Инструкция по ручному развертыванию Kafka в кластерном режиме для Wisla

В данном руководстве описано как подготовить систему и развернуть Kafka в кластерном режиме для работы с Wisla в отказоустойчивом контуре.

Предварительные условия:

1.Архив OpenJDK 17 ()
2.Архив Kafka 2.13-4.1.1

3.Как минимум 3 сервера для развертывания

4.Пользователь wisla

Если пользователь wisla еще не существует в системе его необходимо создать используя следующие команды пошагово.

1.Команда создания пользователя:

```
sudo useradd -d /home/wisla -m -s /bin/bash wisla
```

2.Команда установки пароля:

```
sudo passwd wisla
```

3.Добавление пользователя в файлы sudoers:

```
echo "wisla    ALL=(ALL:ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/wisla
```

4.Настройка системных лимитов для пользователя wisla:

```
# Создаем файл лимитов
cat > /etc/security/limits.d/wisla << 'EOF'
wisla   soft     nofile  32768
wisla   hard     nofile  32768
wisla   soft     nproc   32768
wisla   hard     nproc   32768
EOF
```

5. Добавляем в PAM (если нет):

```
echo "session required pam_limits.so" >> /etc/pam.d/common-session
```

Этап 1. Подготовка каталогов на всех узлах

1.Создание каталогов:

```
sudo mkdir -p /opt/kafka
sudo mkdir -p /var/lib/kafka
sudo mkdir -p /var/log/kafka
```

2.Выдача прав:

```
sudo chown -R wisla:wisla /opt/kafka
sudo chown -R wisla:wisla /var/lib/kafka
sudo chown -R wisla:wisla /var/log/kafka
```

3.Проверка прав:

```
ls -ld /opt/kafka /var/lib/kafka /var/log/kafka
```

Этап 2. Копирование и распаковка Java на всех узлах

1.Скопируем архив Java в домашнюю директорию wisla используя следующую команду:

```
cp OpenJDK17U-jdk_x64_linux_hotspot_17.0.17_10.tar.gz /home/wisla/
```

2.Смена пользователя у архива:

```
sudo chown wisla:wisla /home/wisla/OpenJDK17U-jdk_x64_linux_hotspot_17.0.17_10.tar.gz
```

3.Распаковка архива в /opt/kafka:

- Переходим в каталог kafka используя следующую команду:

  ```
  cd /opt/kafka
  ```

- Распаковка архива:

  ```
  tar -xzf /home/wisla/OpenJDK17U-jdk_x64_linux_hotspot_17.0.17_10.tar.gz
  ```

- Проверка распаковки:

  ```
  ls -la /opt/kafka/
  ```

4.Настройка переменных окружения:

```
export JAVA_HOME="/opt/kafka/jdk-17.0.17+10"
export PATH="$JAVA_HOME/bin:$PATH"
export KAFKA_HOME="/opt/kafka"
```

5.Применение изменений:

```
source ~/.bashrc
```

6.Проверка:

```
java -version
```

Этап 3. Копирование и распаковка Kafka на всех узлах

1.Скопируем архив Kafka в домашнюю директорию wisla используя следующую команду:

```
cp kafka_2.13-4.1.1.tgz /opt/kafka/
```

2.Смена пользователя у архива:

```
sudo chown wisla:wisla /opt/kafka/kafka_2.13-4.1.1.tgz
```

3.Распаковка архива в /opt/kafka:

- Переходим в каталог kafka используя следующую команду:

  ```
  cd /opt/kafka
  ```

- Распаковка архива:

  ```
  tar -xzf kafka_2.13-4.1.1.tgz --strip-components=1
  ```

- Проверка распаковки:

  ```
  ls -la /opt/kafka/bin/
  ```

Этап 4. Генерация уникального ID кластера (только на 1 сервере)

1.Переходим в каталог kafka используя следующую команду:

```
cd /opt/kafka
```

2.Запускаем скрипт генерации ID:

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

Записать полученный ID, например:

Этап 5. Конфигурация кластера на всех узлах

1.Настройка конфигурации на сервере 1:

```
cat > /opt/kafka/config/kraft/server.properties << 'EOF'
# Базовые настройки
process.roles=broker,controller
node.id=1

# Кворум контроллеров (3 узла для отказоустойчивости)
controller.quorum.voters=1@wisla01:9093,2@wisla02:9093,3@wisla03:9093

# Сетевые настройки wisla01
```

```
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
advertised.listeners=PLAINTEXT://wisla01:9092
advertised.controller.listener.name=CONTROLLER

# Директории данных
log.dirs=/var/lib/kafka

# Настройки репликации для 3 узлов
num.partitions=3
default.replication.factor=3
min.insync.replicas=2

# Системные топики (репликация на все узлы)
offsets.topic.replication.factor=3
transaction.state.log.replication.factor=3
transaction.state.log.min.isr=2

# Настройки отказоустойчивости контроллеров
controller.listener.names=CONTROLLER
controller.quorum.election.timeout.ms=1000
controller.quorum.fetch.timeout.ms=2000
controller.quorum.request.timeout.ms=2000

# Настройки брокера
num.io.threads=8
num.network.threads=3
num.replica.fetchers=2

# Настройки логов
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
log.flush.interval.messages=10000
log.flush.interval.ms=1000

# Настройки безопасности (опционально)
# ssl.client.auth=required
# ssl.keystore.location=/path/to/keystore.jks
# ssl.keystore.password=password
# ssl.truststore.location=/path/to/truststore.jks
# ssl.truststore.password=password
EOF
```

