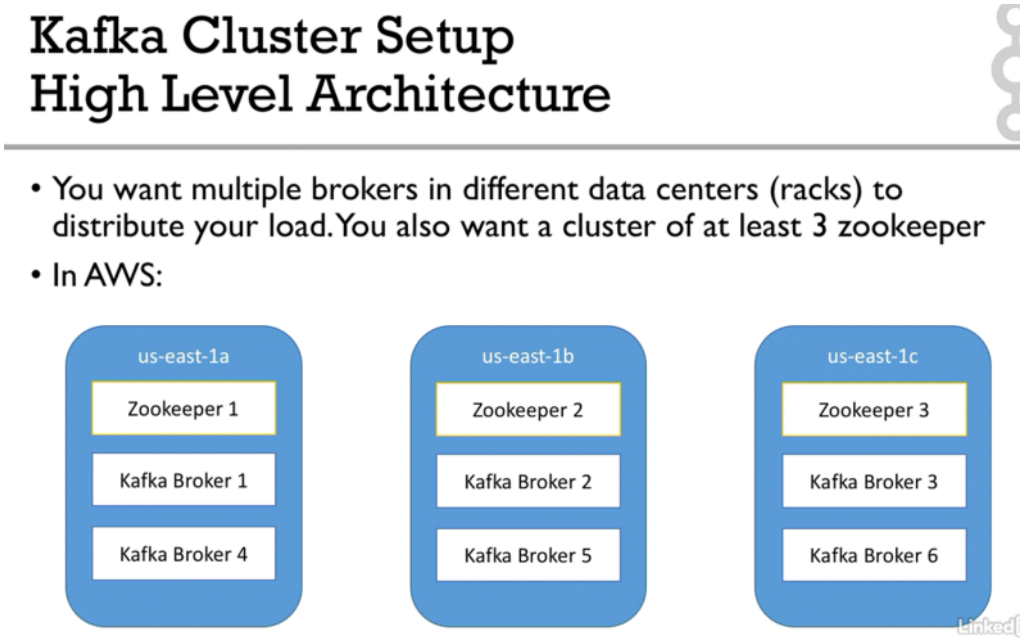
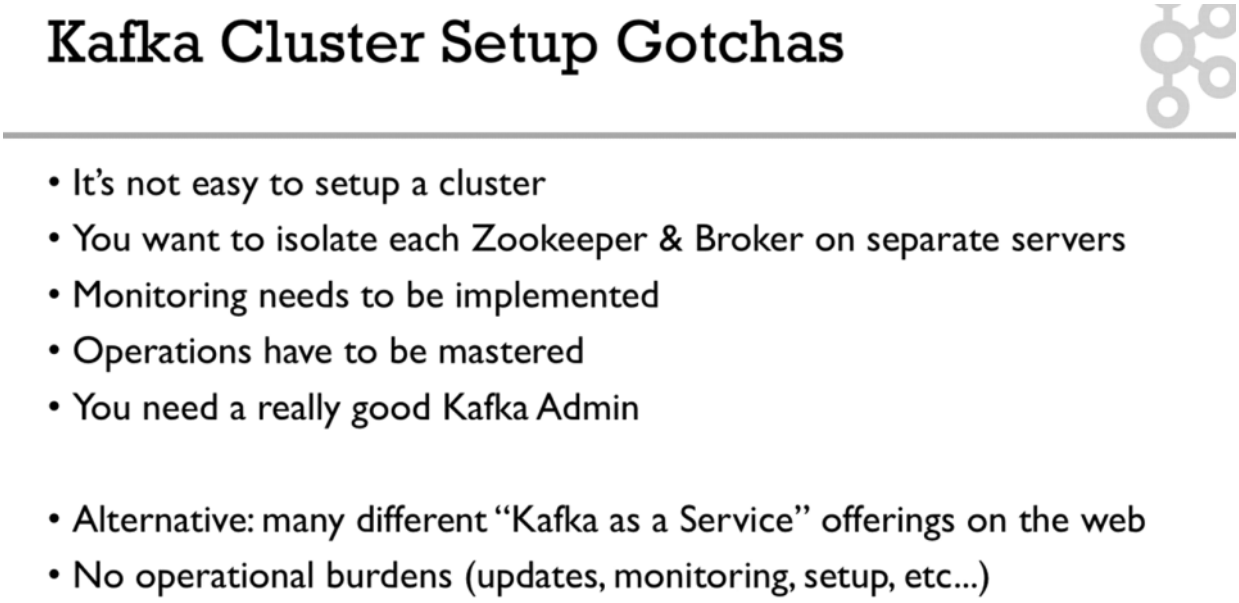
__установка в пром

30 ноября 2020 г. 22:17

должно быть как минимум три ноды





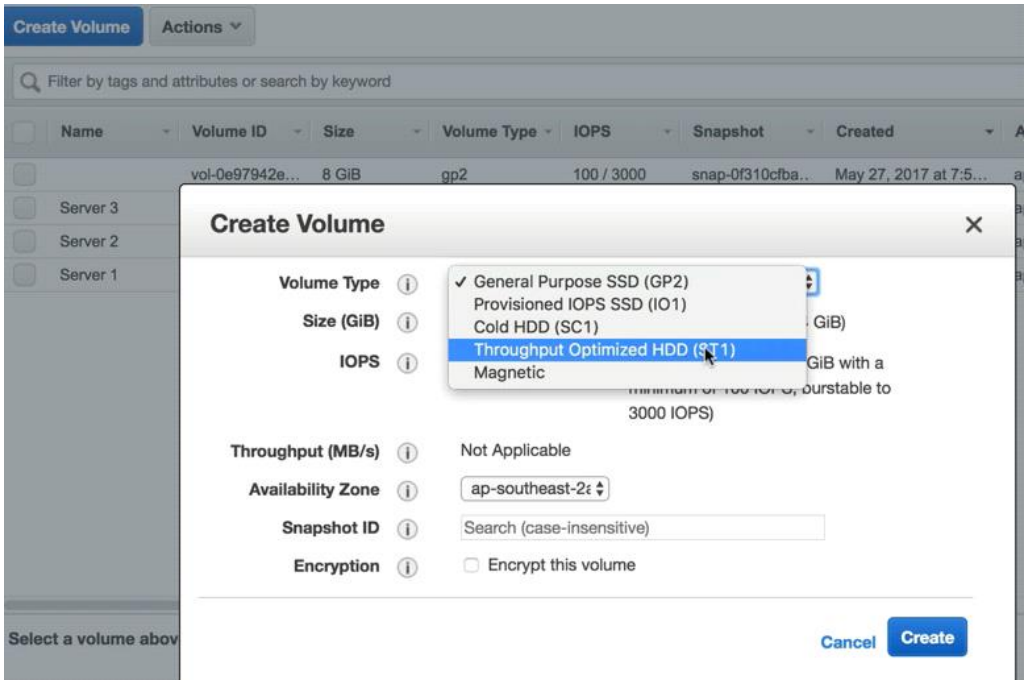
диски под кафку рекомендуется форматировать на XFS и использовать HDD throughput optimized (не SSD)

Hands On: AWS Setup

1. Setup network security to allow Kafka ports (9092)
2. Create and Attach EBS volumes to EC2 Instances (to have a separate drive for Kafka operations)
3. Format the newly attached EBS volumes as XFS (recommended file system for Kafka as per documentation - requires less tuning)
4. Make sure the volume stays mapped on reboot
5. Apply on all machines

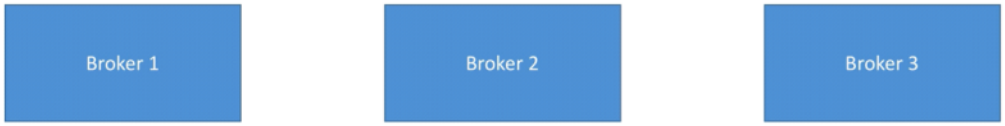
Factors impacting Kafka performance Disks: I/O

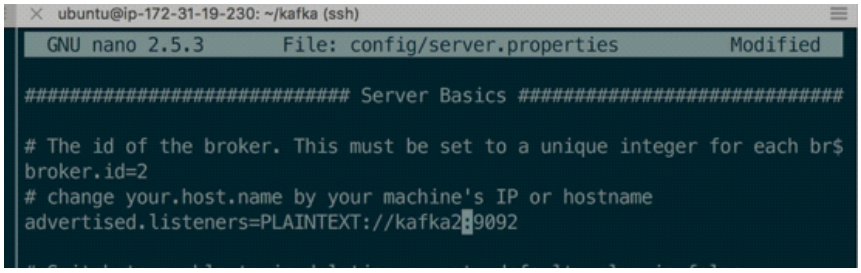
- Reads are done sequentially (as in not randomly), therefore make sure you should a disk type that corresponds well to the requirements
- Format your drives as XFS (easiest, no tuning required)
- If read / write throughput is your bottleneck
  - it is possible to mount multiple disks in parallel for Kafka
  - The config is log.dirs=/disk1/kafka-logs,/disk2/kafka-logs,/disk3/kafka-logs…
- Kafka performance is constant with regards to the amount of data stored in Kafka.
  - Make sure you expire data fast enough (default is one week)
  - Make sure you monitor disk performance



у каждого брокера свой ID в конфиге
и также свой IP/DOMAINNAME тачки в advertized listener

1. Apply same operations to the other machines
2. Edit configurations to make sure the broker ids are different
3. Launch Cluster, observe the logs



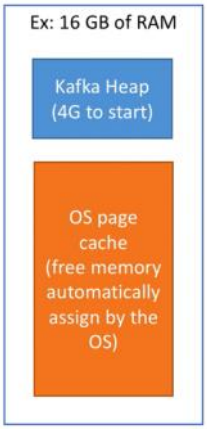


чтобы первая нода поднялась без других: возможно понадобится
на первой ноде сначала поставить min.insync.replicas =1 а потом вернуть обратно ==2

ff

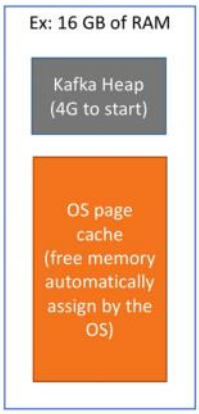
3гб памяти для кафки должно быть немного т.к. кафка напрямую пишет с сети на диск

- Kafka has amazing performance thanks to the page cache which utilizes your RAM
- Understanding RAM in Kafka means understanding two parts:
  - The Java HEAP for the Kafka process
  - The rest of the RAM used by the OS page cache
- Let's understand how both of those should be sized
- Overall, your Kafka production machines should have at least 8GB of RAM to them (the more the better – it's common to have 16GB or 32GB per broker)



Factors impacting Kafka performance
RAM – OS Page Cache

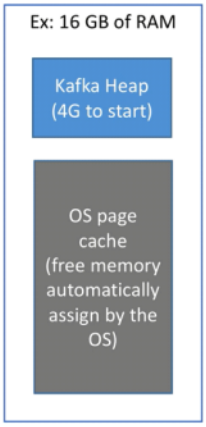
- The remaining RAM will be used automatically for the Linux **OS Page Cache**.
- This is used to buffer data to the disk and this is what gives Kafka an amazing performance
- You don't have to specify anything!
- Any un-used memory will automatically be leveraged by the Linux Operating System and assign memory to the page cache
- Note: Make sure swapping is disabled for Kafka entirely **vm.swappiness=0 or vm.swappiness=1 (default is 60 on Linux)**



export KAFKA_HEAP_OPTS="-Xmx4g"

Factors impacting Kafka performance
RAM – Java Heap

- When you launch Kafka, you specify Kafka Heap Options (KAFKA_HEAP_OPTS environment variable)
- I recommend to assign a MAX amount (-Xmx) of 4GB to get started to the kafka heap:
- `export KAFKA_HEAP_OPTS="-Xmx4g"`
- Don't set –Xms (starting heap size):
  - Let heap grow over time
  - Monitor the heap over time to see if you need to increase Xmx
- Kafka should keep a low heap usage over time, and heap should increase only if you have more partitions in your broker



процессора для кафки должно быть немного т.к. кафка напрямую пишет с сети на диск

Factors impacting Kafka performance CPU

- CPU is usually not a performance bottle neck in Kafka because Kafka does not parse any messages , but can become one in some situations
- If you have SSL enabled, Kafka has to encrypt and decrypt every payload, which adds load on the CPU
- Compression can be CPU bound if you force Kafka to do it. Instead, if you send compressed data, make sure your producer and consumers are the ones doing the compression work (that's the default setting anyway)
- Make sure you monitor Garbage Collection over time to ensure the pauses are not too long

для кафки рекомендуется linux
* hard nofile 100000 также рекомендуется установить лимит на число открытых файлов

Factors impacting Kafka performance Operating System (OS)

- Use Linux or Solaris, running production Kafka clusters on Windows is not recommended.
- Increase the file descriptor limits (at least 100,000 as a starting point) https://unix.stackexchange.com/a/8949/170887
- Make sure only Kafka is running on your Operating System. Anything else will just slow the machine down.

