Backend

Client-Server Architecture

BACKEND IS:

System architecture

Servers

Database

Data analysis

Security

Frameworks

APIs

Scalability

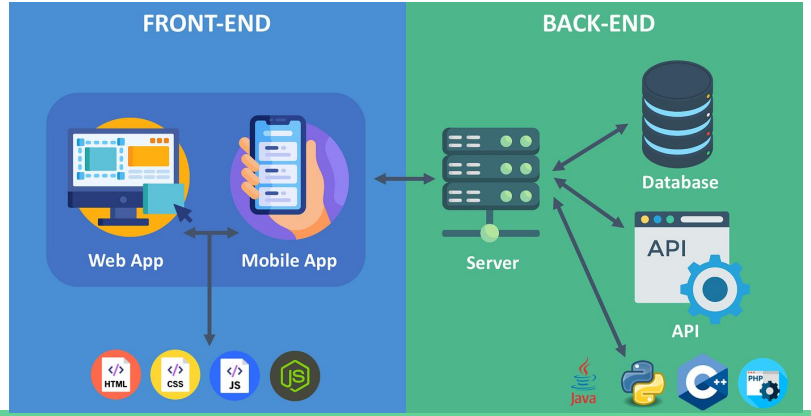
Operating system

Business logic

Frontend vs Backend

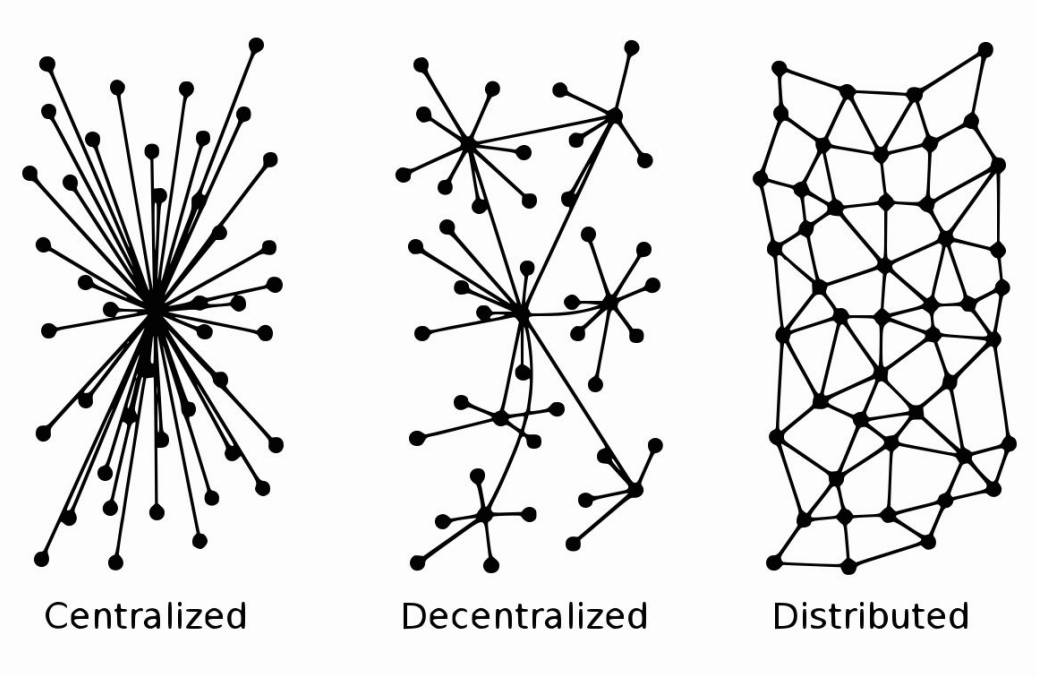
- Displaying data and handling user events
- User experience, design, and interactivity
- Visible for the end-user
- Runs in the user's device

- Data processing, business logic and database interactions
- Functionality, security, and performance
- Not visible to the end-user
- Runs on the server



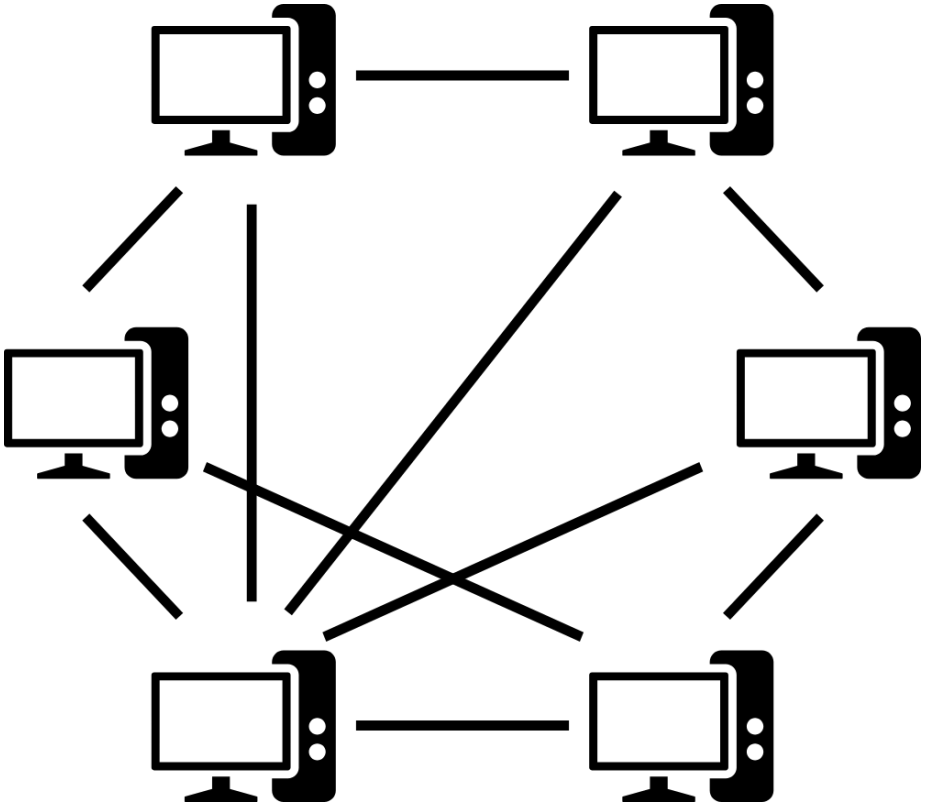
Distributed Computing Systems

- Multiple systems are connected through a network to solve a common problem
- Architecture types:
  - Client-Server
  - Peer-to-Peer (P2P)
  - Grid Computing
  - Cloud Computing



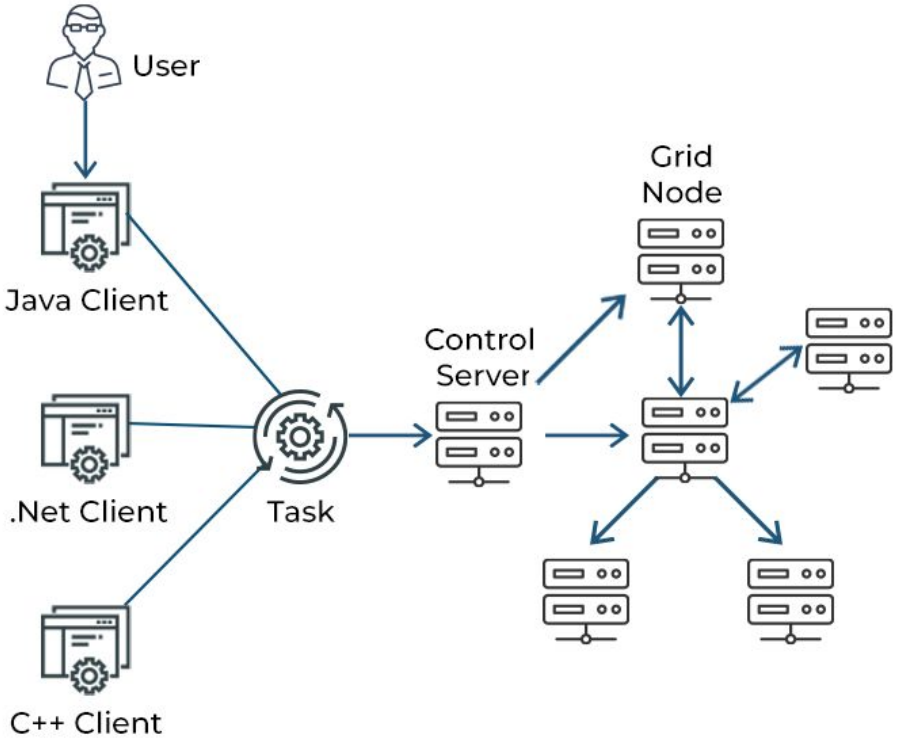
Peer-to-Peer Architecture

- Decentralized distributed computing architecture
- Each node (peer) is a client and a server
- Works without a central coordination
- Horizontal scaling by adding more peers
- Applications:
  - File Sharing (BitTorrent)
  - Voice over IP (Skype)
  - Distributed computing
  - Cryptocurrencies (Bitcoin)



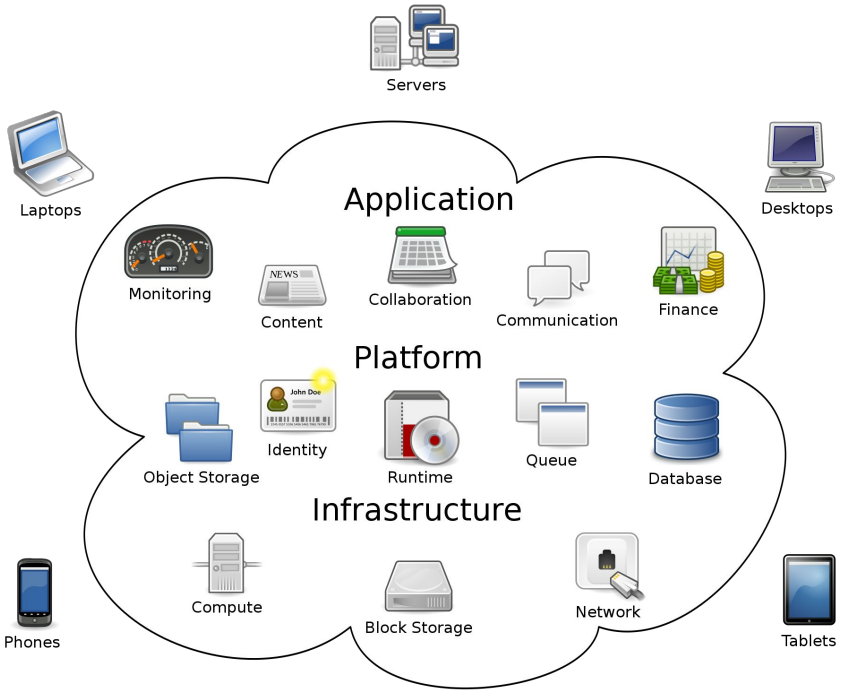
Grid Computing

- Uses distributed computing resources to reach a common goal
- Heterogeneity (variability of components)
- Layer of abstraction hides complexity
- Huge parallel processing potential
- Dynamic resource management
- Applications:
  - Scientific Research (modeling)
  - Engineering (simulations)
  - Commercial Applications (data analysis)



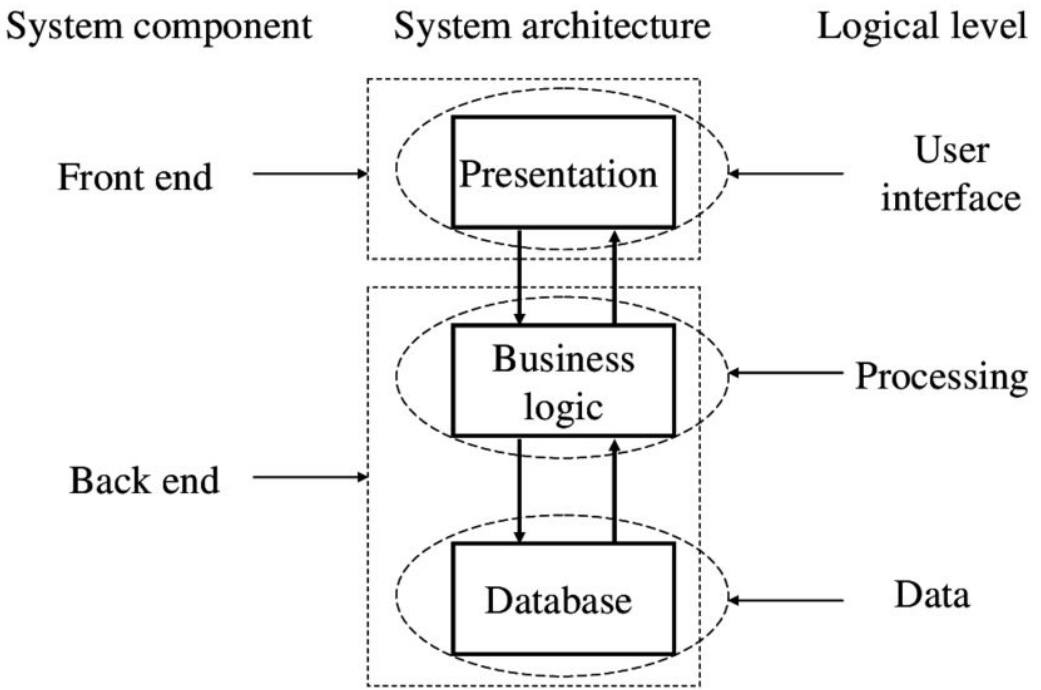
Cloud Computing

- Model for delivering servers, storage, databases, networking, software, etc. over the internet
- Resources are broadly accessed
- Resources are pooled and shared
- Resources are scaled on demand
- Resources are monitored, controlled, and reported
- Types:
  - Software as a Service
  - Platform as a Service
  - Infrastructure as a Service



