Backend Essentials for DevOps Engineers

This document is created by CloudChamp. Please Subscribe!! and Share this document with other DevOps Engineers :]

Backend Overview
In any web application, the backend handles server operations, application logic, and data storage. It's the "behind-the-scenes" functionality, as opposed to the frontend, which is what the user directly interacts with.

Backend Components:
1. **Server**: Hosts and runs the backend application.
2. **Application**: Contains core application logic and APIs.
3. **Database**: Stores and manages user data.
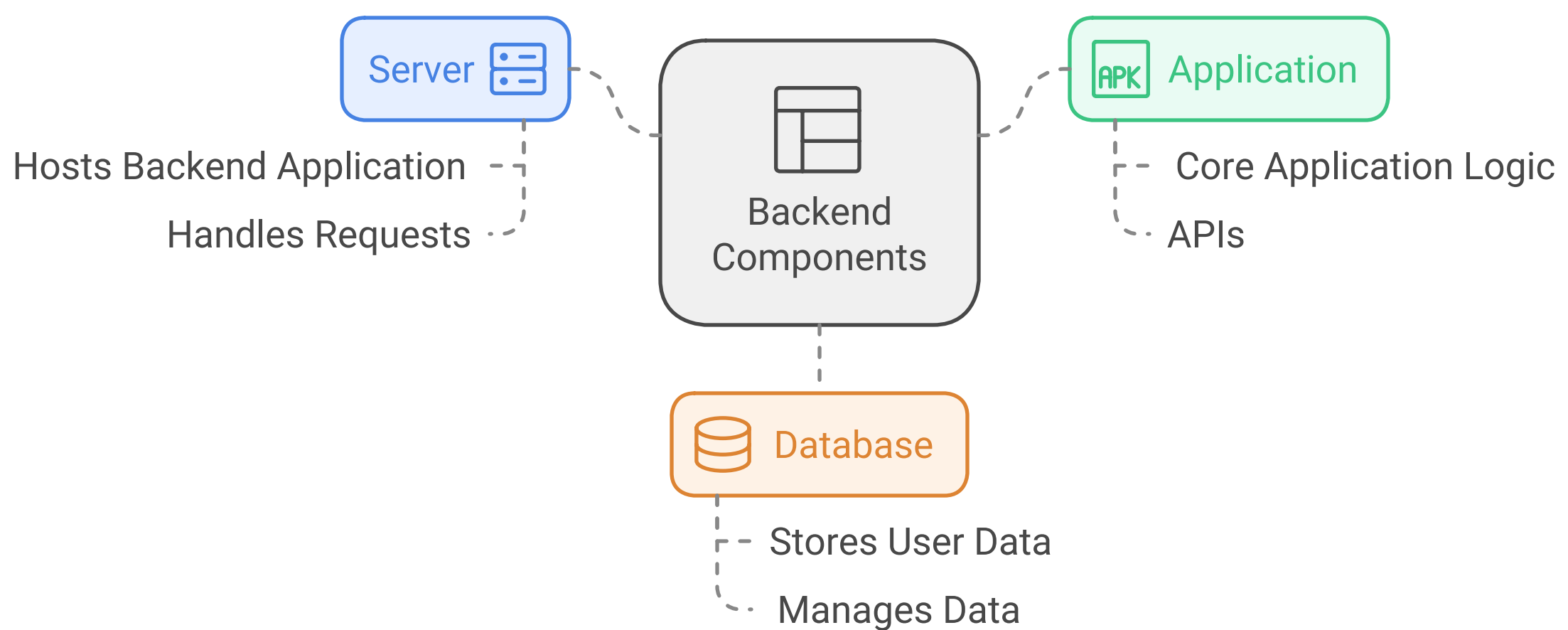
1. Servers
Definition: Servers are high-powered computers that process requests from clients (e.g., browsers, mobile apps). In modern applications, servers are often hosted on cloud platforms.
Types of Servers:
- **Dedicated Servers**: Single-tenant physical servers.
- **Virtual Private Servers (VPS)**: Virtualized servers within a larger physical machine.
- **Cloud Servers**: Scalable virtual servers managed by cloud providers.

Popular Cloud Providers:
- **AWS EC2**: Provides scalable compute capacity.
- **Google Cloud Compute Engine**: High-performance VMs.
- **Azure Virtual Machines**: Provides virtualized servers for Windows and Linux.

