Lecture Week 10: Introduction to Serverless Computing

Dr. Hamed Hamzeh

Outline

- Introduction to Serverless Computing
- Benefits of Serverless Computing
- Serverless Architecture
- Serverless Providers
- Use Cases
- Challenges and Limitations
- Future Trends

Introduction to Serverless Computing

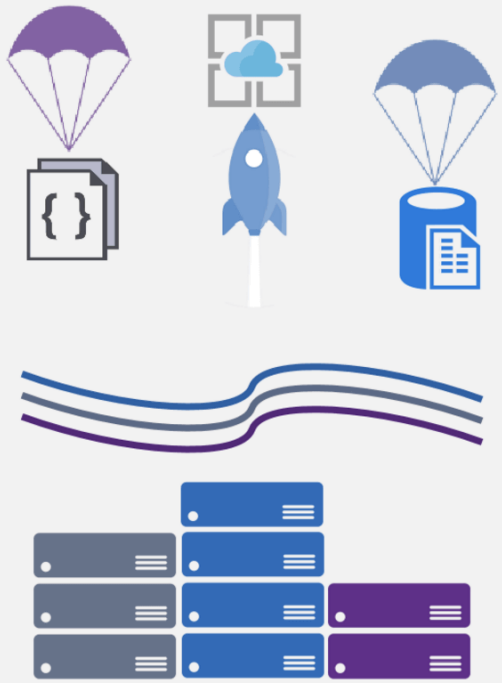
Definition of Serverless Computing

Serverless computing is a cloud computing execution model where the cloud provider dynamically manages the allocation of machine resources.

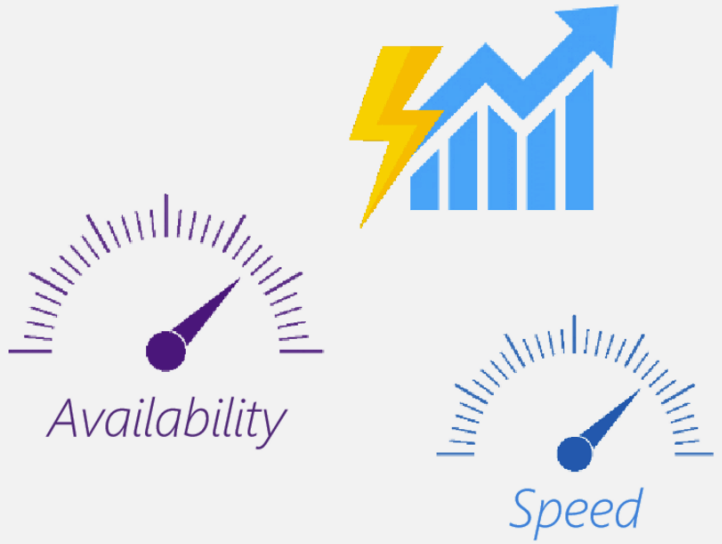
Applications are broken down into smaller, event-driven functions that are executed in response to specific triggers or events.

No server management, auto-scaling, pay-per-use pricing model.