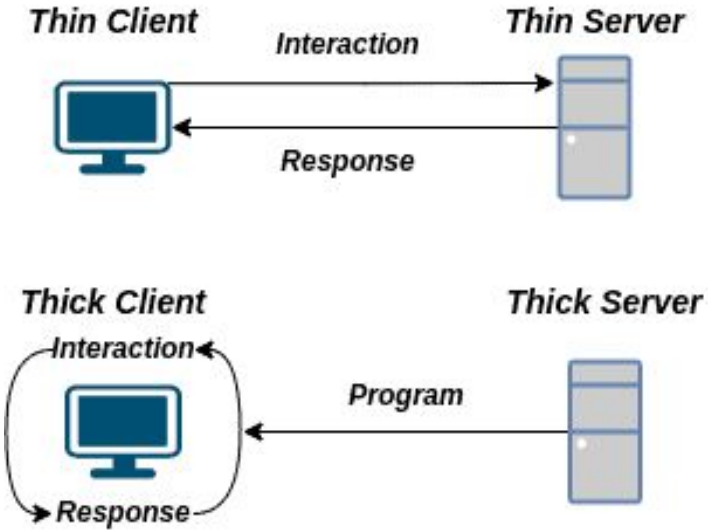
Client-Server Architecture

- Application is divided into client and server
- Client and server communicate over a network using protocols
- Multiple clients can access shared resources on the server



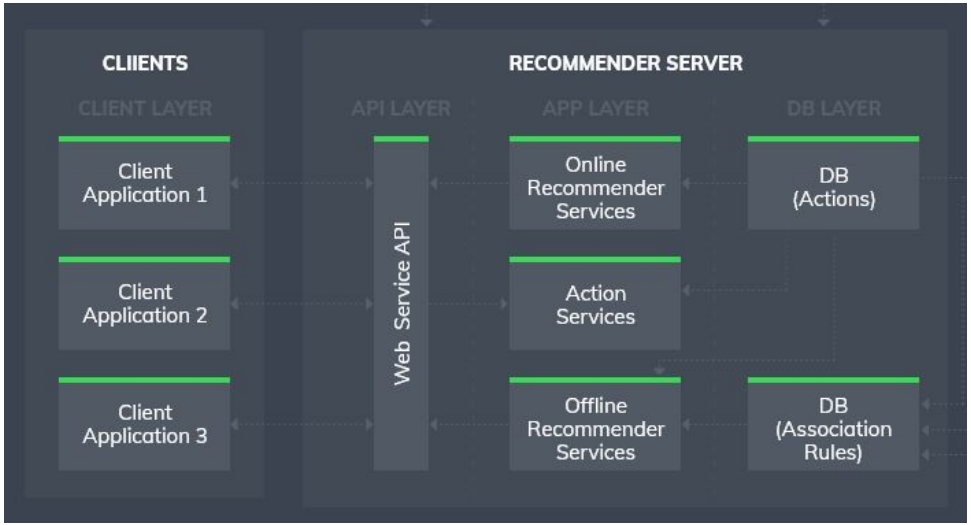
Client (Presentation Layer)

- Provides user interface
- Handles user input
- Displays data received from the server
- Performs client-side validation
- Can implement caching and offline functionality
- Types:
  - Web
  - Mobile
  - Desktop
  - Command-Line
  - Internet of Things (IoT)



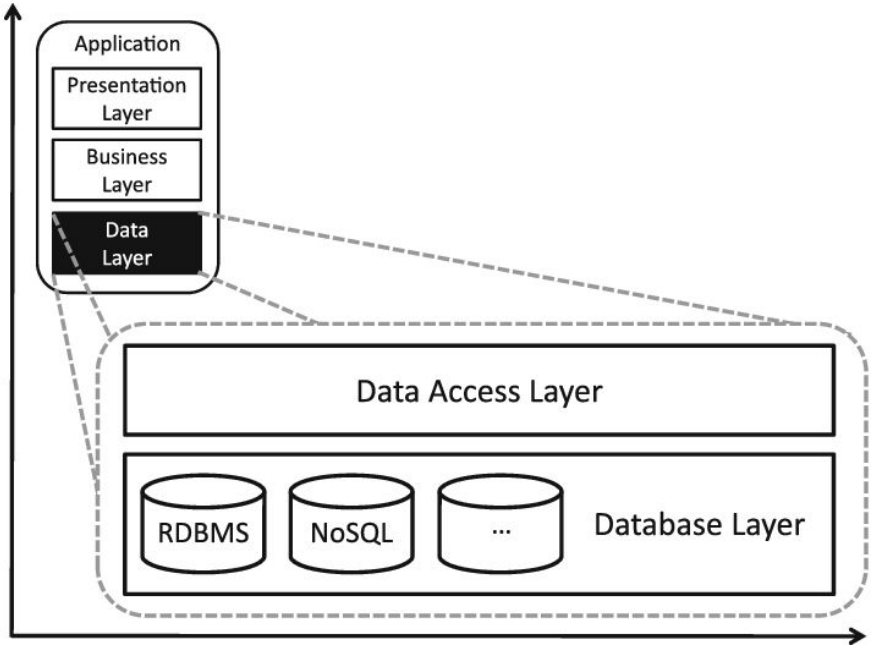
Server (Application Layer)

- Receives and processes requests from clients
- Implements the core business logic
- Performs operations on data
- Structure:
  - Web server (nginx)
  - Application server (puma)
  - Framework (Ruby on Rails)
  - Data Access Layer (ORM, Adapter)



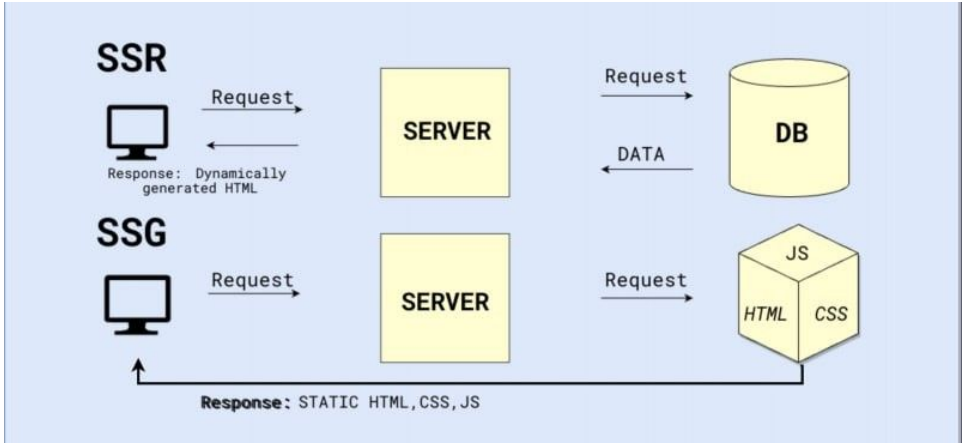
Data Layer

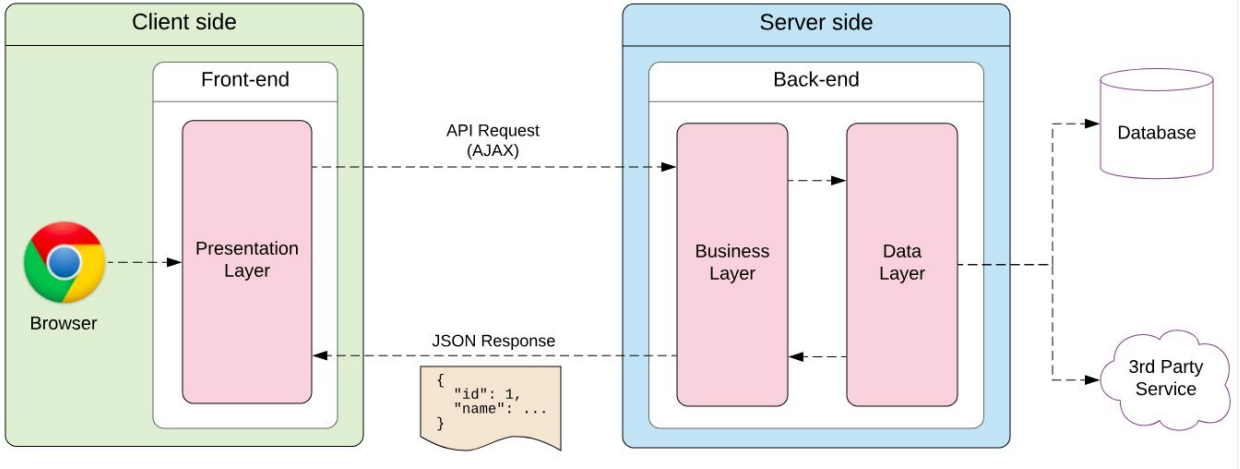
- Managing and storing the application's data
- Ensures data consistency, accuracy, and reliability
- Ensures data security
- Components:
  - Databases
  - File Systems
  - Caching Systems



Client Architecture

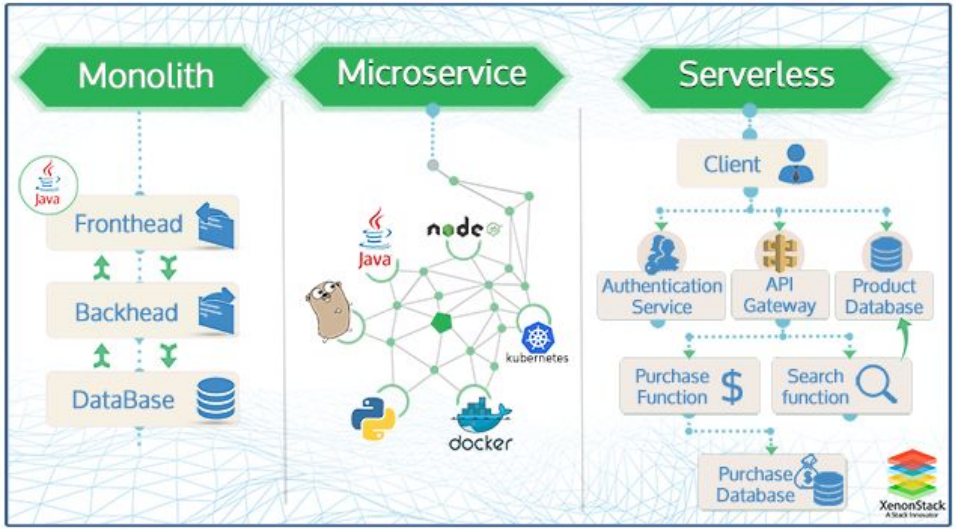
- Server-Side Rendering (SSR)
- Static Site Generation (SSG)
- Single-Page Application (SPA)





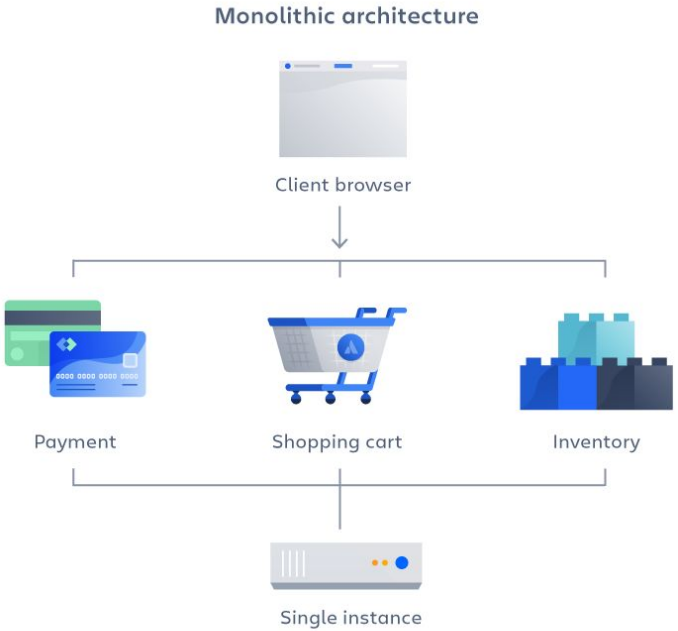
Server Architecture

- Monolithic Architecture
- Microservices Architecture
- Service-Oriented Architecture (SOA)
- Event-Driven Architecture
- Serverless Architecture



