Designing Cloud Solutions

Chathra Serasinghe

Define the Business Case

Before designing any cloud solution, clearly understand *why* the business is moving to the cloud.

Key Questions

- What business problems are we trying to solve?
- Why invest in new cloud infrastructure?
- How does the cloud add measurable business value?

Business Value Drivers

- Improved business agility
- Faster time-to-market
- Increased scalability and availability
- Improved reliability and resilience

Cost & ROI Considerations

- Initial investment
- Migration and modernization costs
- People costs (cloud engineers, consultants)
- Training and upskilling costs

Expected Returns

- Reduced operational and maintenance costs (e.g., data center, hardware)
- Elastic scaling based on demand
- Higher system availability

Define the requirement

A successful cloud design starts with clear technical and operational requirements.

Requirement Categories

- Compute
- Storage
- Database
- Networking
- Security
- Governance
- Performance
- Cloud Operations

Compute

Key Considerations

- Required CPU and memory capacity
- Operating system selection
- Scalability requirements
- High availability and fault tolerance
- Disaster recovery strategy

Design Goals

- Right-size resources to avoid over-provisioning
- Enable horizontal and vertical scaling
- Design for resilience

Storage

Capacity & Growth

- Initial storage size
- Data growth rate over time

Storage Models

- Block storage
- Object storage
- File storage

Performance & Access

- IOPS and throughput requirements
- Access methods (NFS, SMB, iSCSI)

Data Protection

- Backup and recovery
- Disaster recovery
- Encryption and access control

Networking

Bandwidth & Latency

- Application bandwidth needs
- Latency-sensitive workloads

Architecture Decisions

- Single-region vs multi-region deployment
- Edge computing for global users

Network Security

- Firewalls and security groups
- Inbound and outbound traffic control
- Secure connectivity (VPN, Direct Connect, ExpressRoute)

Database

Database Models

- Relational (SQL)
- NoSQL (Key-value, Document, Graph)

Key Design Factors

- Data volume
- Transaction speed and latency
- Scaling requirements
- Monitoring and management

Database Options

- Cloud-native databases (e.g., Aurora)
- Provider-managed databases (e.g., DynamoDB)
- Third-party managed databases(E.g: MongoDB Atlas)

Resilience & DR

- Backup and restore
- Active–Passive or Active–Active architectures

Security

Security Design Principles

- Shared responsibility model
- Defense in depth
- Least privilege access

Data Protection

- Encryption at rest (e.g., KMS)
- Encryption in transit (HTTPS, TLS)

Security Controls

- Identity and Access Management (IAM)
- Network security and firewalls
- Auditing and logging
- Vulnerability management
- API and endpoint security

Business Continuity

- Disaster recovery planning
- High availability architectures

Proactive Security

Continuous Monitoring

- Network, compute, storage, and application metrics

Security Intelligence

- Analyze logs and telemetry
- Detect abnormal patterns

Automated Response

- Proactively block suspicious activity
- Reduce incident response time

Governance - Requirements

Cost Governance

- Track usage by team or application
- Enforce budgets and spending limits

Resource Governance

- Resource tagging
- Standardized provisioning

API & Service Governance

- Access control
- Policy enforcement

Performance Considerations

Application-Level Factors

- Code structure and efficiency
- Runtime
- Threading

Infrastructure Factors

- Network bandwidth and latency
- Region and availability zone placement
- Database response times

