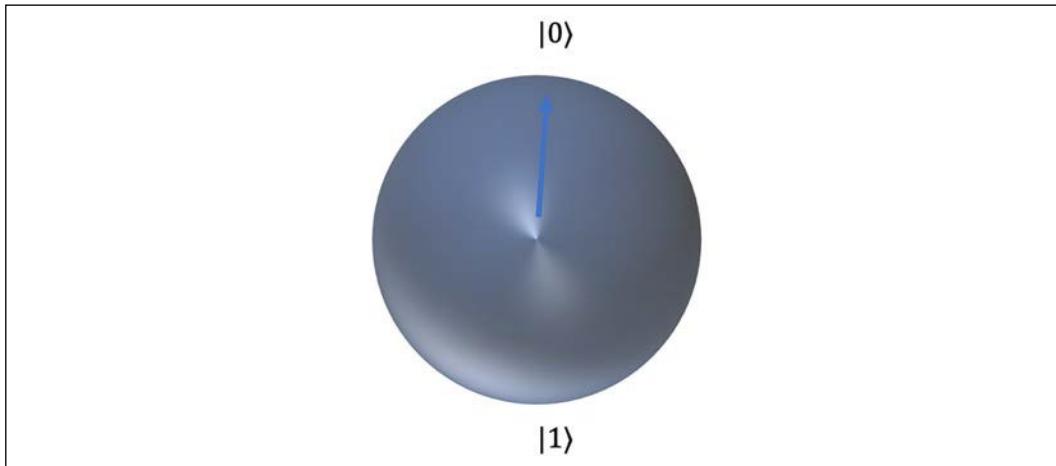


Figure 16.1: Bloch sphere – abstract representation of a qubit

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

Equation 16.1: two dimensional representations of $|0\rangle$ and $|1\rangle$

	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

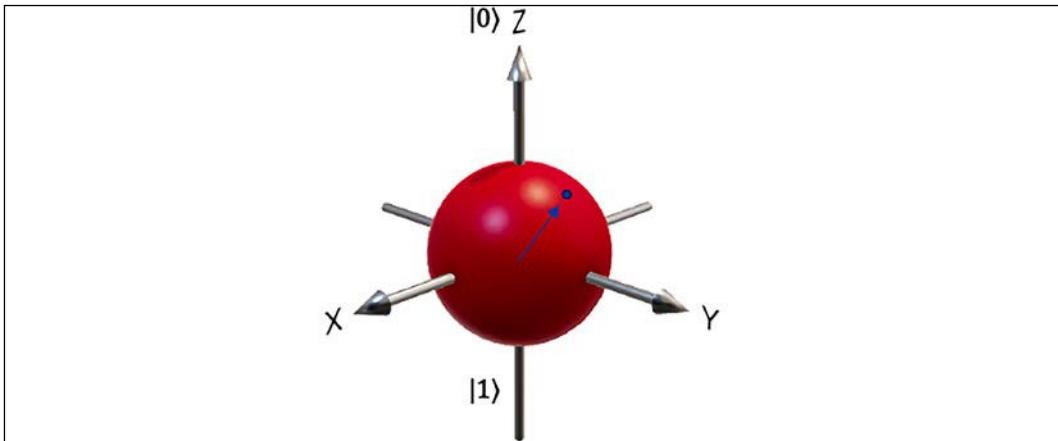


Figure 16.2: 3D representation of an electron's spin along with its axes

$$\begin{array}{c} \text{---} \square X \text{---} \\ \text{Pauli-X gate} \end{array} \left\{ \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right\}$$

Figure 16.3: Pauli-X gate

$$\begin{array}{c} \text{---} \square Y \text{---} \\ \text{Pauli-Y gate} \end{array} \left\{ \begin{array}{cc} 0 & -i \\ i & 0 \end{array} \right\}$$

Figure 16.4: Pauli-Y gate

$$\begin{array}{c} \text{---} \square Z \text{---} \\ \text{Pauli-Z gate} \end{array} \left\{ \begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array} \right\}$$

Figure 16.5: Pauli-Z gate

A quantum circuit diagram showing a Hadamard gate (H) on a single wire. To the right of the gate is the expression $\frac{1}{\sqrt{2}}$. To the right of that is a matrix enclosed in curly braces:

$$\left\{ \begin{array}{cc} 1 & 1 \\ 1 & -1 \end{array} \right\}$$

Figure 16.6: Hadamard gate

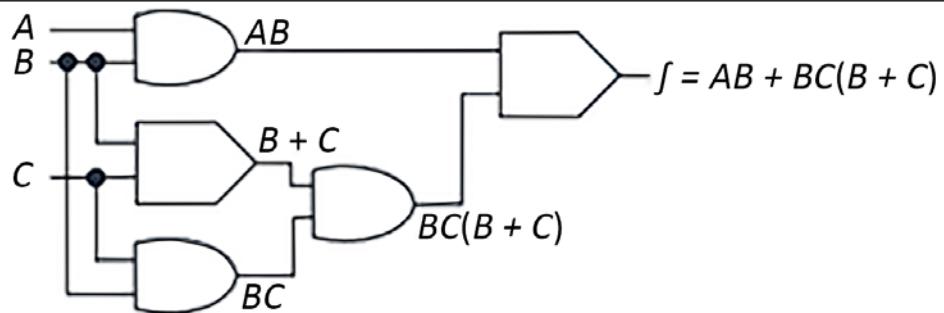


Figure 16.7: Operation in a Boolean circuit

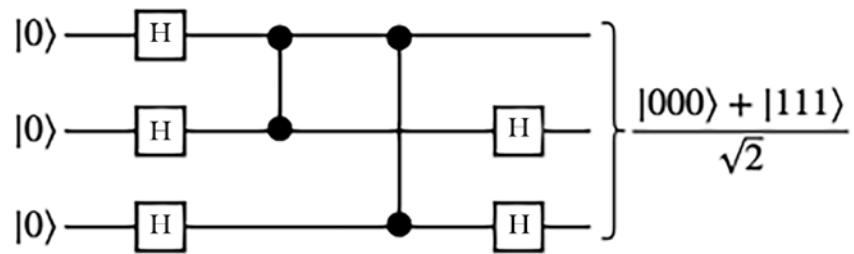


Diagram 16.8: Operation in a quantum circuit

Chapter 17

Rearchitecting Legacy Systems

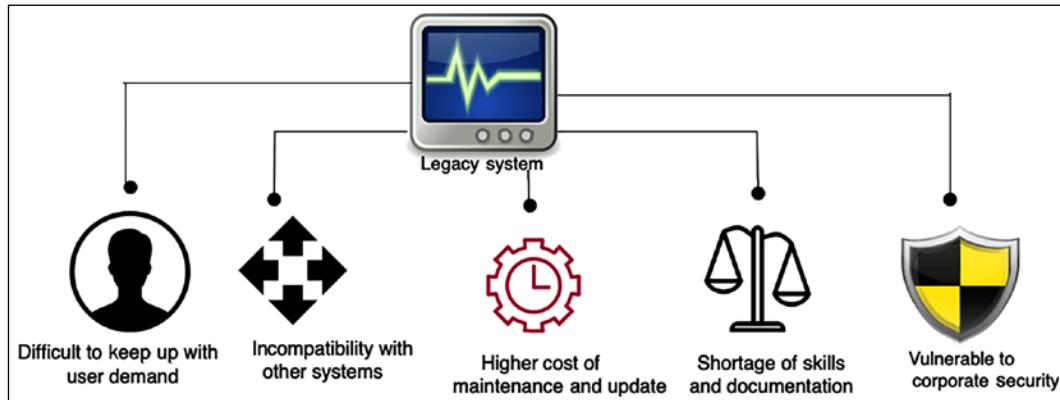


Figure 17.1: Challenges with a legacy system

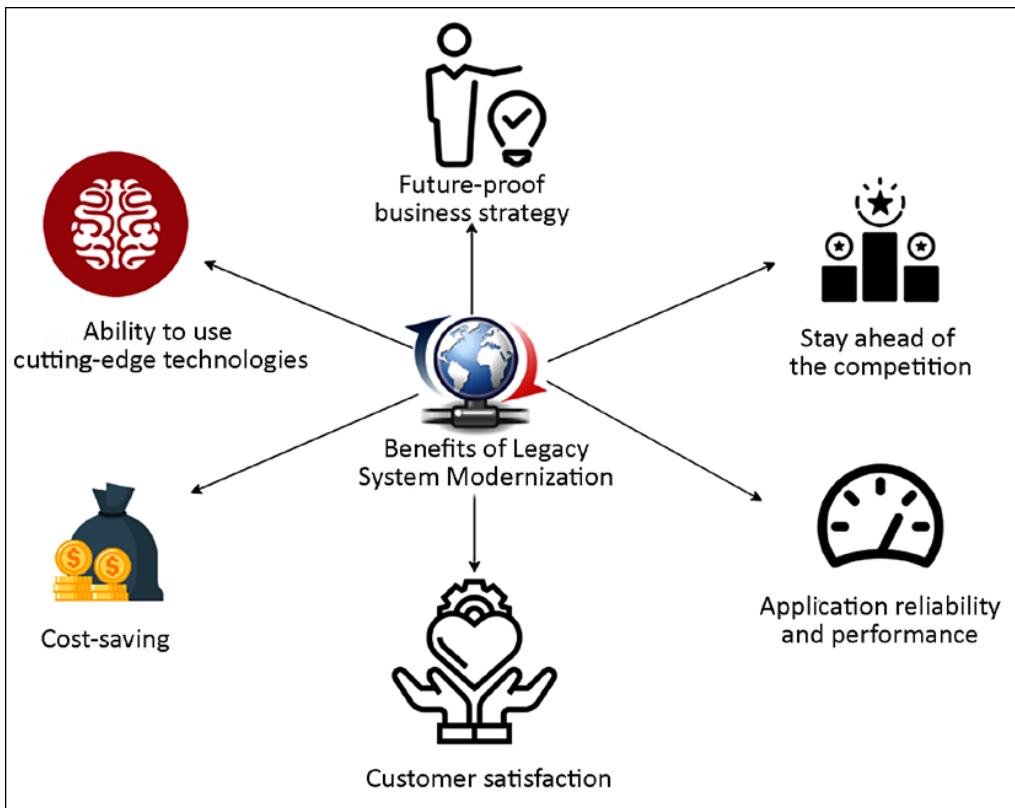


Figure 17.2: Benefits of legacy system modernization

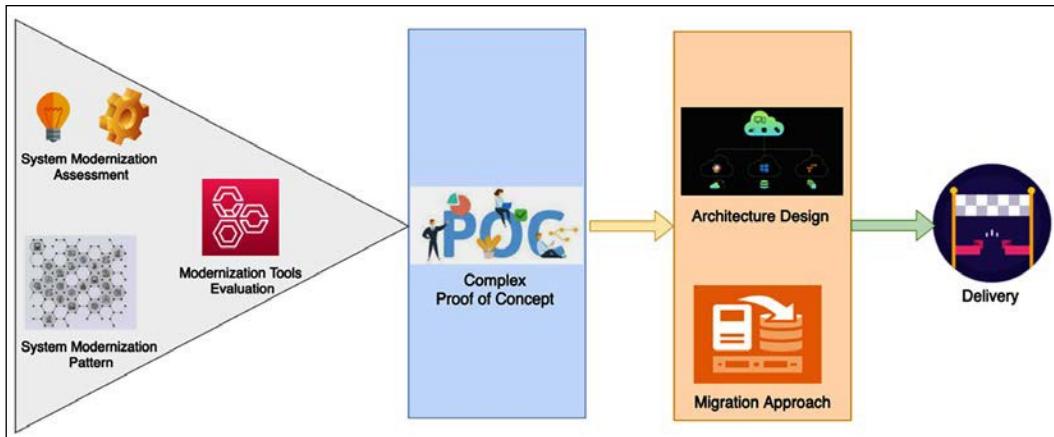


Figure 17.3: The legacy system modernization approach

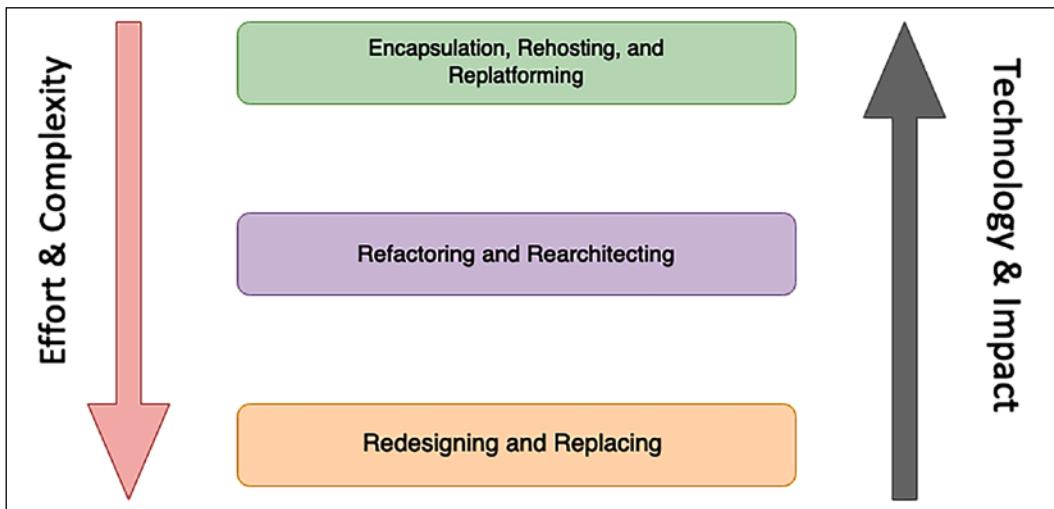


Figure 17.4: Legacy system modernization techniques

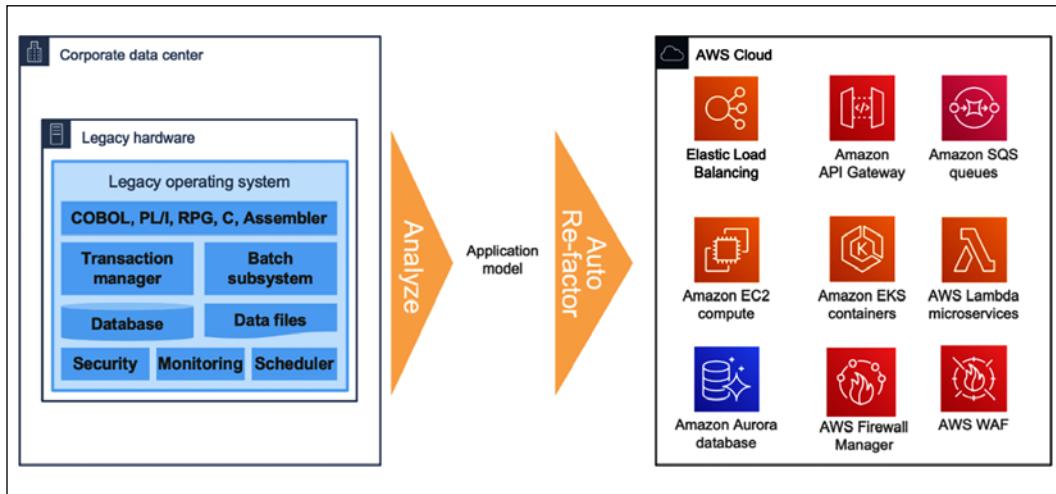


Figure 17.5: Legacy mainframe system modernization to the cloud

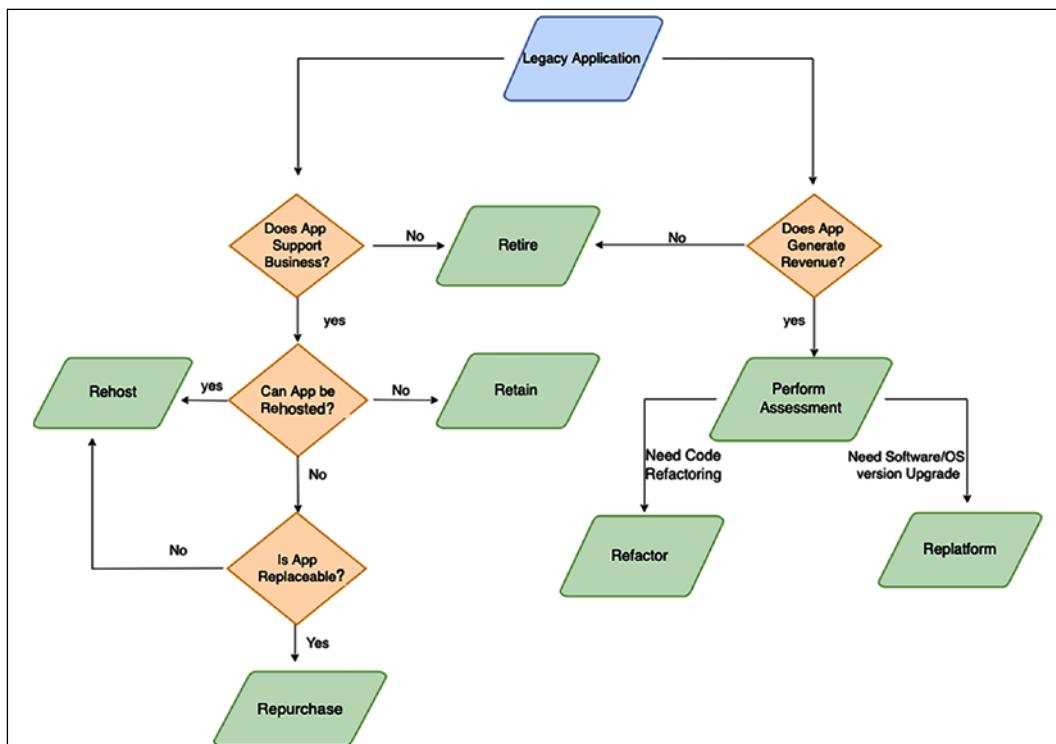


Figure 17.6: Cloud migration path for legacy system modernization