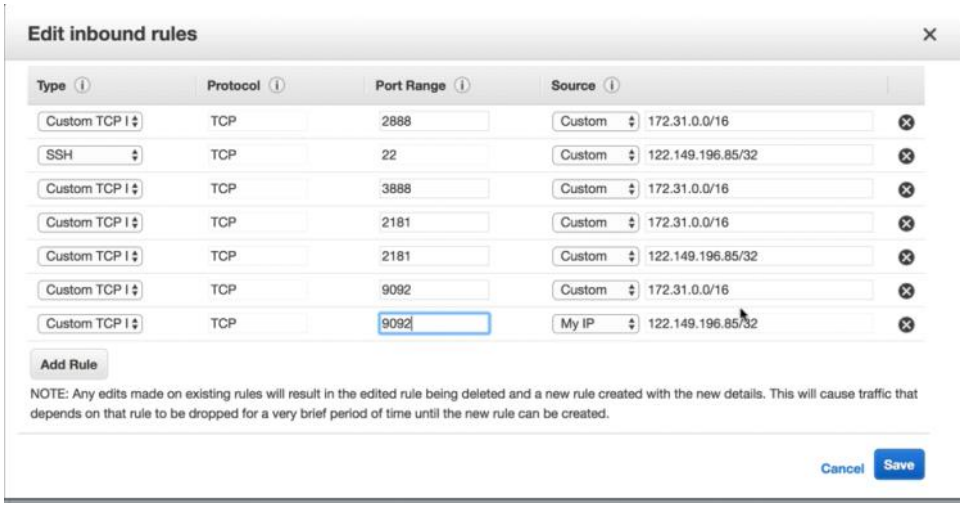
- Make sure you have enough file handles opened on your servers, as Kafka opens 3 file descriptor for each topic-partition-segment that lives on the Broker.
- Make sure you use Java 8
- You may want to tune the GC implementation: (see in the resources)
- Set Kafka quotas in order to prevent unexpected spikes in usage

мелкие но важные настройки для прода

- auto.create.topics.enable=true => set to false in production
- background.threads=10 => increase if you have a good CPU
- delete.topic.enable=false => your choice
- log.flush.interval.messages => don't ever change. Let your OS do it
- log.retention.hours=168 => Set in regards to your requirements
- message.max.bytes=1000012 => increase if you need more than 1MB
- min.insync.replicas=1 => set to 2 if you want to be extra safe
- num.io.threads=8 => ++if your network io is a bottleneck
- num.network.threads=3 => ++if your network is a bottleneck

- num.recovery.threads.per.data.dir=1 => set to number of disks
- num.replica.fetchers=1 => increase if your replicas are lagging
- offsets.retention.minutes=1440 => after 24 hours you lose offsets
- unclean.leader.election.enable=true => false if you don't want data loss
- zookeeper.session.timeout.ms=6000 => increase if you timeout often
- broker.rack=null => set your to availability zone in AWS
- default.replication.factor=1 => set to 2 or 3 in a kafka cluster
- num.partitions=1 => set from 3 to 6 in your cluster
- quota.producer.default=10485760 => set quota to 10MBs
- quota.consumer.default=10485760 => set quota to 10MBs

открыть порты



скрипт предварительной настройки

```
#!/bin/bash

# execute commands as root
sudo su

# Attach the EBS volume in the console, then
```

```
# view available disks
lsblk

# we verify the disk is empty - should return "data"
file -s /dev/xvdf

# Note on Kafka: it's better to format volumes as xfs:
# https://kafka.apache.org/documentation/#filesystems
# Install packages to mount as xfs
apt-get install -y xfsprogs

# create a partition
fdisk /dev/xvdf

# format as xfs
mkfs.xfs -f /dev/xvdf

# create kafka directory
mkdir /data/kafka
# mount volume
mount -t xfs /dev/xvdf /data/kafka
# add permissions to kafka directory
chown -R ubuntu:ubuntu /data/kafka
# check it's working
df -h /data/kafka

# EBS Automount On Reboot
cp /etc/fstab /etc/fstab.bak # backup
echo '/dev/xvdf /data/kafka xfs defaults 0 0' >> /etc/fstab

# reboot to test actions
reboot
sudo service zookeeper start
```

скрипт установки

```
#!/bin/bash

# Make sure you have done the steps from 5 on all machines
# we repeat the steps from 6

# Add file limits configs - allow to open 100,000 file descriptors
echo "* hard nofile 100000
* soft nofile 100000" | sudo tee --append /etc/security/limits.conf
sudo reboot
sudo service zookeeper start
sudo chown -R ubuntu:ubuntu /data/kafka

# edit the config
rm config/server.properties
# MAKE SURE TO USE ANOTHER BROKER ID
nano config/server.properties
# launch kafka - make sure things look okay
bin/kafka-server-start.sh config/server.properties

# Install Kafka boot scripts
sudo nano /etc/init.d/kafka
sudo chmod +x /etc/init.d/kafka
sudo chown root:root /etc/init.d/kafka
# you can safely ignore the warning
sudo update-rc.d kafka defaults

# start kafka
sudo service kafka start
# verify it's working
nc -vz localhost 9092
# look at the logs
cat /home/ubuntu/kafka/logs/server.log
# make sure to fix the __consumer_offsets topic
bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka --config min.insync.replicas=1 --topic __consumer_offsets --alter
```

```
# read the topic on broker 1 by connecting to broker 2!
bin/kafka-console-consumer.sh --bootstrap-server kafka2:9092 --topic first_topic --from-beginning


# DO THE SAME FOR BROKER 3

# After, you should see three brokers here
bin/zookeeper-shell.sh localhost:2181
ls /kafka/brokers/ids

# you can also check the zoonavigator UI
```

_ КОНФИГ ФАЙЛ

3 декабря 2020 г. 11:28

настройки сервера в кластере

https://github.com/piyushcse29/Kafka/blob/master/code/kafka/server.properties
############################## Server Basics ##############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1 у каждой ноды кафки свой отдельный номер
# change your.host.name by your machine's IP or hostname
advertised.listeners=PLAINTEXT://kafka1:9092 указать внешний ip или доменное имя моей ноды с кафкой

# Switch to enable topic deletion or not, default value is false
delete.topic.enable=true

############################## Log Basics ##############################

# A comma seperated list of directories under which to store log files
log.dirs=/data/kafka

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=8 дефолтное число партиций на топик (брокеров 3 или 4)
# we will have 3 brokers so the default replication factor should be 2 or 3
default.replication.factor=3 число реплик сообщения (должно быть ровно 3)
# number of ISR to have in order to minimize data loss
min.insync.replicas=2 для producer ack

############################## Log Retention Policy ##############################

# The minimum age of a log file to be eligible for deletion due to age
# this will delete data after a week
log.retention.hours=168 сообщения хранятся одну неделю, неважно прочитаны или нет (см log cleanup)

# The maximum size of a log segment file. When this size is reached a new log segment will be created.
log.segment.bytes=1073741824 ==1гб если меньше то log_compaction чаще
?может быть это связано с тем насколько часть "падают" мои сервисы, чтобы они могли поднятся и восстановить свое состояние они будут читать из compacted topic

# The interval at which log segments are checked to see if they can be deleted according
# to the retention policies
log.retention.check.interval.ms=300000 каждые 5мин будем проверять не пора ли удалить сегмент в соответствии с policy

############################## Zookeeper ##############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka каталог в zk

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000

############################## Other ##############################

# I recommend you set this to false in production.
# We'll keep it as true for the course
auto.create.topics.enable=true для dev среды

мелкие но важные настройки для прода

- auto.create.topics.enable=true => set to false in production
- background.threads=10 => increase if you have a good CPU
- delete.topic.enable=false => your choice
- log.flush.interval.messages => don't ever change. Let your OS do it
- log.retention.hours=168 => Set in regards to your requirements
- message.max.bytes=1000012 => increase if you need more than 1MB
- min.insync.replicas=1 => set to 2 if you want to be extra safe
- num.io.threads=8 => ++if your network io is a bottleneck
- num.network.threads=3 => ++if your network is a bottleneck

- num.recovery.threads.per.data.dir=1 => set to number of disks
- num.replica.fetchers=1 => increase if your replicas are lagging
- offsets.retention.minutes=1440 => after 24 hours you lose offsets
- unclean.leader.election.enable=true => false if you don't want data loss
- zookeeper.session.timeout.ms=6000 => increase if you timeout often
- broker.rack=null => set your to availability zone in AWS
- default.replication.factor=1 => set to 2 or 3 in a kafka cluster
- num.partitions=1 => set from 3 to 6 in your cluster
- quota.producer.default=10485760 => set quota to 10MBs
- quota.consumer.default=10485760 => set quota to 10MBs

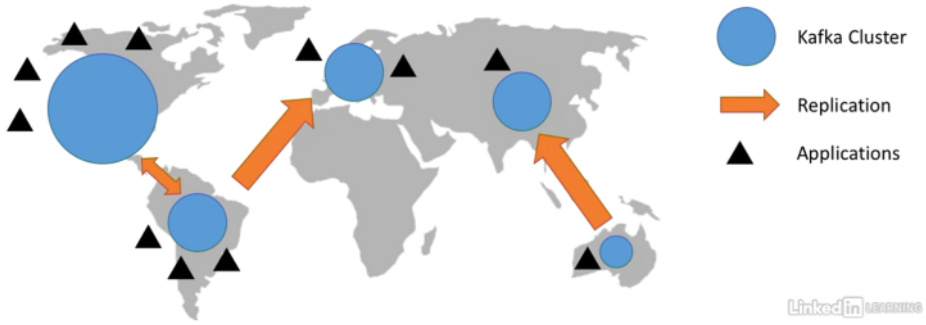
multi-cluster

30 ноября 2020 г. 22:34

гео-шардинг делается через multi-cluster с репликацией между ними

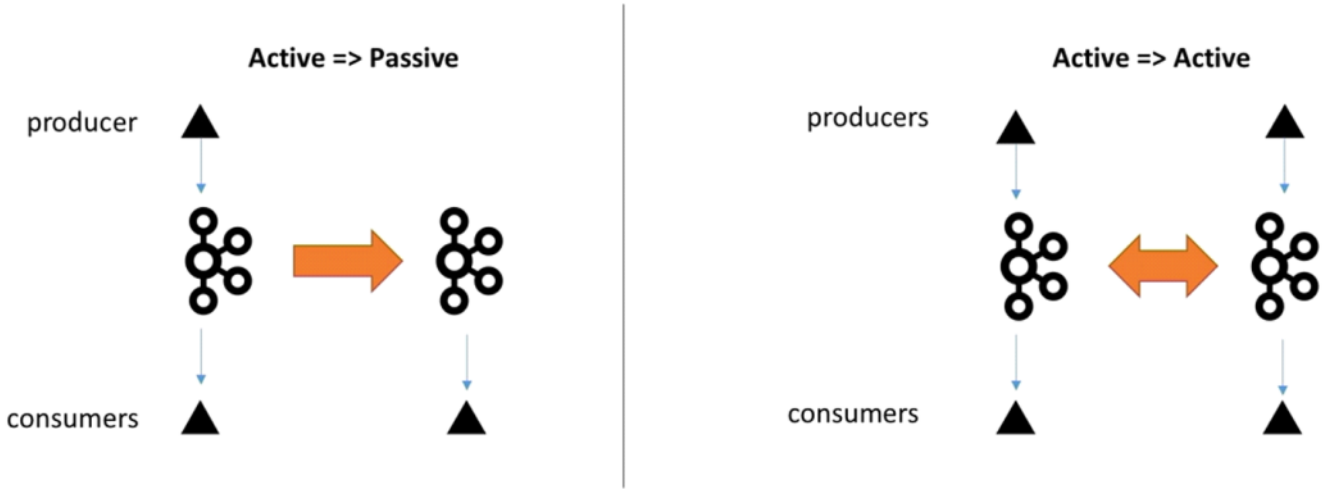
Kafka Multi Cluster + Replication

- Kafka can only operate well in a single region
- Therefore, it is very common for enterprises to have Kafka clusters across the world, with some level of replication between them



- A replication application at its core is just a consumer + a producer
- There are different tools to perform it:
  - Mirror Maker - open source tool that ships with Kafka
  - Netflix uses Flink – they wrote their own application
  - Uber uses uReplicator – addresses performance and operations issues with MM
  - Comcast has their own open source Kafka Connect Source
  - Confluent has their own Kafka Connect Source (paid)
- Overall, try these and see if it works for your use case before writing your own

- There are two designs for cluster replication:



- Active => Active:
  - You have a global application
  - You have a global dataset

- Active => Passive:
  - You want to have an aggregation cluster (for example for analytics)
  - You want to create some form of disaster recovery strategy (*it's hard)
  - Cloud Migration (from on-premise cluster to Cloud cluster)

