Kafka Admin/Ops

❖ Topics are added and modified using the topic tool

❖ *Replication factor* - replicas count amount of Kafka Brokers needed

  ❖ use replication factor of at least 3 (or 2)

    ❖ survive outages, head-room for upgrades and maintenance -ability to bounce servers

```
#!/usr/bin/env bash

cd ~/kafka-training

kafka/bin/kafka-topics.sh \
    --create \
    --zookeeper localhost:2181 \
    --replication-factor 3 \
    --partitions 23 \
    --topic stock-prices \
    --config min.insync.replicas=2
```

Partition count - how much topic log will get sharded

  ❖ determines broker count – if you have a partition count of 3, but have 5 servers, 2 not host topic log

  ❖ consumers parallelism - active consumer count in consumer group

2

Modifying Topics

❖ You can modify topic configuration

❖ You can add partitions

   ❖ ***existing data partition don't change!***

   ❖ Consumers semantics could break, data is not moved from existing partitions to new partitions

❖ You can use ***bin/kafka-topics.sh —alter*** to modify a topic

   ❖ add partitions - ***you can't remove partitions!***

   ❖ ***you can't change replication factor!***

   ❖ modify config or ***delete*** it

❖ You can use ***bin/kafka-topics.sh —delete*** to delete a topic

   ❖ Has to be enabled in Kafka Broker config - ***delete.topic.enable=true***

Review of Kafka Topic Tools

Create Topic

```
#!/usr/bin/env bash

cd ~/kafka-training

## Create a new Topic
kafka/bin/kafka-topics.sh \
        --create \
        --zookeeper localhost:2181 \
        --replication-factor 2 \
        --partitions 3 \
        --topic stock-prices \
        --config min.insync.replicas=1 \
        --config retention.ms=60000
```

Describe Topic

```
#!/usr/bin/env bash
cd ~/kafka-training

# Describe existing topic
kafka/bin/kafka-topics.sh \
        --describe \
        --topic stock-prices \
        --zookeeper localhost:2181
```

Delete Topic

```
#!/usr/bin/env bash
cd ~/kafka-training

# Delete Topic
kafka/bin/kafka-topics.sh \
    --delete \
    --zookeeper localhost:2181 \
    --topic stock-prices
```

4

Alter Topic

```
alter-topic.sh  ×

1    #!/usr/bin/env bash
2
3    cd ~/kafka-training
4
5    ## Alter the topic
6    kafka/bin/kafka-topics.sh --alter \
7         --zookeeper localhost:2181 \
8         --partitions 13 \
9         --topic stock-prices \
10        --config min.insync.replicas=2 \
11        --delete-config retention.ms
12
```

❖ Changes *min.insync.replicas* from 1 to 2

❖ Changes partition count (*partitions*) from 3 to 13

❖ Use *—delete-config* to delete *retention.ms* configuration

5

Modifying Topics with Alter

```
$ bin/delete-topic.sh
```
Topic stock-prices is marked for deletion.
```
$ bin/create-topic.sh
```
Created topic "stock-prices".
```
$ bin/describe-topic.sh
```
Topic:stock-prices

Topic: stock-prices	Partition: 0	Leader: 1	Replicas: 1,2	Isr:1,2
Topic: stock-prices	Partition: 1	Leader: 2	Replicas: 2,0	Isr:2,0
Topic: stock-prices	Partition: 2	Leader: 0	Replicas: 0,1	Isr:0,1

```
$ bin/alter-topic.sh
```
Adding partitions succeeded
```
$ bin/describe-topic.sh
```
Topic:stock-prices

Topic: stock-prices	Partition: 0	Leader: 1	Replicas: 1,2	Isr:1,2
Topic: stock-prices	Partition: 1	Leader: 2	Replicas: 2,0	Isr:2,0
Topic: stock-prices	Partition: 2	Leader: 0	Replicas: 0,1	Isr:0,1
Topic : stock-prices	Partition: 11	Leader: 0	Replicas: 0,1	Isr:0,1
Topic : stock-prices	Partition: 12	Leader: 1	Replicas: 1,0	Isr:1,0

6

- Kafka Clustering detects Kafka broker shutdown or failure

  - Elects new partition leaders

  - For maintenance shutdowns Kafka supports graceful shutdown

- Graceful shutdown optimizations - ***controlled.shutdown.enable=true***

  - Topic logs data synced to disk = faster log recovery on restart by avoiding log recovery and checksum validation

  - Partitions are migrated to other Kafka brokers

  - Clean, fast leadership transfers, reduces partitions unavailability

  - Controlled shutdown fails if replicas on broker do not have in-sync replicas on another server

