1. Install OpenJDK 21 on RHEL

RHEL 9 provides **OpenJDK 21** via AppStream.

```
# Update packages
sudo dnf update -y

# Search available JDKs
sudo dnf search openjdk

# Install JDK 21
sudo dnf install -y java-21-openjdk java-21-openjdk-devel
```

Verify:

```
java -version
javac -version
```

Expected output:

```
openjdk version "21"
```

Set JAVA_HOME (optional but recommended):

```
JAVA_BIN=$(readlink -f $(which java))
JAVA_HOME=$(dirname $(dirname "$JAVA_BIN"))
echo "export JAVA_HOME=$JAVA_HOME" | sudo tee /etc/profile.d/java.sh
sudo chmod +x /etc/profile.d/java.sh
source /etc/profile.d/java.sh
```

Download Kafka 3.8.0

download the latest stable version, **Kafka 3.8.0**, which is fully compatible with OpenJDK 21 and supports ZooKeeper.

1. **Download Kafka 3.8.0:**

   ```
   cd /tmp
   ```

2. `wget https://downloads.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz`

3. **Extract the archive:**

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

4. `sudo tar -xzf kafka_2.13-3.8.0.tgz -C /opt/kafka --strip-components=1`

5. **Set appropriate permissions:**

   ```
   sudo useradd -r -m -s /sbin/nologin kafka
   ```

6. `sudo chown -R kafka:kafka /opt/kafka`

7. **Start ZooKeeper:**

   ```
   sudo -u kafka /opt/kafka/bin/zookeeper-server-start.sh -daemon /opt/kafka/config/zookeeper.properties
   ```

8. **Start Kafka Broker:**

   ```
   sudo -u kafka /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties
   ```

9. **Verify Kafka is running:**

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

10. **Create a test topic:**

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

11. **Produce a message to the test topic:**

    ```
    sudo -u kafka /opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
    ```

    ```
    Type a message and press Enter.
    ```

12. **Consume the message from the test topic:**

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh
--bootstrap-server localhost:9092 --topic test –from-
beginning
```

(OPTIONAL) set up systemd services for ZooKeeper and Kafka on your RHEL VM so they auto-start at boot.

Create a systemd unit for ZooKeeper

```
sudo tee /etc/systemd/system/zookeeper.service > /dev/null
<<EOF
[Unit]
Description=Apache ZooKeeper server
After=network.target

[Service]
Type=simple
User=kafka
ExecStart=/opt/kafka/bin/zookeeper-server-start.sh /opt/
kafka/config/zookeeper.properties
ExecStop=/opt/kafka/bin/zookeeper-server-stop.sh
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
EOF
```

Create a systemd unit for Kafka Broker

```
sudo tee /etc/systemd/system/kafka.service > /dev/null <<EOF
[Unit]
Description=Apache Kafka server
After=zookeeper.service

[Service]
Type=simple
```

```
User=kafka
ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/
config/server.properties
ExecStop=/opt/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
EOF
```

Reload systemd and enable services

```
sudo systemctl daemon-reload
sudo systemctl enable zookeeper
sudo systemctl enable kafka
```

Start the services

```
sudo systemctl start zookeeper
sudo systemctl start kafka
```

Verify they are running

```
sudo systemctl status zookeeper
sudo systemctl status kafka
```

You should see `active (running)` for both.

Test Kafka

Create a topic:

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

Produce messages:

```
sudo -u kafka /opt/kafka/bin/kafka-console-producer.sh --
broker-list localhost:9092 --topic test
```

Consume messages:

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh --
bootstrap-server localhost:9092 --topic test —from-beginning
```

Step-by-Step Setup: Kafka in KRaft Mode on RHEL

Prepare Kafka installation

If you already downloaded & extracted Kafka 3.8.0:

```
cd /tmp
wget https://downloads.apache.org/kafka/3.8.0/
kafka_2.13-3.8.0.tgz

sudo mkdir -p /opt/kafka
sudo tar -xzf kafka_2.13-3.8.0.tgz -C /opt/kafka --strip-
components=1

sudo useradd -r -m -s /sbin/nologin kafka
sudo chown -R kafka:kafka /opt/kafka
```

Create KRaft config file

Kafka provides a template in `config/kraft/`.
We'll use `server.properties`.

```
sudo cp /opt/kafka/config/kraft/server.properties /opt/kafka/
config/kraft/server.properties.backup
```