- **Replicating doesn't preserve offsets, just data!**

kafka connector: replicator используется для соеденения кластеров между собой

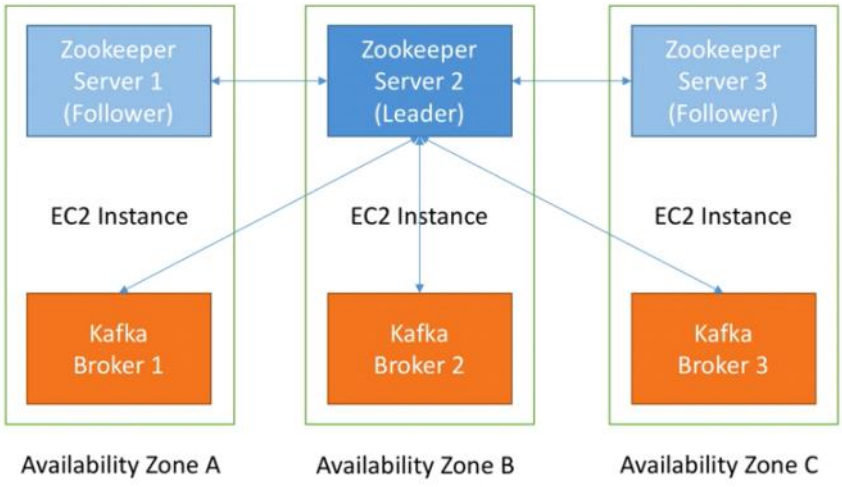
kafka cluster

2 декабря 2020 г. 19:59

для dev среды zk на том же сервере что kafka

Kafka Cluster Architecture

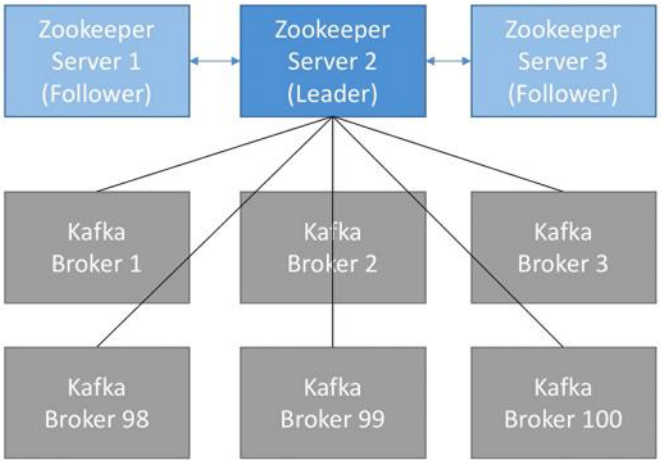
- We will co-locate Zookeeper and Kafka for cost-saving purposes (not recommend for production deployments)
- Knowledge is still applicable to production deployments



Note on IPs and DNS *mostly AWS

- Your Zookeeper & Kafka **must** know their hostname and / or IP
- These are not supposed to change over time, even after a reboot, otherwise your setup will be broken.
- Options:
  1. Use an Elastic Public IP == constant public IP
     you will be able to access your cluster from the outside (e.g. laptop)
  2. Use a secondary ENI == constant private IP **(this course)**
     you will **not** be able to access your cluster from the outside
     you will only be able to access the cluster from within your network
  3. Use DNS names (private or public) == no need to keep fixed IP
     public means you can access instances from outside
     private means you **can't** access instance from outside

Kafka Cluster Size Discussions 100 brokers



- In the case of 100 brokers:
  - Your cluster is fully distributed and can handle tremendous volumes, even using commodity hardware.
  - Zookeeper may be under pressure because of many open connections, so you need to increase the Zookeeper instance performance
  - Cluster management is a full time job (make sure no broker acts weirdly)
  - It is not recommended to have a replication factor of 4 or more, and this would incur network communications within the brokers. Leave it as 3
  - Scale horizontally only when a bottleneck is reached (network, disk i/o, cpu, ram)

ноды кластера кафки должны быть рядом (в соседних ЦОДах) ??задержка не более
так как сеть оказывает ключевое воздействие на кафку

Factors impacting Kafka performance Network

- Latency is key in Kafka
  - Ensure your Kafka instances are your Zookeeper instances are geographically close!!!
  - Do not put one broker in Europe and the other broker in the US
  - Having two brokers live on the same rack is good for performance, but a big risk if the rack goes down.
- Network bandwidth is key in Kafka
  - Network will be your bottleneck.
  - Make sure you have enough bandwidth to handle many connections, and TCP requests.
  - Make sure your network is high performance
- Monitor network usage to understand when it becomes a bottleneck

ноды установить в различные avialability zones

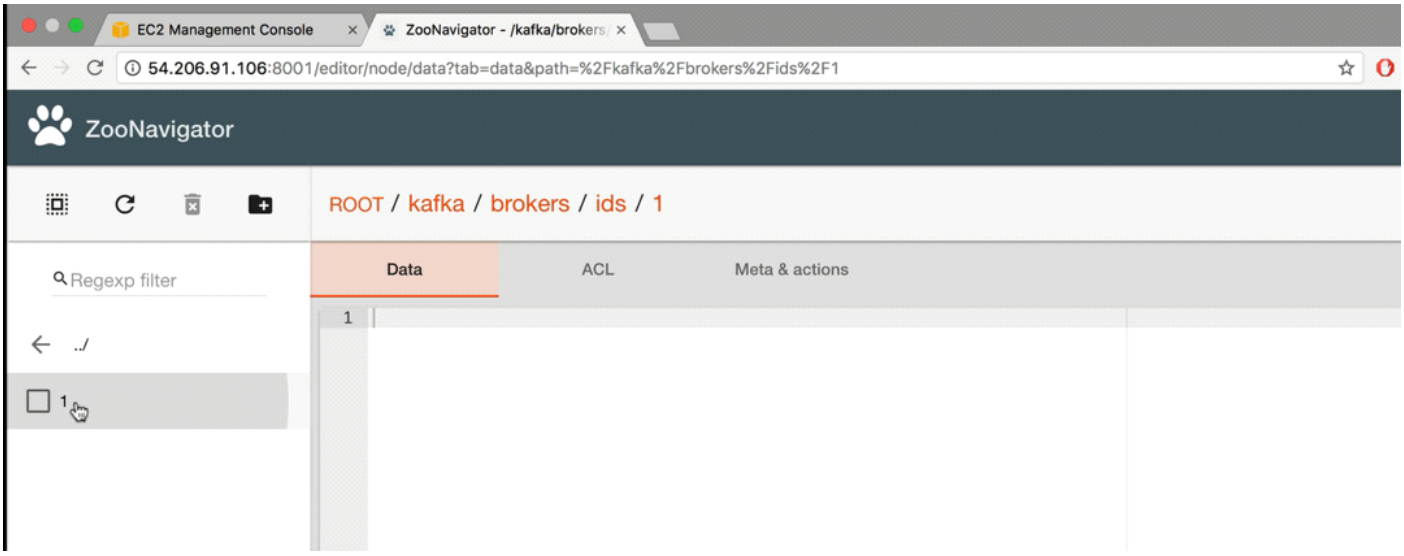
Running Kafka on AWS in Production

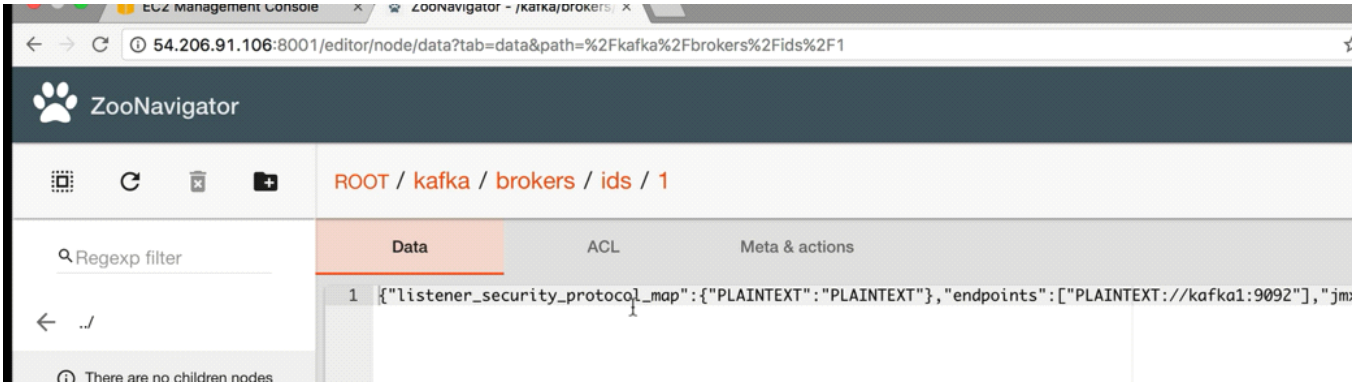
- Separate your instances between different availability zones
- Use st1 EBS volumes for the best price / performance ratio
- Mount multiple EBS volumes to the same broker if you need to scale
- Use *r4.xlarge* or *m4.2xlarge* if you're using EBS (these instances are EBS optimized). You can use something smaller but performance may degrade
- Setup DNS names for your brokers / fixed IPs so that your clients aren't affected if you recycle your EC2 instances

ZooKeeper

3 декабря 2020 г. 13:23

кафка хранит URL своего лидера в ZK





проверитм что кафка работает

3 декабря 2020 г. 13:37

nc -vz localhost 9092 еще один вариант как проверить что порт открыт

sudo apt -y install wget ca-certificates zip net-tools vim nano tar netcat

https://www.alexgur.ru/articles/4822/

```
ubuntu@ip-172-31-9-1:~/kafka$ nc -vz localhost 9092
Connection to localhost 9092 port [tcp/*] succeeded!
```

После установки и настройки фаервола стоит убедиться, что он действительно работает. Лучший способ это сделать - отправить запрос на интересующий порт со сторонней машины. Если запрос прошёл - то порт открыт. Если вернулся отказ или запрос оборвался, то порт закрыт и фаервол работает верно.

Для такой проверки потребуется любой компьютер с Unix/Linux и утилита NetCat "nc" (она часто идёт из коробки со многими дистрибутивами). Эта утилита позволяет стучать как в TCP, так и в UDP порты. И имеет довольно понятный синтаксис команд. Взглянем на пример использования:

> nc -zv ip_адрес порт

Ключ "z" означает, что не надо отправлять никаких данных. Если его не поставить, то после ввода команды сначала будет дан ответ удалось или не удалось соединиться с портом, а потом будет пуста строка для отправки данных. В таком случае набираем данные руками и жмём Enter для отправки. Но такой подход редко используется, обычно достаточно проверить доступность порта, поэтому ставят ключ "z".

Ключ "v" заставляет команду выводить информацию о результате сканирования. Продемонстрируем работу на примере (доступ к 80-ому TCP порту на ip адресе 8.8.8.8):

> **alex@alex-pc:~$** nc -zv 8.8.8.8 80
>
> Connection to 8.8.8.8 80 port [tcp/http] succeeded!

Если нужно пощупать UDP порт, то добавляем ключ "-u":

> nc -zvu ip_адрес порт

запишем и получим из кафки через CLI

```
ubuntu@ip-172-31-9-1:~/kafka$ bin/kafka-console-producer.sh --broker-list kafka1:9092 --topic first_topic
new data
finally should be working!
^Cubuntu@ip-172-31-9-1:~/kafkabin/kafka-console-consumer.sh --bootstrap-server kafka1:9092 --topic first_topic --from-beginning
how are you doing broker 1?
hello
finally should be working!
new data
```

смотрим логи кафки на сервере кафки (также в логах смотрим какая из нод является сейчас лидером кафки)

```
ubuntu@ip-172-31-9-1:~/kafka$ sudo service kafka start
ubuntu@ip-172-31-9-1:~/kafka$ tail -f logs/server.log
[2017-05-27 12:05:02,251] INFO [Group Metadata Manager on Br
oker 1]: Loading offsets and group metadata from __consumer_
offsets-36 (kafka.coordinator.GroupMetadataManager)
[2017-05-27 12:05:02,251] INFO [Group Metadata Manager on Br
oker 1]: Finished loading offsets from __consumer_offsets-36
 in 0 milliseconds. (kafka.coordinator.GroupMetadataManager)
```

проверим что ZK работает

```
~/udemy-kafka-setup zookeeper-shell 52.65.231.26
:2181
Connecting to 52.65.231.26
Welcome to ZooKeeper!
JLine support is disabled

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
ls /
[zookeeper, kafka]
ls /kafka
[cluster, controller_epoch, controller, brokers, admin, isr_
change_notification, consumers, config]
```

* rolling restart of brokers

4 декабря 2020 г. 11:17

Rolling Restart of Brokers

- For most of the Kafka changes you will operate, you will need to perform a roll restart of your Kafka Brokers.
- Doing a manual Roll Restart means:
  - Shutdown down a broker
  - Updating that broker
  - Starting the broker
  - Wait for the cluster to be stable
  - Repeat
- It would be better if we could automate this!

Updating Kafka Configurations Solution 1

- We often need to update or tweak Kafka configurations, even when Kafka is running in production.
- Let's learn how to do that!
- Updating Kafka Configurations requires the following steps:
  - Update server.properties
  - Roll Restart every broker
- Let's go ahead and change the number of min.insync.replicas=2

установим Jolokia

Roll Restart of Brokers
Installing Jolokia Agent

- We can use the kafka-utils by Yelp to perform automated roll restarts.

- Let's install the Jolokia Agents on our Brokers
- Let's start by the first one!