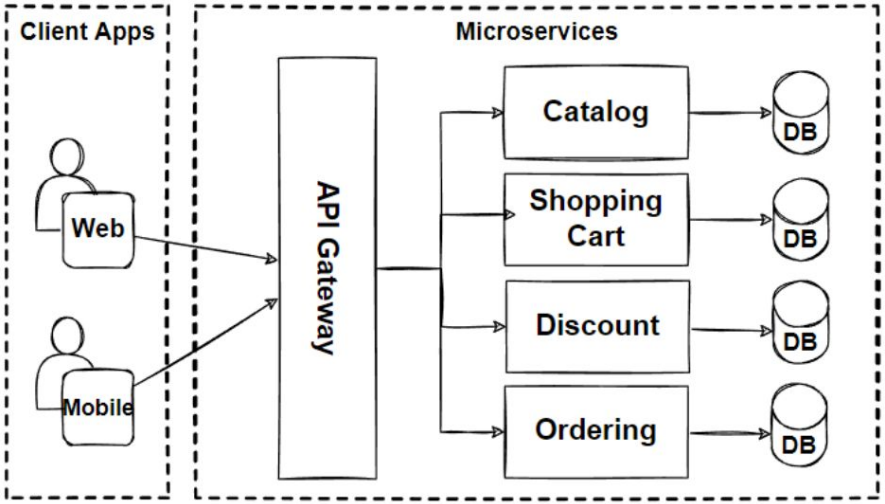
Monolithic Architecture

- Application is developed as a single unit
- Simple deployment
- A lot of dependencies
- Scaling is done by allocating an entire app



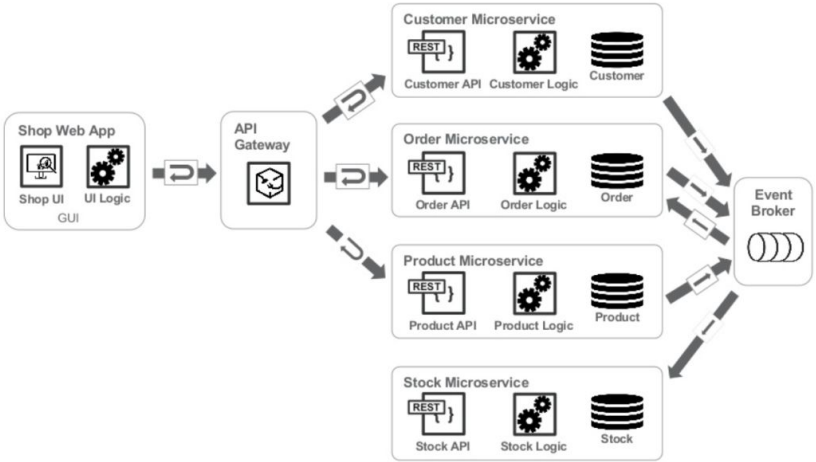
Microservices Architecture

- Application is decomposed into smaller services
- Services communicate through APIs
- Services developed, deployed, and scaled independently
- Services can use different technologies
- System is resilient because of isolation



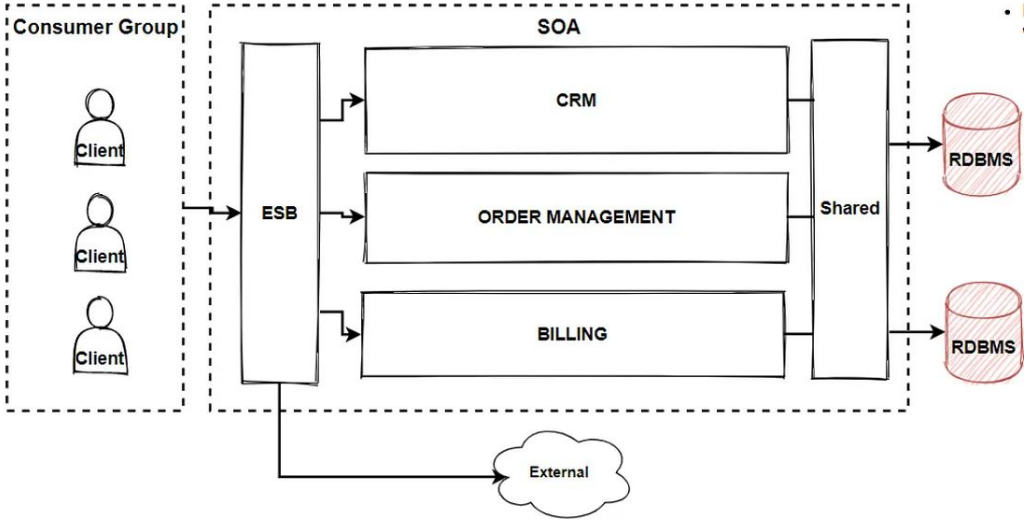
Event-Driven Architecture

- Flow of information is based on the occurrence of events and the reactions to those events
- Components:
  - Event - change in the system or business domain
  - Event Producer
  - Event Consumer
  - Event Broker - mediator of the communication
- Components interact asynchronously
- Events are processed in real-time
- Event ordering may cause racing



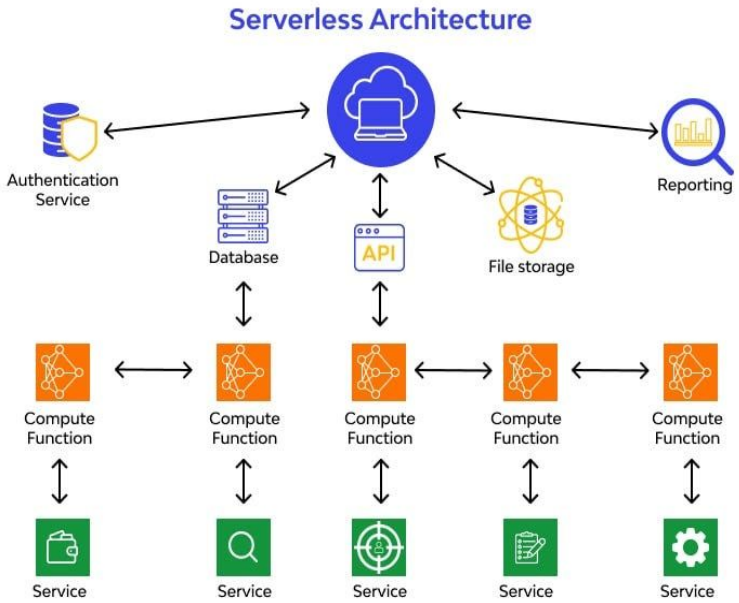
Service-Oriented Architecture

- Software components are designed as reusable services
- Services communicate through standardized interfaces
- Communication via ESB
- Services can be reused as independent components



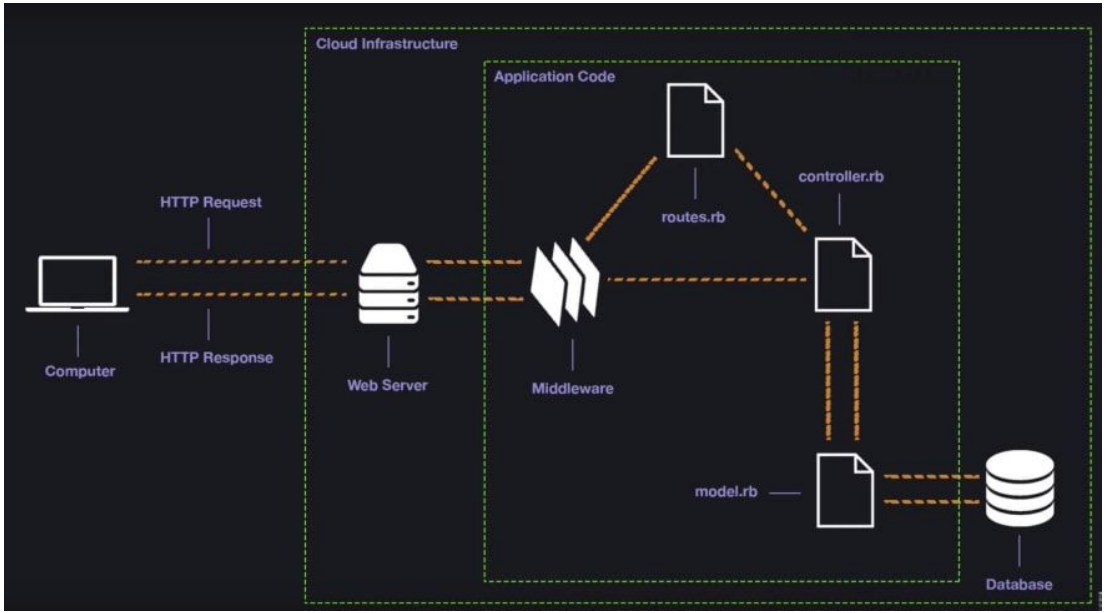
Serverless Architecture

- Serverless architecture is a cloud computing model
- Functions are stateless
- Function-as-a-Service (FaaS)
- Scales automatically in response to demand
- Pay only for the resources consumed during function execution
- Allows to focus on writing application code rather than managing infrastructure
- Functions may experience latency during cold starts
- Dependency on proprietary platforms



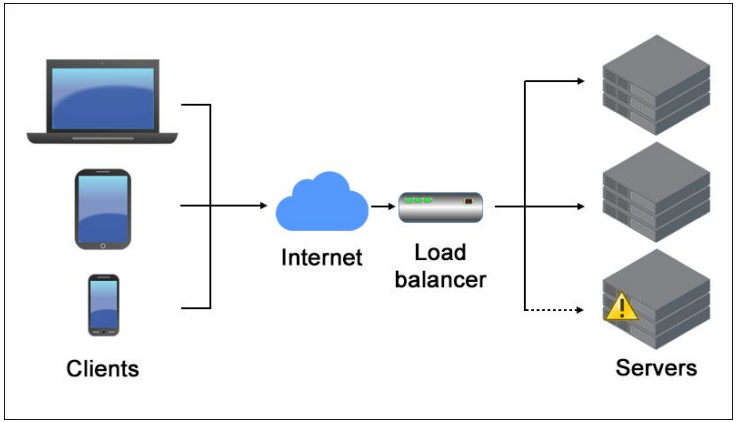
Server Components

- Web Server
  - Load Balancer
  - Reverse Proxy
  - Cache Server
- Application Server
  - API Gateway
    - Middleware
    - Router
    - Authentication
  - Service
    - Authorization
    - Business Logics
    - Data Access
- Database



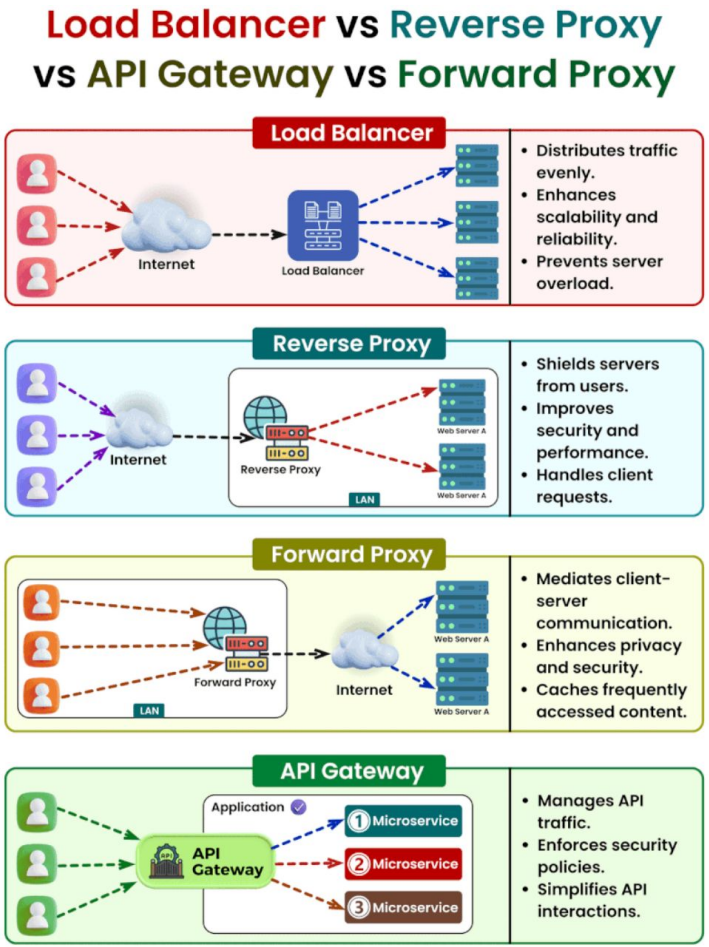
Load Balancer

- Distributes requests across backend servers
- Checks the health and availability of servers
- Maintains session persistence
- Offloads SSL/TLS encryption and decryption
- Algo:
  - Round Robin
  - Least Connections
  - Weighted Round Robin
  - IP Hash
- Type:
  - Hardware
  - Software
  - Cloud