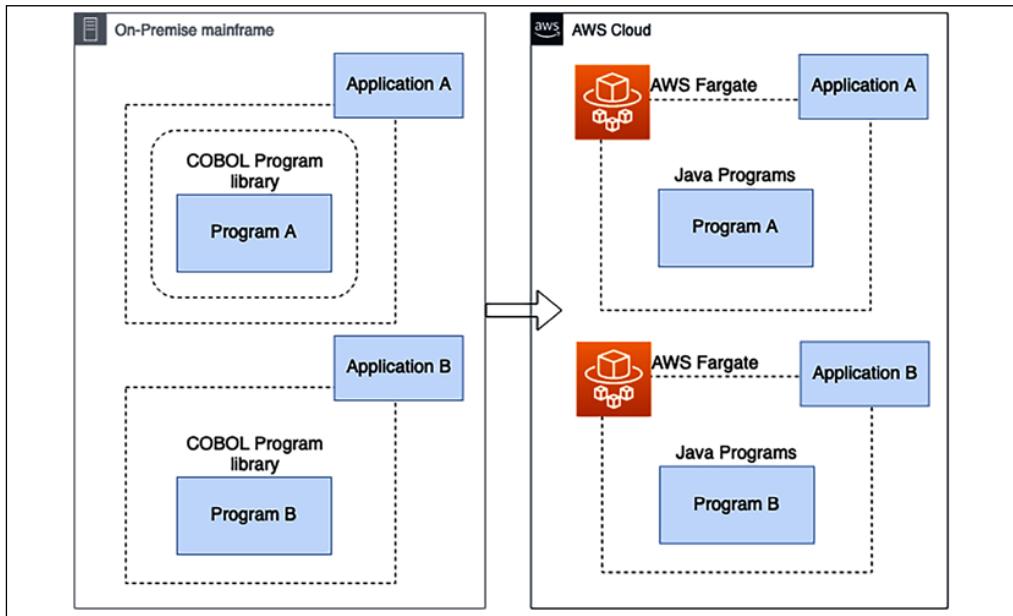


Figure 17.7: Mainframe modernization for a standalone application

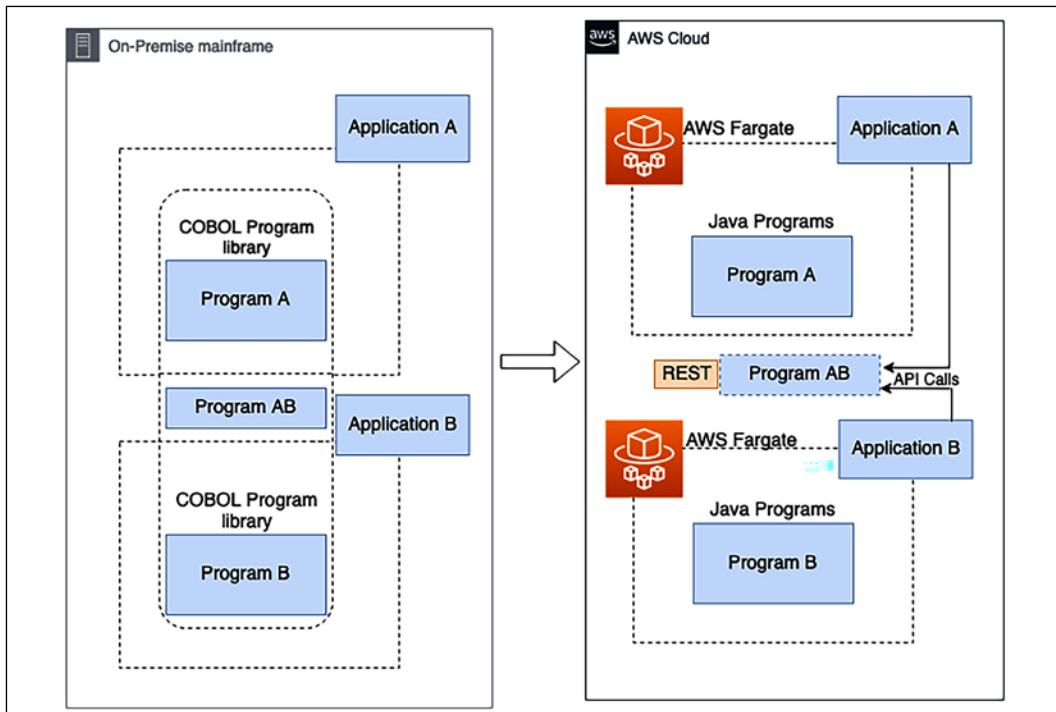


Figure 17.8: Migration of shared program applications using a standalone API

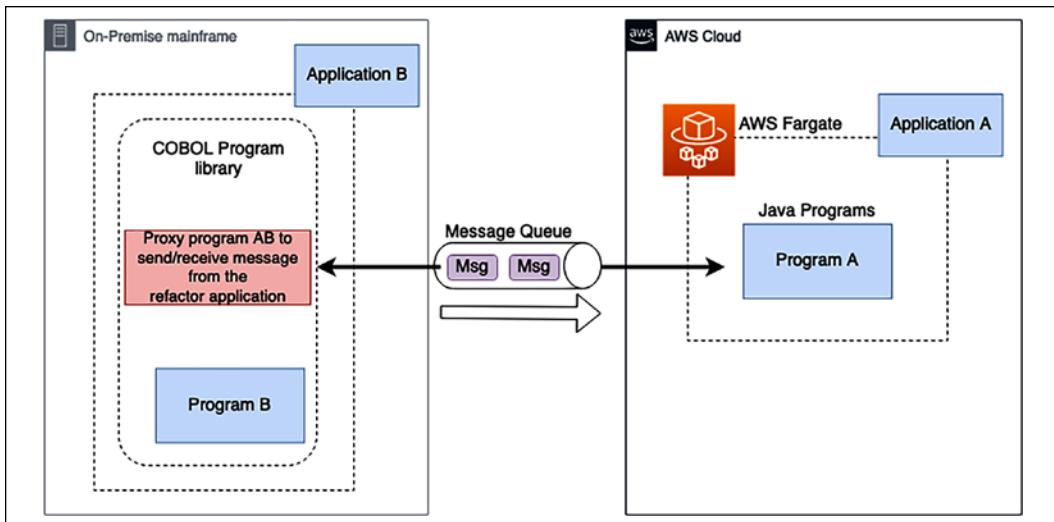


Figure 17.9: Migration of shared program applications using a message queue

Chapter 18

Solution Architecture Document

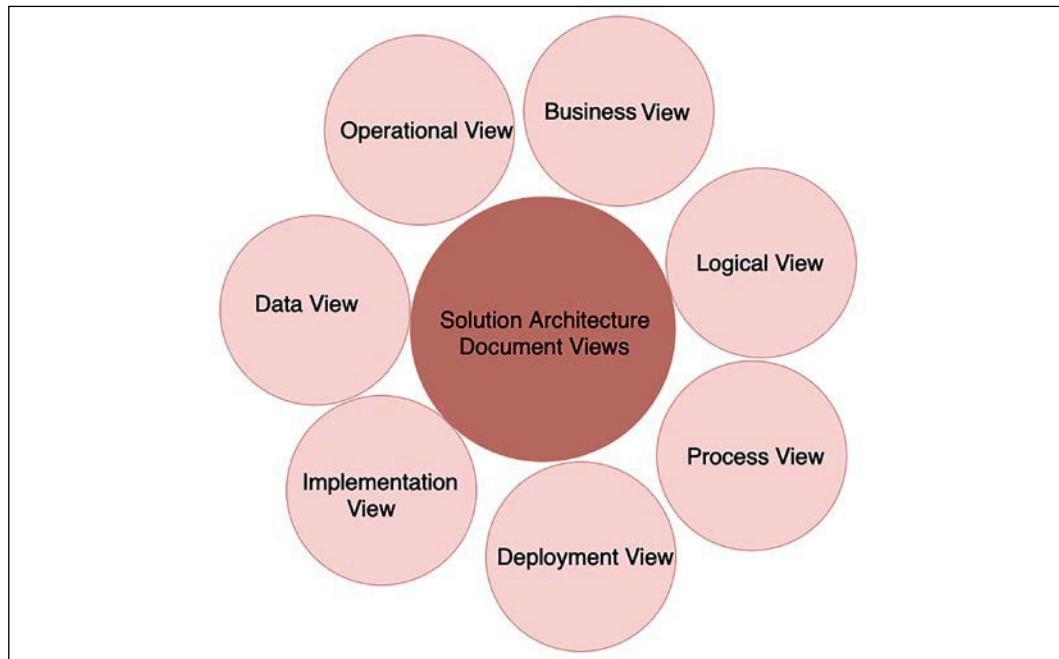


Figure 18.1: SAD views

Contents	
1. Solution Overview	
1.1 Solution Purpose	
1.2 Solution Scope	
1.2.1 In Scope	
1.2.2 Out of Scope	
1.3 Solution Assumptions	
1.4 Solution Constraints	
1.5 Solution Dependencies	
1.6 Key Architecture Decisions	
2. Business Context	
2.1 Business Capabilities	
2.2 Key Business Requirements	
2.2.1 Key Business Processes	
2.2.2 Business Stakeholders	
2.3 Non-Functional Requirements	
2.3.1 Scalability	
2.3.2 Availability and Reliability	
2.3.3 Performance	
2.3.4 Portability	
2.3.5 Security	
3. Conceptual Solution Overview	
3.1 Conceptual and Logical Architecture	
4. Solution Architecture	
4.1 Information Architecture	
4.1.1 Information components	
4.2 Application Architecture	
4.2.1 Application components	
4.3 Data Architecture	
4.3.1 Data Flow and Context	
4.4 Integration Architecture	
4.4.1 Interface Component	
4.5 Infrastructure Architecture	
4.5.1 Infrastructure Component	
4.6 Security Architecture	
4.6.1 Identity and Access Management	
4.6.2 Application Threat Model	
5. Solution Implementation	
5.1 Development	
5.2 Deployment	
5.3 Data Migration	
5.4 Application Decommissioning	
6. Solution Management	
6.1 Operational Management	
6.1.1 Monitoring and Alert	
6.1.2 Support and Incident Management	
6.1.3 Disaster Recovery	
6.2 User On-boarding	
6.2.1 User system requirement	
7. Appendix	
7.1 Open Items	
7.2 Proof of Concept findings	

