SMS DOCUMENTATION

Below is a **clear, point-wise, easy-to-understand explanation** of **Nginx, Kafka, MySQL, Redis, Tomcat, Jenkins, and CI/CD**, including **real-life examples**, **simple architecture diagrams**, and finally **SMS sending flow diagrams** (Submit → ACK → DLR).

1. NGINX

What is NGINX?

NGINX is a **web server and traffic controller**.
It is the **first point of entry** when a request comes from a user or client.

What does NGINX do?

- Receives requests from users
- Sends requests to the correct backend server
- Distributes load between servers
- Improves speed and security

Why is it needed?

- Protects backend servers
- Handles thousands of requests at the same time
- Prevents server overload

Key Functions

- Reverse Proxy
- Load Balancer
- SSL Termination
- Static Content Serving

Easy Real-Life Example

Think of **NGINX as a receptionist** in an office:

- Visitors come to the reception
- Receptionist decides which department to send them to
- Employees do not deal with crowd directly

Simple Example

When you send an SMS request:

- Request hits **NGINX**
- NGINX forwards it to the application server

Simple Architecture Diagram

```
User
  |
  v
[ NGINX ]
  |
  +--> App Server 1
  |
  +--> App Server 2
```

2. KAFKA

What is Kafka?

Kafka is a **message queue system** that handles **large volumes of data** safely and reliably.

What problem does Kafka solve?

- Prevents data loss
- Handles traffic spikes
- Allows systems to work independently

What does Kafka do?

- Stores messages temporarily
- Handles millions of messages per second
- Sends messages to consumers when they are ready
- Processes data asynchronously (not immediately blocking)

Key Concepts

- Producer: Sends message
- Topic: Message category
- Consumer: Reads message
- Broker: Kafka server

Real-Life Example

Kafka is like a **courier hub**:

- Senders drop parcels
- Parcels wait safely
- Delivery boys pick them up when available

Simple Example

In SMS:

- Message is added to Kafka
- SMS system sends messages to operators one by one
- Even if operator is slow, messages are not lost

Architecture Diagram

```
Producer
   |
   v
[ Kafka Topic ]
   |
   v
Consumer
```

3. MySQL

What is MySQL?

MySQL is a **database** used to store data **permanently**.

What type of data is stored?

- User details
- SMS records
- Delivery status
- Billing information

Why is MySQL needed?

- Keeps records safe
- Data is available even after restart
- Supports reports and audits

Easy Real-Life Example

MySQL is like a **ledger book** in an office:

- Every transaction is written
- Records are never erased accidentally
- Can be checked anytime

Simple Example

When SMS is sent:

- Message ID, number, status are saved in MySQL

Architecture Diagram

```
Application
     |
     v
  [ MySQL ]
```

4. Redis

What is Redis?

Redis is a **fast in-memory data store** (data is kept in RAM).

Why is Redis used?

- Very fast
- Reduces load on database
- Used for temporary data

What data is stored in Redis?

- OTPs
- Session data
- Rate limits
- Frequently used information

Easy Real-Life Example

Redis is like a **sticky note on your desk**:

- Important info kept nearby
- Faster than searching files
- Temporary but very useful

Simple Example

When OTP is sent:

- OTP stored in Redis
- Verified quickly without checking database

Architecture Diagram

```
Application
   |
   +--> Redis (Cache)
   |
   +--> MySQL (DB)
```

5. Tomcat

What is Tomcat?

Tomcat is a **Java application server** that runs the **main business logic**.

What does Tomcat do?

- Validates requests
- Applies business rules
- Connects to Redis, MySQL, Kafka
- Sends responses back

Why is Tomcat needed?

- Core processing engine
- Handles APIs and logic
- Connects all components

Easy Real-Life Example

Tomcat is like a **kitchen in a restaurant**:

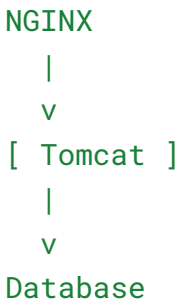
- Takes order
- Prepares food
- Sends food to customer

Simple Example

For SMS:

- Validates mobile number
- Checks balance
- Pushes message to Kafka

Architecture Diagram



6. Jenkins

What is Jenkins?

Jenkins is an **automation tool** used for building and deploying software.

What does Jenkins automate?

- Code compilation
- Testing
- Deployment to servers

Why is Jenkins needed?

- Avoids manual mistakes
- Faster releases
- Consistent deployments

Easy Real-Life Example

Jenkins is like a **washing machine**:

- Clothes go in
- Wash, rinse, dry happens automatically
- No manual effort every time

Simple Example

When developer updates code:

- Jenkins builds it
- Tests it
- Deploys to server automatically

Architecture Diagram

```
Developer Code
     |
     v
  Jenkins
     |
     v
  Build → Test → Deploy
```

7. CI/CD (Continuous Integration & Continuous Deployment)

What is CI (Continuous Integration)?

- Developers regularly push code
- Code is automatically tested

What is CD (Continuous Deployment)?

- Tested code is automatically deployed to server

Why CI/CD is important?

- Faster development
- Fewer bugs
- Reliable deployments

Real-Life Example

Easy Real-Life Example

CI/CD is like a **factory assembly line**:

- Every part is checked
- Faulty items are rejected early
- Final product is delivered automatically

Simple Example

For SMS platform:

- Code change → test → deploy without downtime

HOW ALL COMPONENTS WORK TOGETHER (SMS EXAMPLE)

1. **Client sends SMS request**
2. **NGINX receives request**
3. **Tomcat validates and processes**
4. **Redis checks cache (OTP, limits)**
5. **MySQL stores message record**
6. **Kafka queues the message**
7. **SMS sent to operator**
8. **DLR received and updated**

CI/CD Flow Diagram

```
Code Commit
    |
    v
 Automated Test
    |
    v
 Build
    |
    v
 Deploy to Server
```

COMPLETE SMS SENDING FLOW (IMPORTANT)

SMS Architecture Overview

```
Client / API User
      |
      v
   NGINX
      |
      v
   Tomcat
      |
      +--> Redis (Cache)
      |
      +--> MySQL (Store Data)
      |
      v
   Kafka
      |
      v
 SMSC / Operator
```

SMS SUBMIT → ACK → DLR FLOW

1. SMS Submit Flow

```
Client
  |
  v
NGINX
  |
  v
Tomcat (Validate & Process)
  |
  v
Kafka (Queue)
  |
  v
SMSC / Operator
```

2. ACK (Acknowledgement) Flow

```
SMSC
  |
  v
Kafka
  |
  v
Tomcat
  |
  v
Client (ACK Received)
```

3. DLR (Delivery Report) Flow

```
User Mobile
   |
   v
 Operator
   |
   v
 Kafka
   |
   v
 Tomcat
   |
   +--> Update MySQL
   |
   v
 Client Receives DLR
```

SMPP vs HTTP FLOW (SHORT COMPARISON)

SMPP Flow

```
Client (SMPP Bind)
   |
   v
SMSC
   |
   v
DLR via SMPP
```

HTTP Flow

```
Client
  |
  v
HTTP API
  |
  v
Application
  |
  v
SMSC
```

SUMMARY (ONE-LINE EACH)

- **NGINX**: Entry gate and traffic controller
- **Kafka**: Message highway
- **MySQL**: Permanent storage
- **Redis**: Fast cache memory
- **Tomcat**: Application brain
- **Jenkins**: Automation engine
- **CI/CD**: Automated delivery pipeline