The agents and the client library can be downloaded directly from our maven repository, too:

Artifact	Download
WAR-Agent	jolokia-war-1.6.0.war
WAR-Agent (unsecured)	jolokia-war-unsecured-1.6.0.war
Osgi-Agent	jolokia-osgi-1.6.0.jar
Osgi-Agent (full bundle)	jolokia-osgi-bundle-1.6.0.jar
Mule-Agent	jolokia-mule-1.6.0-agent.jar
JVM-Agent	jolokia-jvm-1.6.0-agent.jar
Java Client Library	jolokia-client-java-1.6.0.jar

```
2018-08-09 12:10:36 (1.11 MB/s) - 'jolokia-agent.jar' saved [461859/461859]

[ec2-user@ip-172-31-1-31 jolokia]$ ll
total 452
-rw-rw-r-- 1 ec2-user ec2-user 461859 Aug  9 12:10 jolokia-agent.jar
[ec2-user@ip-172-31-1-31 jolokia]$
```

```
 GNU nano 2.3.1                File: /etc/systemd/system/kafka.service

[Unit]
Description=Kafka
After=network.target

[Service]
User=ec2-user
Group=ec2-user
Environment="KAFKA_HEAP_OPTS=-Xmx256M -Xms128M"
$fka-0-8-2.yml -javaagent:/home/ec2-user/jolokia/jolokia-agent.jar=host=*"
ExecStart=/home/ec2-user/kafka/bin/kafka-server-start.sh /home/ec2-user/kafka.properties
SuccessExitStatus=143

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

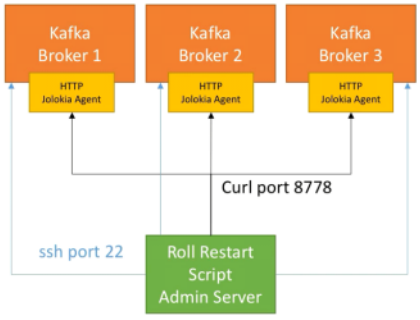
проверим что работает

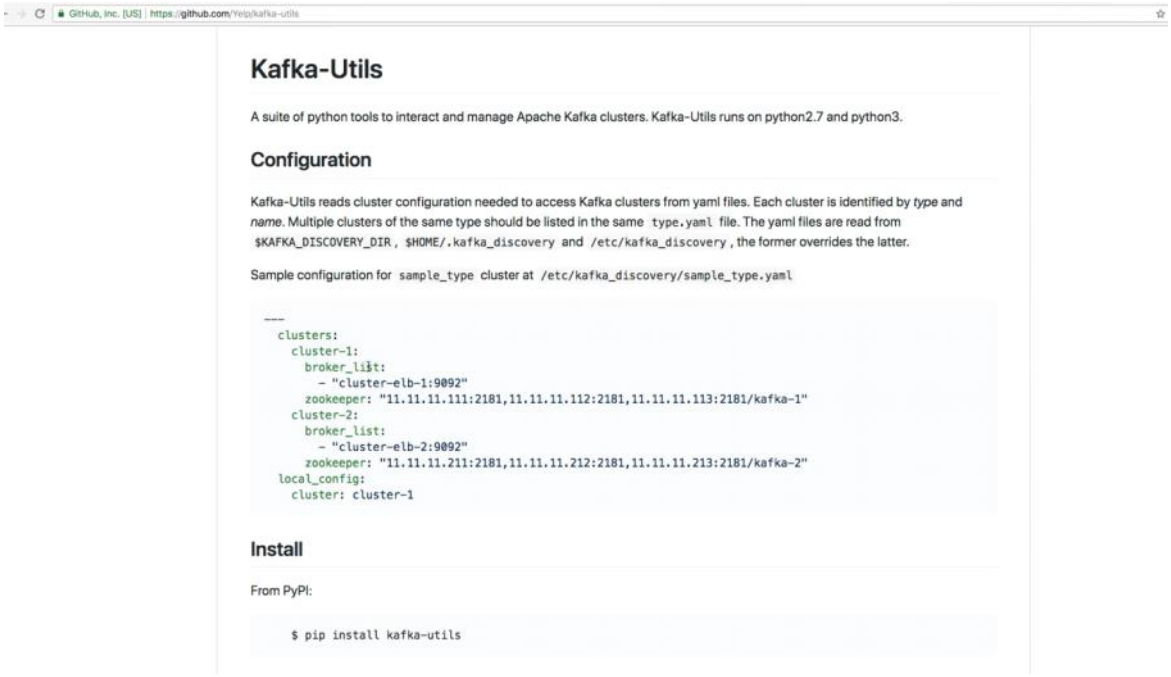
```
# restart Kafka
sudo systemctl daemon-reload
sudo systemctl restart kafka

# Test jolokia
sudo yum install -y jq
curl
localhost:8778/jolokia/read/kafka.server:type=KafkaRequestHandlerPool
,name=RequestHandlerAvgIdlePercent | jq
```

установим yelp

Rolling Restart of Brokers Installing Yelp Tools

- The yelp tools will be installed on the administration machine
- We'll generate a SSH key
- We'll need to copy our SSH public keys to the brokers
- We'll test SSH is working from the administration machine
- We'll test kafka roll restart using Kafka-Tools by Yelp!





Kafka-Utils

A suite of python tools to interact and manage Apache Kafka clusters. Kafka-Utils runs on python2.7 and python3.

Configuration

Kafka-Utils reads cluster configuration needed to access Kafka clusters from yaml files. Each cluster is identified by *type* and *name*. Multiple clusters of the same type should be listed in the same `type.yaml` file. The yaml files are read from `$KAFKA_DISCOVERY_DIR`, `$HOME/.kafka_discovery` and `/etc/kafka_discovery`, the former overrides the latter.

Sample configuration for `sample_type` cluster at `/etc/kafka_discovery/sample_type.yaml`

```
---
  clusters:
    cluster-1:
      broker_list:
        - "cluster-elb-1:9092"
      zookeeper: "11.11.11.111:2181,11.11.11.112:2181,11.11.11.113:2181/kafka-1"
    cluster-2:
      broker_list:
        - "cluster-elb-2:9092"
      zookeeper: "11.11.11.211:2181,11.11.11.212:2181,11.11.11.213:2181/kafka-2"
  local_config:
    cluster: cluster-1
```

Install

From PyPI:

```
    $ pip install kafka-utils
```

```
[ec2-user@ip-172-31-31-225 ~]$ kafka-rolling-restart --cluster-type kafka
```

```
[ec2-user@ip-172-31-31-225 ~]$ kafka-rolling-restart --cluster-type kafka --start-command "systemctl start
kafka" --stop-command "systemctl stop kafka" --check-count 3
```

* updating kafka configurations

4 декабря 2020 г. 12:24

Updating Kafka Configurations

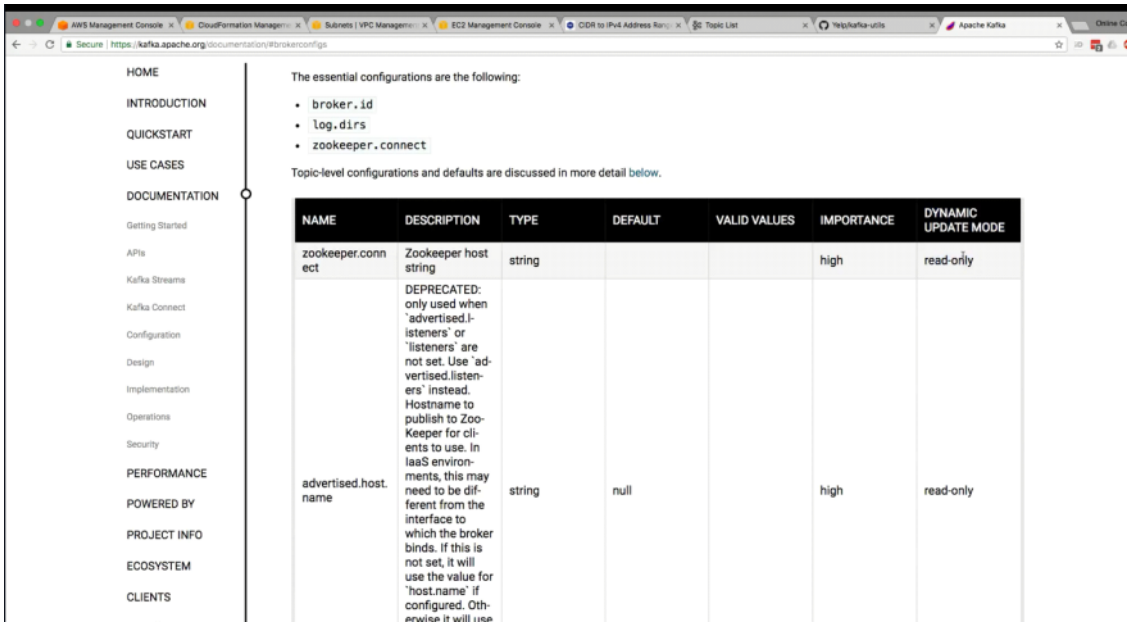
The following order will be followed for priority of config:

1. Dynamic per-broker config stored in ZooKeeper
2. Dynamic cluster-wide default config stored in ZooKeeper
3. Static broker config from server.properties
4. Kafka default, see https://kafka.apache.org/documentation/#brokerconfigs

Hands-On

- Let's learn how to use the command `kafka-configs`

help



HOME
INTRODUCTION
QUICKSTART
USE CASES
DOCUMENTATION
Getting Started
APIs
Kafka Streams
Kafka Connect
Configuration
Design
Implementation
Operations
Security
PERFORMANCE
POWERED BY
PROJECT INFO
ECOSYSTEM
CLIENTS

The essential configurations are the following:

- `broker.id`
- `log.dirs`
- `zookeeper.connect`

Topic-level configurations and defaults are discussed in more detail below.

NAME	DESCRIPTION	TYPE	DEFAULT	VALID VALUES	IMPORTANCE	DYNAMIC UPDATE MODE
zookeeper.connect	Zookeeper host string	string			high	read-only
advertised.host.name	DEPRECATED: only used when `advertised.listeners` or `listeners` are not set. Use `advertised.listeners` instead. Hostname to publish to ZooKeeper for clients to use. In IaaS environments, this may need to be different from the interface to which the broker binds. If this is not set, it will use the value for `host.name` if configured. Otherwise it will use	string	null		high	read-only

3.1.1 Updating Broker Configs

From Kafka version 1.1 onwards, some of the broker configs can be updated without restarting the broker. See the `Dynamic Update Mode` column in Broker Configs for the update mode of each broker config.

- `read-only` : Requires a broker restart for update
- `per-broker` : May be updated dynamically for each broker
- `cluster-wide` : May be updated dynamically as a cluster-wide default. May also be updated as a per-broker value for testing.

To alter the current broker configs for broker id 0 (for example, the number of log cleaner threads):

```
--bootstrap-server localhost:9092 --entity-type brokers --entity-name 0 --alter --add-config log.cleaner.threads=2
```

пример скрипта

source of code and full documentation is here:
https://kafka.apache.org/documentation/#dynamicbrokerconfigs

KAFKA_HOST=172.31.33.33:9092

To alter the current broker configs for broker id 1 (for example, the number of log cleaner threads):
bin/kafka-configs.sh --bootstrap-server $KAFKA_HOST --entity-type brokers --entity-name 1 --alter --add-config log.cleaner.threads=2

To describe the current dynamic broker configs for broker id 1:
bin/kafka-configs.sh --bootstrap-server $KAFKA_HOST --entity-type brokers --entity-name 1 --describe

To delete a config override and revert to the statically configured or default value for broker id 1 (for example, the number of log cleaner threads):
bin/kafka-configs.sh --bootstrap-server $KAFKA_HOST --entity-type brokers --entity-name 1 --alter --delete-config log.cleaner.threads

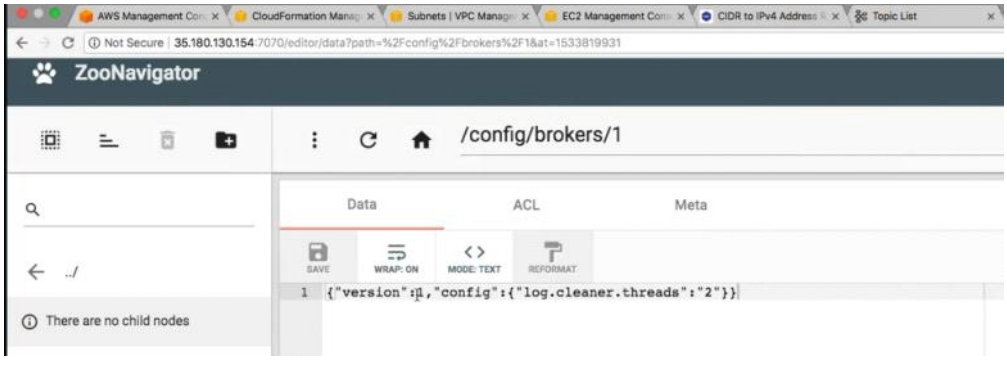
Some configs may be configured as a cluster-wide default to maintain consistent values across the whole cluster. All brokers in the cluster will process the cluster default update. For example, to update log cleaner threads on all brokers:
bin/kafka-configs.sh --bootstrap-server $KAFKA_HOST --entity-type brokers --entity-default --alter --add-config log.cleaner.threads=2

To describe the currently configured dynamic cluster-wide default configs:
bin/kafka-configs.sh --bootstrap-server $KAFKA_HOST --entity-type brokers --entity-default --describe

проверим что настройки применились: вар1