Figure 18.2: Structure of a SAD

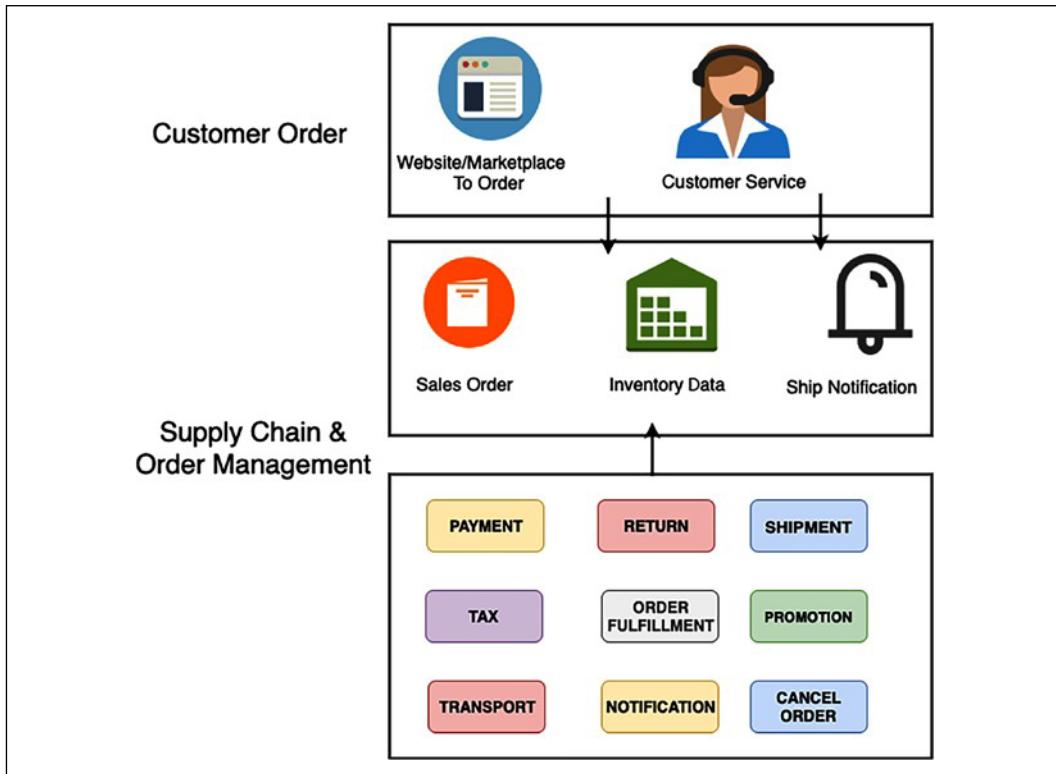


Figure 18.3: Solution overview of an e-commerce platform

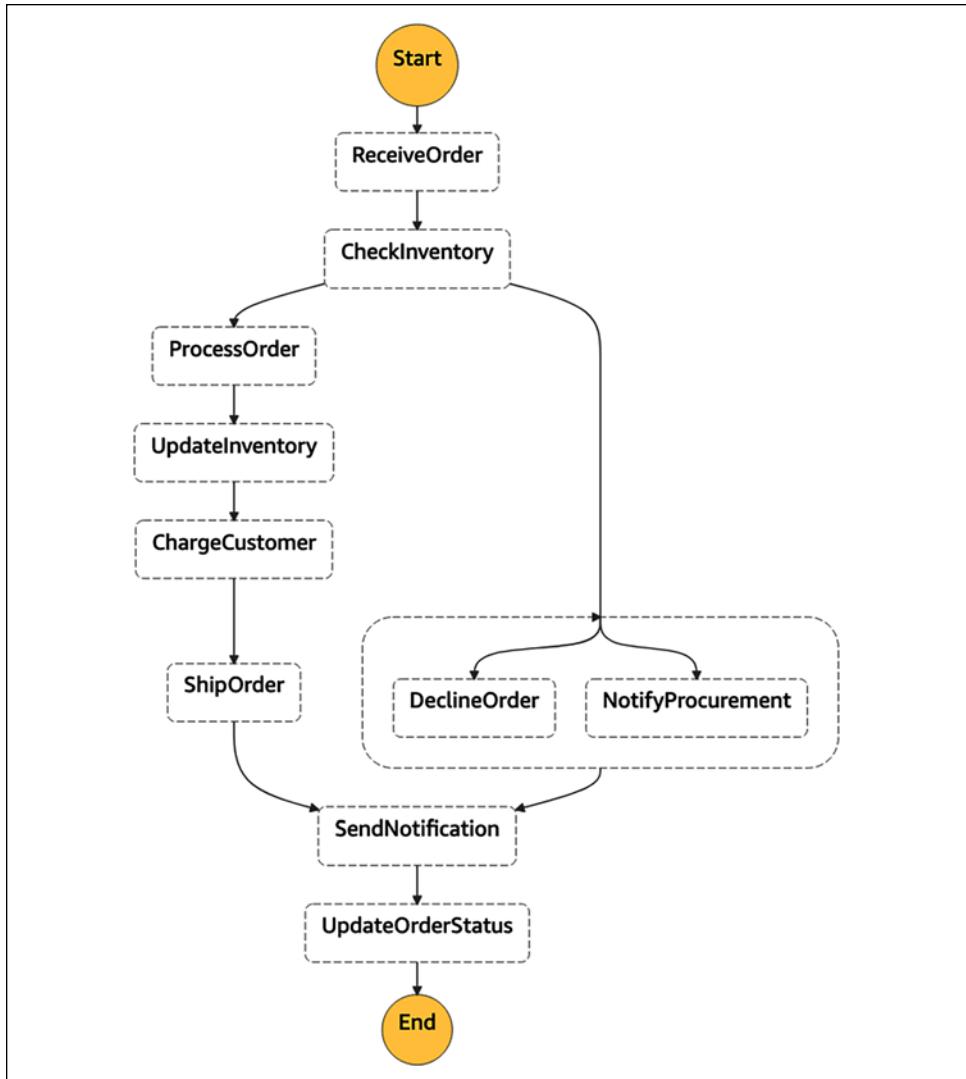


Figure 18.4: Business process diagram of an e-commerce platform

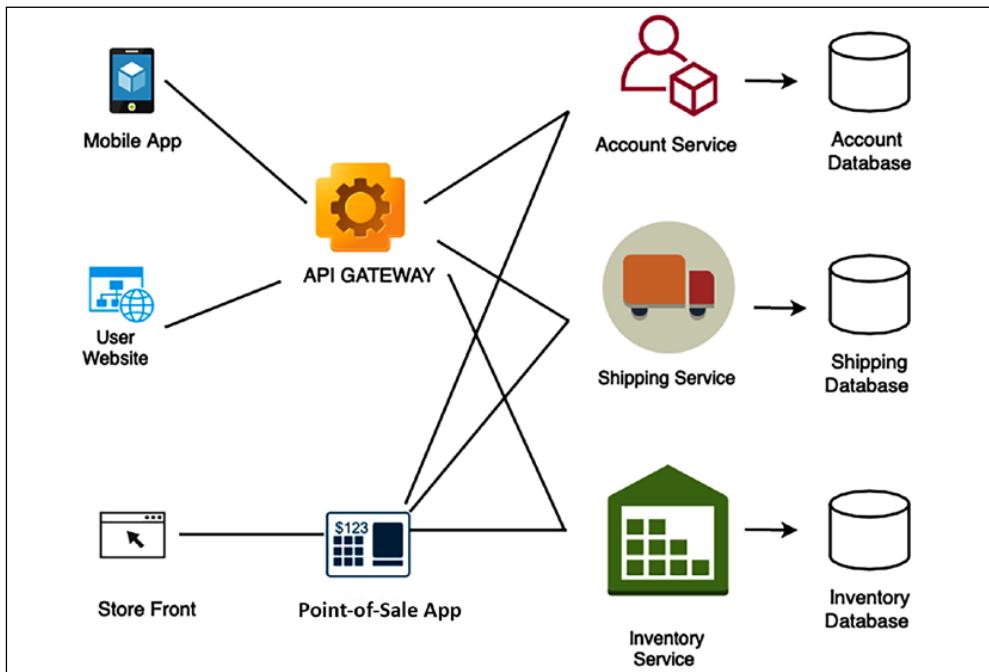