Serverless computing

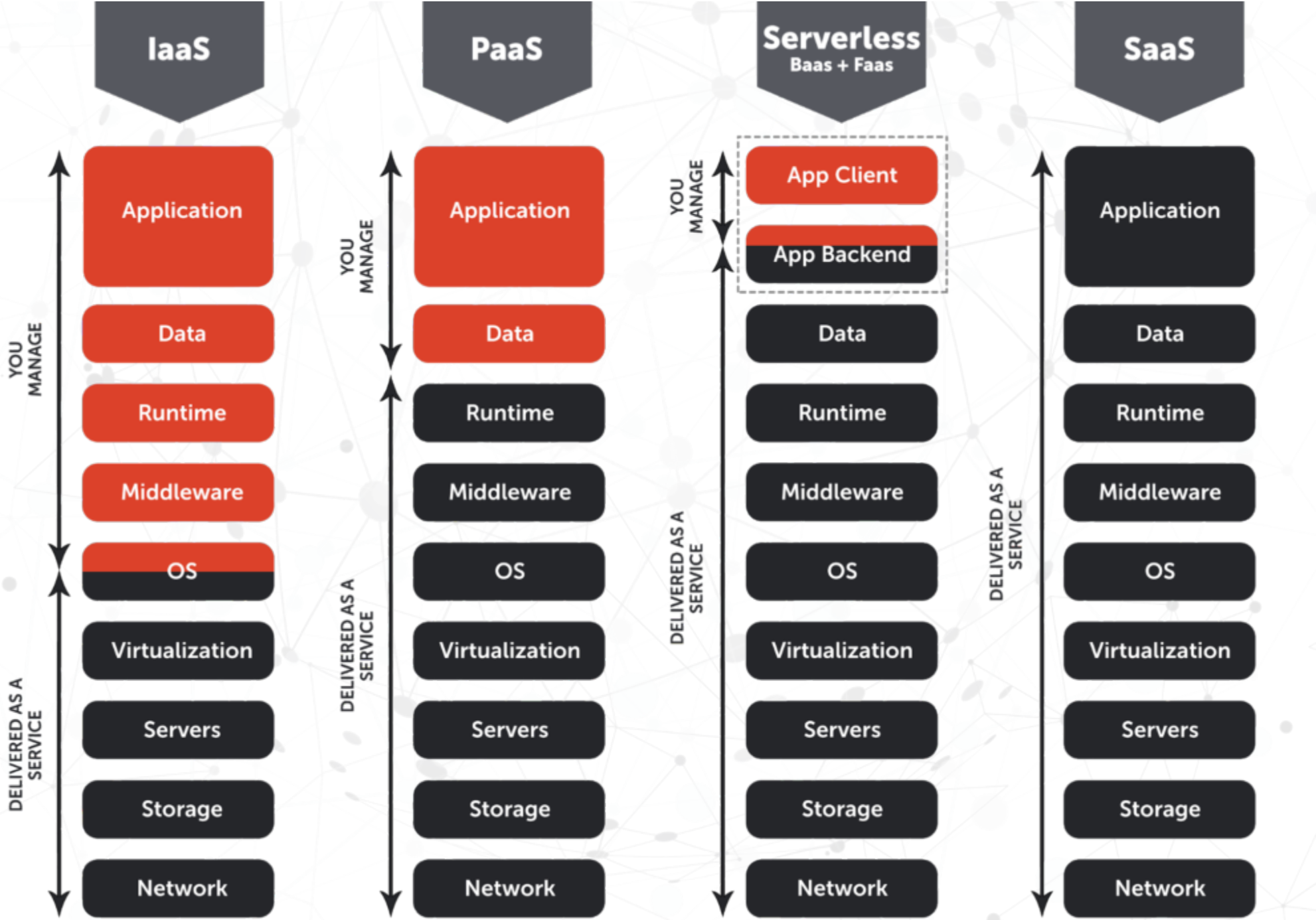


Servers are
fully-abstracted



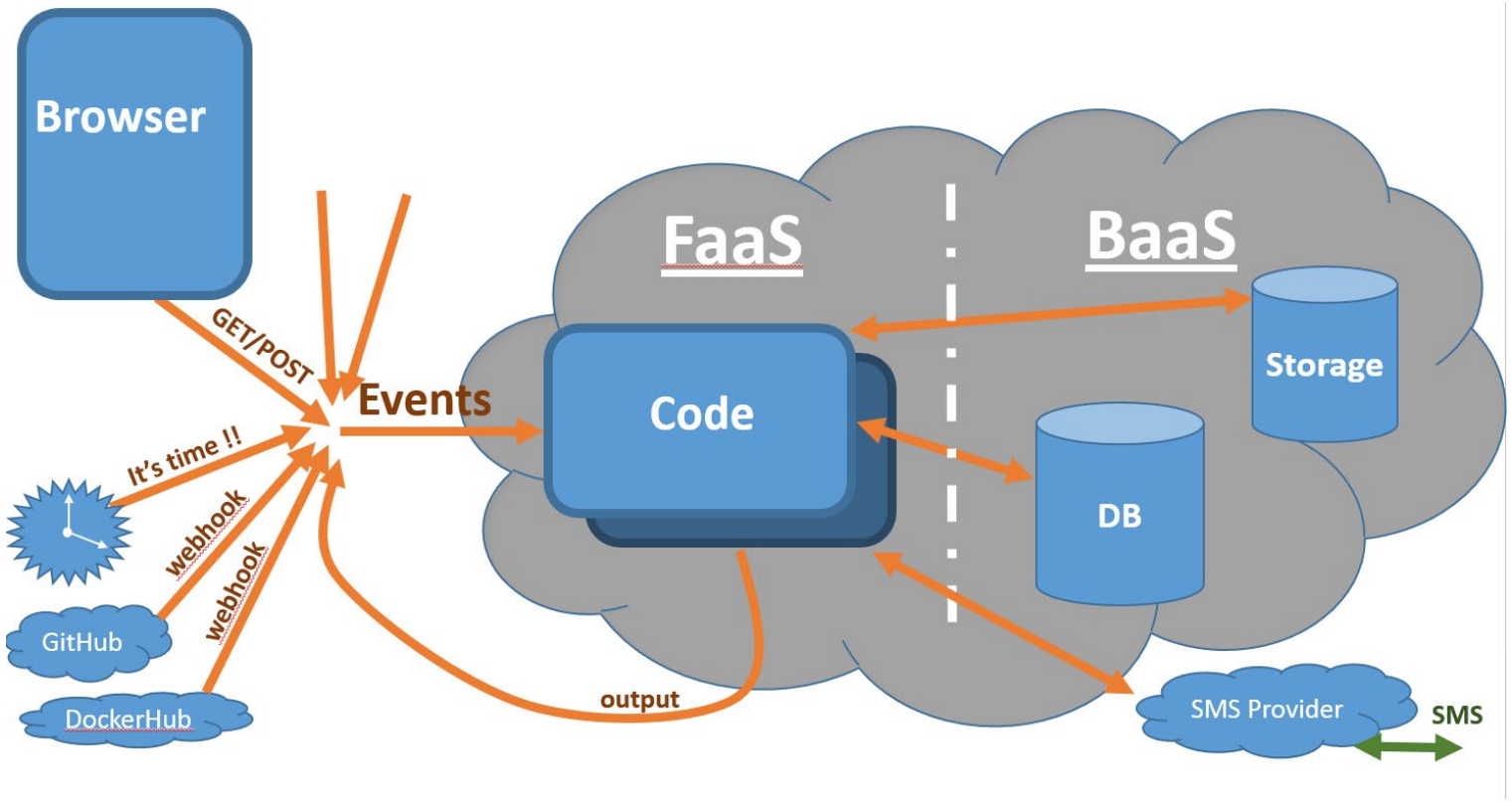
Scaling is event-driven
not resource-driven

Pay only
for what you use


IaaS
PaaS
Serverless
Baas + Faas
SaaS
YOU MANAGE
DELIVERED AS A SERVICE
Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Network
App Client
App Backend

Serverless Architecture

Events and Functions

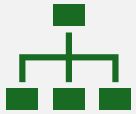


Stateless Functions

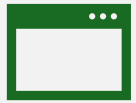
Stateless functions are the building blocks of serverless applications. They are ephemeral and do not maintain state between invocations.

Each function is designed to perform a specific task or handle a single event, promoting modularity and scalability.

