Kafka Full Guide - All Details Included

1. What is Kafka?

Kafka is an open-source distributed event streaming platform designed to handle high-throughput, fault-tolerant, real-time data streams. It is widely used for event-driven architectures, data pipelines, and microservices communication.

2. Kafka Architecture & Components

Kafka's architecture consists of:
- **Producers:** Applications that send messages to Kafka.
- **Topics & Partitions:** Kafka topics are divided into partitions for scalability.
- **Consumers & Consumer Groups:** Consumers read messages in parallel.
- **Brokers & Clusters:** A Kafka cluster is composed of multiple brokers.
- **ZooKeeper:** Manages metadata, leader election, and broker coordination.

3. Kafka Producer

Kafka producers send messages to topics. They can control partitioning, acks, and message compression.
Example of sending messages using Kafka Producer CLI:

```
bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092
>Hello Kafka!
>Hello Again!
```

4. Kafka Consumer

Kafka consumers read messages from topics. They use consumer groups to parallelize message processing.
Example of consuming messages from the beginning:

```
bin/kafka-console-consumer.sh --topic my-topic --from-beginning --bootstrap-server localhost:9092
```

5. Creating a Kafka Topic

Kafka topics store messages and are divided into partitions. To create a topic:

```
bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1
```

6. Listing Kafka Topics

To list all available topics in Kafka:

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

7. Kafka Streams API

Kafka Streams API is used for real-time stream processing. Example of transforming a stream:

```
java
KStream<String, String> source = builder.stream("input-topic");
KStream<String, String> transformed = source.mapValues(value -> value.toUpperCase());
transformed.to("output-topic");
```

8. Kafka Connect

Kafka Connect allows integration with external systems like databases, cloud storage, and more.
Example of running a connector:

```
json
{
  "name": "jdbc-source-connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "connection.url": "jdbc:mysql://localhost:3306/mydb",
    "table.whitelist": "users",
    "mode": "incrementing",
    "incrementing.column.name": "id",
    "topic.prefix": "mysql-"
  }
}
```

9. Kafka Security

Kafka security includes authentication (SSL/SASL), authorization (ACLs), and encryption.
Example: Enabling SSL encryption in Kafka:

```
listeners=SSL://:9093
ssl.keystore.location=/etc/kafka/secrets/kafka.keystore.jks
ssl.keystore.password=secret
```

10. Monitoring Kafka

Kafka monitoring can be done using JMX, Prometheus, and Grafana.
Example: Exposing Kafka metrics for Prometheus:

```
KAFKA_OPTS="-javaagent:/usr/prometheus/jmx_prometheus_javaagent-0.3.1.jar=7071:/usr/prom
etheus/kafka.yml"
```

11. Kafka vs RabbitMQ vs ActiveMQ

Comparison of Kafka with RabbitMQ and ActiveMQ:
- **Kafka:** Best for event-driven, high-throughput messaging.
- **RabbitMQ:** Best for request-response messaging and task queues.
- **ActiveMQ:** Best for legacy enterprise messaging systems.

12. Running Kafka in Docker

Example docker-compose file to run Kafka and ZooKeeper in Docker:

```
yaml
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"
  kafka:
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
```

13. Kafka Message Delivery Semantics

Kafka provides three message delivery guarantees:
- **At most once:** Messages may be lost.
- **At least once:** Messages may be duplicated.
- **Exactly once:** Ensures no message loss and no duplication.

14. Advanced Kafka Optimization

Performance tuning for Kafka includes:
- **Producer:** Use batch size, linger.ms, and compression.
- **Consumer:** Increase fetch size and optimize offset commits.
- **Brokers:** Tune segment sizes and retention policies.

15. What's Next?

Once comfortable with Kafka basics, explore:
- Kafka Schema Registry (Avro, Protobuf, JSON)
- Kafka Transactions & Exactly-Once Processing
- Kafka Multi-Cluster Replication (MirrorMaker)
- Kafka in Kubernetes using Strimzi
- Kafka and Flink for real-time analytics