Figure 18.5: Conceptual architecture diagram of an e-commerce platform

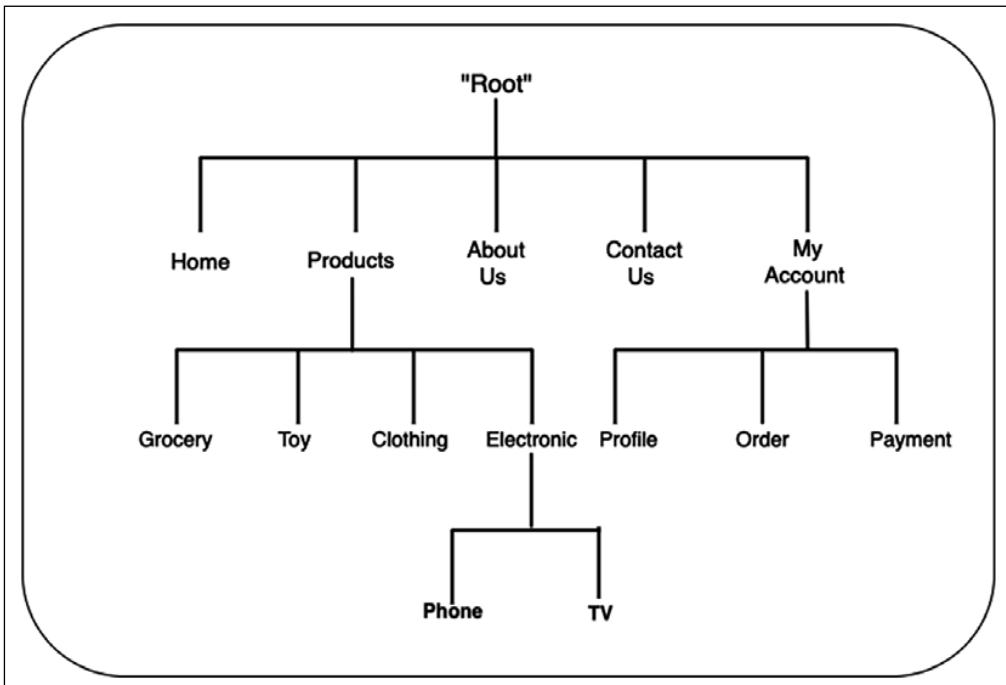


Figure 18.6: Informational architecture diagram of an e-commerce platform

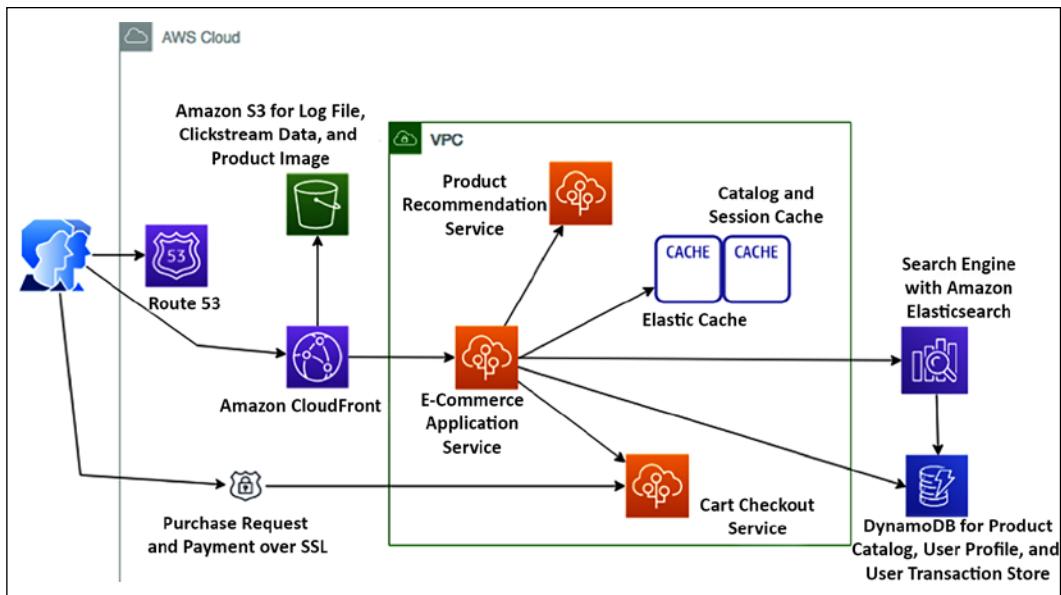


Figure 18.7: Application architecture diagram of an e-commerce platform

