Two Producers and two Consumers KAFKA using zookeeper

1. Create the Topic

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

Verify:

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

2. Start Two Producers (in separate terminals)

Producer 1

```
/opt/kafka/bin/kafka-console-producer.sh --topic test --
bootstrap-server localhost:9092
```

Producer 2

```
/opt/kafka/bin/kafka-console-producer.sh --topic test --
bootstrap-server localhost:9092
```

3. Start Two Consumers (in separate terminals)

Option A: Same Consumer Group (messages load-balanced)

Consumer 1

```
/opt/kafka/bin/kafka-console-consumer.sh --topic test --
bootstrap-server localhost:9092 --group test-group
```

Consumer 2

```
/opt/kafka/bin/kafka-console-consumer.sh --topic test --
bootstrap-server localhost:9092 --group test-group
```

Messages will be split between them.

Option B: Different Consumer Groups (both get all messages)

Consumer 1

```
/opt/kafka/bin/kafka-console-consumer.sh --topic test --
bootstrap-server localhost:9092 --group group1
```

Consumer 2

```
/opt/kafka/bin/kafka-console-consumer.sh --topic test --
bootstrap-server localhost:9092 --group group2
```

Both will see **every message** produced.

Ip addr show

My vm ip 192.168.114.128

If you want to delete a topic

Opt/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic <topic-name>

Kafka Connect with Postgres Connector

Step 1 — Install PostgreSQL on RHEL

```
# Update system packages
sudo dnf update -y

# Install PostgreSQL server & contrib
sudo dnf install -y postgresql-server postgresql-contrib

# Initialize PostgreSQL database
sudo postgresql-setup --initdb

# Start and enable service
sudo systemctl enable postgresql
sudo systemctl start postgresql
```

Step 2 — Configure PostgreSQL for Password Auth

```
sudo vi /var/lib/pgsql/data/pg_hba.conf
```
Change lines like this:

```
local   all    all                                    md5
host    all    all             127.0.0.1/32              md5
host    all    all             ::1/128                   md5
```

Restart PostgreSQL:

```
sudo systemctl restart postgresql
```

Step 3 — Create Database, User, and Table

```
# Login as postgres superuser
sudo -u postgres psql
```

Inside psql:

```
CREATE USER kafkauser WITH PASSWORD 'kafkapass';
CREATE DATABASE kafkadb OWNER kafkauser;

\c kafkadb

CREATE TABLE customer (
    id SERIAL PRIMARY KEY,
    firstname TEXT,
    lastname  TEXT,
    email     TEXT
);

INSERT INTO customer (firstname, lastname, email)
VALUES ('Alice','Wonder','alice@example.com'),
       ('Bob','Marley','bob@example.com');

GRANT CONNECT ON DATABASE kafkadb TO kafkauser;
GRANT USAGE ON SCHEMA public TO kafkauser;
GRANT SELECT ON customer TO kafkauser;
```

```
\q
```
Test connection:

```
PGPASSWORD=kafkapass psql -U kafkauser -d kafkadb -h
localhost -c "SELECT * FROM customer;"
```

Step 4 — Download & Install Kafka

```
# Download Kafka (3.8.0 as example)
curl -O https://downloads.apache.org/kafka/3.8.0/
kafka_2.13-3.8.0.tgz

# Extract
tar -xvzf kafka_2.13-3.8.0.tgz
sudo mv kafka_2.13-3.8.0 /opt/kafka
```

Step 5 — Start Zookeeper & Kafka Broker

```
# Start Zookeeper
/opt/kafka/bin/zookeeper-server-start.sh -daemon /opt/kafka/
config/zookeeper.properties

# Start Kafka broker
/opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/
config/server.properties
```

Step 6 — Create Kafka Topic for Postgres Table

```
/opt/kafka/bin/kafka-topics.sh \
  --create \
  --topic test_customer \
  --bootstrap-server localhost:9092 \
  --partitions 1 \
  --replication-factor 1
```

Check topic:

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

At this point you have:
PostgreSQL installed and ready
Database + user + `customer` table
Kafka installed and running
Kafka topic `test_customer` created

Step 7 — Install JDBC Driver & Connector

1. Create plugin folder:

```
sudo mkdir -p /opt/kafka/plugins/jdbc
```

2. Download PostgreSQL JDBC driver

```
curl -o /tmp/postgresql-42.6.0.jar https://
jdbc.postgresql.org/download/postgresql-42.6.0.jar
sudo mv /tmp/postgresql-42.6.0.jar /opt/kafka/plugins/jdbc/
```