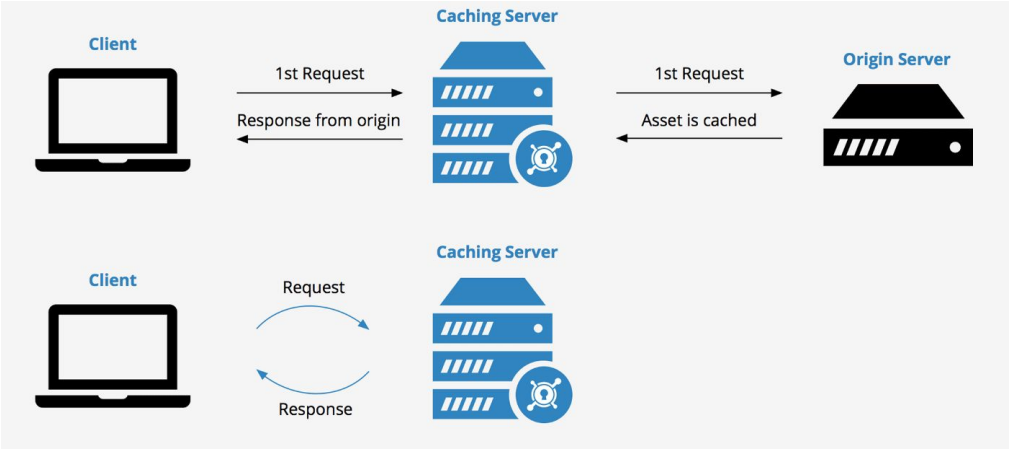
Reverse Proxy

- Special case of load balancer
- Distributes incoming traffic
- Terminates SSL/TLS encryption
- Caching
- Handles static data
- Protecting against attacks such as DDoS, SQL injection, and cross-site scripting (XSS)
- Examples:
  - Nginx
  - Apache HTTP Server



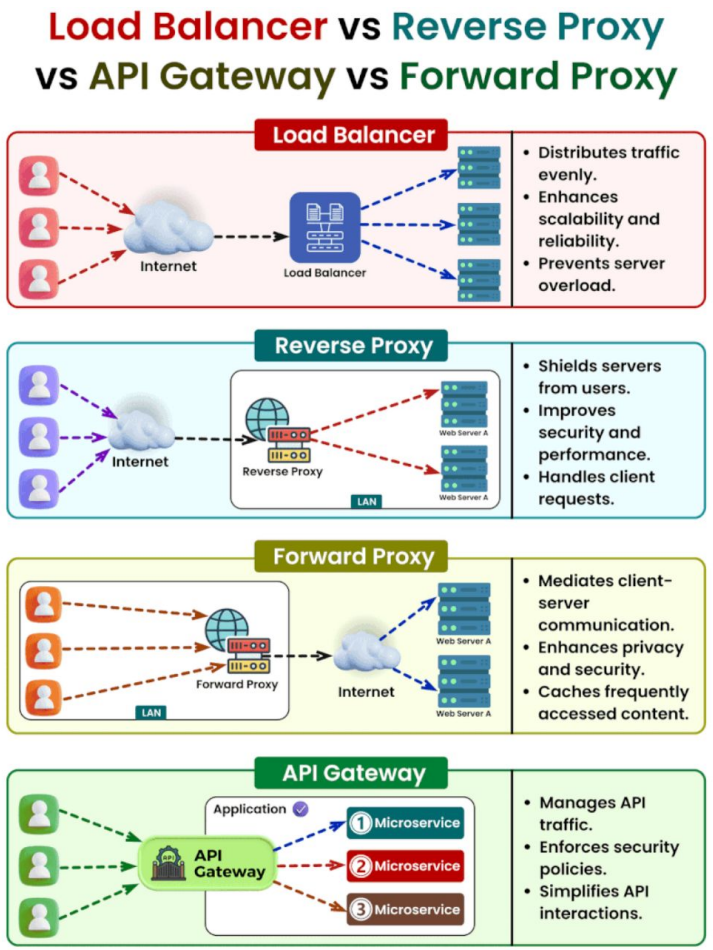
Cache Server

- Stores copies of frequently accessed data
- Improves response times and reduces server load
- Implements strategies to invalidate or expire cache:
  - Time-Based Invalidation
  - Event-Based Invalidation
  - Version-Based Invalidation



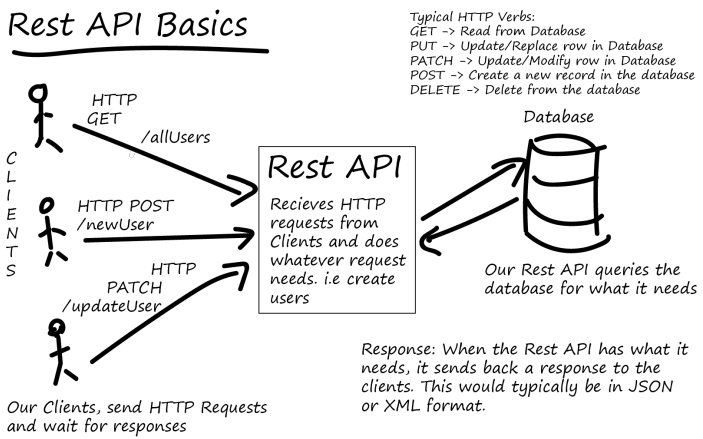
API Gateway

- Entry point for APIs
- Centralized management, security, and access control
- Enforces authentication
- Limits the number of requests
- Transforms incoming requests or responses to match the requirements
- Logs API requests and responses
- Supports API versioning and documentation
- Components:
  - Middlewares
  - Router
- Types:
  - Server
  - Microservice
  - Module



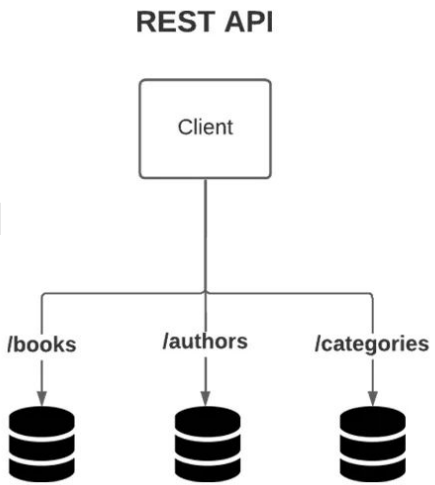
RESTful API

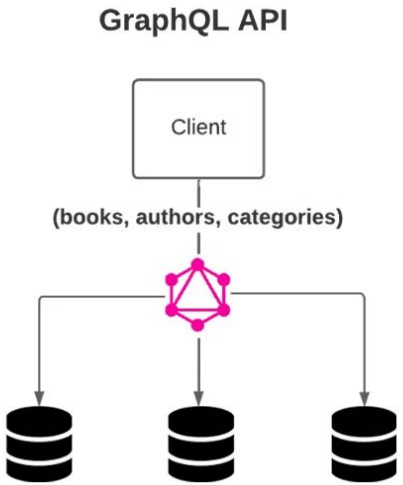
- REST (Representational State Transfer)
- Resources are identified by URIs
- CRUD (Create, Read, Update, Delete) operations are mapped to standard HTTP methods: GET, POST, PUT, DELETE
- The server does not store client state between requests
- Uniform and well-defined interface, using standard HTTP methods and representations (JSON or XML)
- Messages exchanged between clients and servers include metadata that describes how to process them
- Hypermedia as the Engine of Application State (HATEOAS)

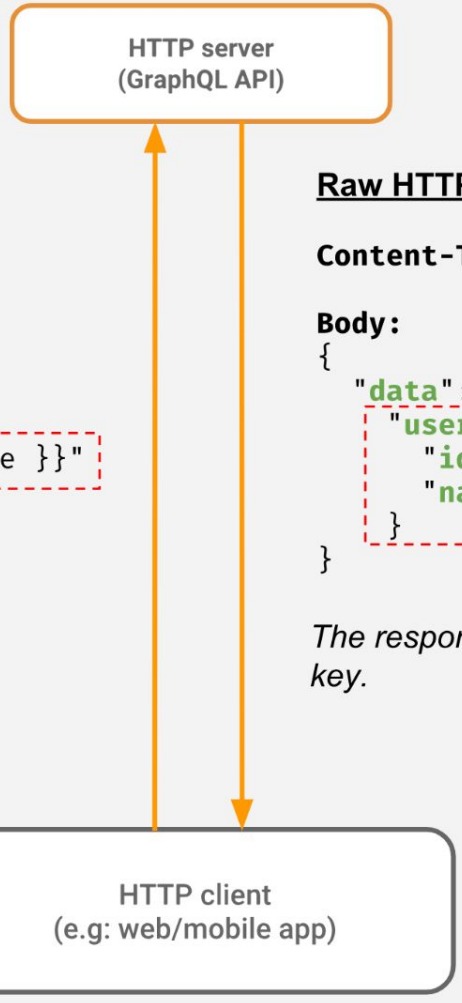


GraphQL

- Clients query data from a single endpoint
- Queries mirror the shape of the requested data
- APIs are defined by a strongly typed schema
- Clients specify the structure and shape of the data they require
- Supports nested queries







Raw HTTP request

Method: POST
URL: https://api.com/graphql
Content-Type: application/json

Body:

```
{
  "query": "query { user (id: 1) { id name }}"
}
```

The GraphQL query is sent as a string inside a JSON object.

Raw HTTP response

Content-Type: application/json

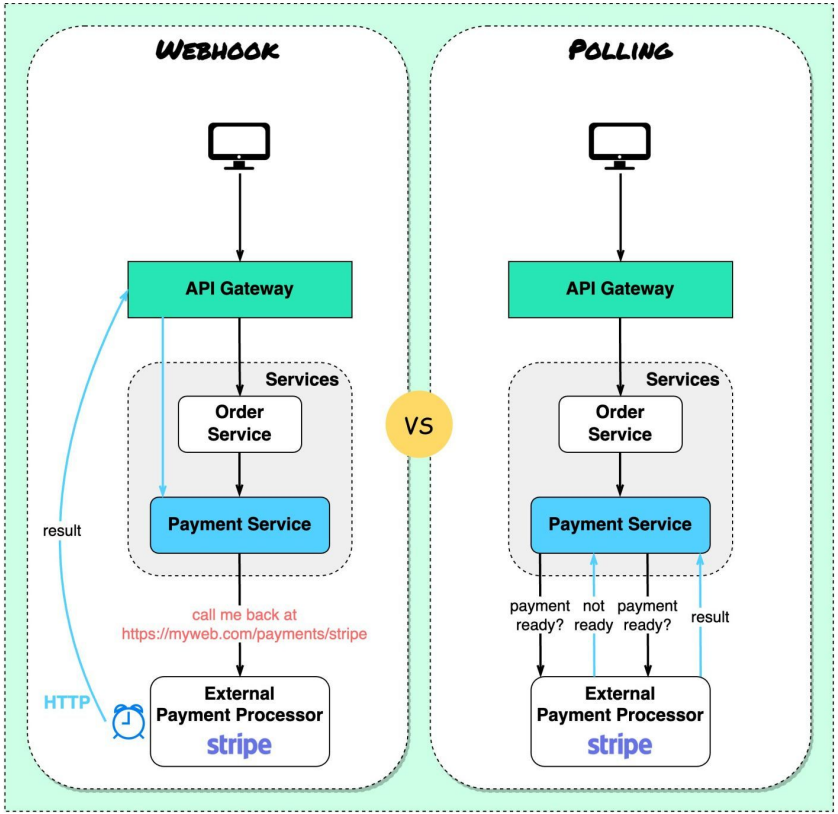
Body:

```
{
  "data": {
    "user": {
      "id": 1,
      "name": "Elmo"
    }
}
```

*The response object is inside the **data** key.*

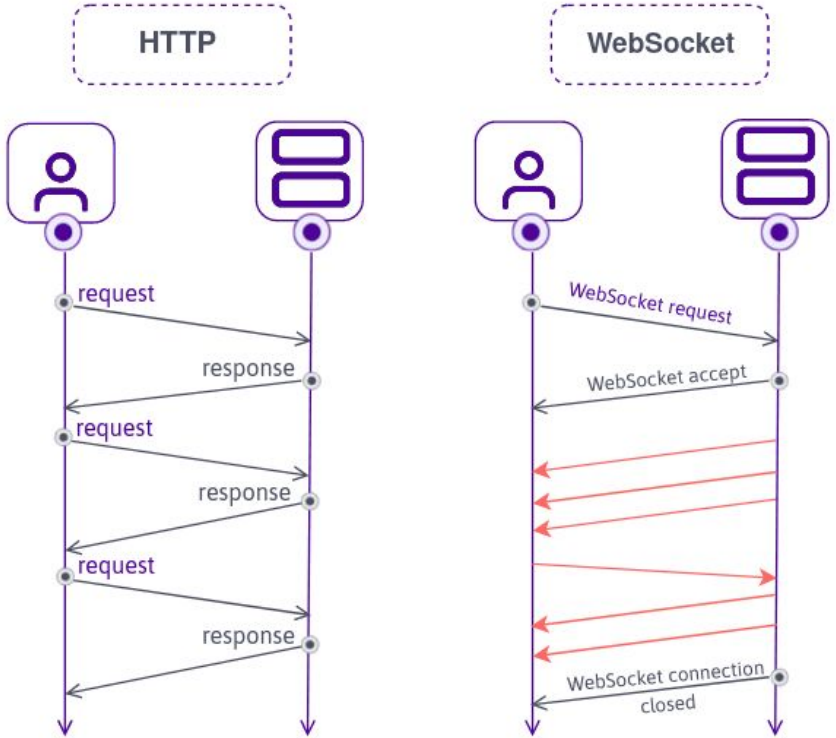
Webhooks

- User-defined HTTP callbacks triggered by specific events
- Event-driven communication between systems
- Webhooks operate asynchronously
- Contains payload with relevant data
- Use cases:
  - Notifications
  - Data Synchronization
  - Workflow Automation