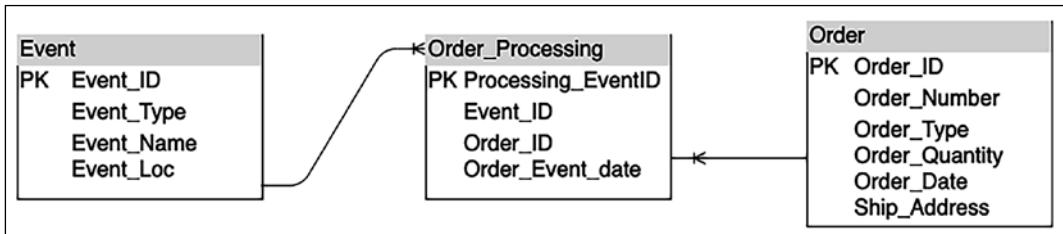


Figure 18.8: ERD of an e-commerce platform

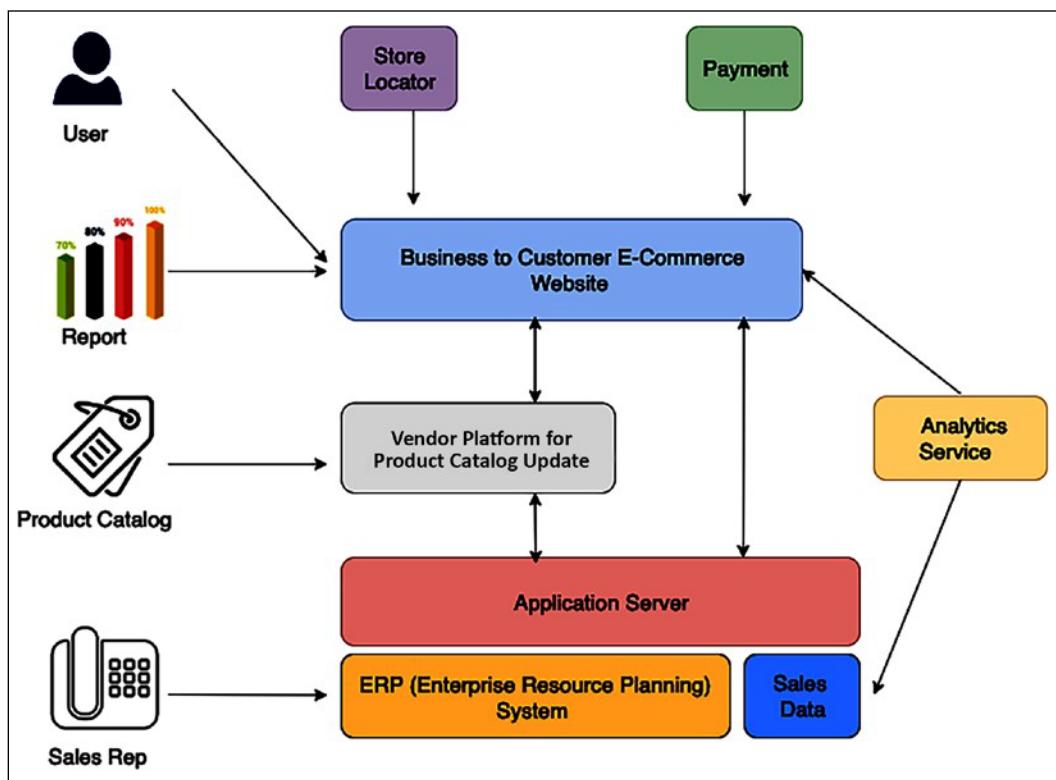


Figure 18.9: Integration architecture diagram of an e-commerce platform

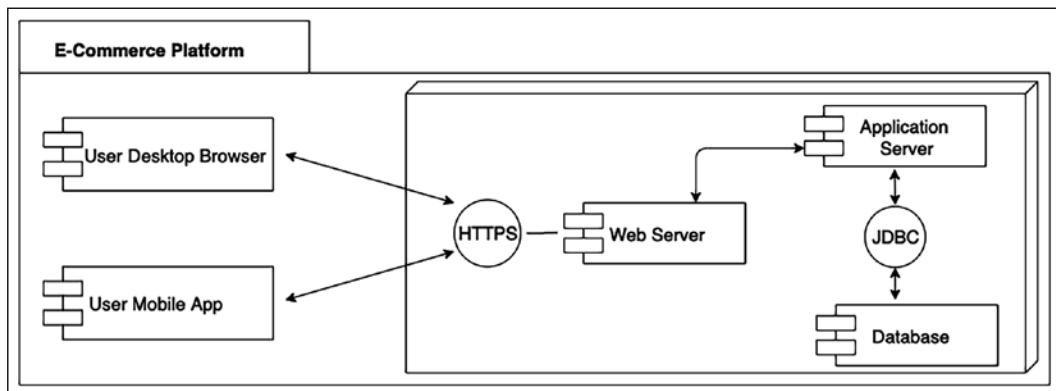


Figure 18.10: Deployment diagram of an e-commerce platform

Chapter 19

