Над текстом нажмите кнопку "**Создать стенд**". До 10 минут создается стенд. Каждый стенд создан под определенный модуль курса, поэтому при выполнении нижеследующих инструкций, нужно запустить отсюда стенд.

Где получить логин/пароль? В настройках профиля или над текстом по кнопке "**Доступы**".

Практический стенд этого урока состоит из одной ноды. Как зайти? Тут.

1. Копируем архив с кафкой в домашний каталог:

```
cp /opt/kafka_2.13-2.7.0.tgz ~/
```

В случае, если что-то пойдет не так, вы сможете удалить каталог полностью и взять чистый архив для того, чтобы начать практику с начала. Если вы его случайно удалили, можно скачать архив по ссылке:

```
wget https://archive.apache.org/dist/kafka/2.7.0/kafka_2.13-2.7.0.tgz
```

Распакуем архив и перейдем в каталог с приложением:

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

2. Первым делом мы запускаем Zookeeper

```
./bin/zookeeper-server-start.sh config/zookeeper.properties
```

3. Запускаем кластер

```
cat << 'EOF' > kafka-distributed-start.sh
#!/usr/bin/env bash
for i in {0..2}
do
  cp ./config/server.properties "./config/server$i.properties"
  sed -i "s/broker.id=0/broker.id=$i/g" ./config/server$i.properties
  echo "listeners=PLAINTEXT://:909$i" >> ./config/server$i.properties
  mkdir /tmp/kafka-logs$i
  sed -i "s/log.dirs=\/tmp\/kafka-logs/log.dirs=\/tmp\/kafka-logs$i/g"
./config/server$i.properties
  (./bin/kafka-server-start.sh config/server$i.properties &)
done
EOF
```

```
chmod +x kafka-distributed-start.sh
./kafka-distributed-start.sh
```

4. Создаем топик

```
./bin/kafka-topics.sh --create --topic registrations --partitions 1
--bootstrap-server localhost:9092 --replication-factor 2
```

5. Проверяем, что топик создан (обратите внимание на значение Replicas в выводе команды)

```
./bin/kafka-topics.sh --describe --topic registrations --bootstrap-server
localhost:9092
```

6. Запускаем процесс балансировки партиций (в данном случае мы предполагаем, что партиции были созданы на 0 и 1 брокерах. Если у вас партиции были созданы на других брокерах, следует скорректировать значения массива "replicas" в json файле `topic-to-move.json`). Обратите внимание на флаг `--throttle` (bytes/sec). Здесь мы ограничиваем сетевой трафик 100 килобитами. Важно, что мы можем изменить значение троттлинга прямо в момент запущенной балансировки. Это позволяет нам динамически корректировать троттлинг, исходя из сетевых метрик и нагрузки на диски.

```
cat << 'EOF' > topic-to-move.json
{"version":1,"partitions":[{"topic":"registrations","partition":0,"replica
s":[1,2],"log_dirs":["any","any"]}]}
EOF
./bin/kafka-reassign-partitions.sh \
--bootstrap-server localhost:9092 \
--execute \
--throttle 12500 \
--reassignment-json-file topic-to-move.json
```

7. Дожидаемся завершения балансировки партиций. Если балансировка завершена для всех партиций, мы должны получить значение 0 в выводе команды ниже (т.е. у нас не осталось партиций, находящихся в процессе балансировки).

```
./bin/kafka-reassign-partitions.sh \
--bootstrap-server localhost:9092 \
--verify \
```

```
--reassignment-json-file topic-to-move.json | grep "is still in progress"
| wc -l
```

8. Проверяем на каких брокерах находятся реплики (обратите внимание на значение Replicas в выводе команды)

```
./bin/kafka-topics.sh --describe --topic registrations --bootstrap-server
localhost:9092
```

Интересный факт.

Непосредственно в процессе балансировки партиций, вы можете увидеть, что фактор репликации переносимой в данный момент партиции увеличился. Это происходит из-за того, что брокер-контроллер выполняет балансировку посредством добавления новых реплик в список реплик каждой из партиций, увеличивая коэффициент репликации. После этого новые реплики копируют все существующие сообщения для всех партиций с текущей ведущей реплики. В зависимости от размера партиций на диске, эта операция может занять немало времени, так как возникают затраты на копирование данных по сети в новые реплики. После завершения репликации контроллер удаляет старые реплики из списка реплик, уменьшая коэффициент репликации до первоначального значения.

Небольшой трюк.

Скрипт `kafka-reassign-partitions.sh` требует от нас заранее подготовленного списка топиков и партиций для балансировки. Иногда мы хотим провести балансировку для всех топиков в кластере. Это можно легко сделать, например, скриптом ниже. Важно понимать, что это опасная операция, и выполнять ее стоит, только если вы знаете, что делаете.

```
echo '{"topics": [' > topics-to-move.json && ./bin/kafka-topics.sh
--bootstrap-server localhost:9092 --list | perl -ne 'chomp;print
"{\"topic\": \"$_\"},\n"' >> topics-to-move.json && truncate --size=-2
topics-to-move.json && echo '],"version":1}' >>  topics-to-move.json
```

```
./bin/kafka-reassign-partitions.sh \
--generate \
--bootstrap-server localhost:9092 \
--topics-to-move-json-file topics-to-move.json \
--broker-list "0,1,2" > full-reassignment-file.json
cat full-reassignment-file.json | grep version | head -n 1 > rollback.json

cat full-reassignment-file.json | grep version | tail -n 1 >
reassignment.json
./bin/kafka-reassign-partitions.sh \
--bootstrap-server localhost:9092 \
--execute \
--throttle 12500 \
--timeout 300000 \
--reassignment-json-file reassignment.json
./bin/kafka-reassign-partitions.sh \
--bootstrap-server localhost:9092 \
--verify \
--reassignment-json-file reassignment.json
```

9. Останавливаем брокеры и потом Zookeeper

```
kill -s ${SIGNAL:-TERM} $(ps ax | grep -i 'kafka\.Kafka' | grep
'server[0-2].properties' | grep java | grep -v grep | awk '{print $1}' |
head -n1)

./bin/zookeeper-server-stop.sh config/zookeeper.properties
```