```
[ec2-user@ip-172-31-1-31 kafka]$ journalctl -u kafka | grep log.
cleaner.threads
```

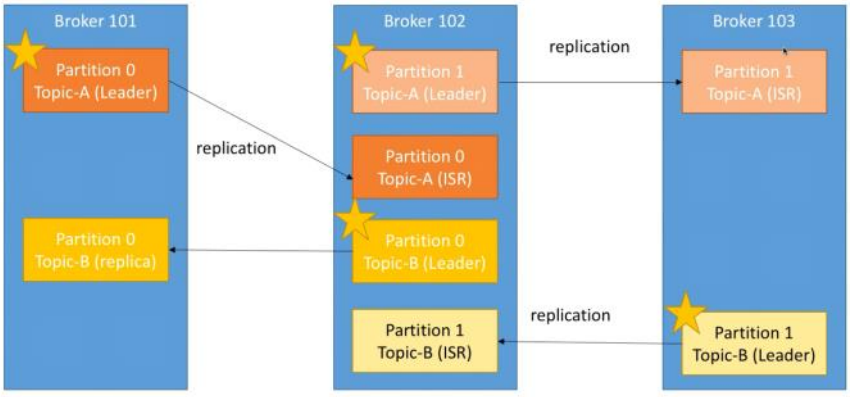
проверим что настройки применились: вар2

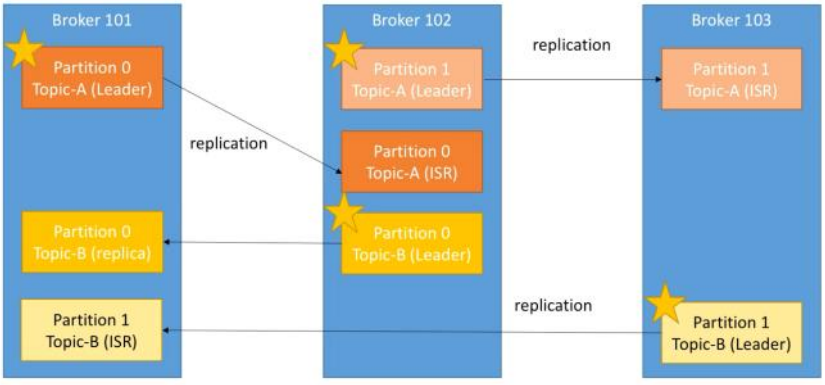


* rebalancing partitions (вручную)

4 декабря 2020 г. 12:38

Rebalancing Partitions
Moving Partitions from 102 => 101





Rebalancing Partitions Manually

- To rebalance partitions we have to create a "partition assignment" JSON file.
- We will go ahead and generate a partition assignment file, and then rebalance our cluster
- We'll run the assignment using the kafka-reassign-partitions.sh command
- All of this will be done manually at first, in the next lectures we will explore tools to do so in an automated way

Partitions by Broker

Broker	# of Partitions	# as Leader	Partitions	Skewed?	Leader Skewed?

Partitions by Broker

Broker	# of Partitions	# as Leader	Partitions	Skewed?	Leader Skewed?
1	2	1	(0,1)	false	false
2	1	0	(1)	false	false
3	1	1	(0)	false	false

Broker	# of Partitions	# as Leader	Partitions	Skewed?	Leader Skewed?
1	1	1	(0)	false	false
2	2	1	(0,1)	false	false
3	1	0	(1)	false	false

пример скрипта

```
export ZK_HOST="172.31.9.21:2181"
export KAFKA_HOST="172.31.33.33:9092"

# create a kafka topic with 2 partition and 2 RF
bin/kafka-topics.sh --zookeeper $ZK_HOST --create --topic second_topic --partitions 2 --replication-factor 2

# describe to see the partition assignment
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic second_topic

# Example:
# Topic:second_topic     PartitionCount:2        ReplicationFactor:2     Configs:
#         Topic: second_topic     Partition: 0    Leader: 2       Replicas: 2,3   Isr: 2,3
#         Topic: second_topic     Partition: 1    Leader: 3       Replicas: 3,1   Isr: 3,1

# Produce some messages
bin/kafka-console-producer.sh --broker-list $KAFKA_HOST --topic second_topic

# Consume the messages
bin/kafka-console-consumer.sh --bootstrap-server $KAFKA_HOST --topic second_topic --from-beginning

# Create an assignment
nano topics.json
bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST  --topics-to-move-json-file topics.json --broker-list "1,2,3" --generate
```

команда сгенерит и предложит два варианта reassignment

```
[ec2-user@ip-172-31-31-225 kafka]$ bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST  --topics-to-move-json-
file topics.json --broker-list "1,2,3" --generate
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should
configure the number of parallel GC threads appropriately using -XX:ParallelGCThreads=N
Current partition replica assignment
{"version":1,"partitions":[{"topic":"second_topic","partition":1,"replicas":[1,2],"log_dirs":["any","any"]},{"to
pic":"second_topic","partition":0,"replicas":[3,1],"log_dirs":["any","any"]}]}

Proposed partition reassignment configuration
{"version":1,"partitions":[{"topic":"second_topic","partition":1,"replicas":[2,3],"log_dirs":["any","any"]},{"to
pic":"second_topic","partition":0,"replicas":[1,2],"log_dirs":["any","any"]}]}
```

```
# copy the proposed assignment into a file:
nano reassignment.json
bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST --reassignment-json-file reassignment.json --execute

# verify the status
bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST --reassignment-json-file reassignment.json --verify

# describe topic:
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic second_topic
```

```
# run preferred replica election
bin/kafka-preferred-replica-election.sh --zookeeper $ZK_HOST

# describe topic
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic second_topic

# Consume the messages
bin/kafka-console-consumer.sh --bootstrap-server $KAFKA_HOST --topic second_topic --from-beginning
```

topics.json пример assignment file

содержит топики которые мы будем перемещать между партициями

