complete step-by-step Schema Registry setup for **RHEL VM** with Kafka 3.8 and ZooKeeper, using **sudo for all commands**.

1. Update system & install prerequisites

```
sudo dnf -y update
sudo dnf -y install java-17-openjdk wget tar curl jq nano
```

sudo dnf install -y wget

2. Download and extract Confluent Platform

```
cd /opt
sudo wget https://packages.confluent.io/archive/7.6/
confluent-7.6.0.tar.gz
sudo tar -xvzf confluent-7.6.0.tar.gz
sudo mv confluent-7.6.0 confluent
```

3. Start ZooKeeper

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

4. Start Kafka broker

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

5. Configure Schema Registry

1. copy default config:

```
cd /opt/confluent
sudo cp etc/schema-registry/schema-registry.properties .
```

2. edit config:

```
sudo nano schema-registry.properties
```
update lines:

```
listeners=http://0.0.0.0:8081
kafkastore.bootstrap.servers=PLAINTEXT://localhost:9092
kafkastore.topic=_schemas
```
save & exit.

6. Start Schema Registry

```
cd /opt/confluent
sudo bin/schema-registry-start etc/schema-registry/schema-registry.properties &
```

7. Verify Schema Registry

```
sudo curl http://localhost:8081/subjects
```

- returns `[ ]` if no schemas are registered yet.

8. Register a test schema

1. create schema file:

```
sudo tee /opt/confluent/user.avsc > /dev/null <<'EOF'
{
  "type": "record",
  "name": "User",
  "namespace": "com.example",
  "fields": [
    {"name": "id", "type": "int"},
    {"name": "name", "type": "string"},
    {"name": "email", "type": "string"}
  ]
}
EOF
```

2. **register schema:**

```
sudo curl -X POST -H "Content-Type: application/
vnd.schemaregistry.v1+json" \
--data "{\"schema\": $(jq -Rs . < /opt/confluent/user.avsc)}"
\

http://localhost:8081/subjects/User/versions

  http://localhost:8081/subjects/User/versions
```

3. verify registration:

```
sudo curl http://localhost:8081/subjects
```

- should return: `["User"]`

```
TroubleShooting (Optional)
```

Kill any leftover processes

```
sudo pkill -f kafka
sudo pkill -f zookeeper
sudo pkill -f java
```

Ensure directories exist & have correct permissions

```
sudo mkdir -p /opt/kafka/data
sudo mkdir -p /opt/kafka-logs
sudo chown -R $(whoami) /opt/kafka/data /opt/kafka-logs
```

Ensure Java 17+ is installed

```
java -version
```

- if not installed:

```
sudo dnf install -y java-21-openjdk
```

Start ZooKeeper first

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

- wait a few seconds for ZK to initialize:

```
sleep 5
```

Start Kafka broker without JMX

```
sudo env KAFKA_JMX_OPTS= bin/kafka-server-start.sh config/
server.properties &
```

Verify Kafka is running

```
sudo ss -lntp | grep 9092
tail -f /opt/kafka/logs/server.log
```

- Kafka should now be running on **port 9092**
- There should be **no JMX or Prometheus errors**

- `KAFKA_JMX_OPTS=` ensures **no JMX agent is loaded**.
- Make sure
- `listeners=PLAINTEXT://0.0.0.0:9092` in `server.properties`.
- Start **ZooKeeper first**, then Kafka.
- Verify kafka-server-start.sh has no JMX Agents
- Sudo vi opt/kafka/bin/kafka-server-start.sh
- Check file save and exit

Step-by-Step setup from the beginning on RHEL/CentOS for a **secure Kafka (SASL_SSL with SCRAM)**, **Schema Registry**, and **ACLs**.

This will give you a fully working environment with a test producer/consumer.

Step 1: Start ZooKeeper

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

Step 2: Generate keystore & truststore for Kafka

```
# Create keystore
sudo keytool -genkey -alias localhost -keyalg RSA -keystore /
opt/kafka/secrets/kafka.server.keystore.jks \
  -storepass brokerpass -keypass brokerpass -dname
"CN=localhost"

# Create truststore
sudo keytool -keystore /opt/kafka/secrets/
kafka.server.truststore.jks \
  -alias CARoot -import -file <(keytool -export -alias
localhost -keystore /opt/kafka/secrets/
kafka.server.keystore.jks -storepass brokerpass -rfc) \
  -storepass brokerpass -noprompt
```