2.Настройка конфигурации на сервере 2:

```
cat > /opt/kafka/config/kraft/server.properties << 'EOF'
# Базовые настройки
process.roles=broker,controller
node.id=2

# Кворум контроллеров
controller.quorum.voters=1@node1:9093,2@node2:9093,3@node3:9093

# Сетевые настройки node2
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
```

```
advertised.listeners=PLAINTEXT://node2:9092
advertised.controller.listener.name=CONTROLLER

# Директории данных
log.dirs=/var/lib/kafka

# Настройки репликации
num.partitions=3
default.replication.factor=3
min.insync.replicas=2

# Системные топики
offsets.topic.replication.factor=3
transaction.state.log.replication.factor=3
transaction.state.log.min.isr=2

# Настройки контроллеров
controller.listener.names=CONTROLLER
controller.quorum.election.timeout.ms=1000
controller.quorum.fetch.timeout.ms=2000
controller.quorum.request.timeout.ms=2000

# Настройки брокера
num.io.threads=8
num.network.threads=3
num.replica.fetchers=2

# Настройки логов
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
log.flush.interval.messages=10000
log.flush.interval.ms=1000
EOF
```

3.Настройка конфигурации на сервере 3:

```
cat > /opt/kafka/config/kraft/server.properties << 'EOF'
# Базовые настройки
process.roles=broker,controller
node.id=3

# Кворум контроллеров
controller.quorum.voters=1@wisla01:9093,2@wisla02:9093,3@wisla03:9093

# Сетевые настройки wisla03
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
advertised.listeners=PLAINTEXT://wisla03:9092
advertised.controller.listener.name=CONTROLLER

# Директории данных
log.dirs=/var/lib/kafka

# Настройки репликации
num.partitions=3
```

```
default.replication.factor=3
min.insync.replicas=2

# Системные топики
offsets.topic.replication.factor=3
transaction.state.log.replication.factor=3
transaction.state.log.min.isr=2

# Настройки контроллеров
controller.listener.names=CONTROLLER
controller.quorum.election.timeout.ms=1000
controller.quorum.fetch.timeout.ms=2000
controller.quorum.request.timeout.ms=2000

# Настройки брокера
num.io.threads=8
num.network.threads=3
num.replica.fetchers=2

# Настройки логов
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
log.flush.interval.messages=10000
log.flush.interval.ms=1000
EOF
```

4.Проверка конфигураций:

```
cat /opt/kafka/config/kraft/server.properties
```

Этап 6. Инициализация хранилищ на всех узлах

1.На каждом сервере по очереди

- Экспорт кластер ID:

```
export CLUSTER_ID="wisla-3node-cluster-12345"
```

- Переходим в каталог kafka:

```
cd /opt/kafka
```

- Добавляем кластер ID:

```
./bin/kafka-storage.sh format \
  -t $CLUSTER_ID \
  -c /opt/kafka/config/kraft/server.properties
```

- Проверяем

```
cat /var/lib/kafka/meta.properties
```

Этап 7. Создаем сервис службы Kafka на всех узлах

1.Создаем Unit:

```
sudo cat > /etc/systemd/system/kafka.service << 'EOF'
[Unit]
Description=Apache Kafka Service (Kraft mode) - Wisla 3-Node Cluster
After=network.target
Wants=network.target

[Service]
Type=simple
User=wisla
Group=wisla
WorkingDirectory=/opt/kafka
Environment="JAVA_HOME=/opt/kafka/jdk-17.0.17+10"
Environment="PATH=/opt/kafka/jdk-17.0.17+10/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Environment="CLUSTER_ID=wisla-3node-cluster-12345"
Environment="KAFKA_HEAP_OPTS=-Xmx4G -Xms4G"
Environment="KAFKA_JVM_PERFORMANCE_OPTS=-server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -
XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -Djava.awt.headless=true"
Environment="KAFKA_OPTS=-Dlog4j.configuration=file:/opt/kafka/config/log4j.properties"

# Инициализация хранилища при первом запуске
ExecStartPre=/bin/bash -c "if [ ! -f /var/lib/kafka/meta.properties ]; then echo 'Инициализация хранилища для кластера...' && /opt/kafka/
bin/kafka-storage.sh format -t $CLUSTER_ID -c /opt/kafka/config/kraft/server.properties; fi"

# Запуск Kafka
ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/kraft/server.properties

# Остановка
ExecStop=/opt/kafka/bin/kafka-server-stop.sh

# Логи
StandardOutput=journal
StandardError=journal
SuccessExitStatus=0 143

# Перезапуск при сбоях
Restart=on-failure
RestartSec=10

# Ограничения
LimitNOFILE=65536
LimitNPROC=65536

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

2.Создание конфигурации log4j:

```
cat > /opt/kafka/config/log4j.properties << 'EOF'
log4j.rootLogger=INFO, stdout, kafkaAppender

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%d] %p %m (%c)%n
```

```
log4j.appender.kafkaAppender=org.apache.log4j.DailyRollingFileAppender
log4j.appender.kafkaAppender.DatePattern='.'yyyy-MM-dd-HH
log4j.appender.kafkaAppender.File=/var/log/kafka/kafka.log
log4j.appender.kafkaAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.kafkaAppender.layout.ConversionPattern=[%d] %p %m (%c)%n

log4j.logger.kafka=INFO
log4j.logger.kafka.network.RequestChannel$=WARN
log4j.logger.kafka.server.KafkaApis=WARN
log4j.logger.org.apache.zookeeper=WARN
log4j.logger.org.I0Itec.zkclient=WARN
EOF
```

Этап 8. Создание скриптов управления кластером

1. Создаем скрипт запуска кластера на первом сервере:

```
sudo cat > /usr/local/bin/kafka-cluster-start << 'EOF'
#!/bin/bash
echo "Запуск кластера Kafka на всех узлах..."

# Запуск на локальном узле
systemctl start kafka
echo "Node $(hostname): Kafka запущен"

# Если настроен SSH-доступ к другим узлам, можно добавить:
# ssh wisla@wisla02 "systemctl start kafka"
# ssh wisla@wisla03 "systemctl start kafka"
# echo "wisla02: Kafka запущен"
# echo "wisla03: Kafka запущен"

echo "Кластер Kafka (3 узла) запущен"
echo "Для проверки выполните: kafka-cluster-status"
EOF
```

2.Создаем скрипт остановки кластера на первом сервере:

```
sudo cat > /usr/local/bin/kafka-cluster-stop << 'EOF'
#!/bin/bash
echo "Остановка кластера Kafka..."

# Остановка в правильном порядке (сначала реплики, потом лидеры)
systemctl stop kafka
echo "Node $(hostname): Kafka остановлен"

echo "Кластер Kafka остановлен"
EOF
```

3.Создание скрипта проверки статуса на первом сервере:

```
sudo cat > /usr/local/bin/kafka-cluster-status << 'EOF'
#!/bin/bash
echo "=== Статус кластера Kafka (3 узла) ==="
echo ""
```

```
echo "Локальный узел ($(hostname)):"
systemctl status kafka --no-pager | grep -A 3 "Active:"

echo ""
echo "Для проверки репликации выполните на любом узле:"
echo "  cd /opt/kafka && ./bin/kafka-topics.sh --describe --bootstrap-server wisla01:9092"
echo ""
echo "Для просмотра логов: sudo journalctl -u kafka -f"
EOF
```

4.Создание скрипта управления топиками:

```
sudo cat > /usr/local/bin/kafka-topic-manage << 'EOF'
#!/bin/bash
KAFKA_HOME="/opt/kafka"
BOOTSTRAP_SERVER="wisla01:9092,wisla02:9092,wisla03:9092"

case $1 in
    create)
        echo "Создание топика: $2"
        $KAFKA_HOME/bin/kafka-topics.sh --create \
            --bootstrap-server $BOOTSTRAP_SERVER \
            --replication-factor 3 \
            --partitions 3 \
            --topic $2
        ;;
    list)
        echo "Список топиков:"
        $KAFKA_HOME/bin/kafka-topics.sh --list \
            --bootstrap-server $BOOTSTRAP_SERVER
        ;;
    describe)
        echo "Описание топика: $2"
        $KAFKA_HOME/bin/kafka-topics.sh --describe \
            --bootstrap-server $BOOTSTRAP_SERVER \
            --topic $2
        ;;
    delete)
        echo "Удаление топика: $2"
        $KAFKA_HOME/bin/kafka-topics.sh --delete \
            --bootstrap-server $BOOTSTRAP_SERVER \
            --topic $2
        ;;
    *)
        echo "Использование: $0 {create|list|describe|delete} [topic-name]"
        echo "Пример:"
        echo "  $0 create wisla-topic"
        echo "  $0 list"
        echo "  $0 describe wisla-topic"
        echo "  $0 delete wisla-topic"
        ;;
esac
EOF
```

5.Предоставление прав на исполнение:

```
sudo chmod +x /usr/local/bin/kafka-*
```

6. Настройка /etc/hosts

```
sudo cat >> /etc/hosts << 'EOF'
# Kafka Cluster Nodes
192.168.1.101 Wisla01
192.168.1.102 Wisla02
192.168.1.103 Wisla03
EOF
```

7.Настройка брандмауэра (Опционально):

```
sudo ufw allow 9092/tcp  # Client connections
sudo ufw allow 9093/tcp  # Controller connections
sudo ufw allow 9094/tcp  # Inter-broker connections (если настроены)
```

Этап 9. Запуск кластера

1.Перезагрузка systemd на всех узлах:

```
sudo systemctl daemon-reload
```

2.Добавление в автозапуск на всех узлах:

```
sudo systemctl enable kafka
```

3.Запуск кластера (ВАЖНО: запускать на всех узлах в течение 30 секунд):

```
# На wisla01:
sudo systemctl start kafka

# На wisla02 (в течение 30 секунд после wisla01):
sudo systemctl start kafka

# На wisla03 (в течение 30 секунд после wisla02):
sudo systemctl start kafka
```

4.Проверка статуса службы:

```
sudo systemctl status kafka
```

Дополнительная проверка:
Так же после установки и запуска службы Kafka можно создать скрипт проверки статусов работы кластера Kafka.
1.Создать файл скрипта командой:

```
sudo nano /usr/local/bin/kafka-cluster-monitor
```

2.Предоставить права на исполнение:

```
sudo chmod +x /usr/local/bin/kafka-cluster-monitor
```

3.Добавить скрипт в файл

```
#!/bin/bash
echo "========================================"
```

```
echo "   Мониторинг кластера Kafka (3 узла)   "
echo "========================================="
echo ""

echo "1. СТАТУС СЛУЖБ:"
for node in cluster01 cluster02 cluster03; do
    if [ "$node" = "cluster01" ]; then
        status=$(sudo systemctl is-active kafka 2>/dev/null || echo "unknown")
    else
        status=$(ssh wisla@$node "sudo systemctl is-active kafka 2>/dev/null" 2>/dev/null || echo "unknown")
    fi

    case $status in
        active) color="\e[32m" ;;
        inactive|failed) color="\e[31m" ;;
        unknown) color="\e[33m" ;;
        *) color="\e[33m" ;;
    esac
    echo -e "    $node: ${color}${status}\e[0m"
done

echo -e "\n2. СПИСОК БРОКЕРОВ:"
cd /opt/kafka 2>/dev/null
if [ -f /opt/kafka/bin/kafka-broker-api-versions.sh ]; then
    /opt/kafka/bin/kafka-broker-api-versions.sh \
      --bootstrap-server cluster01:9092 2>/dev/null | grep -E "^cluster" | head -3
else
    echo "   Kafka не установлен локально"
fi

echo -e "\n3. СПИСОК ТОПИКОВ:"
cd /opt/kafka 2>/dev/null
if [ -f /opt/kafka/bin/kafka-topics.sh ]; then
    /opt/kafka/bin/kafka-topics.sh \
      --list --bootstrap-server cluster01:9092 2>/dev/null | head -10
else
    echo "   Kafka не установлен локально"
fi

echo -e "\n4. ПРОВЕРКА РЕПЛИКАЦИИ:"
cd /opt/kafka 2>/dev/null
if [ -f /opt/kafka/bin/kafka-topics.sh ]; then
    result=$(/opt/kafka/bin/kafka-topics.sh \
      --describe --bootstrap-server cluster01:9092 \
      --under-replicated-partitions 2>/dev/null)
    if echo "$result" | grep -q "Topic"; then
        echo "     Есть недореплицированные партиции"
    else
        echo "    Все партиции реплицированы"
    fi
else
    echo "   Kafka не установлен локально"
fi
```

```
echo -e "\n5. ПОСЛЕДНИЕ СОБЫТИЯ:"
for node in cluster01 cluster02 cluster03; do
    if [ "$node" = "cluster01" ]; then
        errors=$(sudo journalctl -u kafka --since '5 minutes ago' 2>/dev/null | grep -E 'ERROR|WARN' | tail -2)
    else
        errors=$(ssh wisla@$node "sudo journalctl -u kafka --since '5 minutes ago' 2>/dev/null | grep -E 'ERROR|WARN' | tail -2" 2>/dev/null)
    fi

    if [ -n "$errors" ]; then
        echo "   $node:"
        echo "$errors" | sed 's/^/     /'
    fi
done

echo -e "\n========================================="
echo "   Кластер готов к работе!"
echo "========================================="
```

4.Запустить скрипт для проверки

```
./kafka-cluster-monitor
```