Now edit the config:

```
sudo tee /opt/kafka/config/kraft/server.properties > /dev/
null <<EOF
process.roles=broker,controller
node.id=1
controller.quorum.voters=1@localhost:9093
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
```

```
advertised.listeners=PLAINTEXT://localhost:9092
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT
:PLAINTEXT
inter.broker.listener.name=PLAINTEXT
log.dirs=/tmp/kraft-combined-logs
EOF
```

Format the storage directory

Kafka requires formatting before the first start.

```
sudo -u kafka /opt/kafka/bin/kafka-storage.sh format \
  -t $(/opt/kafka/bin/kafka-storage.sh random-uuid) \
  -c /opt/kafka/config/kraft/server.properties
```

Start Kafka manually (test)

```
sudo -u Kafka /opt/kafka/bin/kafka-server-start.sh -daemon /
opt/kafka/config/kraft/server.properties
```

Verify:

```
ps -ef | grep kafka
```

Create a systemd service for Kafka (KRaft mode) (OPTIONAL)

```
sudo tee /etc/systemd/system/kafka.service > /dev/null <<EOF
[Unit]
Description=Apache Kafka (KRaft mode)
After=network.target

[Service]
Type=simple
User=Kafka
ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/
config/kraft/server.properties
ExecStop=/opt/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal
LimitNOFILE=100000

[Install]
```

```
WantedBy=multi-user.target
EOF
```

Enable and start the service

```
sudo systemctl daemon-reload
sudo systemctl enable kafka
sudo systemctl start kafka
sudo systemctl status kafka
```

Test Kafka

Create a topic:

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

Produce messages:

```
sudo -u kafka /opt/kafka/bin/kafka-console-producer.sh --
broker-list localhost:9092 --topic test
```

Consume messages:

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh --
bootstrap-server localhost:9092 --topic test --from-beginning
```

Running **Kafka in KRaft mode** without ZooKeeper.

Multi-node Kafka Cluster in KRaft Mode

Create **3 nodes** (broker + controller roles), but you can scale to 5 or more.

Prepare directories on each node

Run these on **all nodes**:

```
sudo mkdir -p /opt/kafka
sudo useradd -r -m -s /sbin/nologin kafka
sudo chown -R kafka:kafka /opt/kafka
```

Download Kafka **3.8.0**

```
cd /tmp
wget https://downloads.apache.org/kafka/3.8.0/
kafka_2.13-3.8.0.tgz
sudo tar -xzf kafka_2.13-3.8.0.tgz -C /opt/kafka --strip-
components=1
sudo chown -R kafka:kafka /opt/kafka
```

Assign node IDs and roles

Each server must have a **unique node.id**.

- Node 1 → `node.id=1`
- Node 2 → `node.id=2`
- Node 3 → `node.id=3`

All will act as **broker+controller**.

Edit server.properties on each node

Example (Node 1 at `192.168.1.101`):

```
sudo tee /opt/kafka/config/kraft/server.properties > /dev/
null <<EOF
process.roles=broker,controller
node.id=1
controller.quorum.voters=1@192.168.1.101:9093,2@192.168.1.102
:9093,3@192.168.1.103:9093
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
advertised.listeners=PLAINTEXT://192.168.1.101:9092
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT
:PLAINTEXT
inter.broker.listener.name=PLAINTEXT
log.dirs=/tmp/kraft-combined-logs
EOF
```

- Change `node.id` and `advertised.listeners` for each node.

- Node 2 → `node.id=2`, `192.168.1.102`
- Node 3 → `node.id=3`, `192.168.1.103`

Format storage on each node

Each node must be formatted **with the same cluster ID**.

Generate a cluster ID (on one node):

```
/opt/kafka/bin/kafka-storage.sh random-uuid
```

Suppose it prints:

```
pKxQ8XJZQ9yV2Yt3p9Xh4A
```

Format all nodes with the same ID:

```
sudo -u kafka /opt/kafka/bin/kafka-storage.sh format \
  -t pKxQ8XJZQ9yV2Yt3p9Xh4A \
  -c /opt/kafka/config/kraft/server.properties
```

Start Kafka on all nodes

You can start manually first:

```
sudo -u kafka /opt/kafka/bin/kafka-server-start.sh -daemon /
opt/kafka/config/kraft/server.properties
```

Or via `systemd` (like we set up earlier).

Verify the cluster

Check logs (`/opt/kafka/logs/`) for:

```
KafkaServer id=1 started (kafka.server.KafkaServer)
```

Verify controllers:

```
sudo -u kafka /opt/kafka/bin/kafka-metadata-quorum.sh --
bootstrap-server 192.168.1.101:9092 describe --status
```

You should see a leader controller and other voters.

Test Kafka Cluster

Create a topic (on any node):