Example Command (AWS EC2):

```
aws ec2 run-instances --image-id ami-0123456789 --count 1 --instance-type
t2.micro
```

2. Databases
Databases store and retrieve data used by applications. They are classified into two main types:
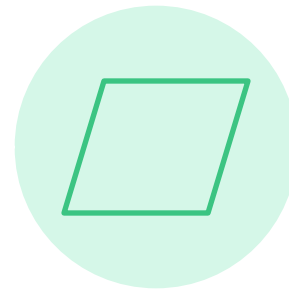
Types of Databases:

Choose the right database type for your application needs

Relational Databases

Structured data
organization

NoSQL Databases

Flexible schema for
unstructured data

- **SQL Databases**: Structured data stored in tables with rows and columns. Supports complex queries.
    - **Examples**: MySQL, PostgreSQL, SQL Server.
- **NoSQL Databases**: Flexible schema for unstructured data. Better suited for large-scale, dynamic data.
    - **Examples**: MongoDB (document store), DynamoDB (key-value store).

SQL Query Example:

```
SELECT * FROM products WHERE category = 'electronics';
```

MongoDB Query Example:

```
db.products.find({ category: 'electronics' })
```

Real-World Example:
- **E-commerce**: MySQL for user accounts and transactions; MongoDB for fast, flexible product catalog searches.

3. Backend Development Essentials

Programming Languages: Backend applications are written in languages that support server-side operations.
- **Python**: Often used with Django and Flask.
- **JavaScript**: Node.js for asynchronous processing.
- **Java**: Common in large-scale enterprise applications.

Frameworks:
- **Django (Python)**: Full-stack framework, useful for complex applications.
- **Express (Node.js)**: Lightweight, flexible, used for APIs and microservices.
- **Spring Boot (Java)**: Enterprise-grade applications.

Package Managers:
- **pip** for Python: Manages libraries like **psycopg2** for PostgreSQL.
- **npm** for Node.js: Manages packages like **express** for web APIs.

Example Commands:
- Install Django in Python: **pip install django**
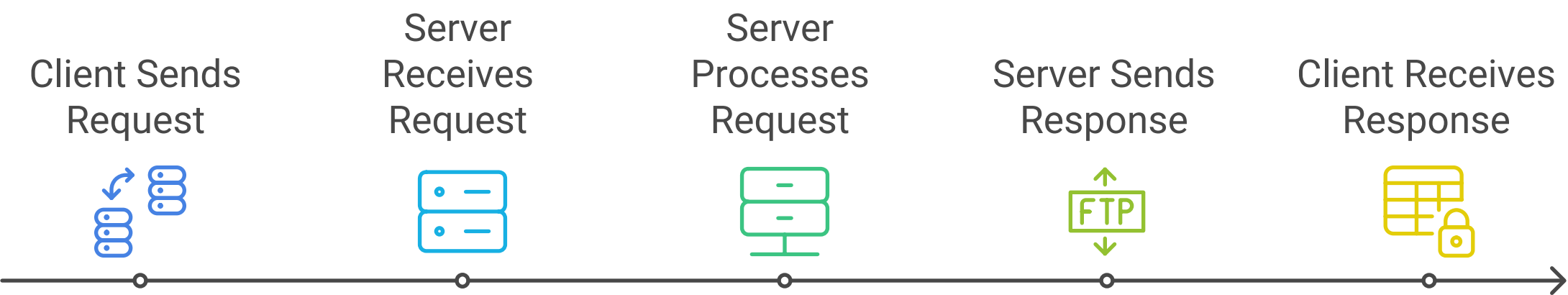- Install Express in Node.js: **npm install express**

4. Request-Response Cycle and APIs

Definition: The request-response cycle is the process by which a client requests data from a server, and the server sends a response.

API (Application Programming Interface): APIs define how data is exchanged between client and server, enabling them to communicate. APIs may use **REST**, **GraphQL**, or **gRPC** protocols.

HTTP Methods:

Request-Response Cycle and API Communication

Client Sends Request	Server Receives Request	Server Processes Request	Server Sends Response	Client Receives Response

- **GET**: Retrieve data.
- **POST**: Submit data.
- **PUT**: Update existing data.
- **DELETE**: Remove data.