Microservices and Serverless Architecture

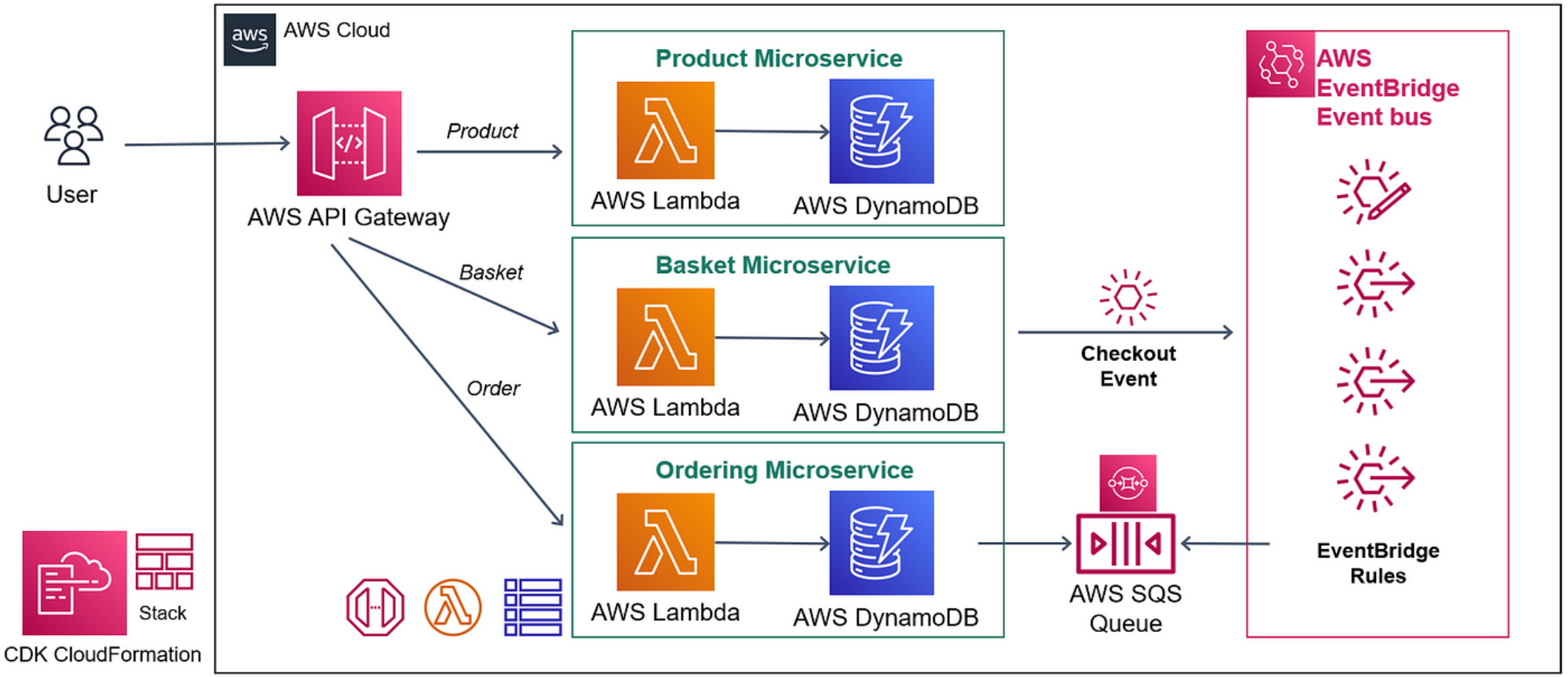


Serverless architecture aligns well with the microservices architectural style, where applications are composed of small, independent services.



Each function in a serverless application can be considered a microservice, offering flexibility, scalability, and ease of maintenance.

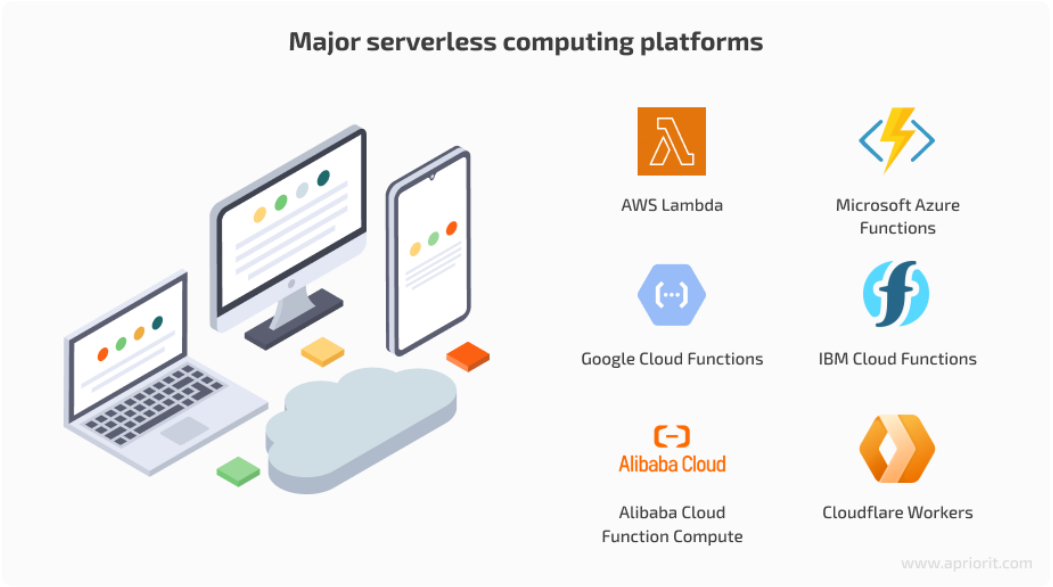
Example


AWS Cloud
User
AWS API Gateway
Product
Basket
Order
Product Microservice
AWS Lambda
AWS DynamoDB
Basket Microservice
AWS Lambda
AWS DynamoDB
Ordering Microservice
AWS Lambda
AWS DynamoDB
Checkout Event
AWS SQS Queue
AWS EventBridge Event bus
EventBridge Rules
Stack
CDK CloudFormation

Serverless Providers

Introduction to Major Serverless Providers

- Major cloud providers offer serverless computing services that enable developers to build, deploy, and scale applications without managing servers.
- These providers offer a range of features, integrations, and pricing models tailored to different use cases and requirements.



aws Lambda

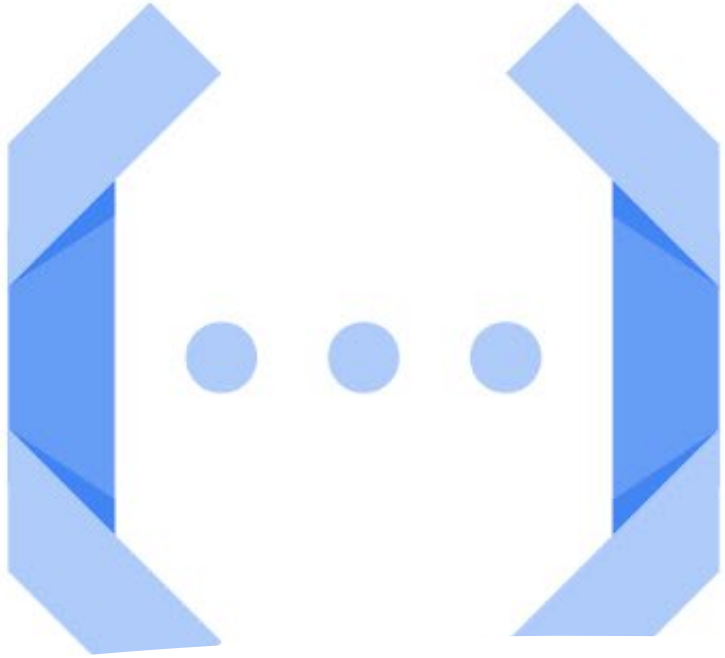
- AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS).
- It allows developers to run code in response to events without provisioning or managing servers.
- Supports multiple programming languages including Node.js, Python, Java, and .NET.
- Integrates seamlessly with other AWS services such as Amazon S3, Amazon DynamoDB, and Amazon API Gateway.
- Offers a pay-per-use pricing model, charging only for the compute time consumed during function execution.