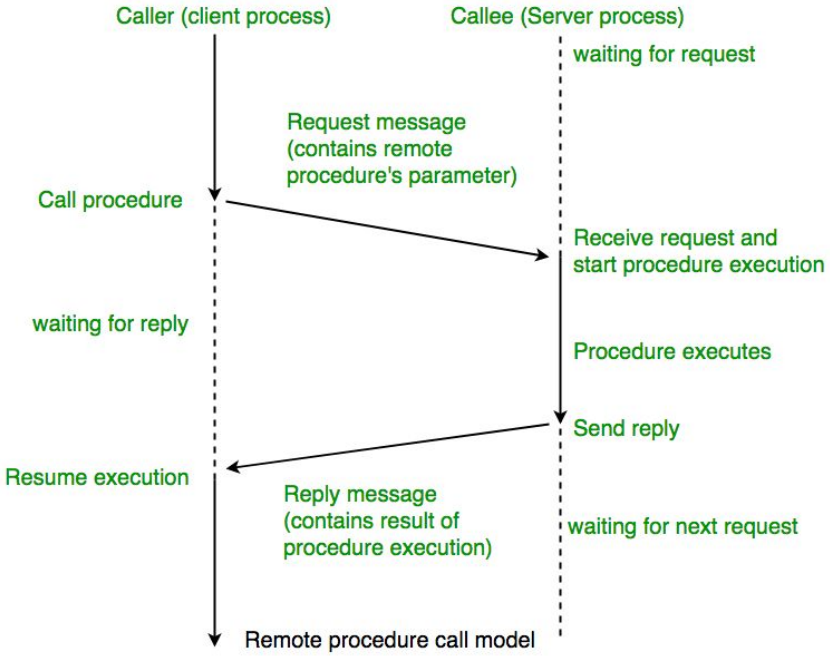
WebSockets

- Bidirectional communication between clients and servers
- Provides low-latency communication
- Can be secured using TLS/SSL encryption
- Use cases:
  - Real-Time Data Update
  - Multiplayer Gaming
  - Live Broadcasting



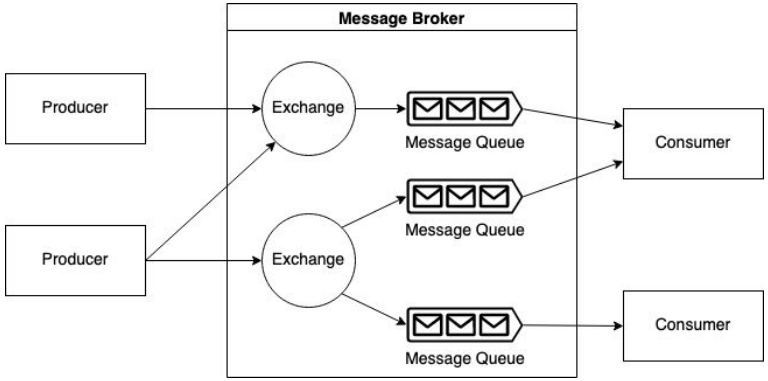
RPC

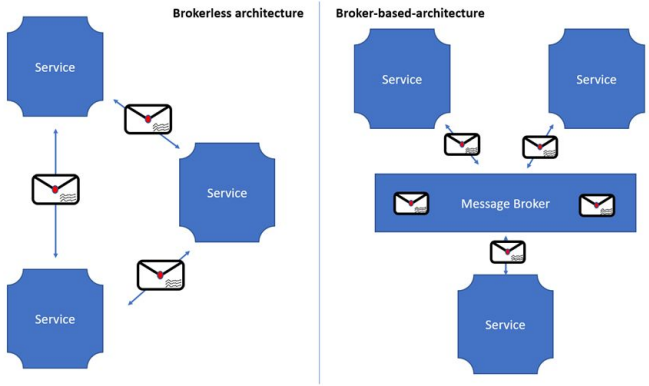
- Remote Procedure Call (RPC)
- Invoke functions or procedures on a remote server as if they were local
- Hides the complexities of network communication
- Language-Independent
- Protocols:
  - gRPC
  - JSON-RPC
  - XML-RPC



Message Brokers

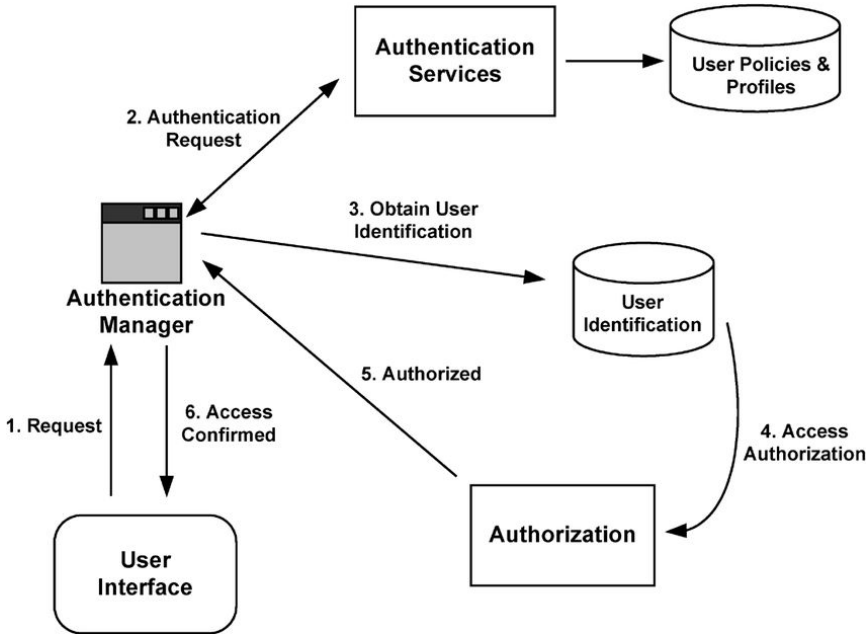
- Provides a message queue where senders can publish messages and receivers can subscribe
- Routes messages from producers to the appropriate consumers
- Ensures message durability by storing messages persistently until they are consumed
- Allows messages to be transformed or enriched
- Types:
  - Broker-Based: RabbitMQ, Kafka
  - Brokerless (Pub/Sub): MQTT, NATS
- Use cases:
  - Event-Driven Architecture
  - Microservices communication





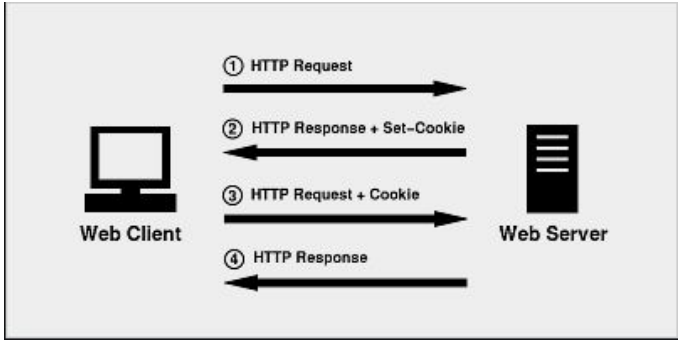
Authentication

- Authentication is the process of verifying the identity of a user
- Users provide an identifier to authenticate themselves (login, email)
- Users provide credentials to prove their identity (password, token)
- Authentication factors:
  - Knowledge (password)
  - Possession (security token)
  - Inherence (biometric data)
  - Multi-Factor Authentication (MFA)



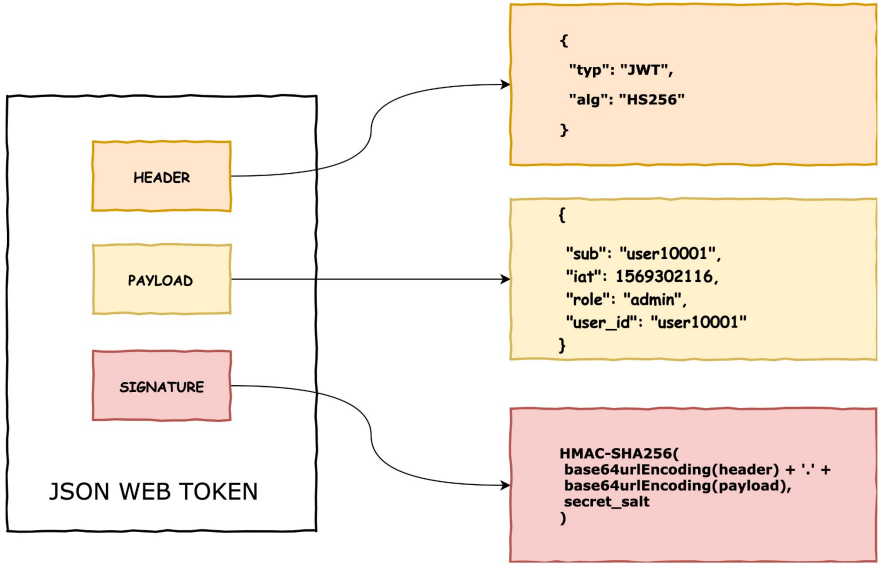
Session

- Sessions are initiated when a user authenticates
- The system assigns a unique session identifier (session ID)
- Sessions can be terminated by logging out or after a predefined period of inactivity
- Storage types:
  - Client side (cookies, local storage)
  - Server side (database, in-memory cache)
- Use cases:
  - Authentication
  - Personalization
  - User Storage
  - Web Analytics



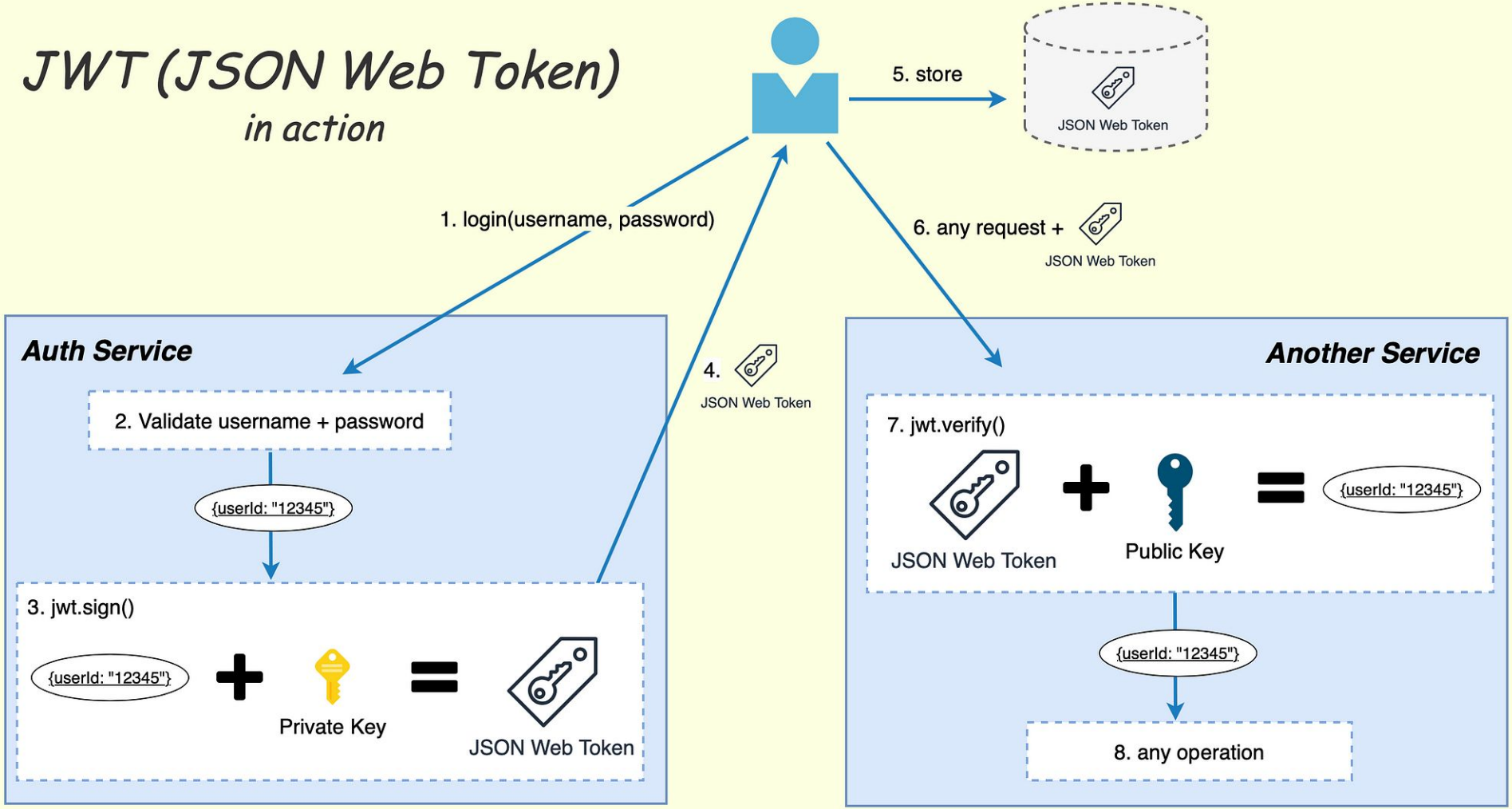
JWT

- JSON Web Tokens (JWT) are a compact, URL-safe means of representing claims to be transferred between two parties
- JWTs contain all the necessary information about the user or session
- JWTs can be signed using a secret key or asymmetric algorithm
- JWTs can include an expiration timestamp
- Public keys can be shared by JWKS