```
{
  "version": 1,
  "topics": [
    { "topic": "second_topic" }
  ]
}
```

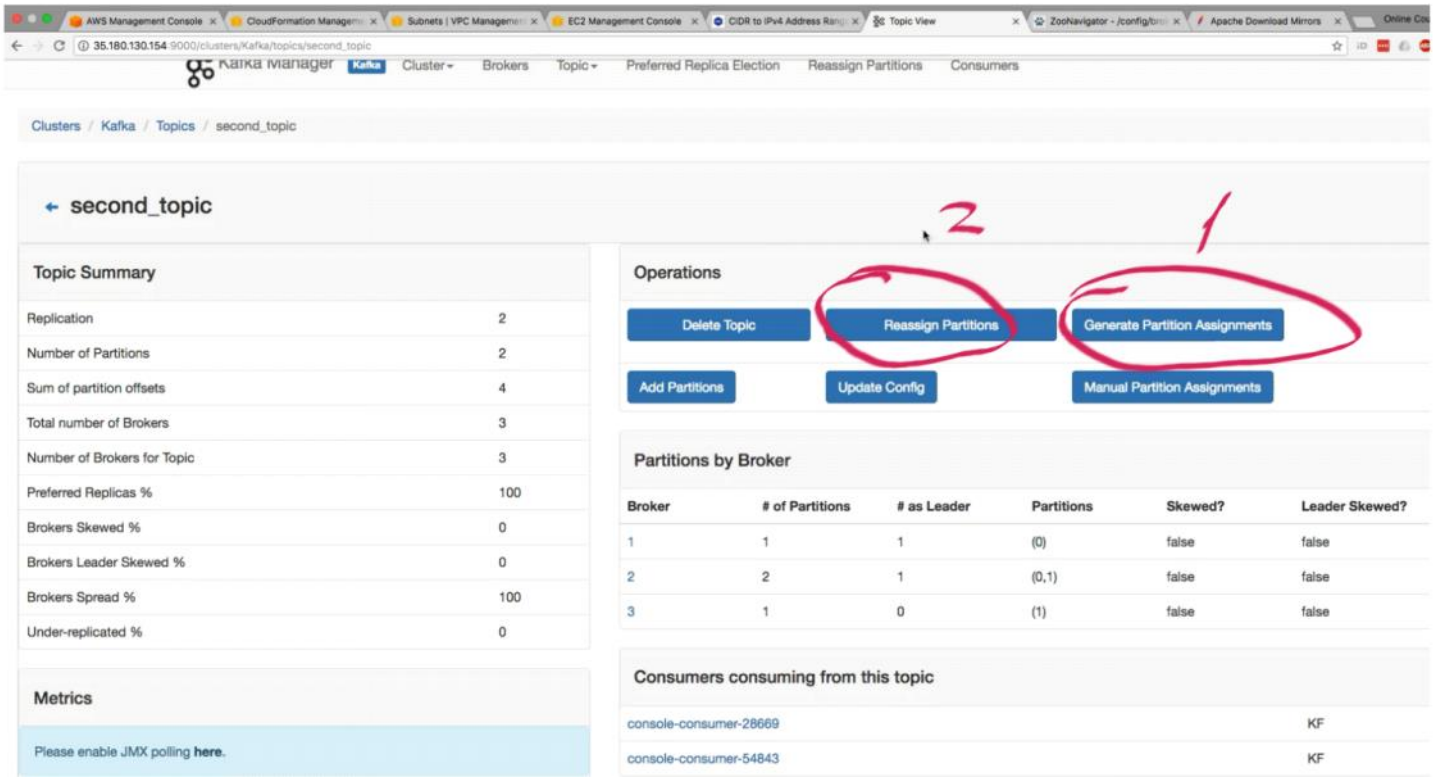
reassignment.json пример сгенеренного reassignment

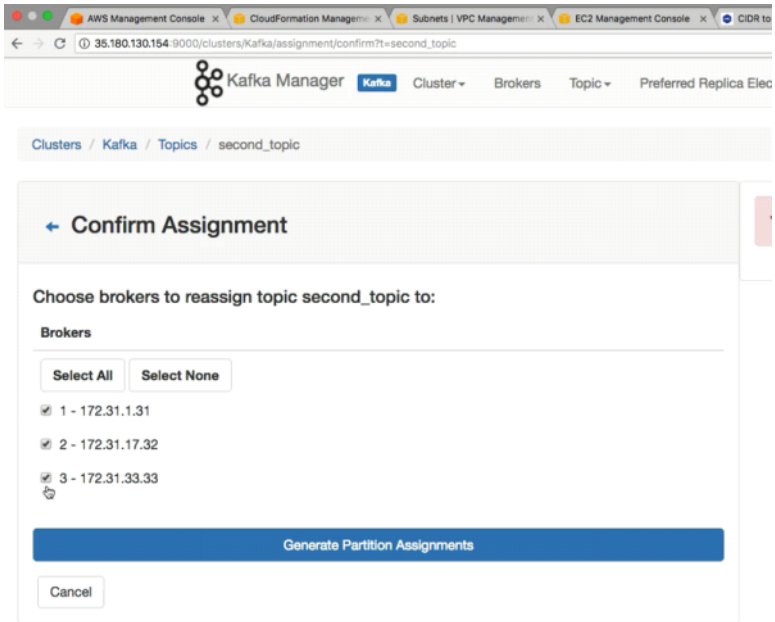
текст сгенерился а вставил в файл его вручную

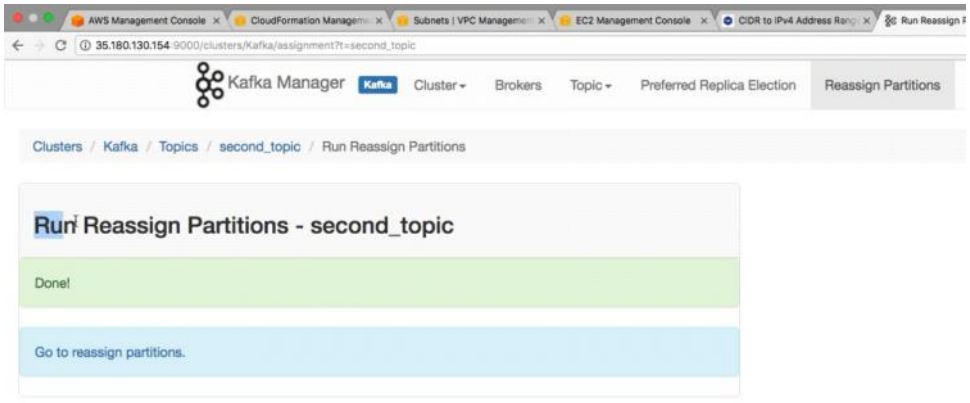
* rebalancing partitions (автоматически)

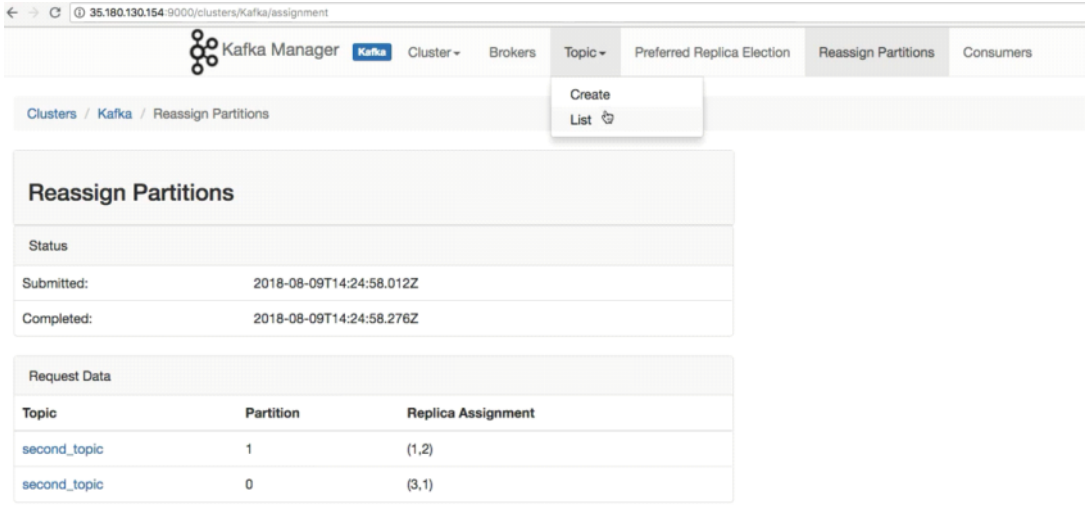
4 декабря 2020 г. 13:37

можно сделать реассаймент черех кафка манагер

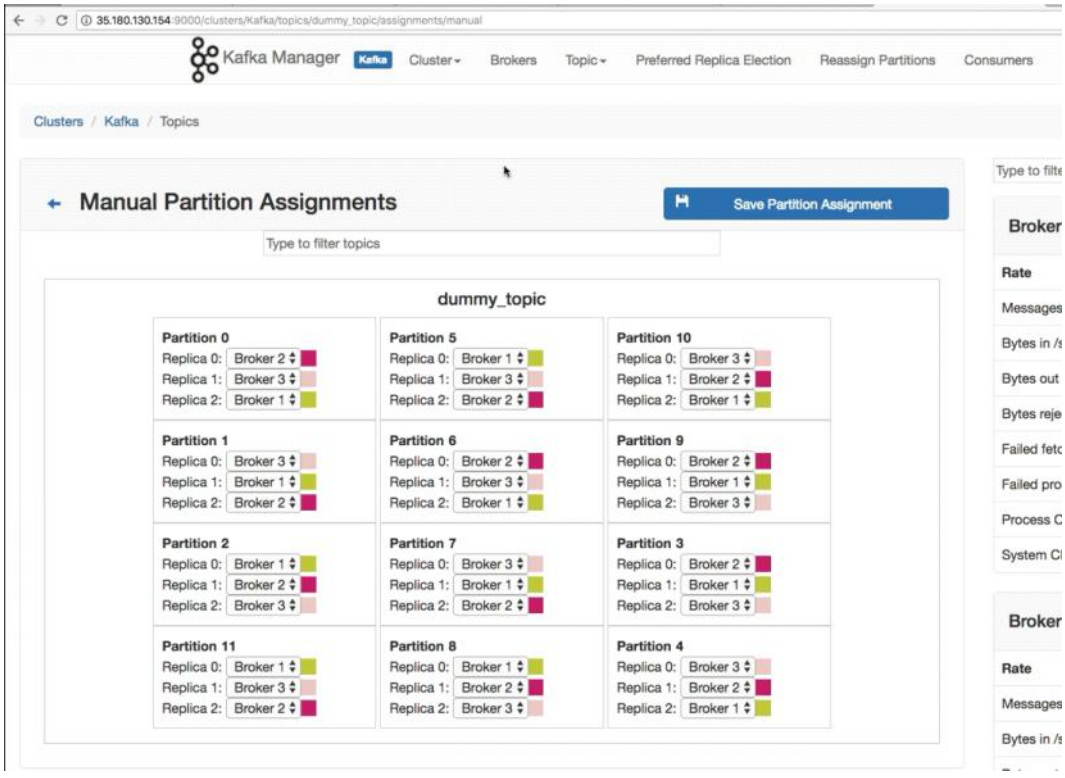








можно не генерить файл а как бы вручную

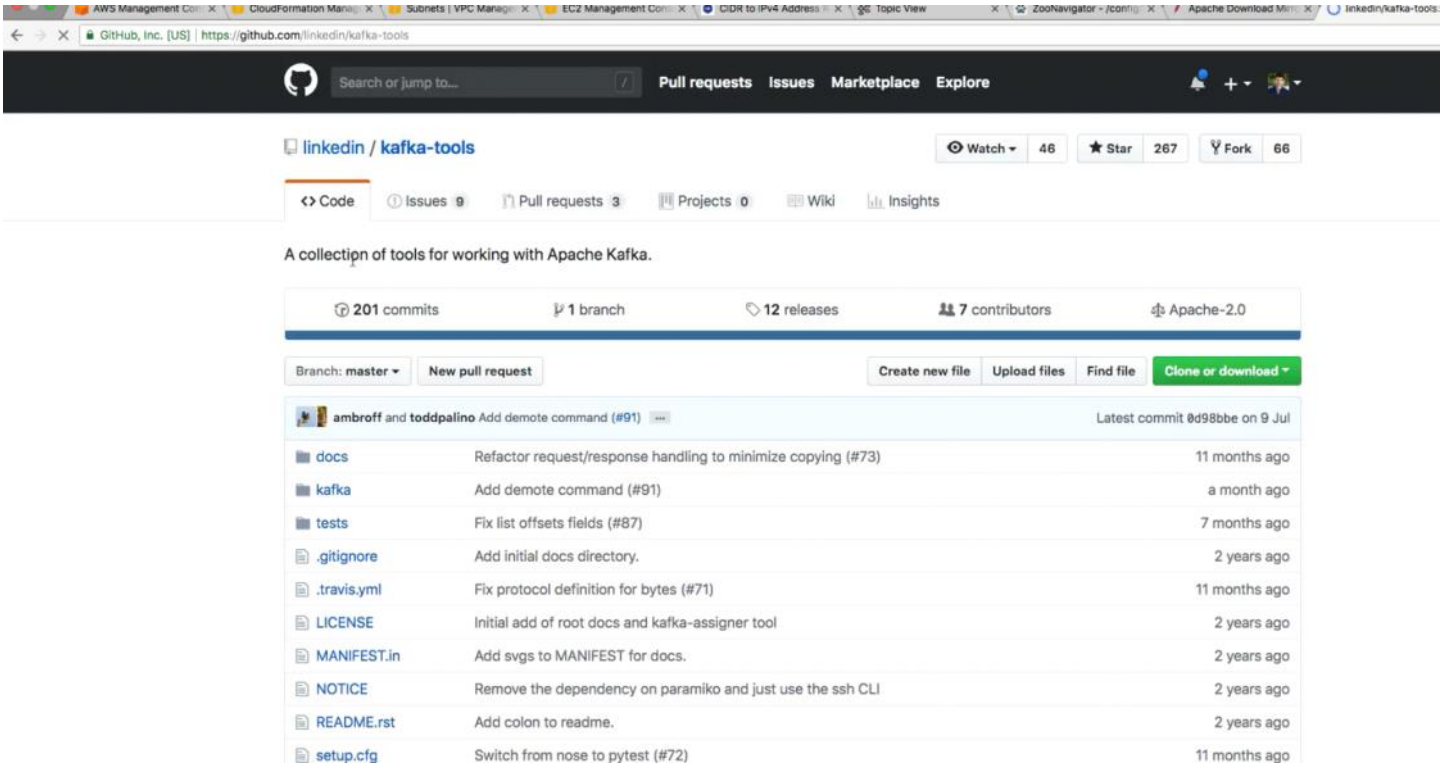


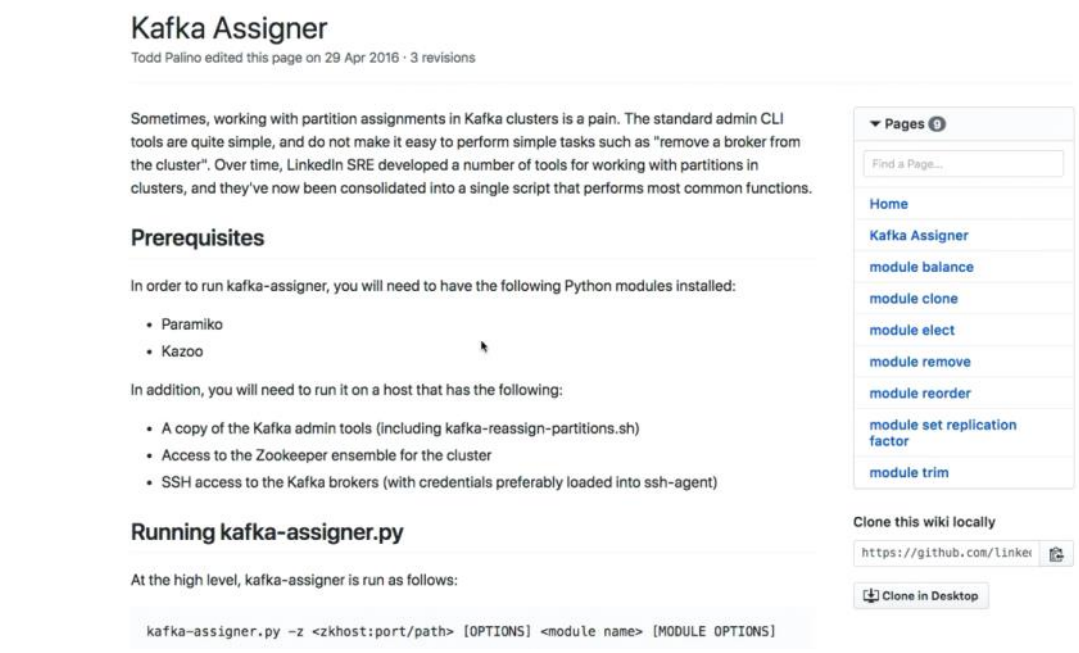
* rebalance cluster

4 декабря 2020 г. 14:12

Rebalancing Partitions Using a CLI (LinkedIn tools)

- Kafka Tools by LinkedIn allows you to automate and generate the partitions JSON file automatically
- Let's install and use the LinkedIn CLI tools to rebalance our cluster!





Kafka Assigner

Todd Palino edited this page on 29 Apr 2016 · 3 revisions

Sometimes, working with partition assignments in Kafka clusters is a pain. The standard admin CLI tools are quite simple, and do not make it easy to perform simple tasks such as "remove a broker from the cluster". Over time, LinkedIn SRE developed a number of tools for working with partitions in clusters, and they've now been consolidated into a single script that performs most common functions.

Prerequisites

In order to run kafka-assigner, you will need to have the following Python modules installed:

- Paramiko
- Kazoo

In addition, you will need to run it on a host that has the following:

- A copy of the Kafka admin tools (including kafka-reassign-partitions.sh)
- Access to the Zookeeper ensemble for the cluster
- SSH access to the Kafka brokers (with credentials preferably loaded into ssh-agent)

Running kafka-assigner.py

At the high level, kafka-assigner is run as follows:

```
kafka-assigner.py -z <zkhost:port/path> [OPTIONS] <module name> [MODULE OPTIONS]
```

Pages

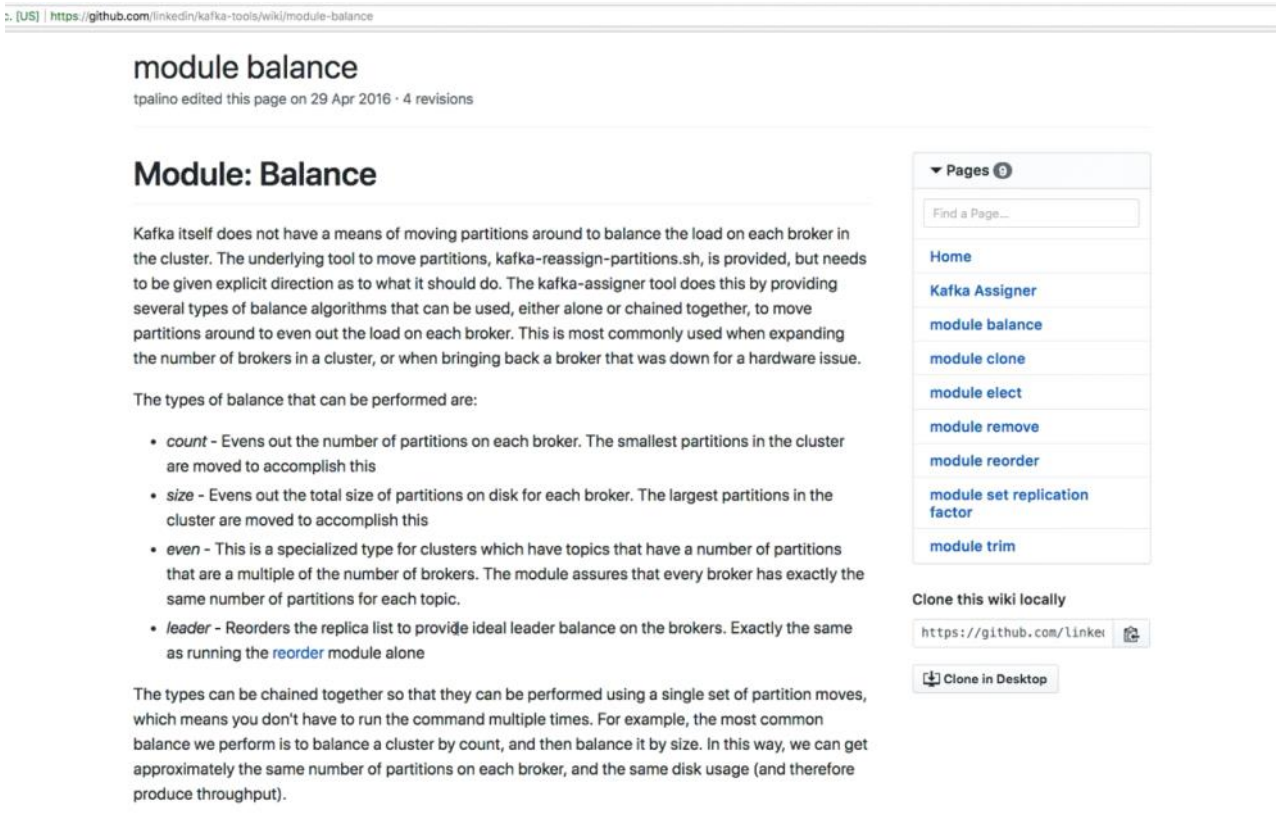
Find a Page...

Home
Kafka Assigner
module balance
module clone
module elect
module remove
module reorder
module set replication factor
module trim

Clone this wiki locally

https://github.com/linke

Clone in Desktop



https://github.com/linkedin/kafka-tools/wiki/module-balance

module balance

tpalino edited this page on 29 Apr 2016 · 4 revisions

Module: Balance

Kafka itself does not have a means of moving partitions around to balance the load on each broker in the cluster. The underlying tool to move partitions, kafka-reassign-partitions.sh, is provided, but needs to be given explicit direction as to what it should do. The kafka-assigner tool does this by providing several types of balance algorithms that can be used, either alone or chained together, to move partitions around to even out the load on each broker. This is most commonly used when expanding the number of brokers in a cluster, or when bringing back a broker that was down for a hardware issue.

The types of balance that can be performed are:

- *count* - Evens out the number of partitions on each broker. The smallest partitions in the cluster are moved to accomplish this
- *size* - Evens out the total size of partitions on disk for each broker. The largest partitions in the cluster are moved to accomplish this
- *even* - This is a specialized type for clusters which have topics that have a number of partitions that are a multiple of the number of brokers. The module assures that every broker has exactly the same number of partitions for each topic.
- *leader* - Reorders the replica list to provide ideal leader balance on the brokers. Exactly the same as running the reorder module alone

The types can be chained together so that they can be performed using a single set of partition moves, which means you don't have to run the command multiple times. For example, the most common balance we perform is to balance a cluster by count, and then balance it by size. In this way, we can get approximately the same number of partitions on each broker, and the same disk usage (and therefore produce throughput).

Pages

Find a Page...

Home
Kafka Assigner
module balance
module clone
module elect
module remove
module reorder
module set replication factor
module trim

Clone this wiki locally

https://github.com/linke

Clone in Desktop

скрипт установки linkedin tools