DynamoDB
KINESIS

Azure Functions

- Azure Functions is a serverless compute service offered by Microsoft Azure.
- Enables developers to build and deploy event-driven, scalable applications without managing infrastructure.
- Supports multiple programming languages including C#, JavaScript, Python, and PowerShell.
- Integrates with various Azure services such as Azure Blob Storage, Azure Cosmos DB, and Azure Event Grid.
- Provides flexible pricing options including consumption-based and premium plans.

Azure Functions



Google Cloud Functions

- Google Cloud Functions is a serverless compute service provided by Google Cloud Platform (GCP).
- Allows developers to write and deploy event-driven functions that automatically scale in response to demand.
- Supports programming languages such as Node.js, Python, and Go.
- Integrates seamlessly with other GCP services including Google Cloud Storage, Google Cloud Pub/Sub, and Google Cloud Firestore.
- Offers a pay-as-you-go pricing model, charging only for the resources used during function execution.

Cloud Functions

Event-driven serverless applications

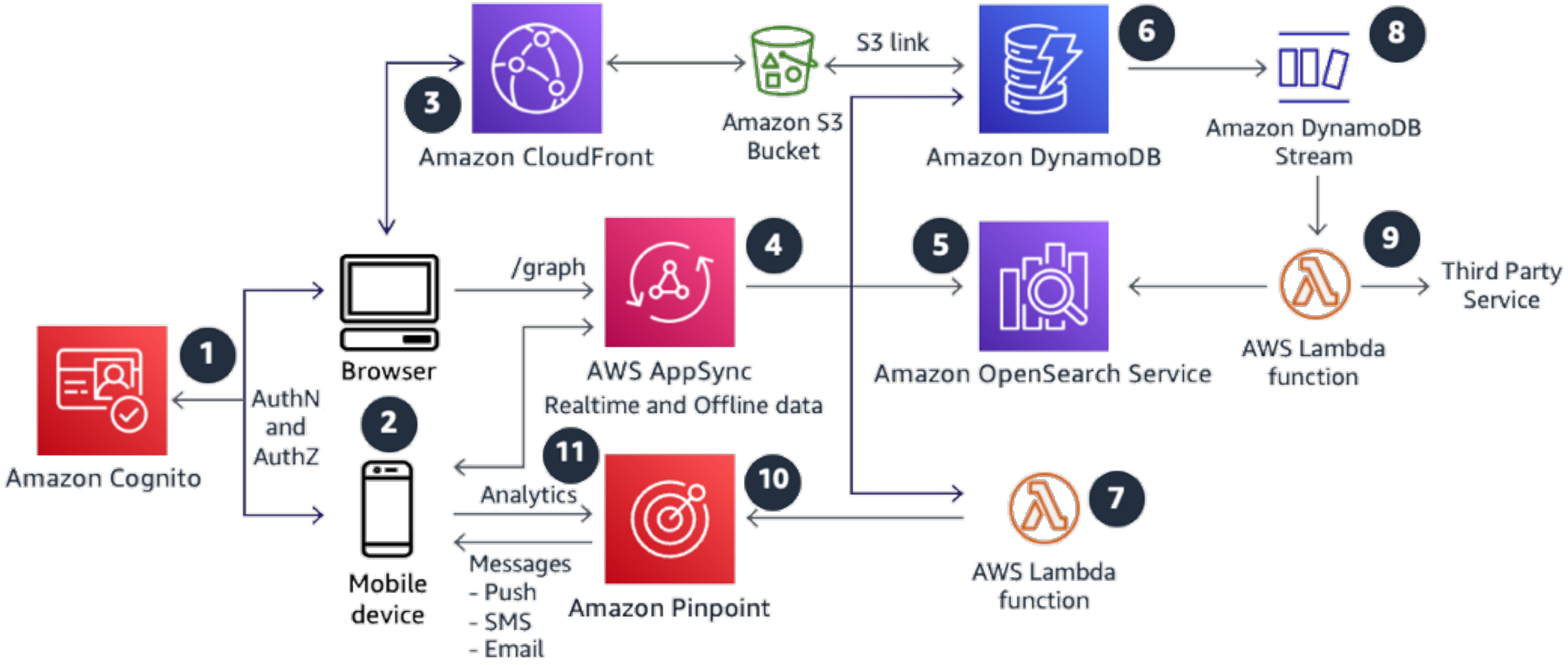
Use cases

Web and Mobile Applications

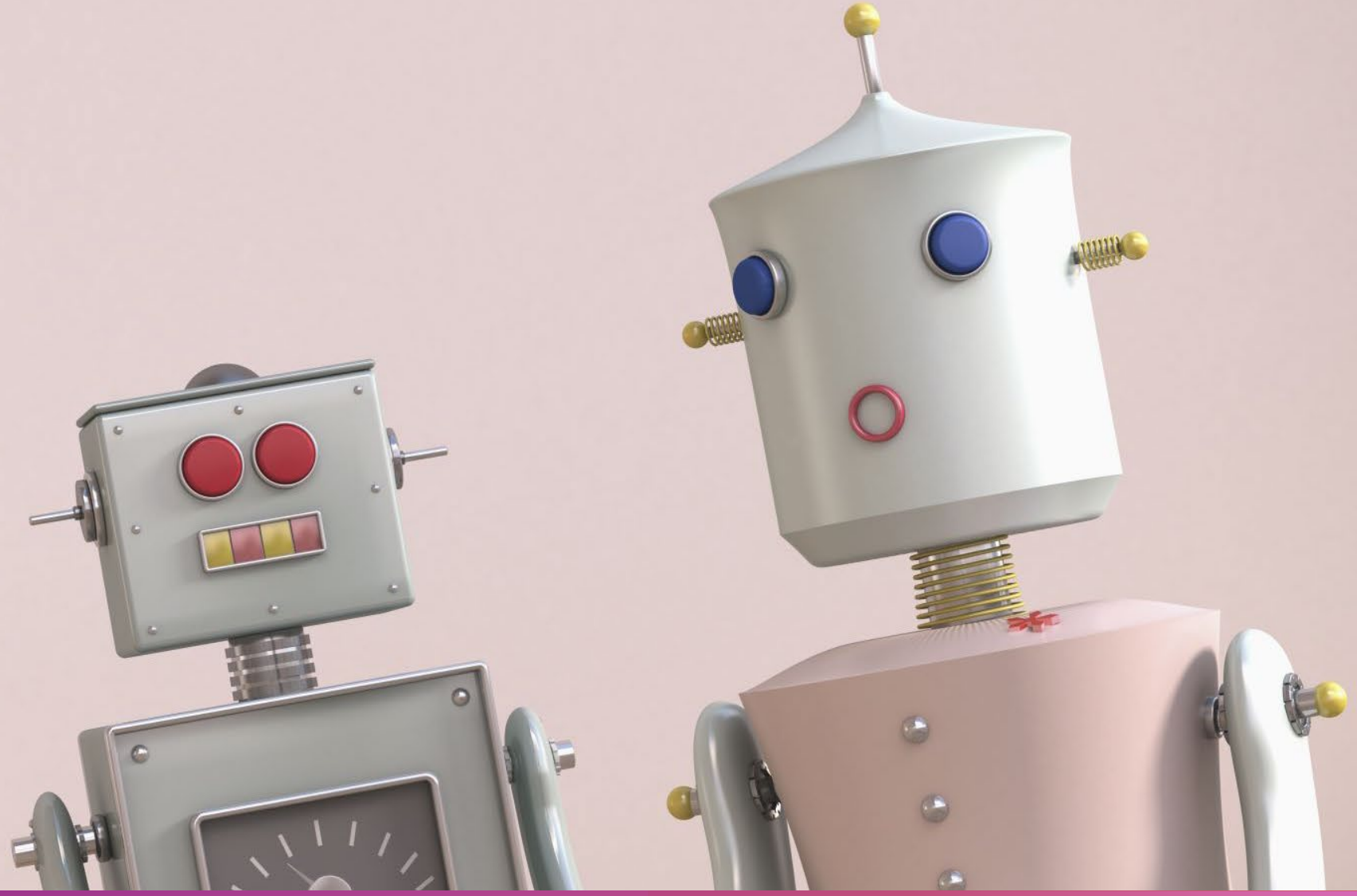
Serverless computing is well-suited for web and mobile applications that require scalable and cost-effective backend services.

