1. Скачиваем архив с кафкой (обратите внимание, что теперь мы скачиваем версию 2.4.0 и 2.7.0). Вы можете пропустить скачивание версии 2.7.0, если она уже есть на вашей виртуальной машине:

```
wget https://archive.apache.org/dist/kafka/2.4.0/kafka_2.13-2.4.0.tgz

tar -xzf kafka_2.13-2.4.0.tgz
wget https://archive.apache.org/dist/kafka/2.7.0/kafka_2.13-2.7.0.tgz

tar -xzf kafka_2.13-2.7.0.tgz
```

2. Первым делом мы запускаем Zookeeper

```
rm -rf /tmp/zookeeper
~/kafka_2.13-2.4.0/bin/zookeeper-server-start.sh
~/kafka_2.13-2.4.0/config/zookeeper.properties
```

3. Запускаем **два** брокера Кафки версии 2.4

```
cat << 'EOF' > kafka-distributed-start.sh
#!/usr/bin/env bash
for i in {0..1}
do
  cp ~/kafka_2.13-2.4.0/config/server.properties
~/kafka_2.13-2.4.0/config/server$i.properties
  sed -i "s/broker.id=0/broker.id=$i/g"
~/kafka_2.13-2.4.0/config/server$i.properties
  echo "listeners=PLAINTEXT://:909$i" >>
~/kafka_2.13-2.4.0/config/server$i.properties
  rm -rf /tmp/kafka-logs$i
  mkdir /tmp/kafka-logs$i
  sed -i "s/log.dirs=\/tmp\/kafka-logs/log.dirs=\/tmp\/kafka-logs$i/g"
~/kafka_2.13-2.4.0/config/server$i.properties
  (~/kafka_2.13-2.4.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server$i.properties &)
done
EOF
chmod +x kafka-distributed-start.sh
./kafka-distributed-start.sh
```

4. Создаем топик

```
~/kafka_2.13-2.4.0/bin/kafka-topics.sh --create --topic registrations
--partitions 1 --bootstrap-server localhost:9091 --replication-factor 2
```

5. Проверяем, что топик создан

```
~/kafka_2.13-2.4.0/bin/kafka-topics.sh --describe --topic registrations
--bootstrap-server localhost:9091 | head -n1
```

6. Опционально – запускаем продюсера и консьюмера. Далее они нам не понадобятся, но по ним мы будем наблюдать, как чувствуют себя клиенты во время обновления кластера (обновление можно проводить без даунтайма).

```
~/kafka_2.13-2.4.0/bin/kafka-run-class.sh
org.apache.kafka.tools.ProducerPerformance  \
--topic=registrations  \
--num-records=100000000 \
--throughput=100 \
--record-size=100 \
--producer-props \
  bootstrap.servers=localhost:9090,localhost:9091 \
  batch.size=64000 \
--print-metrics
~/kafka_2.13-2.4.0/bin/kafka-console-consumer.sh --topic registrations
--bootstrap-server localhost:9090,localhost:9091 --from-beginning
```

7. Мы будем обновлять кластер с версии 2.4.0 до версии 2.7.0, поэтому открываем инструкцию https://kafka.apache.org/documentation/#upgrade_2_7_0.

8. Зафиксируем текущую версию протокола брокеров и проведем rolling restart. Обратите внимание, что запускать мы будем уже бинарный файл версии 2.7.0. Мы будем явно вызывать команду `kill` (аналогичный вызов используется в скрипте `./bin/kafka-server-stop.sh`, но он убивает сразу всех брокеры на сервере. В боевых условиях, когда все брокеры у вас живут на отдельных серверах, лучше использовать скрипт `./bin/kafka-server-stop.sh`). После каждого рестарта будем проверять, что в топике были выбраны лидеры для всех партиций и только после этого переходить к следующему рестарту. Вы также можете ориентироваться на работу клиентов для определения момента, когда возможен рестарт следующего брокера.

```
for i in {0..1}; do echo "inter.broker.protocol.version=2.4" >>
~/kafka_2.13-2.4.0/config/server$i.properties; echo
"log.message.format.version=2.4" >>
~/kafka_2.13-2.4.0/config/server$i.properties; done
```

```
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server0.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server0.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server1.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server1.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
```

9. Обновим версию протокола брокеров и опять проведем rolling restart

```
for i in {0..1}; do sed -i
"s/inter.broker.protocol.version=2.4/inter.broker.protocol.version=2.7/g"
~/kafka_2.13-2.4.0/config/server$i.properties; done
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server0.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server0.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server1.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server1.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
```

10. Обновим версию формата клиентов и проведем последний рестарт. Кстати, ранее вы могли заметить, что в выводе команды `bin/kafka-topics.sh` была явно зафиксирована версия формата событий (`message.format.version=2.4-IV1`), что позволяло нам не переживать по поводу “внезапных” обновлений версии

протокола событий из-за перехода на новый бинарный файл. Здесь нам пригодился шаг с предварительной фиксацией версии 2.4.0.

```
for i in {0..1}; do sed -i
"s/log.message.format.version=2.4/log.message.format.version=2.7/g"
~/kafka_2.13-2.4.0/config/server$i.properties; done
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server0.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server0.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server1.properties' | grep java | grep -v grep | awk '{print $1}' | head -n1) &&
~/kafka_2.13-2.7.0/bin/kafka-server-start.sh
~/kafka_2.13-2.4.0/config/server1.properties &
~/kafka_2.13-2.7.0/bin/kafka-topics.sh \
--describe \
--bootstrap-server localhost:9091 \
--unavailable-partitions
```