Learning Soft Skills to Become a Better Solution Architect

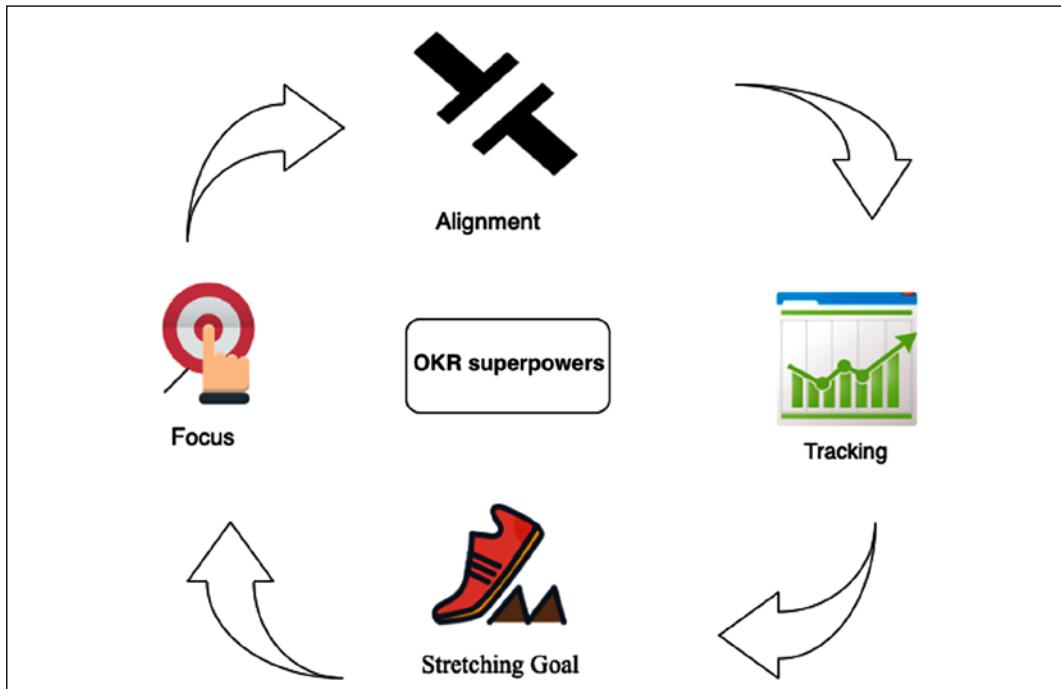


Figure 19.1: Superpowers of OKRs

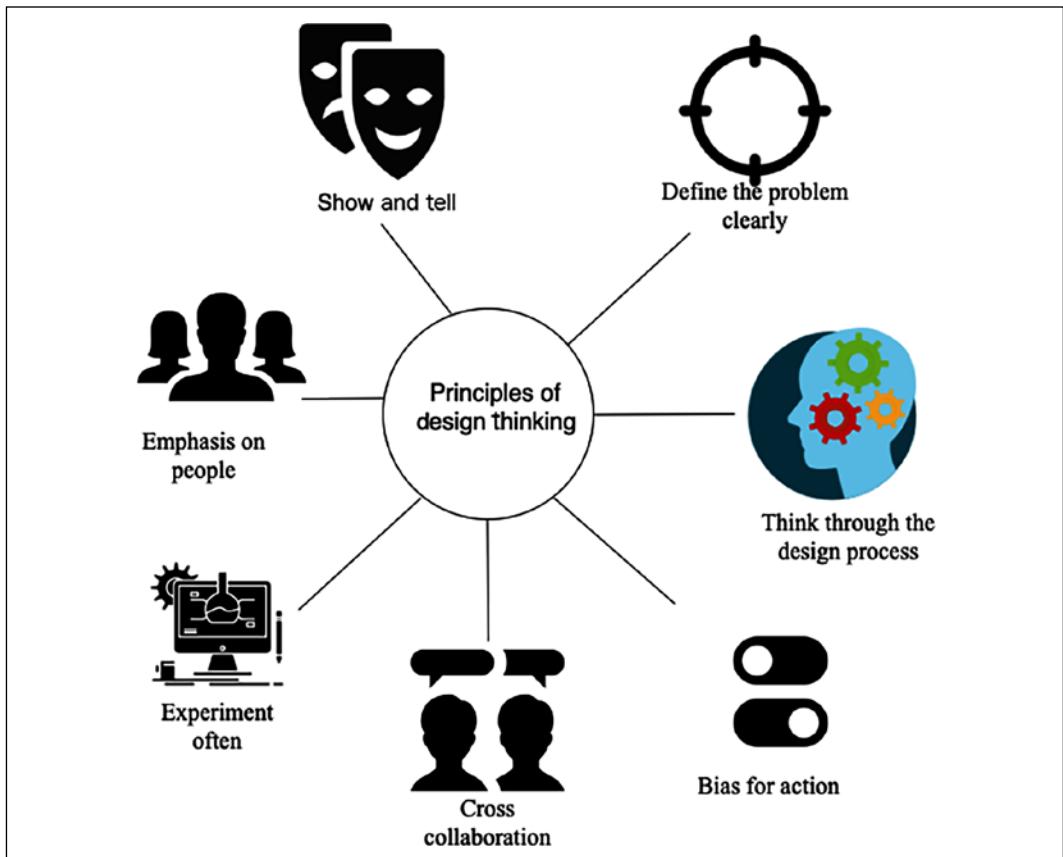


Figure 19.2: Principles of design thinking

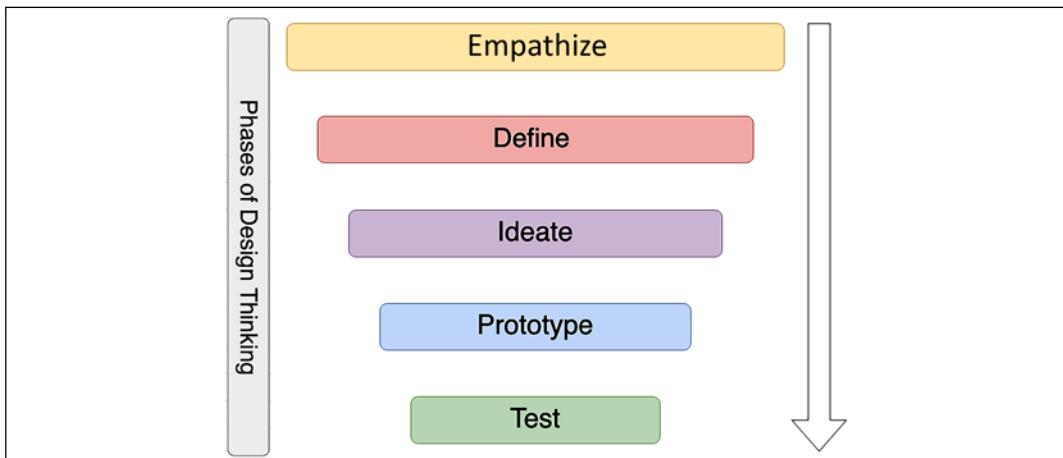


Figure 19.3: Five phases of design thinking