Step 3: Configure `server.properties`

```
sudo tee /opt/kafka/config/server.properties > /dev/null
<<'EOF'
broker.id=1
listeners=SASL_SSL://localhost:9093
advertised.listeners=SASL_SSL://localhost:9093
listener.security.protocol.map=SASL_SSL:SASL_SSL
inter.broker.listener.name=SASL_SSL

log.dirs=/opt/kafka/logs
zookeeper.connect=localhost:2181
```

```
# SSL
ssl.keystore.location=/opt/kafka/secrets/
kafka.server.keystore.jks
ssl.keystore.password=brokerpass
ssl.key.password=brokerpass
ssl.truststore.location=/opt/kafka/secrets/
kafka.server.truststore.jks
ssl.truststore.password=brokerpass

# SASL
security.inter.broker.protocol=SASL_SSL
sasl.enabled.mechanisms=SCRAM-SHA-256
EOF
```

Step 4: Create JAAS for Kafka broker

```
sudo tee /opt/kafka/config/kafka_server_jaas.conf > /dev/null
<<'EOF'
KafkaServer {
  org.apache.kafka.common.security.scram.ScramLoginModule
required
  username="kafkauser"
  password="kafkapass";
};
EOF
```

Step 5: Start Kafka broker with JAAS

```
sudo KAFKA_OPTS="-Djava.security.auth.login.config=/opt/
kafka/config/kafka_server_jaas.conf" \
/opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/
config/server.properties
```

Step 6: Create Kafka user (`kafkauser`)

```
sudo tee /opt/kafka/config/client.properties > /dev/null
<<'EOF'
security.protocol=SASL_SSL
sasl.mechanism=SCRAM-SHA-256
ssl.truststore.location=/opt/kafka/secrets/
kafka.server.truststore.jks
ssl.truststore.password=brokerpass
```

```
sasl.jaas.config=org.apache.kafka.common.security.scram.Scram
LoginModule required username="kafkauser"
password="kafkapass";
EOF

sudo /opt/kafka/bin/kafka-configs.sh \
  --bootstrap-server localhost:9093 \
  --alter \
  --add-config 'SCRAM-SHA-256=[password=kafkapass]' \
  --entity-type users \
  --entity-name kafkauser \
  --command-config /opt/kafka/config/client.properties
```

Step 7: Grant ACLs to `kafkauser`

```
sudo /opt/kafka/bin/kafka-acls.sh \
  --authorizer-properties zookeeper.connect=localhost:2181 \
  --add --allow-principal User:kafkauser \
  --operation All --topic '*' --group '*' --cluster
```

Step 8: Test with producer/consumer

```
# Create topic
sudo /opt/kafka/bin/kafka-topics.sh \
  --bootstrap-server localhost:9093 \
  --command-config /opt/kafka/config/client.properties \
  --create --topic test-topic --partitions 1 --replication-
factor 1

# Start producer
sudo KAFKA_OPTS="-Djava.security.auth.login.config=/opt/
kafka/config/kafka_client_jaas.conf" \
/opt/kafka/bin/kafka-console-producer.sh \
  --broker-list localhost:9093 \
  --topic test-topic \
  --producer.config /opt/kafka/config/client.properties

# In another terminal, start consumer
sudo KAFKA_OPTS="-Djava.security.auth.login.config=/opt/
kafka/config/kafka_client_jaas.conf" \
/opt/kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server localhost:9093 \
```

```
  --topic test-topic \
  --from-beginning \
  --consumer.config /opt/kafka/config/client.properties
```

Anything you type in the producer should appear in the consumer.

Step 9: Configure Schema Registry

```
sudo tee /opt/confluent/etc/schema-registry/schema-
registry.properties > /dev/null <<'EOF'
listeners=https://0.0.0.0:8081

kafkastore.bootstrap.servers=SASL_SSL://localhost:9093
kafkastore.security.protocol=SASL_SSL
kafkastore.sasl.mechanism=SCRAM-SHA-256
kafkastore.sasl.jaas.config=org.apache.kafka.common.security.
scram.ScramLoginModule required username="kafkauser"
password="kafkapass";

ssl.truststore.location=/opt/kafka/secrets/
kafka.server.truststore.jks
ssl.truststore.password=brokerpass

schema.registry.ssl.endpoint.identification.algorithm=
EOF
```

Step 10: Start Schema Registry

```
sudo KAFKA_OPTS="-Djavax.net.ssl.trustStore=/opt/kafka/
secrets/kafka.server.truststore.jks
-Djavax.net.ssl.trustStorePassword=brokerpass" \
/opt/confluent/bin/schema-registry-start -daemon /opt/
confluent/etc/schema-registry/schema-registry.properties
```

Step 11: Test Schema Registry

```
curl -k -u kafkauser:kafkapass https://localhost:8081/
subjects
```

Prometheus + Grafana to monitor your **secure Kafka + Schema Registry** setup on RHEL. configure Prometheus to scrape **Kafka JMX metrics** and Schema Registry metrics, then visualize them in Grafana.