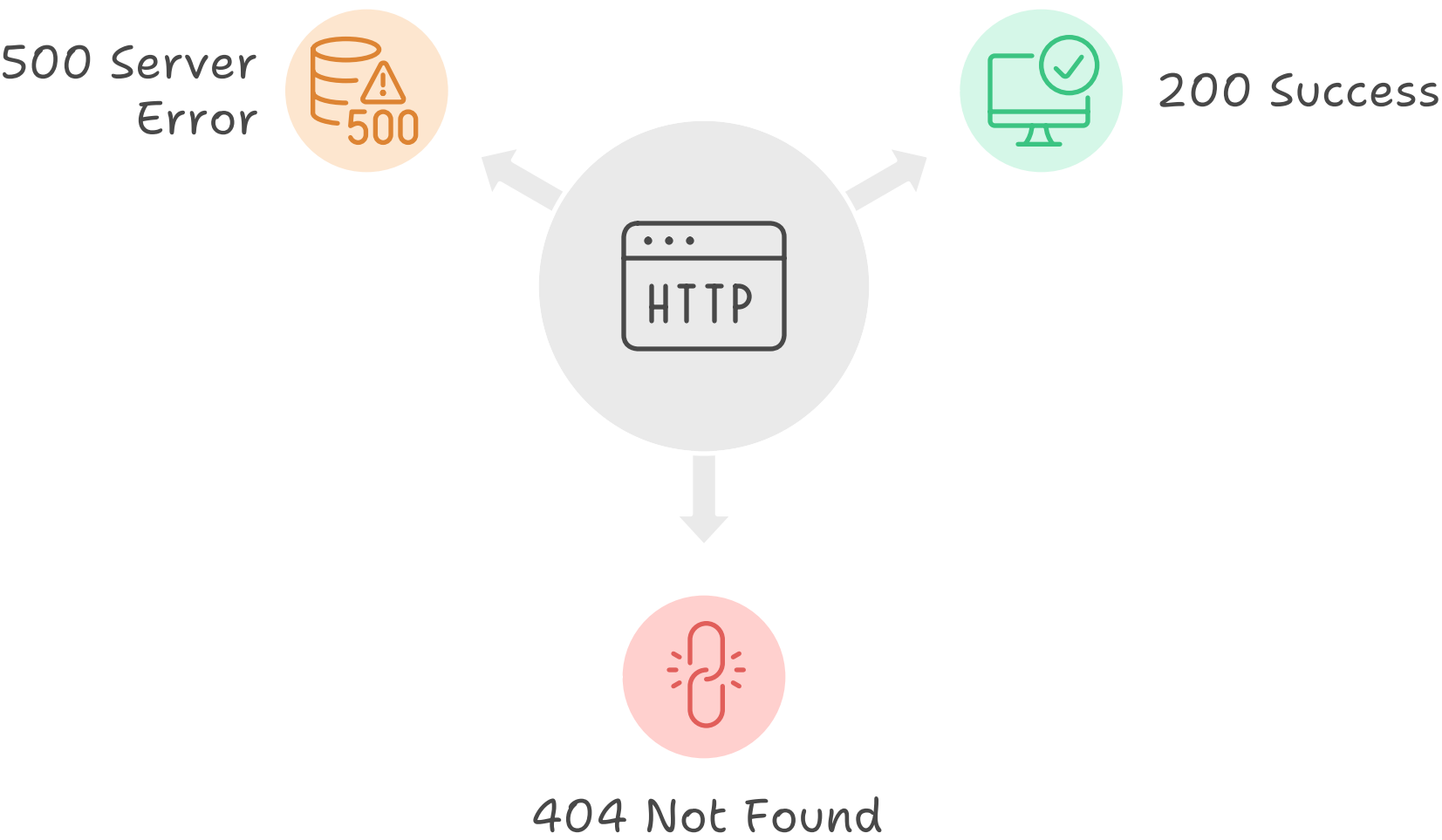**API Request Example**:
- **GET Request in Python**:

```python
import requests
response = requests.get("<https://example.com/api/v1/users>")
print(response.json())
```