Balancing Leadership

❖ When broker stops or *crashes* leadership moves to surviving brokers

  ❖ crashed broker's partitions *transfers* to other replicas

  ❖ If broker restarted becomes a *follower* for all its partitions

    ❖ Recall only *leaders* read and write

bin/kafka-preferred-replica-election.sh —zookeeper host:port

    ❖ **kafka-preferred-replica-election.sh** will rebalance leadership, OR

    ❖ Kafka Broker Config: *auto.leader.rebalance.enable=true*

      ❖ auto-balance leaders on change

Kafka balancing across racks

- Kafka has rack awareness

  - spreads same partition replicas to different racks or AWS AZ (EC2 availability zones)

  - Survive single rack or single AZ outage

  - broker config: **broker.rack=us-west-2a**

- During topic creation, rack constraint used to span replicas to as many racks as possible

  - min(#racks, replication-factor)

- Assignment of replicas to brokers ensures leaders count per brokersame, regardless rack distribution if racks have equal number of brokers

  - if rack has fewer brokers, then each broker in rack will get more replicas

  - keep broker count the same in each rack or AZ

Checking Consumer Position

- Useful to see position of your consumers

  - Especially MirrorMaker consumers

- Tool to show consumer position

  - **bin/kafka-consumer-groups.sh**

- Shows **Topic** and which **Client** (client id) and Consumer (consumer id) from consumer group is working with which Topic **Partition**

  - GUID for Consumer ID based on **client id plus GUID**

- **Shows Lag between Consumer and Log**

- **Shows Lag between Producer and what consumer can see (replicated vs non-replicated)**

```
check-consumer-offsets.sh ×
1   #!/usr/bin/env bash
2
3   cd ~/kafka-training
4
5   BOOTSTRAP_SERVERS="localhost:9092,localhost:9093"
6
7   kafka/bin/kafka-consumer-groups.sh --describe  \
8       --bootstrap-server "$BOOTSTRAP_SERVERS"    \
9       --group StockPriceConsumer
```

❖ Using —describe

❖ Specifies bootstrap server lists not ZooKeeper

❖ Specifies name of ConsumerGroup

❖ Will show lag, etc. for every consumer in group

```
$ bin/check-consumer-offsets.sh
```

TOPIC	PARTITION	CURRENT-LOG	LOG-END-OFFSET	LAG	HOST	CLIENT-ID
Stock-prices	5	910	910	0	/10.0.1.11	green-2
Stock-prices	4	611	611	0	/10.0.1.11	green-1
Stock-prices	2	946	946	0	/10.0.1.11	blue-2
Stock-prices	6	39	39	0	/10.0.1.11	red-0
Stock-prices	8	13	13	0	/10.0.1.11	red-2
Stock-prices	1	13	13	0	/10.0.1.11	blue-1
Stock-prices	3	1534	1534	0	/10.0.1.11	green-0
Stock-prices	7	-	0	-	/10.0.1.11	red-1
Stock-prices	0	611	611	0	/10.0.1.11	blue-0

❖ Shows *Topic* and which *Client* from the consumer group is working with which Topic *Partition* - Note also shows GUID for Consumer ID (not shown)

❖ *Current offset* is what is visible to Consumer (replicated to ISRs)

❖ *Log end* shows what the leader of has written

$ bin/check-consumer-offsets.sh

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	HOST	CLIENT-ID
Stock-prices	1	524	524	0	/10.0.1.11	blue-1
Stock-prices	8	380	524	144	/10.0.1.11	red-2
Stock-prices	7	0	0	0	/10.0.1.11	red-1
Stock-prices	3	2959	3067	108	/10.0.1.11	green-0
Stock-prices	0	909	1122	213	/10.0.1.11	blue-0
Stock-prices	6	1464	1572	108	/10.0.1.11	red-0
Stock-prices	5	1277	1421	144	/10.0.1.11	green-2
Stock-prices	4	934	1122	188	/10.0.1.11	green-1
Stock-prices	2	2464	2993	529	/10.0.1.11	blue-2

❖ Notice Partition 8, the replication is behind Current Offset is behind Log End

❖ Notice how partition 3 has 6x as many records as Partition 1

  ❖ Could be an example of a hot spot!

❖ Notice how Partition 7 has no records so red-2 is idle!

13

Managing Consumer Groups

❖ ConsumerGroupCommand - *kafka-consumer-groups.sh*

  ❖ you can also list, describe, or delete consumer groups

❖ Delete restriction -

  ❖ Only works with older clients

  ❖ No need for new client API because group is deleted automatically when last committed offset for group expires

  ❖ If using older consumers that relied on ZooKeeper then you can use *—delete*

14

14

List Consumers

```bash
#!/usr/bin/env bash

cd ~/kafka-training

BOOTSTRAP_SERVERS="localhost:9092,localhost:9093"

kafka/bin/kafka-consumer-groups.sh --list \
    --bootstrap-server "$BOOTSTRAP_SERVERS"
```

❖   Use —list to get a list of consumers

Expanding Kafka cluster

- ❖ Adding Kafka Brokers to cluster is simple

  - ❖ need unique broker id

  - ❖ new Kafka Brokers are not automatically assigned Topic partitions

  - ❖ You need to migrate partitions to it

- ❖ Migrating Topic Partitions is manually initiated

  - ❖ New Kafka Broker becomes followers of partitions

  - ❖ When it becomes ISR set member, then it gains leadership over partitions assigned to it

  - ❖ Once it becomes leader, existing replica will delete partition data if needed

- ❖ Kafka provides a partition reassignment tool

Kafka Partition Reassignment Tool

- ❖ partition can be moved across brokers

- ❖ avoid hotspots, balance load on brokers

- ❖ you have to look at load on Kafka Broker

  - ❖ use *kafka-consumer-groups.sh*

  - ❖ other admin tools to find hotspots (top, KPIs, etc.)