Functions can be used to handle HTTP requests, process user data, and interact with databases, providing a seamless user experience.

Serverless architectures enable automatic scaling to accommodate fluctuating user loads and minimize infrastructure costs.


Amazon CloudFront
3
Amazon S3 Bucket
S3 link
Amazon DynamoDB
6
Amazon DynamoDB Stream
8
Browser
/graph
AWS AppSync
Realtime and Offline data
4
5
Amazon OpenSearch Service
AWS Lambda function
9
Third Party Service
1
AuthN and AuthZ
Amazon Cognito
2
Mobile device
11
Analytics
Messages
- Push
- SMS
- Email
Amazon Pinpoint
10
AWS Lambda function
7

Example



Chatbots and Virtual Assistants

Challenges and Limitations

Cold Start Problem

Cold start refers to the latency experienced when a serverless function is invoked for the first time or after a period of inactivity.

It can impact application responsiveness and user experience, especially for latency-sensitive workloads.

Vendor Lock-in

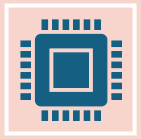
Serverless architectures often rely on proprietary cloud services and APIs, leading to vendor lock-in.

Switching between cloud providers may require significant refactoring and migration efforts, limiting flexibility and portability.

Debugging and Monitoring

- Debugging and monitoring serverless applications can be challenging due to limited visibility into underlying infrastructure.
- Traditional debugging tools may not provide sufficient insights into function execution and performance.

Performance Concerns



Serverless functions are constrained by resource limits and execution environments, which can impact performance for compute-intensive tasks.

Achieving optimal performance requires careful optimization of function code and resource allocation.

Mitigating Challenges

Strategies to Mitigate Cold Start Problem

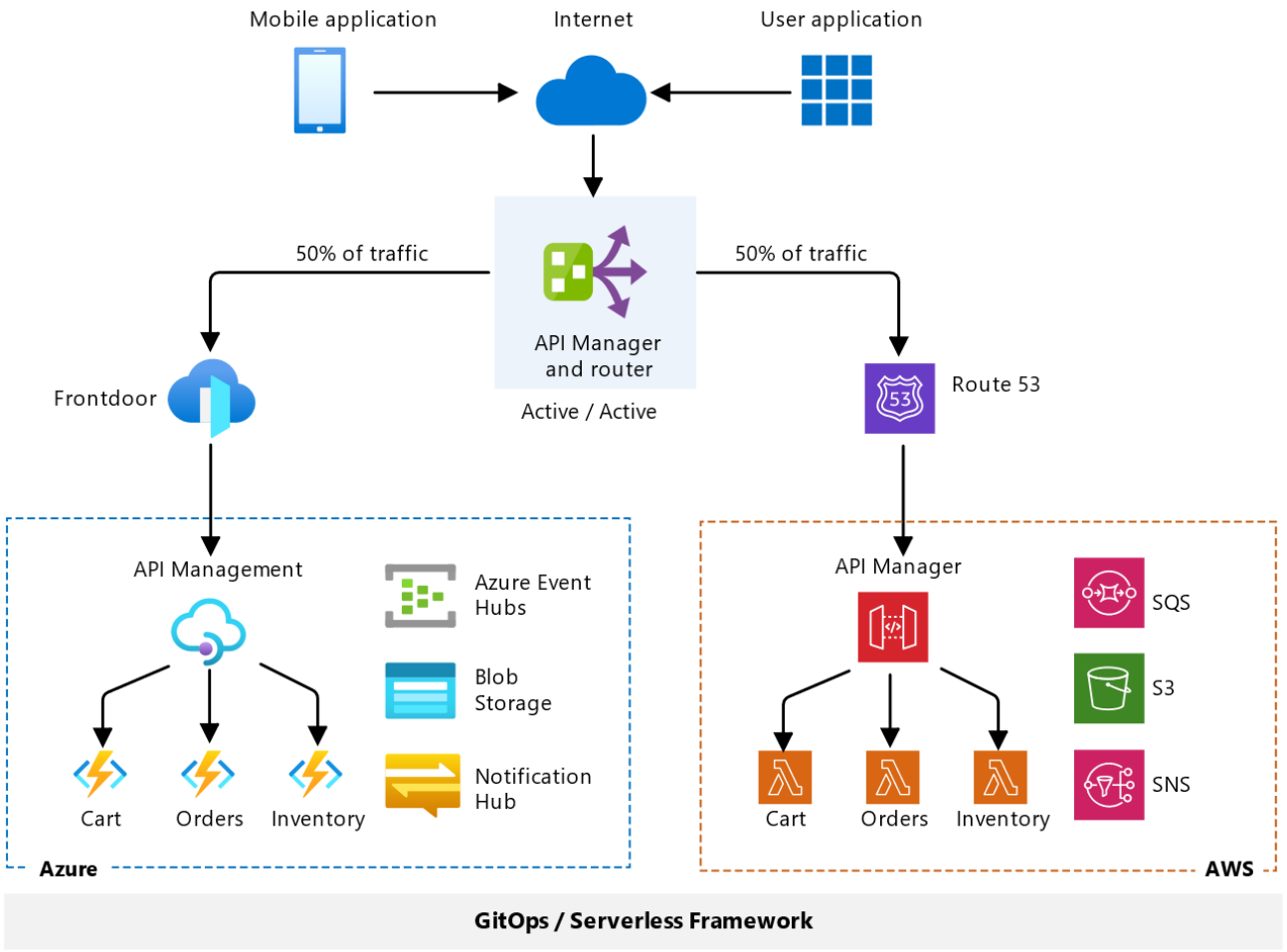
- Implement warm-up techniques such as **periodic function invocations** to keep instances warm and reduce cold start latency.
- **Cache-based solutions** rely on preloading required or commonly used libraries into containers.
- **keep containers alive** for a certain amount of time after function execution is complete