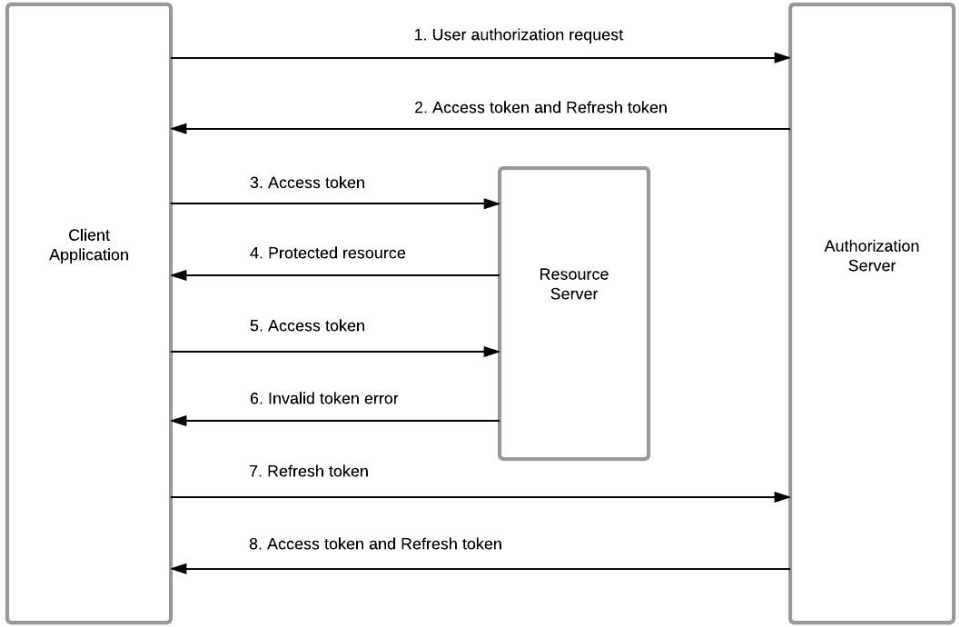
JWT (JSON Web Token)

in action



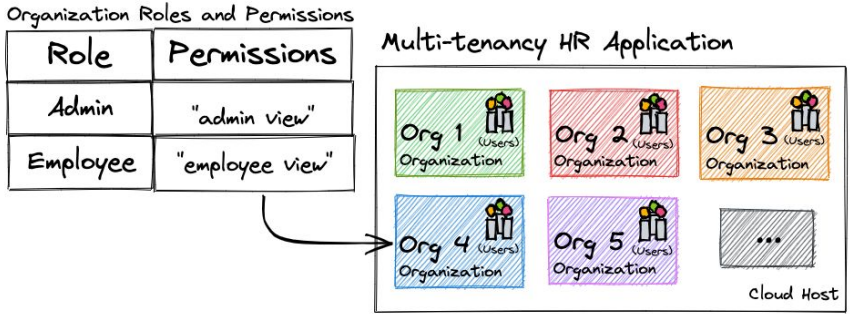
OAuth 2.0

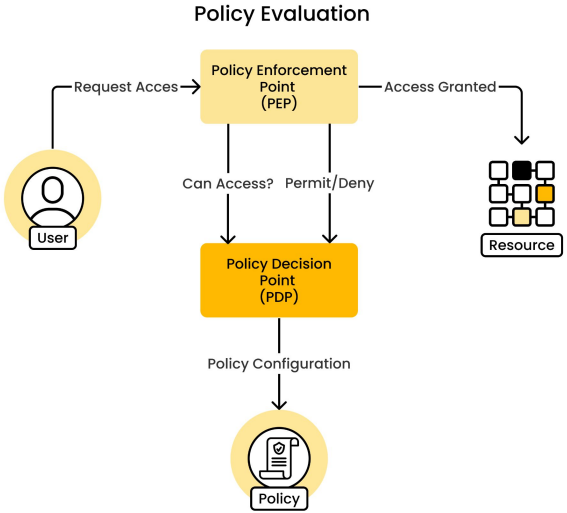
- Authorization protocol
- Users can grant limited access to their resources without sharing their credentials
- Follows standardized protocols and flows
- Supports SSO



Authorization

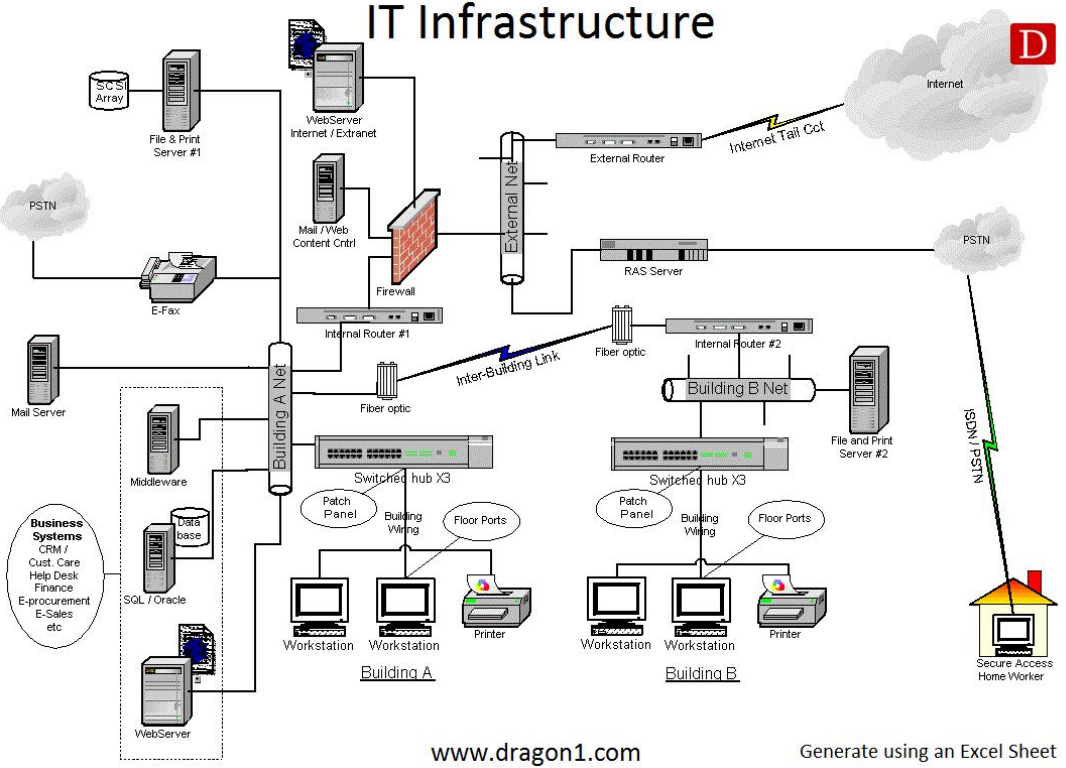
- Authorization is the process of determining whether a user has privileges
- Enforces policies and control access to resources
- Users are assigned roles that represent a set of permissions or privileges
- Prevents unauthorized access
- Ensure compliance with regulatory requirements
- Types:
  - Role-Based
  - Attribute-Based
  - Rule-Based





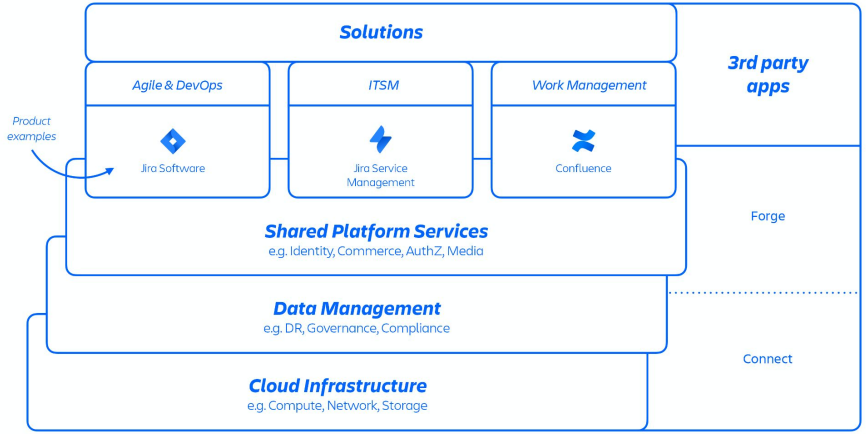
Infrastructure

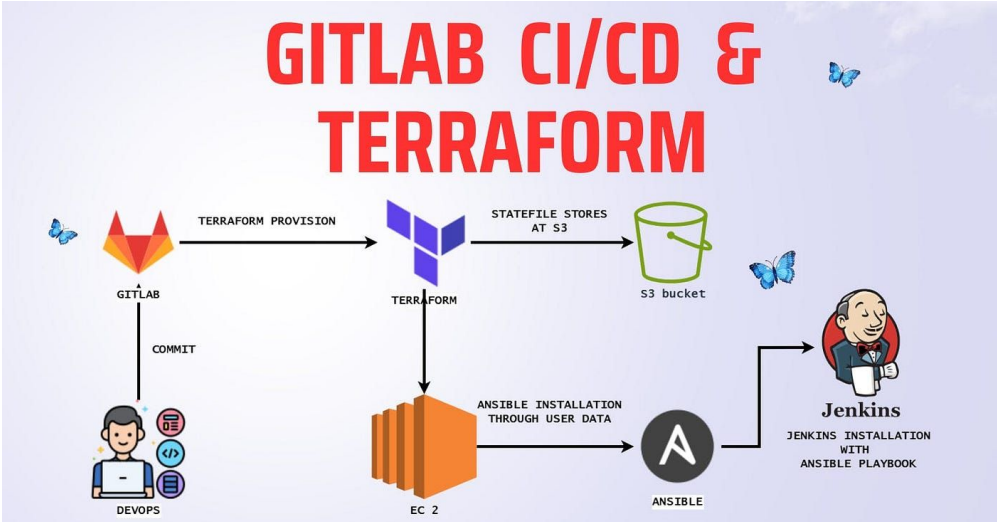
- Integrated framework to support the operation, management, and delivery
- Hardware
  - Servers
  - Networking Equipment
  - Data Centers
- Network
  - Local Area Network (LAN)
  - Virtual Private Network (VPN)
  - Content Delivery Network (CDN)
  - Domain Name System (DNS)
- Software



Software Infrastructure

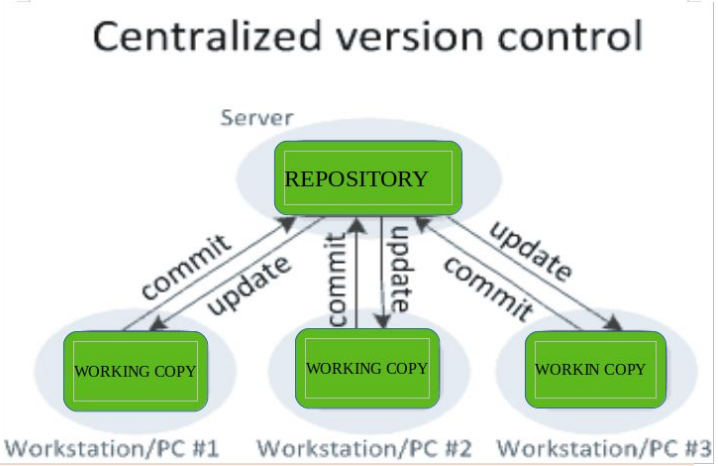
- Development
  - Frameworks and Libraries
  - Web Servers
  - Databases
  - Version Control Systems
  - Integrated Development Environments (IDI
- Delivery
  - Hosting Platform
  - CI/CD Tools
  - Container Orchestration Platforms
- Management
  - Project Management Systems
  - Knowledge Sharing Platforms
  - Communication Platforms
  - Monitoring & Analytics Systems