```
########
#### ADMIN MACHINE
########

export ZK_HOST="172.31.9.21:2181"
export KAFKA_HOST="172.31.33.33:9092"

# install Kafka tools
sudo yum install -y gcc-c++
pip install --user kafka-tools

# check the command is working
kafka-assigner
```

```
# ensure the kafka commands are in .bash_profile
cat << "EOF" >> /home/ec2-user/.bash_profile
DAEMON_PATH=/home/ec2-user/kafka/bin
export PATH=$PATH:$DAEMON_PATH
export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
export JAVA_HOME="/usr/lib/jvm/jre-1.8.0-openjdk"
EOF
source /home/ec2-user/.bash_profile
```

можно сделать реассаймент черех LINKEDIN TOOLS CLI

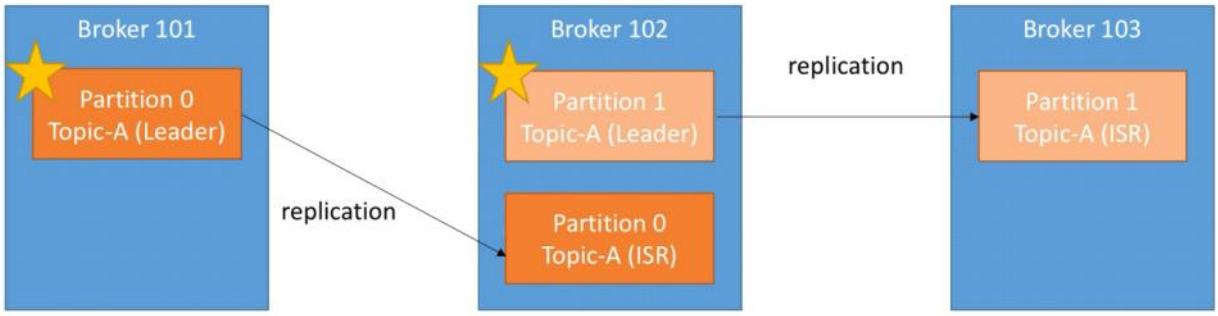
```
# documentation is here: https://github.com/linkedin/kafka-tools/wiki/Kafka-Assigner

# test an assigner (dry run)
kafka-assigner  -z $ZK_HOST --generate balance --types count

# execute an assignment
kafka-assigner  -z $ZK_HOST -e balance --types count
# Documentation for balance module: https://github.com/linkedin/kafka-tools/wiki/module-balance
```

* increasing replication factor (вручную)

4 декабря 2020 г. 14:06

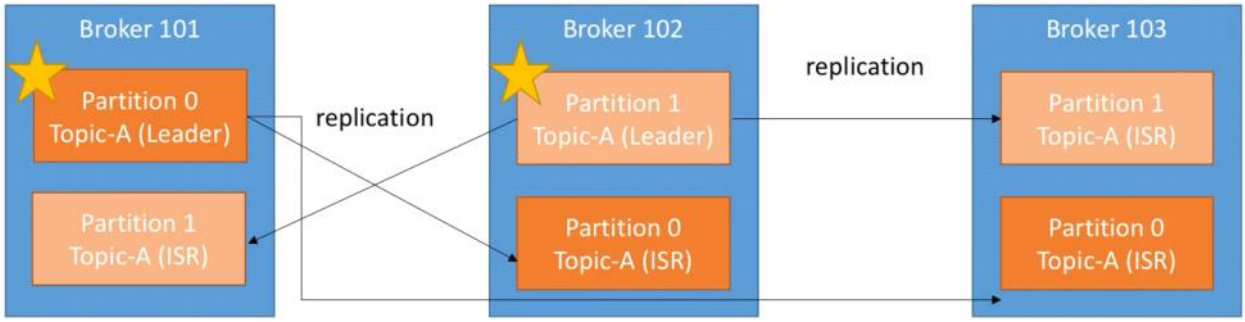
Increasing Replication Factor Manually

- Increasing the replication factor is the same command as rebalancing partitions, but in this case we augment the number of brokers that will hold each partition
- Replication Factor of 2:



- Increasing the replication factor will increase network and disk space usage

- Replication Factor of 3:



скрипт

```
export ZK_HOST="172.31.9.21:2181"
export KAFKA_HOST="172.31.33.33:9092"

# create a kafka topic with 2 partition and 2 RF
bin/kafka-topics.sh --zookeeper $ZK_HOST --create --topic third_topic --partitions 3 --replication-factor 1
```

```
# describe to see the partition assignment
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic

# Topic:third_topic       PartitionCount:3        ReplicationFactor:1     Configs:
#         Topic: third_topic      Partition: 0    Leader: 2       Replicas: 2     Isr: 2
#         Topic: third_topic      Partition: 1    Leader: 3       Replicas: 3     Isr: 3
#         Topic: third_topic      Partition: 2    Leader: 1       Replicas: 1     Isr: 1

# Produce some messages
bin/kafka-console-producer.sh --broker-list $KAFKA_HOST --topic third_topic

# Consume the messages
bin/kafka-console-consumer.sh --bootstrap-server $KAFKA_HOST --topic third_topic --from-
beginning

# Create an assignment
nano topics.json
{
  "version": 1,
  "topics": [
    { "topic": "third_topic" }
  ]
}



bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST  --topics-to-move-json-file topics.json --
broker-list "1,2,3" --generate

# modify, then copy the proposed assignment into a file:  увеличим число реплик
nano reassignment_third_topic.json
{
  "version": 1,
  "partitions": [{
    "topic": "third_topic",
    "partition": 2,
    "replicas": [2,3]
  }, {
    "topic": "third_topic",
    "partition": 1,
    "replicas": [1,2]
  }, {
    "topic": "third_topic",
    "partition": 0,
    "replicas": [3,1]
  }]
}



bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST --reassignment-json-file
reassignment_third_topic.json --execute

# verify the status
bin/kafka-reassign-partitions.sh --zookeeper $ZK_HOST --reassignment-json-file
reassignment_third_topic.json --verify
```

describe topic:
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic

run preferred replica election
bin/kafka-preferred-replica-election.sh --zookeeper $ZK_HOST

describe topic
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic

```
Topic:third_topic       PartitionCount:3        ReplicationFactor:2     Configs:
        Topic: third_topic      Partition: 0    Leader: 3       Replicas: 3,1   Isr: 3,1
        Topic: third_topic      Partition: 1    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: third_topic      Partition: 2    Leader: 2       Replicas: 2,3   Isr: 2,3
```

Consume the messages
bin/kafka-console-consumer.sh --bootstrap-server $KAFKA_HOST --topic third_topic --from-beginning

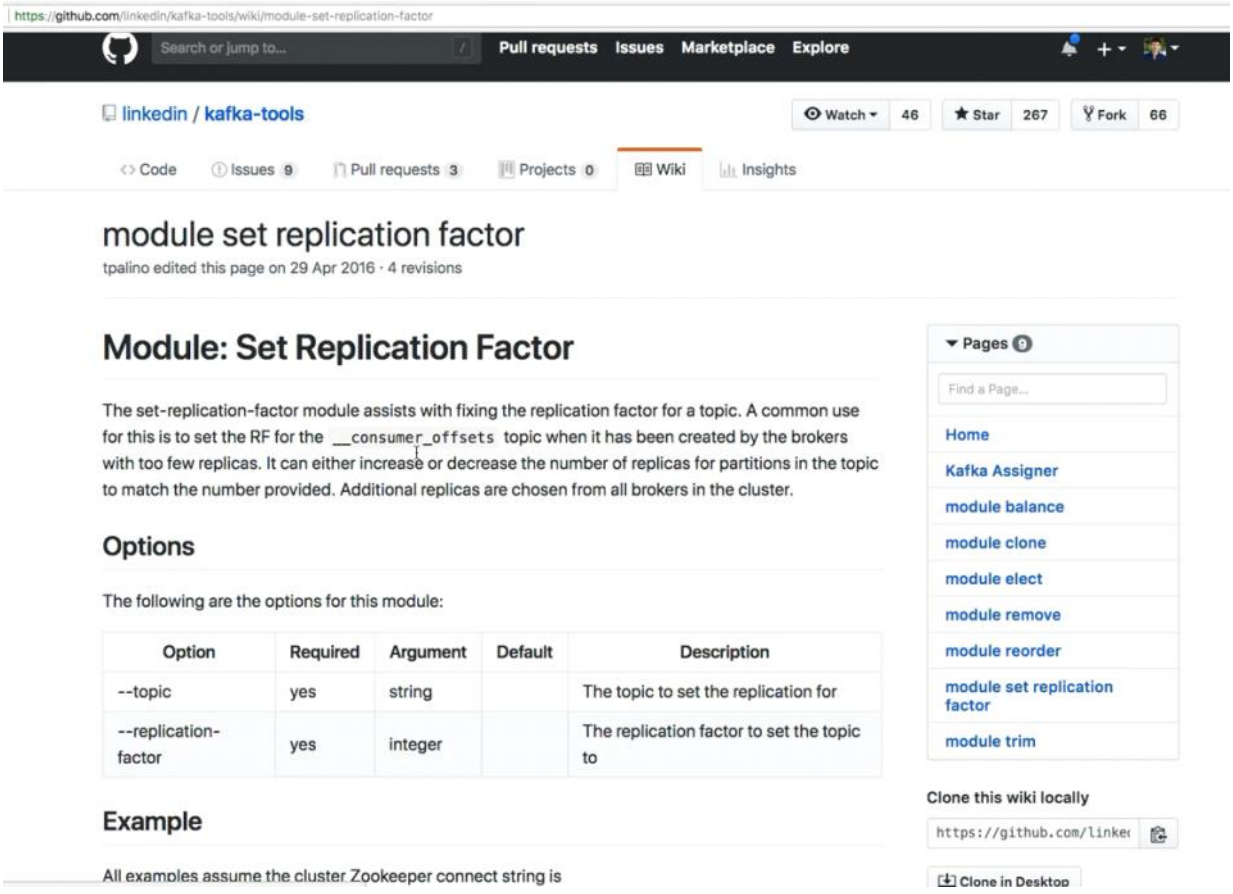
* increasing replication factor (автоматически)

4 декабря 2020 г. 14:23

Increasing Replication Factor Using CLI LinkedIn Tools

- We can automate what we just did using the CLI tools provided by the LinkedIn Kafka Tools
- Let's increase the replication factor of a topic to 3
- Let's decrease the replication factor of a topic 2

нужно установить дополнительный модуль kafka tools: replication factor



скрипт

Documentation for set-replication-factor module: https://github.com/linkedin/kafka-tools/wiki/module-set-replication-factor

```
# describe topic:
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic

# test an assigner
kafka-assigner  -z $ZK_HOST --generate set-replication-factor --topic third_topic --replication-factor 3
# execute an assignment
kafka-assigner  -z $ZK_HOST -e set-replication-factor --topic third_topic --replication-factor 3