3. Download JDBC Connector (Confluent):

```
curl -L -o /tmp/kafka-connect-jdbc.zip https://
d1i4a15mxbxib1.cloudfront.net/api/plugins/confluentinc/kafka-
connect-jdbc/latest/confluentinc-kafka-connect-jdbc-
latest.zip

sudo unzip -o /tmp/kafka-connect-jdbc.zip -d /opt/kafka/
plugins/jdbc
```

OR
Download Manually

Move file from downloads to jdbc

sudo mv confluentinc-kafka-connect-jdbc-10.8.4 /opt/kafka/plugins/jdbc

4. Add plugin path in Kafka Connect config:

```
echo "plugin.path=/opt/kafka/plugins" | sudo tee -a /opt/
kafka/config/connect-distributed.properties
```

Step 8 — Start Kafka Connect Worker

```
/opt/kafka/bin/connect-distributed.sh /opt/kafka/config/
connect-distributed.properties &
```

Check REST API:

```
curl http://localhost:8083/
```

Step 9 — Create JDBC Source Connector Config

```
Sudo -c 'cat > /opt/kafka/config/jdbc-source.json <<EOF
{
  "name": "jdbc-source-connector",
  "config": {
    "connector.class":
"io.confluent.connect.jdbc.JdbcSourceConnector",
    "tasks.max": "1",
    "connection.url": "jdbc:postgresql://localhost:5432/
kafkadb",
    "connection.user": "kafkauser",
    "connection.password": "kafkapass",
    "table.whitelist": "customer",
    "mode": "incrementing",
    "incrementing.column.name": "id",
    "topic.prefix": "test_",
    "poll.interval.ms": "1000"
  }
}
EOF'
```

Deploy connector:

```
curl -X POST -H "Content-Type: application/json" \
    --data @/opt/kafka/config/jdbc-source.json \
    http://localhost:8083/connectors
```

Check status:

```
curl http://localhost:8083/connectors/jdbc-source-connector/
status
```

Step 10 — Consume Data from Kafka

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

You should now see rows from the Postgres `customer` table inside Kafka.

Step 11 — Prepare PostgreSQL Sink Table

We'll create a table to store Kafka messages coming **from a Kafka topic**.

```
sudo -u postgres psql -d kafkadb
```

Inside psql:

```
CREATE TABLE customer_sink (
    id SERIAL PRIMARY KEY,
    firstname TEXT,
    lastname  TEXT,
    email     TEXT
);

GRANT INSERT, UPDATE, SELECT ON customer_sink TO kafkauser;

\q
```

Step 12 — Produce a Test Message into Kafka

We'll manually put some messages into a new Kafka topic `test_customer_sink`.

```
/opt/kafka/bin/kafka-topics.sh \
  --create \
  --topic test_customer_sink \
  --bootstrap-server localhost:9092 \
  --partitions 1 \
  --replication-factor 1
```

Start a producer:

```
/opt/kafka/bin/kafka-console-producer.sh \
    --broker-list localhost:9092 \
    --topic test_customer_sink
```

Type a JSON message (then press Enter):

```
{"firstname":"Charlie","lastname":"Brown","email":"charlie@ex
ample.com"}
```

(Leave the producer running for now or exit with `Ctrl+C`.)

Step 13 — Create JDBC Sink Connector Config

```
sudo tee /opt/kafka/config/jdbc-sink.json > /dev/null <<EOF
{
  "name": "jdbc-sink-connector",
  "config": {
    "connector.class":
"io.confluent.connect.jdbc.JdbcSinkConnector",
    "tasks.max": "1",
    "connection.url": "jdbc:postgresql://localhost:5432/
kafkadb",
    "connection.user": "kafkauser",
    "connection.password": "kafkapass",
    "topics": "test_customer_sink",
    "auto.create": "false",
    "auto.evolve": "false",
    "insert.mode": "insert",
    "table.name.format": "customer_sink"
  }
}
EOF
```

This will save the file at:
`/opt/kafka/config/jdbc-sink.json`

You can verify with:

```
sudo cat /opt/kafka/config/jdbc-sink.json
```

Deploy connector:

```
curl -X POST -H "Content-Type: application/json" \
    --data @~/jdbc-sink.json \
    http://localhost:8083/connectors
```

Check status:

```
curl http://localhost:8083/connectors/jdbc-sink-connector/
status
```

Step 14 — Verify Data in PostgreSQL

Now check if Kafka messages landed in PostgreSQL:

```
PGPASSWORD=kafkapass psql -U kafkauser -d kafkadb -h
localhost -c "SELECT * FROM customer_sink;"
```

You should see:

```
 id | firstname | lastname |        email
----+-----------+----------+---------------------
  1 | Charlie   | Brown    | charlie@example.com
```

KAFKA MONITORING USING Prometheus And Grafana

2. start zookeeper

```
cd /opt/kafka
bin/zookeeper-server-start.sh config/zookeeper.properties
```

leave this running in one terminal (or run it as a systemd service if you want it permanent).