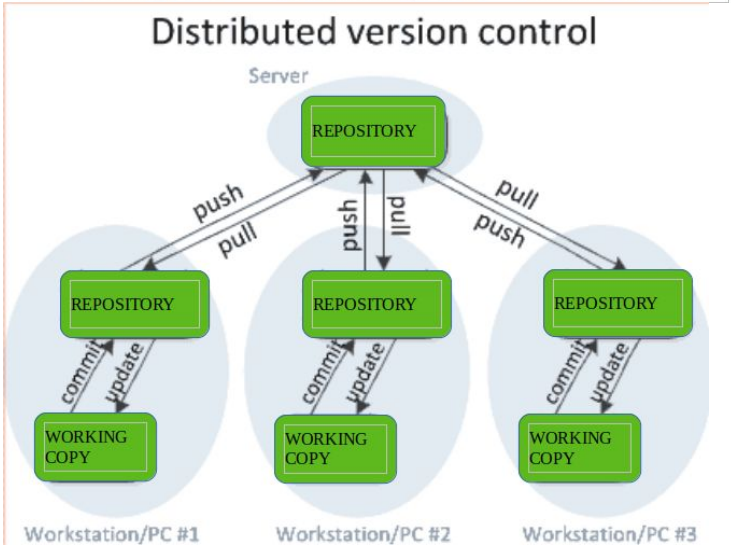




Version Control Systems

- Source code management
- Manage concurrent edits, merging changes, and resolving conflicts
- Release management
- Detailed history of changes
- Backup and recovery mechanism
- Allow code review processes
- Examples:
  - Git (GitHub, GitLab, Bitbucket)
  - Subversion





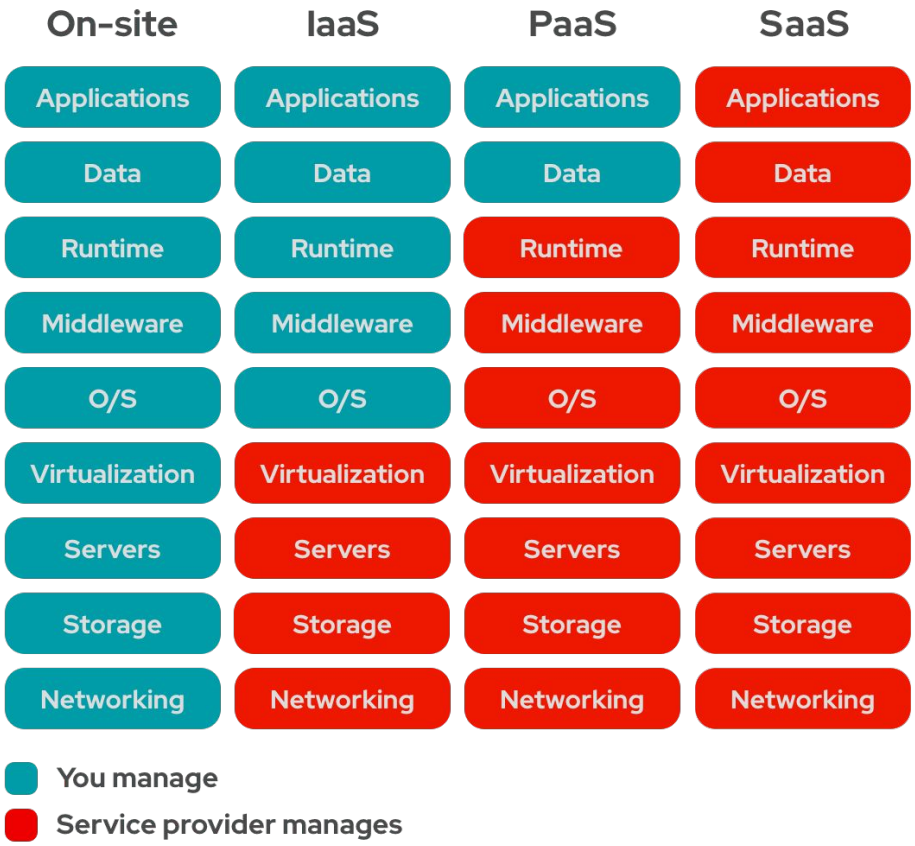
Integrated Development Environments

- Software development facilitator
- Code editing
  - Syntax highlighting
  - Code completion
- Code debugging
  - Breakpoints
  - Inspection
  - Stepping
- Compiling
- Resource Management
  - File management
  - Navigation
- Integrations
- Examples:
  - VSCode
  - RubyMine
  - Vim/Emacs

TYPES OF INTEGRATED DEVELOPMENT ENVIRONMENTS (IDE)	
Type	**Description**
Language-specific IDEs	Designed to support a specific programming language
Multi-language IDEs	Support multiple programming languages
Web development IDEs	Designed for web development
Mobile development IDEs	Designed for developing mobile applications
Cloud-based IDEs	Hosted in the cloud
Platform-specific IDEs	Work best on a specific platform or operating system
Open-source IDEs	Freely available for modification and distribution
Educational IDEs	Designed specifically for learners
Commercial IDEs	Developed by companies and sold commercially

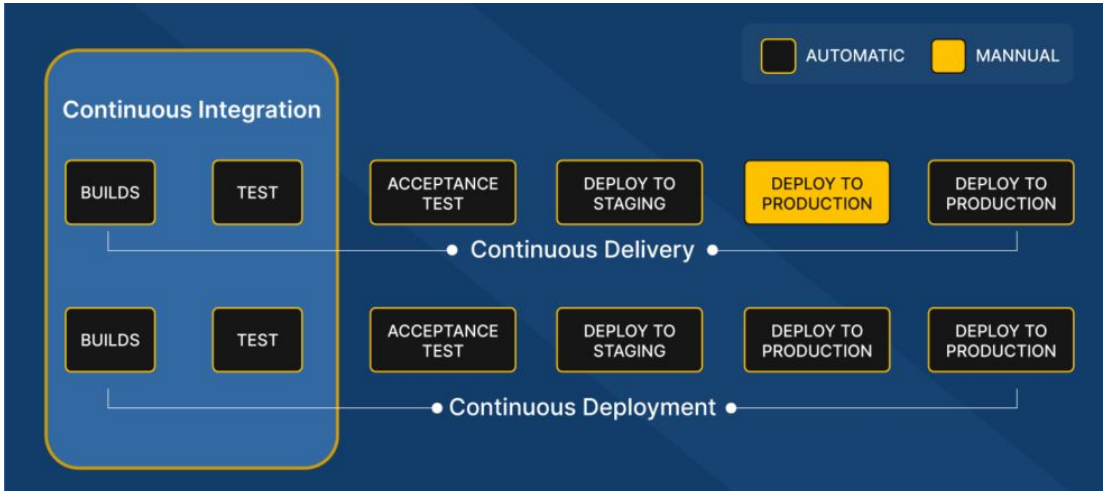
Hosting Platforms

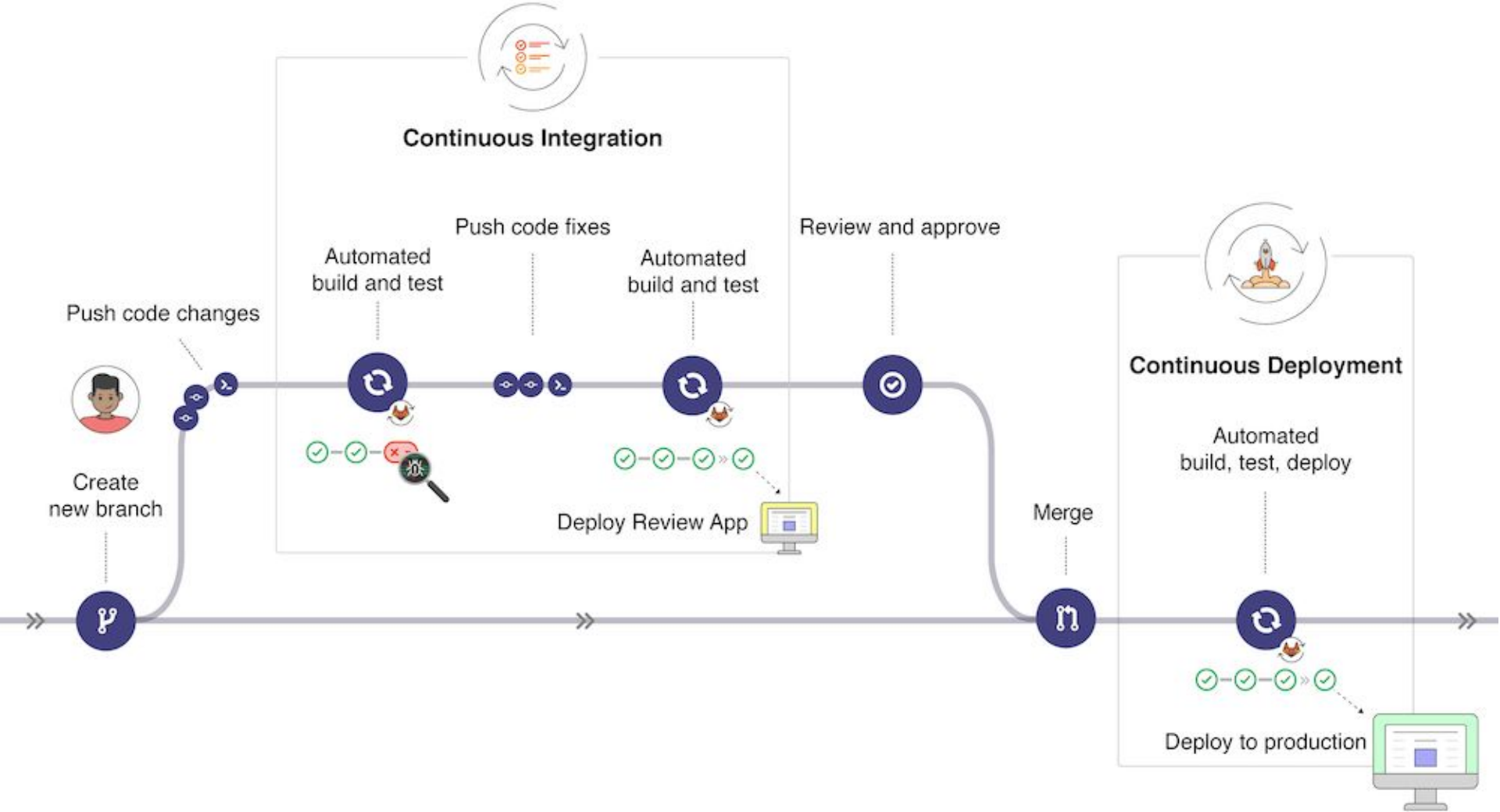
- Provide infrastructure and services for hosting
- Scale up or down based on demand
- Security by firewalls, encryption, and access controls
- Reliability by redundancy, failover mechanisms, and data replication
- Examples:
  - Amazon Web Services (AWS)
  - Microsoft Azure
  - Google Cloud Platform (GCP)
  - Heroku



CI/CD Tools

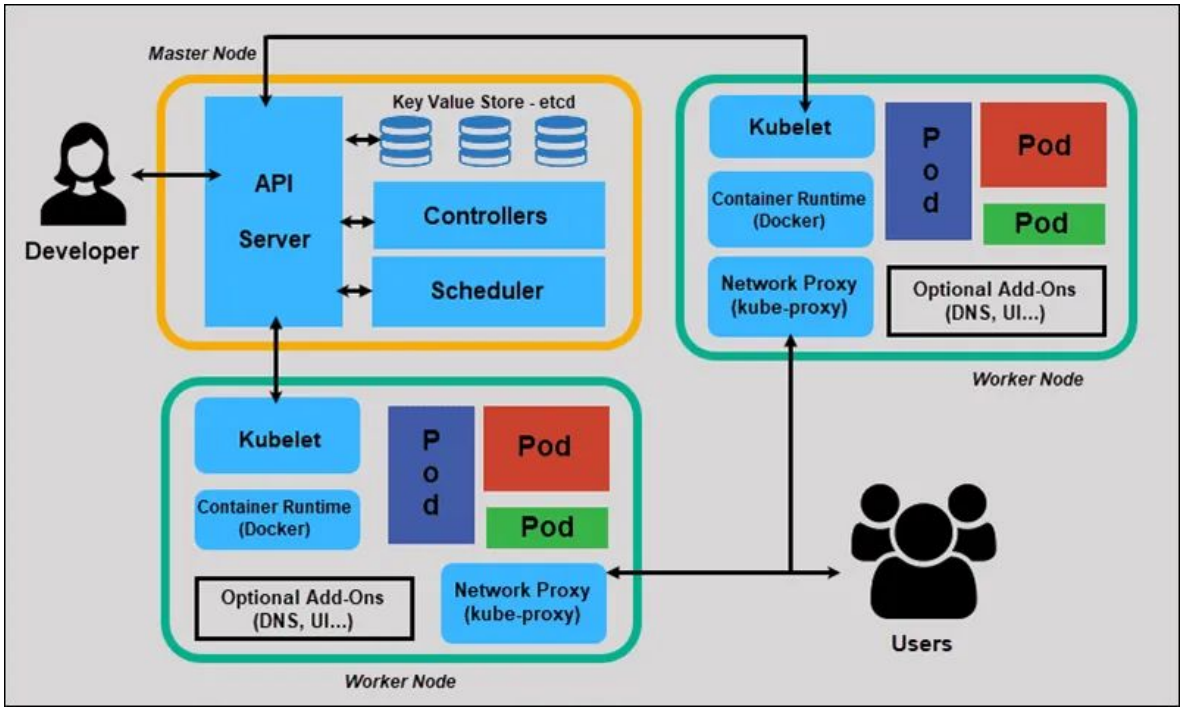
- Continuous Integration/Continuous Deployment
- Automated builds
- Automated testing
- Automate code deployment
- Version Control integration
- Workflow orchestration
- Examples:
  - Jenkins
  - GitLab CI/CD
  - Bitbucket Pipelines