Step 1: Enable JMX metrics in Kafka broker

Edit your Kafka startup command or `server.properties` to include:

```
# Export JMX port for Prometheus JMX exporter
export KAFKA_JMX_OPTS="-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.local.only=false \
-Dcom.sun.management.jmxremote.port=9999 \
-Dcom.sun.management.jmxremote.rmi.port=9999 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-Djava.rmi.server.hostname=localhost"
```

Then restart Kafka broker:

```
sudo KAFKA_OPTS="$KAFKA_JMX_OPTS" /opt/kafka/bin/kafka-
server-start.sh -daemon /opt/kafka/config/server.properties
```

Step 2: Download Kafka JMX Exporter

```
sudo mkdir -p /opt/kafka/jmx-exporter
sudo curl -L -o /opt/kafka/jmx-exporter/kafka-jmx-
prometheus.jar \
https://repo1.maven.org/maven2/io/prometheus/jmx/
jmx_prometheus_javaagent/0.18.0/
jmx_prometheus_javaagent-0.18.0.jar

# Sample config
sudo tee /opt/kafka/jmx-exporter/kafka-2_0_0.yml > /dev/null
<<'EOF'
rules:
  - pattern: 'kafka.server<type=(.+), name=(.+)><>Count'
    name: kafka_$1_$2_count
    type: GAUGE
EOF
```

Step 3: Start Kafka with JMX Exporter

```
sudo KAFKA_OPTS="$KAFKA_JMX_OPTS -javaagent:/opt/kafka/jmx-
exporter/kafka-jmx-prometheus.jar=7071:/opt/kafka/jmx-
exporter/kafka-2_0_0.yml" \
/opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/
config/server.properties
```

- This exposes Kafka metrics on **http://localhost:7071/metrics**

Step 4: Configure 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: 'schema_registry'
    metrics_path: '/metrics'
    static_configs:
      - targets: ['localhost:8081']
EOF
```

Step 5: Start Prometheus

```
sudo /opt/prometheus/prometheus --config.file=/opt/
prometheus/prometheus.yml --web.listen-address=:9090 &
```

- Test: **http://localhost:9090/targets**
- You should see Kafka and Schema Registry metrics being scraped.

Step 6: Install and run Grafana

```
# Download Grafana (RHEL 10 example)
sudo dnf install -y https://dl.grafana.com/enterprise/
release/rpm/grafana-enterprise-10.5.2-1.x86_64.rpm

# Start Grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

- Access **http://localhost:3000**
- Default login: `admin/admin`

Step 7: Add Prometheus as data source

1. Login Grafana → Configuration → Data Sources → Add Prometheus
2. URL: `http://localhost:9090`
3. Save & Test → should succeed

Step 8: Import Kafka dashboards

- Grafana has official dashboards:
    - Kafka Overview: `721`
    - Kafka Consumers: `758`
    - Schema Registry: custom JSON dashboard for `/metrics` endpoint
- Import dashboards → view live Kafka/SR metrics

U**ninstall Kafka** from your CentOS (or RHEL) system.

Since Kafka is usually installed from a **tarball** (not as an RPM package), uninstalling is basically **removing the extracted directory + cleaning up configs/data**.

Here's a clean uninstall process

Step 1: Stop Kafka & ZooKeeper

```
ps aux | grep kafka
ps aux | grep zookeeper
```

Kill running processes:

```
sudo pkill -f kafka.Kafka
sudo pkill -f zookeeper
```

Step 2: Remove Kafka installation

If you installed Kafka in `/opt/kafka`:

```
sudo rm -rf /opt/kafka
```

Step 3: Remove Kafka data logs

By default logs are stored in `/tmp/kafka-logs` and ZooKeeper data in `/tmp/zookeeper`.

```
sudo rm -rf /tmp/kafka-logs /tmp/zookeeper
```

If you changed the `log.dirs` path in `server.properties`, delete that directory instead.

Step 4: Remove systemd service (if created)

If you set up Kafka as a `systemd` service, remove it:

```
sudo systemctl stop kafka
sudo systemctl disable kafka
sudo rm -f /etc/systemd/system/kafka.service
```

For ZooKeeper:

```
sudo systemctl stop zookeeper
sudo systemctl disable zookeeper
sudo rm -f /etc/systemd/system/zookeeper.service
```

Then reload systemd:

```
sudo systemctl daemon-reload
```

Step 5: (Optional) Remove Kafka user

If you created a dedicated user `kafka`:

```
sudo userdel -r kafka
```

At this point, Kafka + ZooKeeper are fully removed from your CentOS/RHEL system.