3. start kafka broker (with jmx exporter)

1. download jmx exporter:

```
cd /opt
```

```
wget https://repo1.maven.org/maven2/io/prometheus/jmx/
jmx_prometheus_javaagent/0.21.0/
jmx_prometheus_javaagent-0.21.0.jar
```

2. create a jmx config:

```
cat <<EOF > /opt/kafka-jmx.yml
rules:
  - pattern: "kafka.server<type=(.+), name=(.+)><>Value"
    name: "kafka_\$1_\$2"
    type: GAUGE
EOF
```

3. start kafka with jmx agent (port **7071**):

```
cd /opt/kafka
export KAFKA_OPTS="-javaagent:/opt/
jmx_prometheus_javaagent-0.21.0.jar=7071:/opt/kafka-jmx.yml"
bin/kafka-server-start.sh config/server.properties
```

4. install prometheus

```
cd /opt
wget https://github.com/prometheus/prometheus/releases/
download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
tar -xvzf prometheus-2.52.0.linux-amd64.tar.gz
mv prometheus-2.52.0.linux-amd64 prometheus
```

create prometheus config:

```
sudo cat <<EOF > /opt/prometheus/prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets: ['localhost:7071']
EOF
```

run prometheus:

```
cd /opt/prometheus
./prometheus –config.file=prometheus.yml
```

open http://localhost:9090

Check If 9090 is Occupied

ss -lntp | grep 9090. or

netstat -tulnp | grep 9090

Run on port 9095

```
cd /opt/prometheus
```

```
./prometheus --config.file=prometheus.yml --web.listen-address="0.0.0.0:9095"
```

curl -v http://localhost:9095

5. install grafana

sudo wget https://dl.grafana.com/oss/release/grafana-12.1.1.linux-arm64.tar.gz

sudo tar -xvzf grafana-12.1.1.linux-arm64.tar.gz

sudo mv grafana-12.1.1 grafana

cd /opt/grafana

sudo ./bin/grafana-server

http://localhost:3000

login: **admin / admin**

6. connect grafana to prometheus

1. go to **grafana → settings → data sources → add data source → prometheus**

2. set url: `http://localhost:9095`
3. save & test

7. import kafka dashboards

- go to **dashboards → import**
- use **id 7587** (kafka overview)
- use **id 10466** (consumer lag)

summary of services

- **zookeeper** → `bin/zookeeper-server-start.sh config/zookeeper.properties`
- **kafka** → `bin/kafka-server-start.sh config/server.properties` (with jmx agent)
- **prometheus** → `/opt/prometheus/prometheus --config.file=prometheus.yml`
- **grafana** → `systemctl start grafana-server`

Step-by-Step Setup: Prometheus + Postgres Exporter(Optional)

1. install postgres exporter

```
cd /opt
wget https://github.com/prometheus-community/postgres_exporter/releases/download/v0.15.0/postgres_exporter-0.15.0.linux-amd64.tar.gz
tar -xvzf postgres_exporter-0.15.0.linux-amd64.tar.gz
mv postgres_exporter-0.15.0.linux-amd64 postgres_exporter
```

2. create monitoring user in postgres

switch to postgres and create a dedicated user:

sudo -i -u postgres psql

inside psql:

-- create db and user

CREATE DATABASE kafkadb;

CREATE USER kafkauser WITH PASSWORD 'kafkapass';

-- grant privileges

GRANT ALL PRIVILEGES ON DATABASE kafkadb TO kafkauser;

-- give monitoring role (needed for exporter)

ALTER USER kafkauser WITH SUPERUSER;

exit with \q.

2. configure postgres exporter

go to exporter folder:

cd /opt/postgres_exporter

set connection string (DSN) for your kafka DB:

export DATA_SOURCE_NAME="postgresql://kafkauser:kafkapass@localhost:5432/kafkadb?sslmode=disable"

run exporter on port 9187:

/opt/postgres_exporter/postgres_exporter

./postgres_exporter —web.listen-address="0.0.0.0:9187"

test:

curl http://localhost:9187/metrics | head

3. add scrape job in prometheus

sudo tee /opt/prometheus/prometheus.yml > /dev/null <<'EOF'

global:

  scrape_interval: 15s

scrape_configs:

  - job_name: 'kafka'

    static_configs:

      - targets: ['localhost:7071']

  - job_name: 'postgres-kafka'

    static_configs:

      - targets: ['localhost:9187']

EOF

restart prometheus:

```
sudo systemctl restart prometheus
sudo systemctl status prometheus
```

4. grafana dashboard

- go to **Grafana → Dashboards → Import**
- import dashboard ID **9628** (Postgres Exporter)
- set datasource = Prometheus
- you'll see stats for `kafkadb`.