Cloud Operations

Operational Requirements

- Monitoring and observability
- Maintenance and patching

Self-Healing Systems

- Automatic failure detection
- Auto-recovery without human intervention

Automation Goals

- Minimize manual operations
- Improve system reliability

Modern Cloud Computing Architectures and Related Concepts

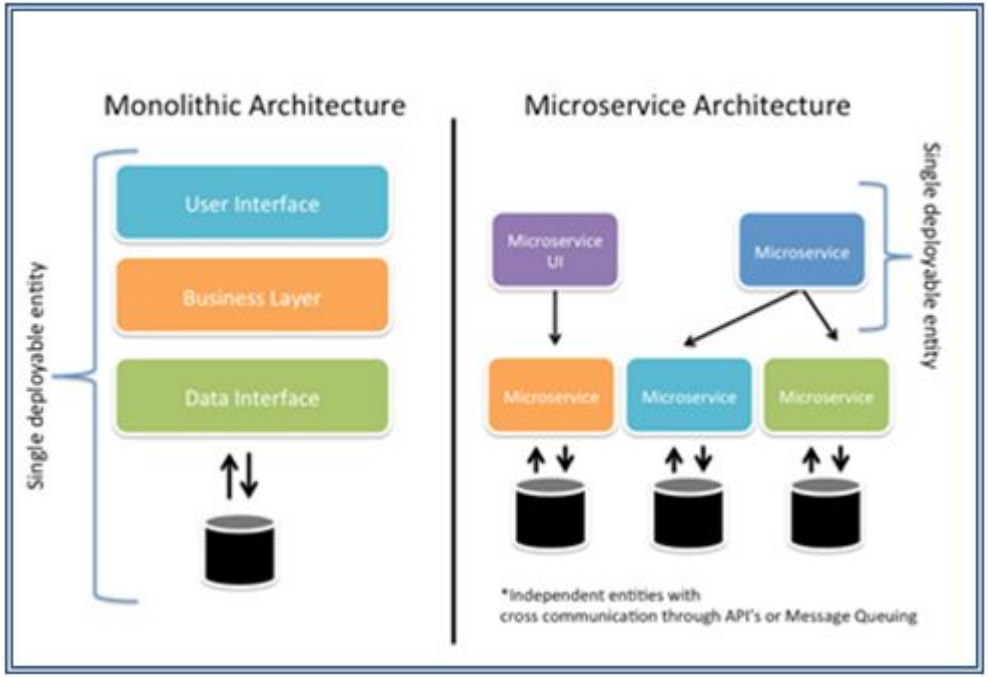
Microservices

Microservices

- Small, independent services
- Communicate via APIs
- Owned by small, autonomous teams

Benefits

- Improved scalability
- Faster development cycles
- Accelerated innovation

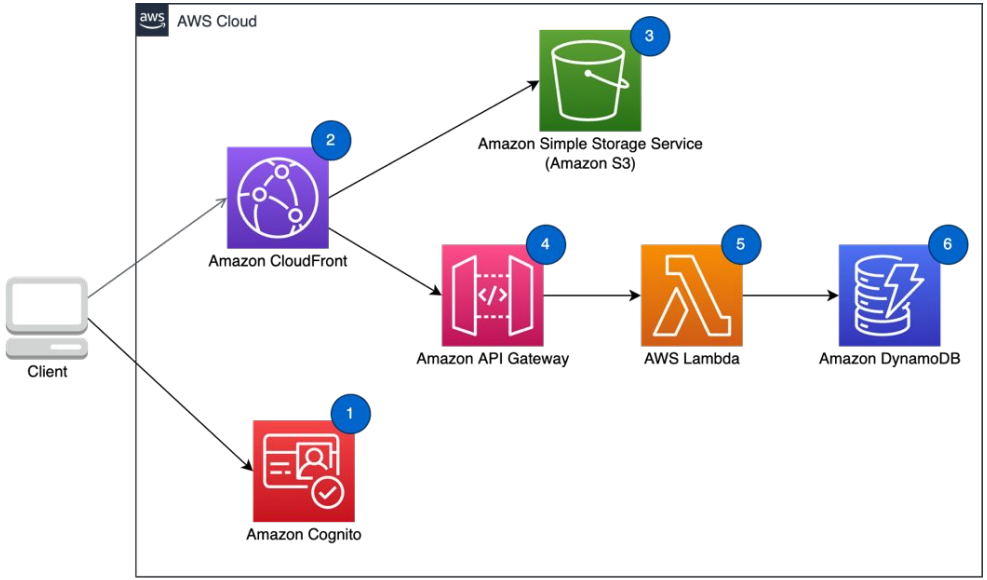


Serverless

- No server management required
- Pay only for execution time
- Automatic scaling

Key Advantages

- Reduced operational overhead
- Cost-efficient for event-driven workloads



Containers and Container Orchestration

Containerization is a software deployment process that bundles an application’s code with all the files and libraries it needs to run on any infrastructure.

- Isolation
- Portability
- Resource Efficiency

Eg:- Docker

Container orchestration is the automated management of the lifecycle of containers. As applications grow to span multiple containers deployed across multiple servers, orchestration becomes essential.