# describe topic:
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic

# we can even decrease the replication factor!
kafka-assigner  -z $ZK_HOST -e set-replication-factor --topic third_topic --replication-factor 1

# describe topic:
bin/kafka-topics.sh --zookeeper $ZK_HOST --describe --topic third_topic
```

* add broker

4 декабря 2020 г. 14:21

Adding a Broker Part I

- We'll add a broker using CloudFormation, which will come preconfigured with:
  - SSH capability
  - JMX Exporter (Prometheus)
  - Jolokia Agent for Rolling Restart
- Then we'll look at Kafka manager:
  - the broker is registered
  - but doesn't have any partitions assigned to it just yet

после добавления брокера можно сделать reassign партиций

- чтобы ее задействовать, так как на ней пока еще вообще нет партиций

```
kafka-assigner  -z $ZK_HOST --execute balance --types even
```

* replace broker

4 декабря 2020 г. 14:50

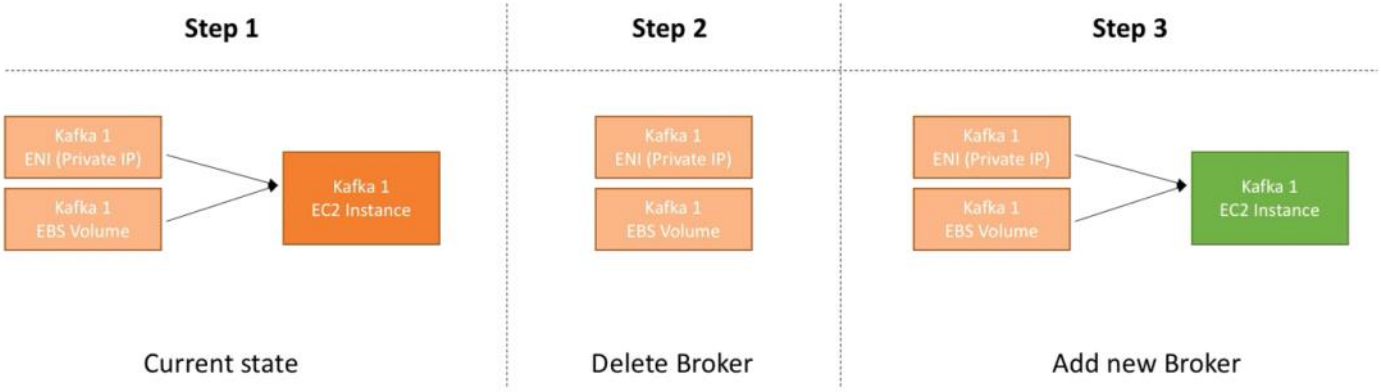
Replacing a Broker While keeping EBS Volume

- If we wanted to upgrade the instance of a broker (operating system, specs, etc…), we should:

- Terminate it
- Create a new machine
- Assign the available EBS volume to the new machine
- Start Kafka on that machine

- It's good because the new Kafka broker will only need to replicate all the data it does not have yet

- We'll do this using CloudFormation!



Replacing a Broker With EBS data loss

- Even if we lose our EBS volume (but it's not advised), it is possible to bring back a broker to life:

- Let's stop our broker
- Let's wipe our EBS volume (same as losing it)
- Let's start our broker again
- Upon joining the cluster, the broker will replicate all missing data

- Kafka Replication Mechanism is amazing for broker recovery!
- This is why you need to ensure you have at least a replication factor of 2

даже если полностью удалить папку с данными на ноде, то при replication factor==2 данные полностью сохранятся и скопируются при подьеме упавшей ноды

```
[ec2-user@ip-172-31-1-31 ~]$ du -sh /data/kafka
5.1M    /data/kafka
[ec2-user@ip-172-31-1-31 ~]$ rm -r -f /data/kafka/*
[ec2-user@ip-172-31-1-31 ~]$ du -sh /data/kafka
0       /data/kafka
[ec2-user@ip-172-31-1-31 ~]$ ls /data/kafka/
[ec2-user@ip-172-31-1-31 ~]$
```

* remove brocker

4 декабря 2020 г. 15:10

удалить ноду не так просто, сначала нужно убрать все партиции с нее

Removing a Broker

- Removing a Broker in Kafka is not as easy as just terminating an instance.
- **This is the most common mistake I've seen at my clients**

- To properly remove a broker:
  - Use tools to move ALL partitions away from that broker
  - Remove the broker

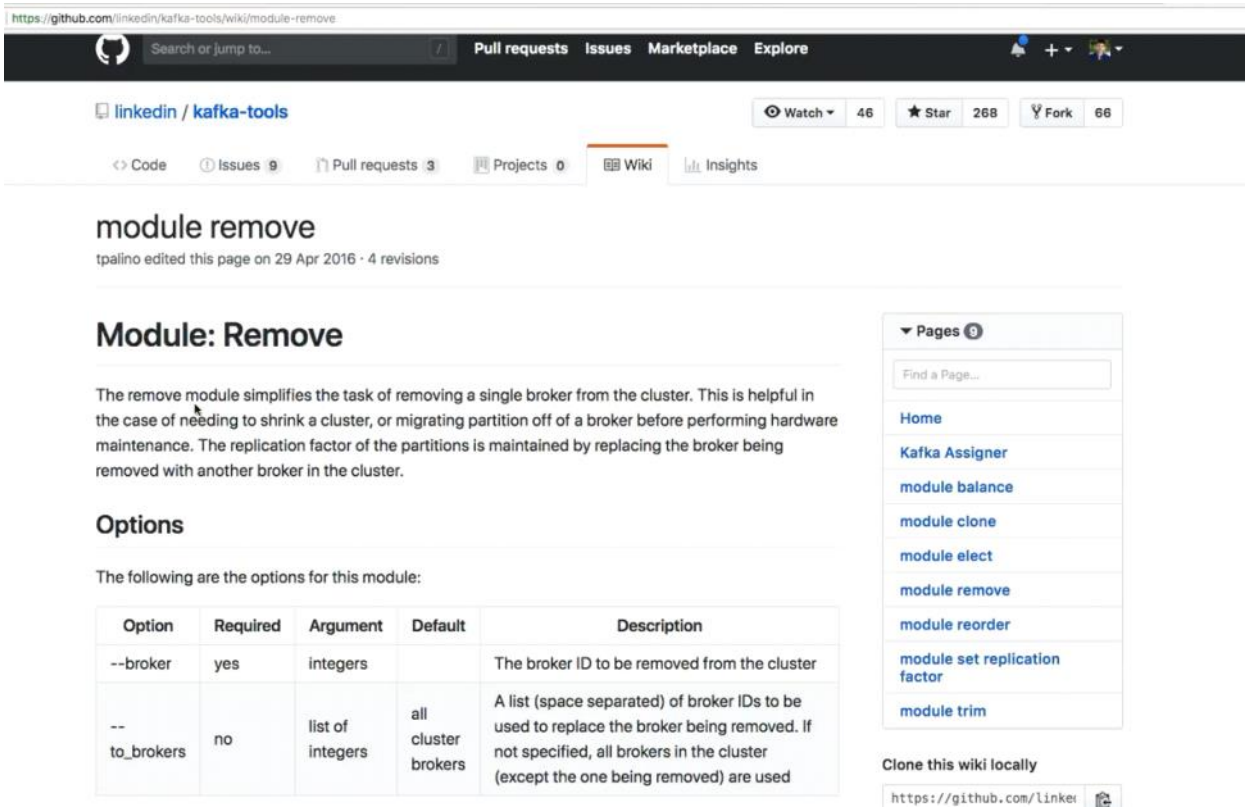
- If the broker is removed first, it's impossible to remove the partitions assignment without tinkering with Zookeeper (and it's really not recommended)

Removing a Broker Hands On

- Let's get started and remove the broker 4

- We'll remove all the partitions from that broker using the LinkedIn Kafka Tools
- We'll terminate the broker instance
- We'll delete the EBS and IP resources

нужно установить дополнительный модуль kafka tools: remove



скрипт

```
########
#### ADMIN MACHINE
########

export ZK_HOST="172.31.9.21:2181"

# Documentation: https://github.com/linkedin/kafka-tools/wiki/module-remove

# Remove broker 4
kafka-assigner -z $ZK_HOST -g remove -b 4

# Execute
kafka-assigner -z $ZK_HOST -e remove -b 4
```

* cluster upgrade

4 декабря 2020 г. 14:50

Why is this a whole section?

- Upgrading Kafka Brokers is by far the most important operation you'll have to do on your Cluster
- It's recommended to upgrade very often in order to keep up with bug fixes / new features / performance improvements
- The more you wait to upgrade a broker, the more painful it will be as there will be many versions jumps

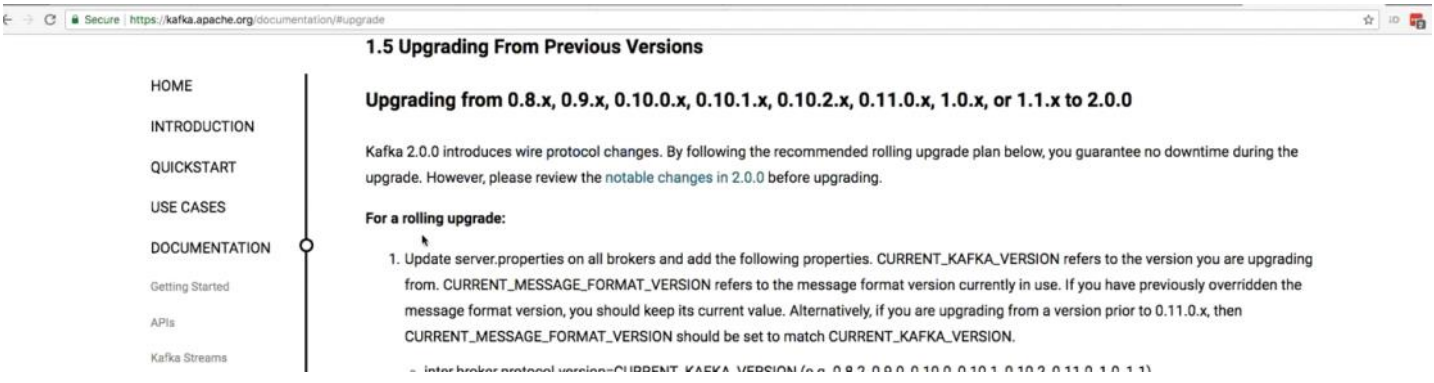
Kafka Upgrades Steps and Documentation.

- Steps we have to perform (roll restart at the end of each step)

1. Setting inter broker and log version to current Kafka Version
2. Upgrade Kafka Binaries
3. Change inter broker protocol version
4. Upgrade Kafka Clients if documentation specifies it (all or most of them to avoid up & down conversion)
5. Upgrade message protocol version

Always read documentation at:
https://kafka.apache.org/documentation/#upgrade

шаг1 обновим конфиги кафки



```
 GNU nano 2.3.1     File: kafka.properties        Modified

auto.create.topics.enable=false
offsets.topic.replication.factor=3

# upgrades
inter.broker.protocol.version=1.1
```

########
KAFKA MACHINE
########

edit kafka properties
nano kafka.properties
inter.broker.protocol.version=1.1
log.message.format.version=1.1

########
ADMIN MACHINE
########
roll restart kafka
kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop kafka" --check-count 2

шаг2 обновим саму прогу кафки

```
[ec2-user@ip-172-31-1-31 ~]$ wget http://mirrors.standaloneinstaller.com/apache/kafka/2.0.0/kafka_2.12-2.0.0.tgz
```

########
KAFKA MACHINE
########

download kafka binaries
cd /home/ec2-user
http://home.apache.org/~rsivaram/kafka-2.0.0-rc3/kafka_2.12-2.0.0.tgz
tar xf kafka_2.12-2.0.0.tgz

update symlink
ln -sfn /home/ec2-user/kafka_2.12-2.0.0 /home/ec2-user/kafka
ll kafka
new option in Kafka 2.0.0!
kafka-topics.sh --version

########
ADMIN MACHINE
########
roll restart kafka
kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop kafka" --check-count 2

шаг3 установим новую версию протокола в конфиге кафки

```
# upgrades
inter.broker.protocol.version=2.0
log.message.format.version=1.1
```

########

```
#### KAFKA MACHINE
########

# edit kafka properties, change inter.broker.protocol.version
nano kafka.properties
# inter.broker.protocol.version=2.0
# log.message.format.version=1.1


########
#### ADMIN MACHINE
########
# roll restart kafka
kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop
kafka" --check-count 2
```

шаг4 обновим все клиентские приложения

```
### CONSUMER / PRODUCERS ####
### UPGRADE CLIENT IS THERE'S A NEW MESSAGE FORMAT FIRST ####

########
#### KAFKA MACHINE
########

# edit kafka properties, edit log.message.format.version
nano kafka.properties
# inter.broker.protocol.version=2.0
# log.message.format.version=2.0


########
#### ADMIN MACHINE
########
# roll restart kafka
kafka-rolling-restart --cluster-type kafka --start-command "systemctl start kafka" --stop-command "systemctl stop
kafka" --check-count 2


########
#### KAFKA MACHINE
########

# optional: remove kafka 1.1.1 binaries
rm -r -f kafka_2.12-1.1.1/
```

шаг5