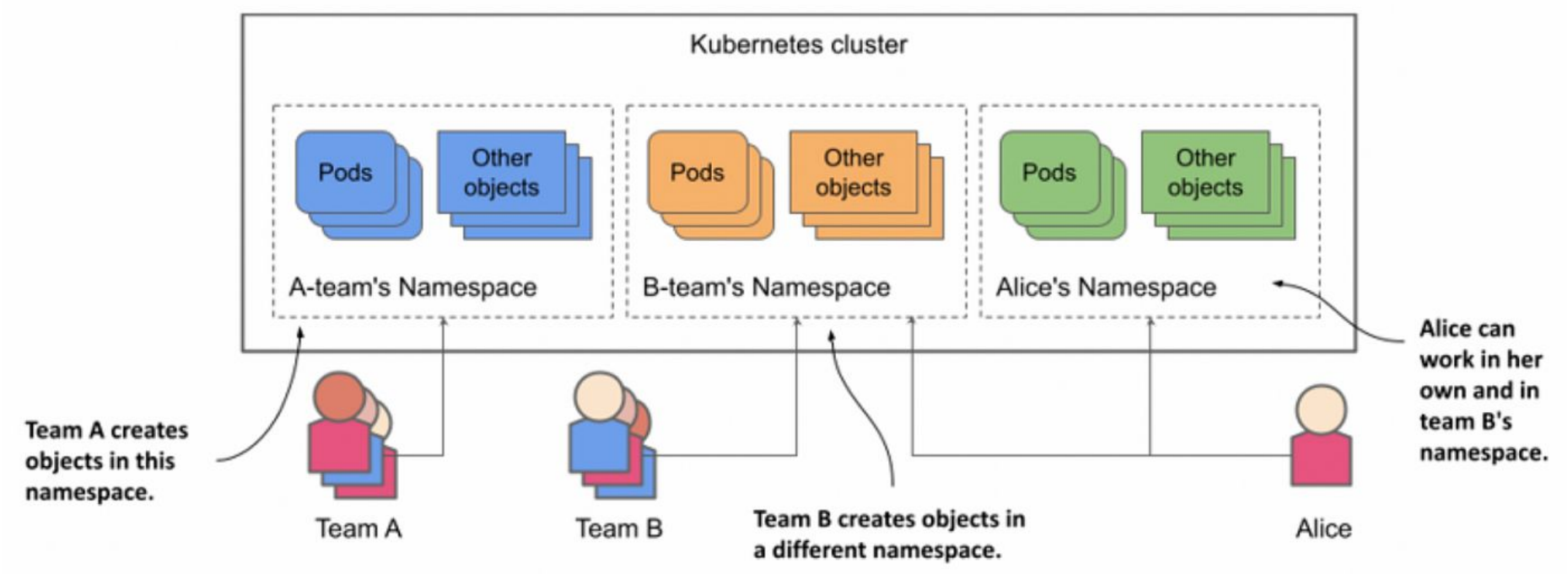

Continuous Integration
Push code changes
Create new branch
Automated build and test
Push code fixes
Automated build and test
Deploy Review App
Review and approve
Merge
Continuous Deployment
Automated build, test, deploy
Deploy to production

Container Orchestration Platforms

- Management of containerized applications
- Scaling management
- Load balancing
- Health monitoring
- Security
- Examples:
  - Kubernetes
  - Docker Swarm
  - Amazon EKS

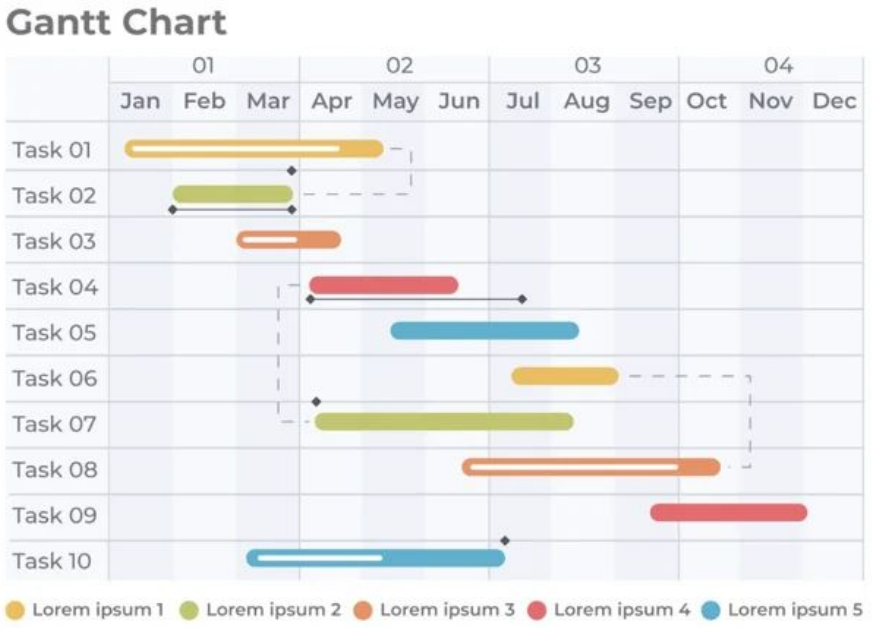


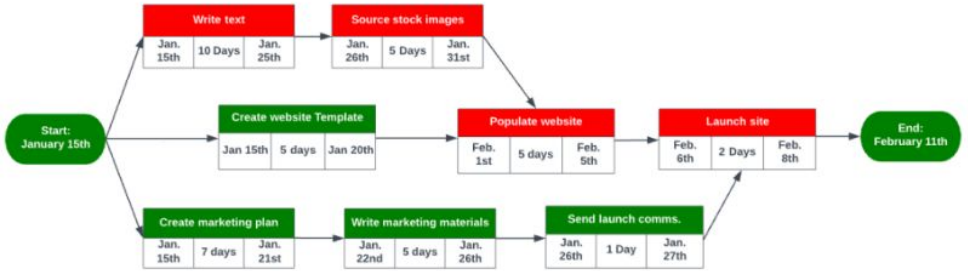
Kubernetes Namespaces



Project Management Systems

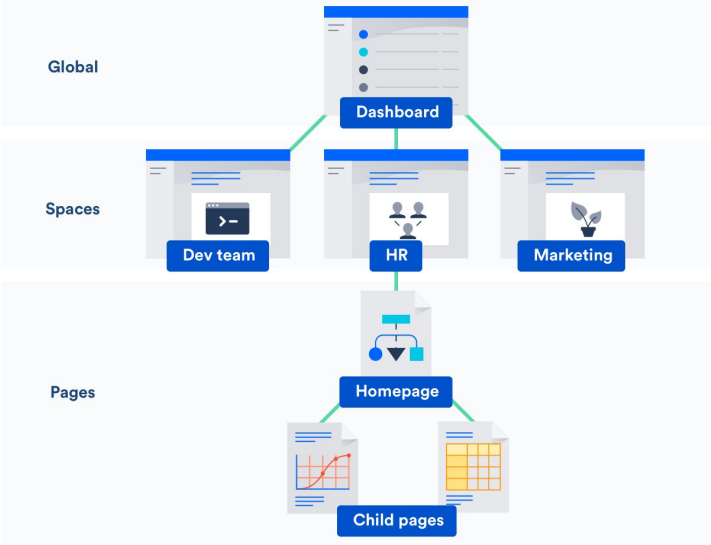
- Task creation and assignment
- Track task status including updates/comments/attachments
- Task scheduling and prioritization
- Workflow management
- Reporting and analytics
- Integrations
- Examples:
  - Jira
  - Trello
  - Asana





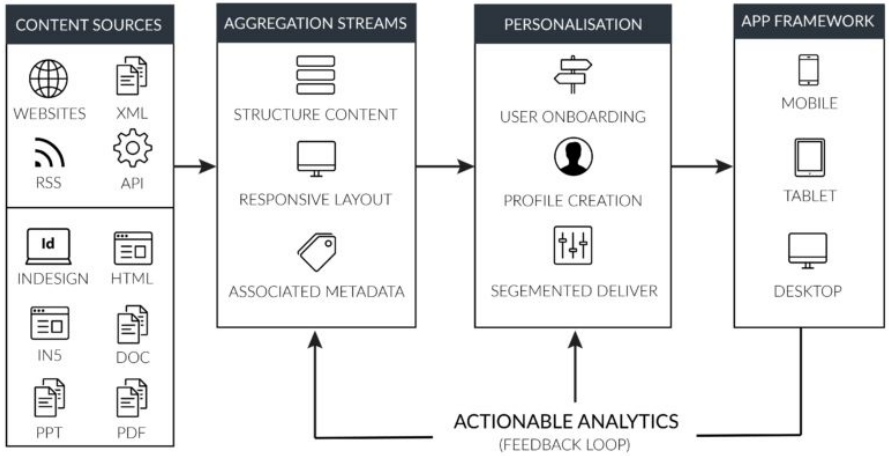
Knowledge Sharing Platforms

- Create, publish, and share various types of content, including articles, documents, presentations, videos, and tutorials
- Search by keywords and categorization
- Comment, rate, and contribute to existing content
- Integration
- Analytics and reporting
- Examples:
  - Confluence
  - Microsoft SharePoint



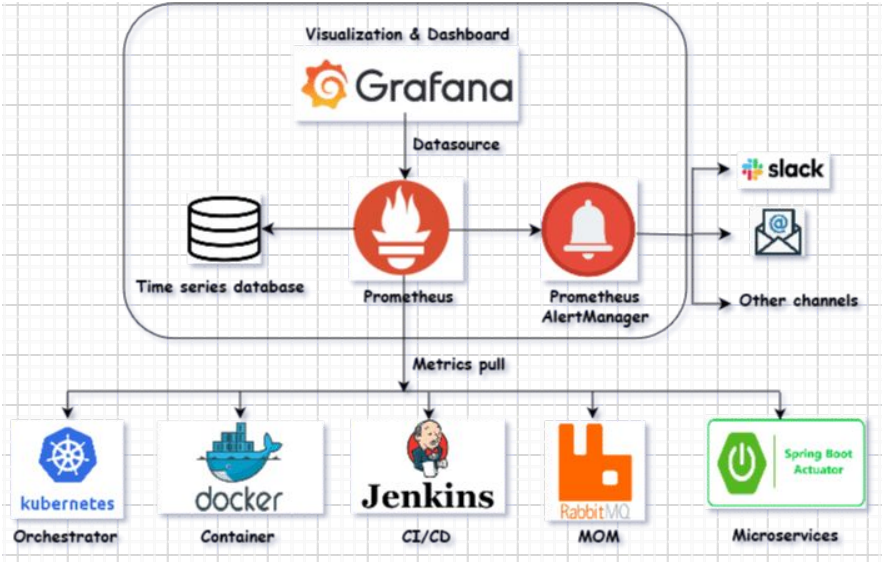
Communication Platforms

- Instant messaging channels and group chats
- Provide voice and video calling features
- Share files, documents, and multimedia content
- Presence and status indicators
- Notifications and alerts
- Advanced search mechanisms
- Encryption, access controls, and compliance standards
- Integration
- Examples:
  - Microsoft Teams
  - Slack
  - Zoom



Monitoring & Analytics Systems

- Collect data from servers, applications, databases, network devices, and user interactions in real-time
- Measure and monitors key performance indicators (KPIs)
- Alerting and notifications
- Visualize data through dashboards, graphs
- Log management
- Monitor system performance
- Integration
- Examples:
  - Prometheus
  - Grafana
  - Datadog



Thank you for your attention!