- Provisioning and Deployment
- Load Balancing and Distribution
- Scaling
- Resource Allocation

Eg:- EKS, ECS

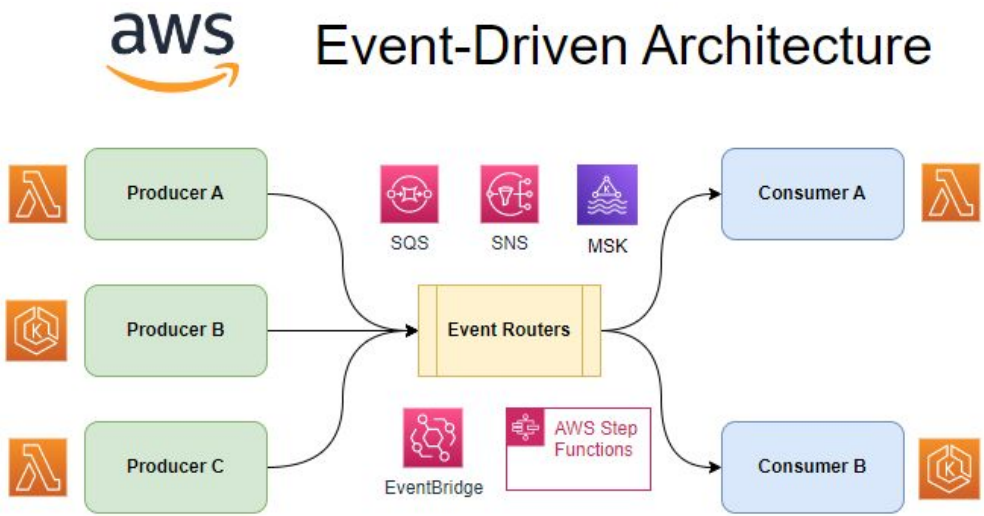
Event Driven Architecture

Core Components

- Events
- Producers
- Consumers
- Event channels

Benefits

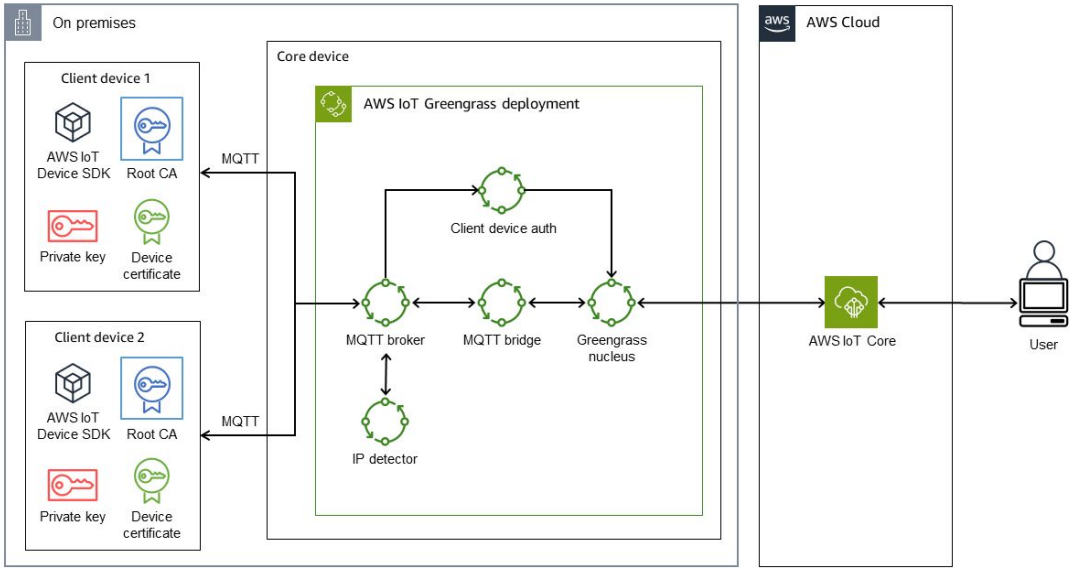
- Loose coupling
- High scalability
- Real-time processing
- Fault tolerance



Edge Computing

- Process data closer to users
- Reduce latency and bandwidth usage
- Enable real-time applications

e.g:-



AI and ML

Compute Needs

- GPU and accelerated instances

Pre-Built AI Models:

- Using pre-built and customizable models provided by cloud services for common tasks like image recognition, natural language processing

Data Collection and Storage:

- Secure and scalable data storage
- Efficient data pipelines

Constructing Cloud Infrastructure

Infrastructure as Code (IaC)

- Terraform
- CloudFormation
- CDK
- SAM
- Serverless Framework

Other Methods

- CLI
- Management Console (manual – not recommended for scale)