```
sudo -u kafka /opt/kafka/bin/kafka-topics.sh --create \
  --topic test \
  --bootstrap-server 192.168.1.101:9092 \
  --partitions 3 \
  --replication-factor 3
```

Produce messages:

```
sudo -u kafka /opt/kafka/bin/kafka-console-producer.sh --
broker-list 192.168.1.101:9092 --topic test
```

Consume messages (from another node):

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh --
bootstrap-server 192.168.1.102:9092 --topic test –from-
beginning
```

Firewall & SELinux Configuration for Kafka Cluster on RHEL

Open required ports on all nodes

Kafka in KRaft mode needs:

- **9092** → Client connections (Producers/Consumers, Brokers)
- **9093** → Controller quorum communication (between brokers/controllers)

Run this on **each RHEL node**:

```
sudo firewall-cmd --permanent --add-port=9092/tcp
sudo firewall-cmd --permanent --add-port=9093/tcp
sudo firewall-cmd --reload
```

Verify:

```
sudo firewall-cmd --list-ports
```
Expected:

```
9092/tcp 9093/tcp
```

Configure SELinux (if enabled)

By default, SELinux may block Kafka since it runs outside system default paths.

Check SELinux mode

```
getenforce
```

- If `Enforcing` → need rules
- If `Permissive` → already less strict
- If `Disabled` → nothing needed

Allow Kafka ports in SELinux

```
sudo semanage port -a -t http_port_t -p tcp 9092
sudo semanage port -a -t http_port_t -p tcp 9093
```
If you see `ValueError: Port already defined`, use `-m` instead of `-a`:

```
sudo semanage port -m -t http_port_t -p tcp 9092
sudo semanage port -m -t http_port_t -p tcp 9093
```

Allow Kafka to bind to network

If Kafka is installed in `/opt/kafka`, you may need to set the right SELinux context:

```
sudo chcon -R -t bin_t /opt/kafka/bin
sudo chcon -R -t usr_t /opt/kafka/config
```
Or to avoid issues in a dev/test cluster, you can set SELinux to permissive mode temporarily:

```
sudo setenforce 0
```
Make it persistent only if necessary (not recommended for production):

```
sudo sed -i 's/^SELINUX=enforcing/SELINUX=permissive/' /etc/
selinux/config
```

Verify connectivity between nodes

From **Node 1** → test connection to Node 2's Kafka port:

```
nc -zv 192.168.1.102 9092
nc -zv 192.168.1.102 9093
```

From **Node 2** → **Node 3**, repeat the test.

If all succeed → nodes can talk to each other.

Restart Kafka services

```
sudo systemctl restart kafka
sudo systemctl status kafka
```

Confirm quorum status

Run on any node:

```
sudo -u kafka /opt/kafka/bin/kafka-metadata-quorum.sh --
bootstrap-server 192.168.1.101:9092 describe --status
```

You should see something like:

```
CurrentVoters: [1,2,3]
LeaderId: 2
```

multi-node Kafka cluster is properly networked & SELinux/firewall ready on RHEL.

Using Kafka Client on Your Cluster

Create a topic with replication

Run this on **any node** (replace with your broker IP):

```
sudo -u kafka /opt/kafka/bin/kafka-topics.sh --create \
  --topic test-topic \
  --bootstrap-server 192.168.1.101:9092 \
  --partitions 3 \
  --replication-factor 3
```

Check if topic created:

```
sudo -u kafka /opt/kafka/bin/kafka-topics.sh --list \
  --bootstrap-server 192.168.1.101:9092
```

Describe topic:

```
sudo -u kafka /opt/kafka/bin/kafka-topics.sh --describe \
  --topic test-topic \
  --bootstrap-server 192.168.1.101:9092
```

You should see **3 partitions** and **3 replicas**.

Start a producer (send messages)

On **Node 1**:

```
sudo -u kafka /opt/kafka/bin/kafka-console-producer.sh \
  --broker-list 192.168.1.101:9092 \
  --topic test-topic
```

Type a few messages, e.g.

```
Hello from Node 1
Kafka cluster test
```

```
Replication working?
```
Press **Enter** after each line.