Response Codes:
- **200**: Success
- **404**: Not Found
- **500**: Server Error

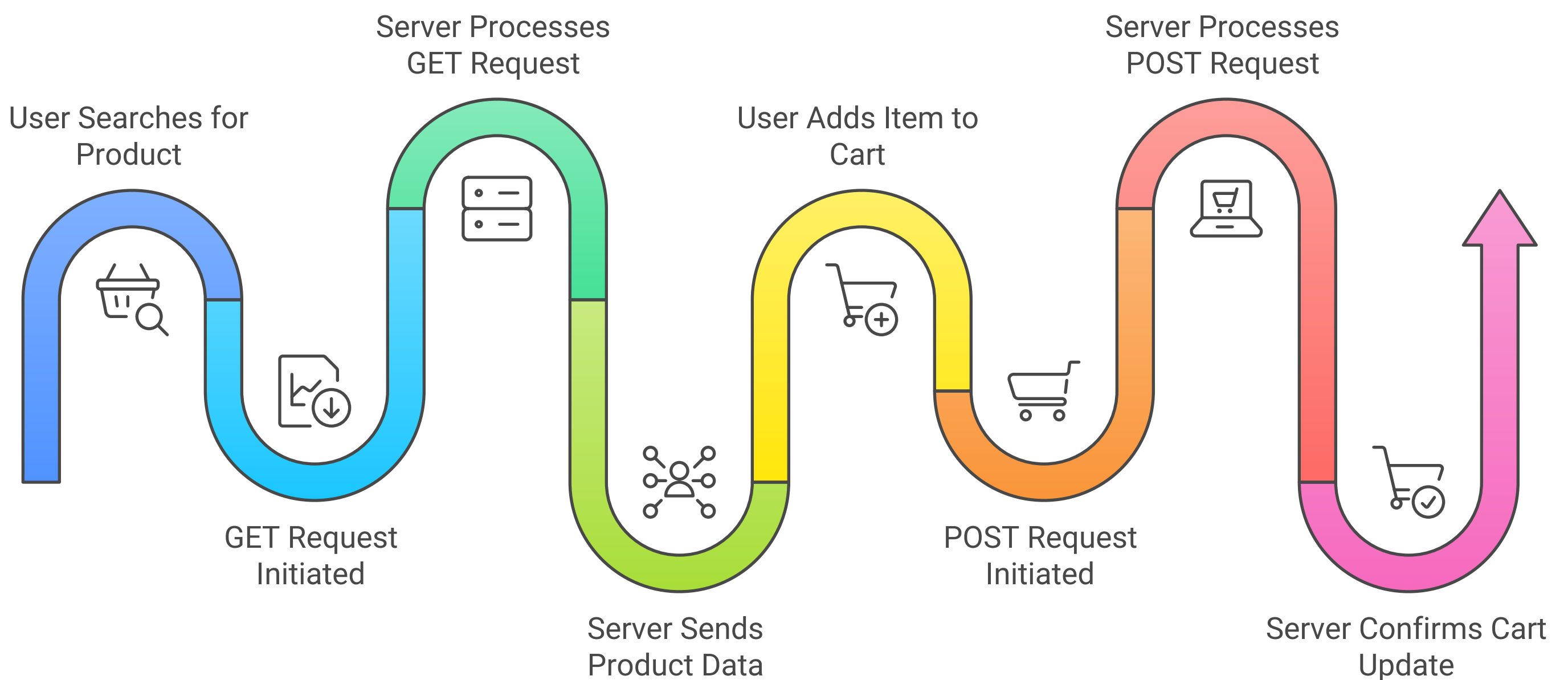## Understanding HTTP Response Codes

500 Server Error

200 Success

HTTP

404 Not Found

Real-World Example:
- **Amazon.com**: Searching for a product initiates a GET request to retrieve items; adding an item to the cart uses a POST request.

Amazon.com Request-Response Cycle

User Searches for
Product

Server Processes
GET Request

GET Request
Initiated

Server Sends
Product Data

User Adds Item to
Cart

POST Request
Initiated

Server Processes
POST Request

Server Confirms Cart
Update

5. Scaling and Load Balancing

As traffic increases, scaling and balancing the load are crucial to maintaining performance.

Scaling Types:
- **Horizontal Scaling**: Adding more servers to handle additional traffic.
- **Vertical Scaling**: Increasing server capacity (CPU, RAM).

Load Balancers: Distribute incoming requests across multiple servers to ensure reliability and uptime. Popular load balancers include **NGINX**, **AWS ELB**.

6. Messaging and Task Queues

For background tasks, messaging systems and task queues are useful, especially for actions like sending notifications.

Messaging Systems:
- **RabbitMQ**: Message-broker for asynchronous communication.
- **Redis**: Can be used as a message broker and caching system.

Real-World Example:
- **Subscription Renewals**: Using RabbitMQ to queue and send reminder emails when subscriptions are about to expire.

7. Monitoring and Logging

For backend stability and performance, monitoring and logging are essential.

Monitoring:
- **Prometheus**: Open-source tool for system and application metrics.
- **Grafana**: Visualization tool for monitoring data.

Logging:
- **ELK Stack** (Elasticsearch, Logstash, Kibana): Centralized logging solution for error tracking and debugging.

Prometheus Configuration:

```
scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['localhost:9100']
```

Real-World Example:

- **High-Traffic Websites**: Monitoring CPU and memory usage across servers to automatically scale when thresholds are crossed.

Additional Backend Tools for DevOps

1. **Containerization**: Docker for packaging applications with dependencies.
1. Learn Docker:
2. **Orchestration**: Kubernetes for managing containerized applications at scale.What is Kubernetes and How it works:
3. **CI/CD**: Jenkins, GitLab CI for automating application deployment.GitLab CICD FREE Course: https://youtu.be/JWXVijJfnHc?si=Gb-YYE-_pP4RYXHU
4. **Security**: Tools like Vault for secrets management, ensuring secure access to sensitive information.

Example Backend Project Structure:

```
myapp/
├────── app/
│    ├────── main.py         # Main application file
│    ├────── config.py       # Configurations
│    └────── requirements.txt # Dependencies
├────── Dockerfile           # Docker container setup
├────── prometheus.yml       # Monitoring configuration
└────── README.md            # Documentation
```

This guide covers essential backend components, from servers and databases to APIs and scaling strategies, providing DevOps engineers with a strong foundation for backend infrastructure management.

Credit: https://www.youtube.com/@cloudchamp?sub_confirmation=1