Multi-Cloud Strategy

Adopt a multi-cloud strategy to mitigate vendor lock-in and increase flexibility.

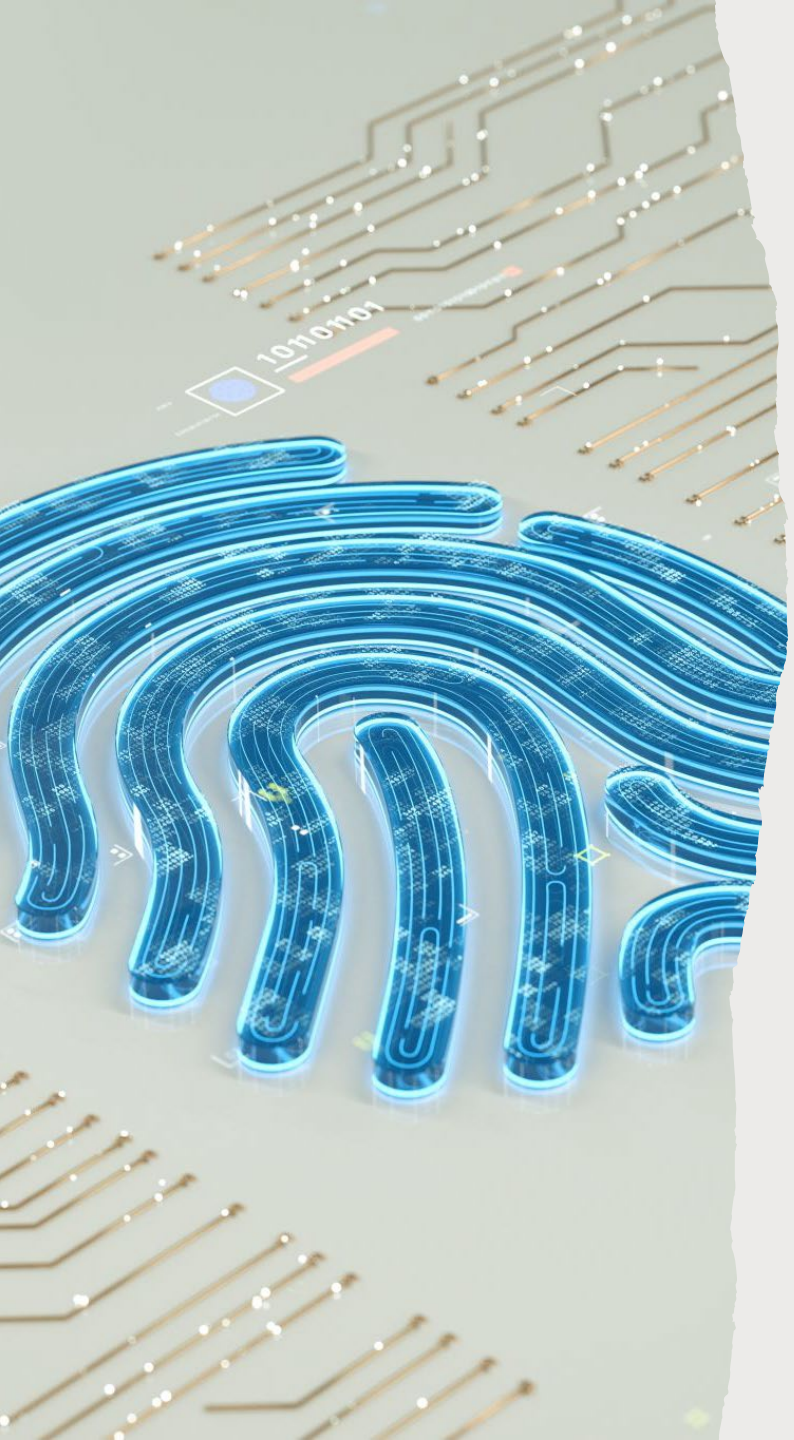
Design applications to be cloud-agnostic by abstracting away provider-specific dependencies and using standard interfaces.