Start a consumer (read messages)

On **Node 2**:

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server 192.168.1.102:9092 \
  --topic test-topic \
  —from-beginning
```

You should see the messages from **Node 1** immediately.

Test multi-node failover

Now kill one broker (say Node 3):

```
sudo systemctl stop kafka
```

Re-check topic status:

```
sudo -u kafka /opt/kafka/bin/kafka-topics.sh --describe \
  --topic test-topic \
  --bootstrap-server 192.168.1.101:9092
```

- You should see leader partitions **moved** to the remaining brokers.
- Messages should still be produced/consumed.

Advanced: Check Quorum Status

On any node:

```
sudo -u kafka /opt/kafka/bin/kafka-metadata-quorum.sh \
  --bootstrap-server 192.168.1.101:9092 describe --status
```

Expected output (example):

```
CurrentVoters: [1,2,3]
LeaderId: 2
HighWatermark: ...
```
This confirms your **controller quorum is healthy**.

Optional: Check Consumer Groups

Run a consumer group:

```
sudo -u kafka /opt/kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server 192.168.1.101:9092 \
  --topic test-topic \
  --group test-group \
  –from-beginning
```

List consumer groups:

```
sudo -u kafka /opt/kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server 192.168.1.101:9092 --list
```
Describe offsets:

```
sudo -u kafka /opt/kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server 192.168.1.101:9092 \
  --describe --group test-group
```

Monitoring Kafka Cluster on RHEL

Check Kafka Logs (basic monitoring)

Kafka writes logs to `/opt/kafka/logs`.
Check for errors:

```
tail -f /opt/kafka/logs/server.log
```
Useful for troubleshooting broker startup, leader election, replication.

Enable JMX Monitoring (for metrics)

Kafka exposes metrics via **Java Management Extensions (JMX)**.

Edit the systemd service for Kafka (`/etc/systemd/system/kafka.service`):

```
[Service]
Environment="KAFKA_HEAP_OPTS=-Xmx1G -Xms1G"
Environment="KAFKA_JMX_OPTS=-Dcom.sun.management.jmxremote \
 -Dcom.sun.management.jmxremote.authenticate=false \
 -Dcom.sun.management.jmxremote.ssl=false \
 -Dcom.sun.management.jmxremote.port=9999 \
 -Djava.rmi.server.hostname=192.168.1.101”
```

Replace `192.168.1.101` with the broker's IP.

Reload and restart:

```
sudo systemctl daemon-reexec
sudo systemctl restart kafka
```

Now Kafka is exposing metrics on **JMX port 9999**.

Install Prometheus JMX Exporter

Prometheus can scrape Kafka metrics via JMX.

1. **Download exporter:**

```
cd /opt
sudo wget https://repo1.maven.org/maven2/io/prometheus/jmx/
jmx_prometheus_javaagent/1.0.1/
jmx_prometheus_javaagent-1.0.1.jar
```

2. **Create config file `/opt/kafka/jmx_config.yml`:**

```
rules:
  - pattern: "kafka.server<type=(.+), name=(.+)PerSec\
\w*><>Count"
    name: kafka_$1_$2_total
    help: Kafka metric $1 $2
    type: COUNTER
  - pattern: "kafka.server<type=(.+), name=(.+)><>(Value)"
    name: kafka_$1_$2
```

```
    help: Kafka metric $1 $2
type: GAUGE
```

Modify Kafka service to include exporter:

```
Environment="KAFKA_OPTS=-javaagent:/opt/
jmx_prometheus_javaagent-1.0.1.jar=7071:/opt/kafka/
jmx_config.yml"
```

Restart Kafka:

```
sudo systemctl restart kafka
```

Now Kafka metrics are available at `http://192.168.1.101:7071/metrics`.

Set up Prometheus

Install Prometheus:

```
sudo dnf install -y prometheus
```

Edit config `/etc/prometheus/prometheus.yml`:

```
scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets:
['192.168.1.101:7071','192.168.1.102:7071','192.168.1.103:707
1']
```

Restart Prometheus:

```
sudo systemctl enable prometheus —now
```

Verify in browser:

```
http://<RHEL-IP>:9090
```

Add Grafana (for visualization)

Install Grafana:

```
sudo dnf install -y grafana
sudo systemctl enable grafana-server —now
```

Open in browser:

```
http://<RHEL-IP>:3000
```

Login with default:

- user: `admin`
- pass: `admin`

Add Prometheus as a datasource:

- URL: `http://localhost:9090`

Import Kafka Dashboards

In Grafana:

1. Go to **Dashboards → Import**
2. Use Kafka dashboard IDs from Grafana.com:
    - **ID 7589** → Kafka Overview
    - **ID 721** → JVM metrics
    - **ID 11271** → Kafka Cluster detailed

Now you can monitor:

- Broker health
- Partition leader distribution
- Under-replicated partitions
- Producer/Consumer throughput
- JVM heap usage

You have a **full monitoring stack**:

- Logs (`server.log`)
- JMX (raw metrics)
- Prometheus (scraping metrics)
- Grafana (beautiful dashboards