Example architecture



Microsoft Azure

Security Considerations

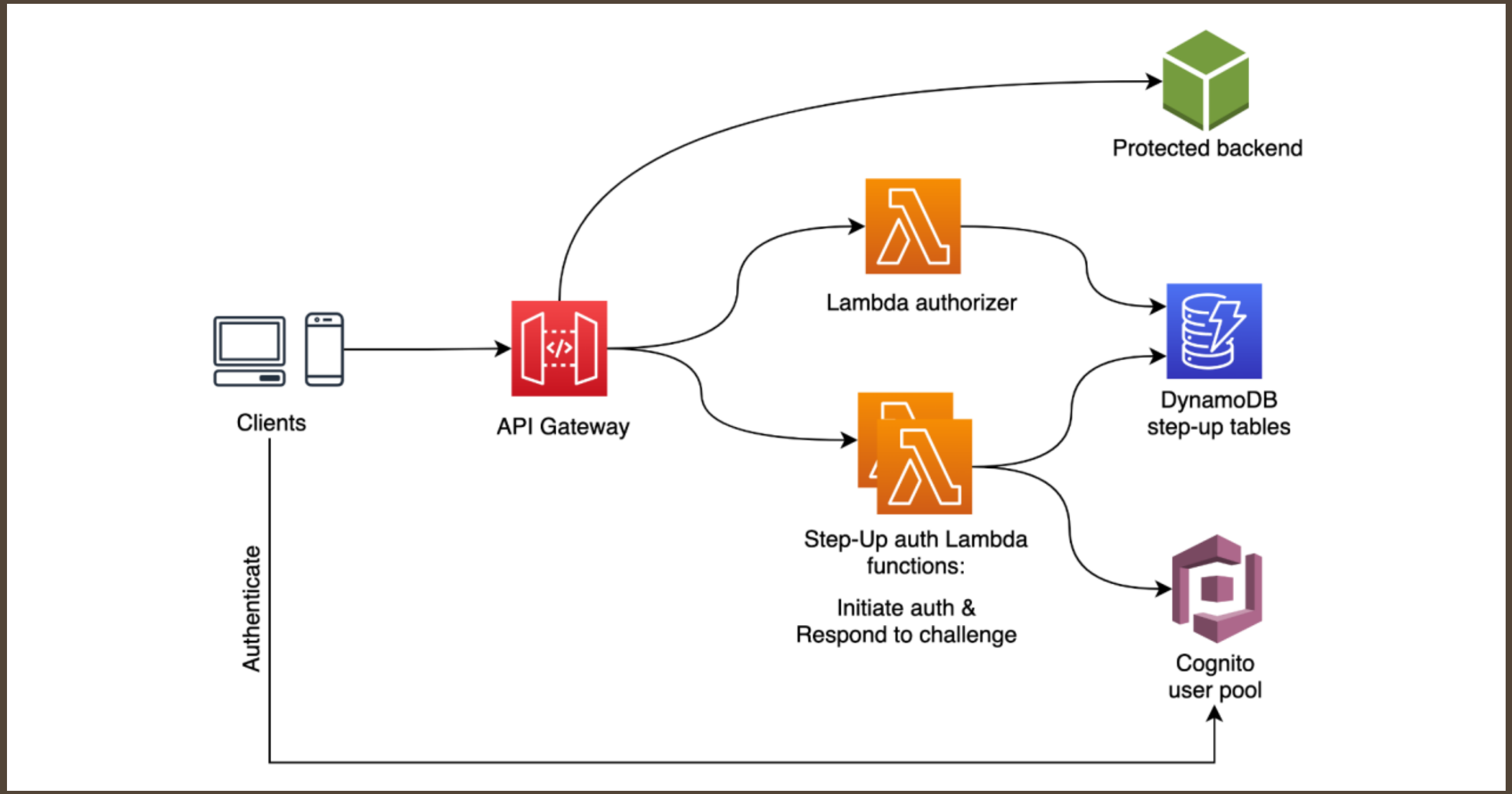


Authentication and Authorization

Authentication ensures that users and services are who they claim to be, typically achieved through mechanisms like tokens, certificates, or biometrics.

Authorization defines what actions users or services are allowed to perform, often managed through role-based access control (RBAC) or attribute-based access control (ABAC).

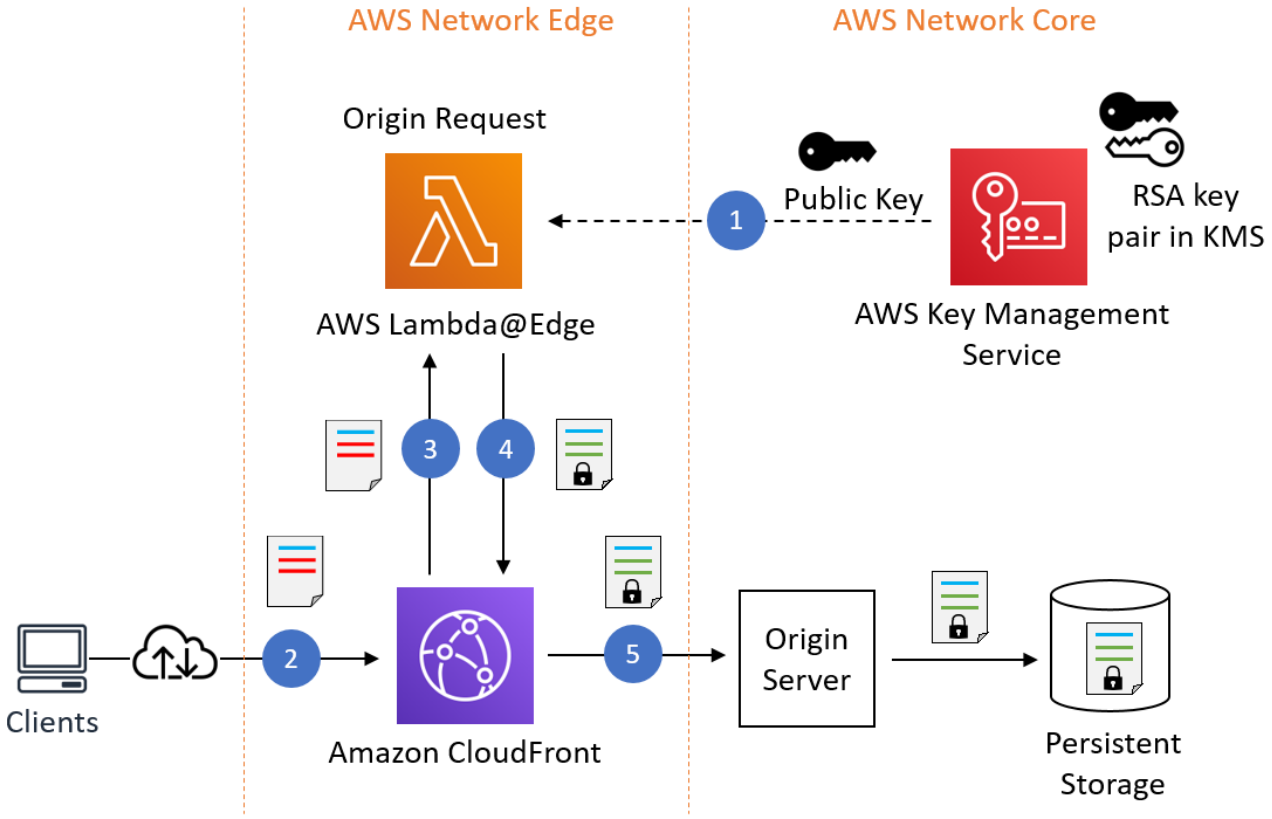
```
// Sample Lambda function for user authentication
exports.handler = async (event) => {
    // Mock authentication logic
    const username = event.username;
    const password = event.password;

    // Check username and password against a database or identity provider
    if (username === 'user@example.com' && password === 'password123') {
        // Authentication successful, generate JWT token
        const token = generateToken(username);
        return { statusCode: 200, body: JSON.stringify({ token }) };
    } else {
        // Authentication failed
        return { statusCode: 401, body: JSON.stringify({ error: 'Invalid credentials' }) };
    }
};

// Mock function to generate JWT token
function generateToken(username) {
    // Implementation to generate JWT token
    return 'mockJWTToken';
}
```


Protected backend
Lambda authorizer
Clients
API Gateway
DynamoDB
step-up tables
Step-Up auth Lambda
functions:
Initiate auth &
Respond to challenge
Cognito
user pool
Authenticate

Data Encryption

Encrypting data at rest and in transit helps protect sensitive information from unauthorized access or interception.

Techniques such as encryption keys, secure sockets layer (SSL)/transport layer security (TLS), and data masking can be employed to safeguard data integrity and confidentiality.



Legend:

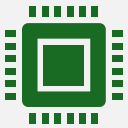
API payload with sensitive data fields in plaintext

API payload with sensitive data fields in ciphertext

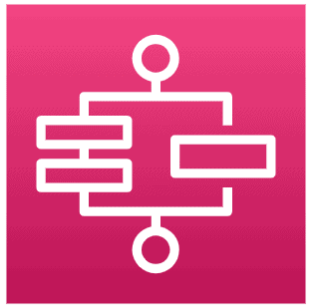

Orchestration

Orchestration Tools

Orchestration tools coordinate the execution of multiple serverless functions and services to accomplish complex tasks or workflows.

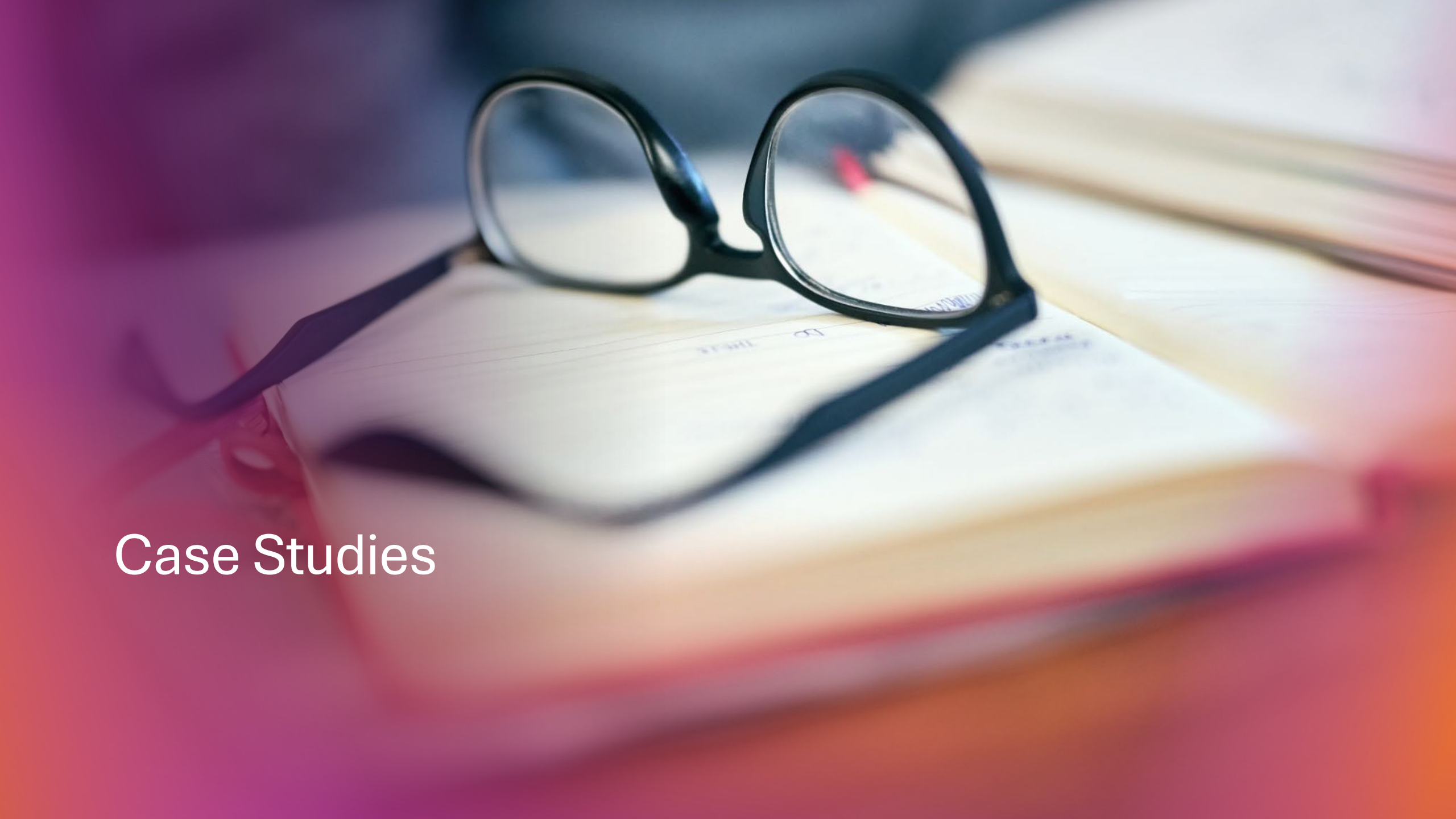


Workflow orchestration platforms like AWS Step Functions, Azure Logic Apps, and Google Cloud Composer provide visual interfaces and state management capabilities to automate business processes and application workflows.



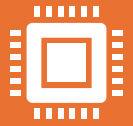
AWS Step Functions

```
{
  "Comment": "Process Order Workflow",
  "StartAt": "ValidateOrder",
  "States": {
    "ValidateOrder": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:ValidateOrderFunction",
      "Next": "CheckInventory"
    },
    "CheckInventory": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:CheckInventoryFunction",
      "Next": "ChargeCustomer"
    },
    "ChargeCustomer": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:ChargeCustomerFunction",
      "Next": "NotifyWarehouse"
    },
    "NotifyWarehouse": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:NotifyWarehouseFunction",
      "Next": "SendConfirmationEmail"
    },
    "SendConfirmationEmail": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:SendConfirmationEmailFunction",
      "End": true
    }
  }
}
```



Case Studies

How Netflix Uses Serverless Computing



Netflix leverages serverless computing for various use cases, including content delivery, recommendation systems, and data processing.

Serverless functions are used to handle traffic spikes during peak hours, optimize content delivery, and personalize user experiences.

Netflix utilizes AWS Lambda, AWS Step Functions, and other serverless services to build scalable, resilient, and cost-effective solutions.