  - ❖ balance as needed

- **GENERATE A PLAN—*generate***

  - Inputs: Topics List, and Kafka Broker List

  - Generates ***reassignment plan*** to move all topic partitions to new Kafka Brokers

- **EXECUTE A PLAN —execute**

  - Input: ***reassignment plan*** (--***reassignment-json-file***)

  - Action: Does partition reassignment using plan

- **CHECK STATUS OF EXECUTE PLAN —verify**

  - Shows status of —execute

  - Outputs: Completed Successfully, Failed or In-Progress

```
reassign-partitions-generate-plan.sh  ×

1   #!/usr/bin/env bash
2   CONFIG=`pwd`/config
3   cd ~/kafka-training
4
5   # Generate Reassignment Plan
6   kafka/bin/kafka-reassign-partitions.sh --generate \
7       --broker-list 0,1,2,3 \
8       --topics-to-move-json-file "$CONFIG/move-topics.json" \
9       --zookeeper localhost:2181 > "$CONFIG/assignment-plan.json"
```

```
move-topics.json  ×

1   { "version": 1,
2     "topics": [
3       { "topic": "stocks" },
4       { "topic": "stock-prices" }]}
```

❖   Added 4th Broker! Now we want it to have some partitions

❖   ***move-topics.json*** - list of topics to move in JSON format

❖   Generates assignment plan which needs to be edited

Tos

```json
assignment-plan.json ×

        partitions  replicas
1    {
2        "version": 1,
3        "partitions": [
4            {
5                "topic": "stocks",
6                "partition": 7,
7                "replicas": [
8                    0,
9                    1,
10                   2
11               ]
12           },
13           {
14               "topic": "stocks",
15               "partition": 2,
16               "replicas": [
17                   3,
18                   2,
19                   0
20               ]
```

❖ Assignment Plan
❖ List of partitions
❖ List of Replicas
❖ Replicas might be removed to new
   Kafka Broker after plan executes
❖ Need to execute plan

20

- After we add a new broker,

  - add it to the —broker-list

  - Run generate plan

  - Execute plan

- To decommission Kafka Broker

  - Remove it from the —broker-list

  - Run generate plan, execute generate plan

```bash
#!/usr/bin/env bash
CONFIG=`pwd`/config
cd ~/kafka-training

# Execute reassignment plan
kafka/bin/kafka-reassign-partitions.sh --execute \
    --reassignment-json-file "$CONFIG/assignment-plan.json" \
    --throttle 100000 \
    --zookeeper localhost:2181
```

- ❖ Executes reassignment plan

- ❖ Use generated plan or use modified generated plan

- ❖ Set throttle rate (optional) so it does not all happen at once

  - ❖ reduces load on Kafka Brokers

22

```
reassign-partitions-verify-plan.sh  ×
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  # Verify executing reassignment plan
6  kafka/bin/kafka-reassign-partitions.sh --verify \
7      --reassignment-json-file "$CONFIG/assignment-plan.json" \
8      --zookeeper localhost:2181
```

❖ Verify/Monitor reassignment plan

❖ Use generated plan or use modified generated plan that is already running

❖ Let's you know when the plan is done

```
reassign-partitions-generate-plan.sh ×

1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  # Generate Reassignment Plan
6  kafka/bin/kafka-reassign-partitions.sh --generate \
7      --broker-list 0,1,2 \
8      --topics-to-move-json-file "$CONFIG/move-topics.json" \
9      --zookeeper localhost:2181 > "$CONFIG/assignment-plan.json"
```

❖ Remove 4th Broker (3)! Now we want it reassign its partitions

❖ Generates assignment plan that moves partitions to 0,1,2

24

Setting quotas

❖ You can configure quotas for client-id and user using ***kafka-configs.sh***

❖ Clients receive an unlimited quota

❖ You can set custom quotas for

    ❖ (user, client-id) pair

    ❖ user

    ❖ client-id

25

```bash
#!/usr/bin/env bash

cd ~/kafka-training

## Add limit for stock_analyst user running as clientId stockConsumer
kafka/bin/kafka-configs.sh --alter \
    --zookeeper localhost:2181 \
    --add-config 'producer_byte_rate=1024,consumer_byte_rate=2048' \
    --entity-type users \
    --entity-name stock_analyst \
    --entity-type clients \
    --entity-name stockConsumer
```
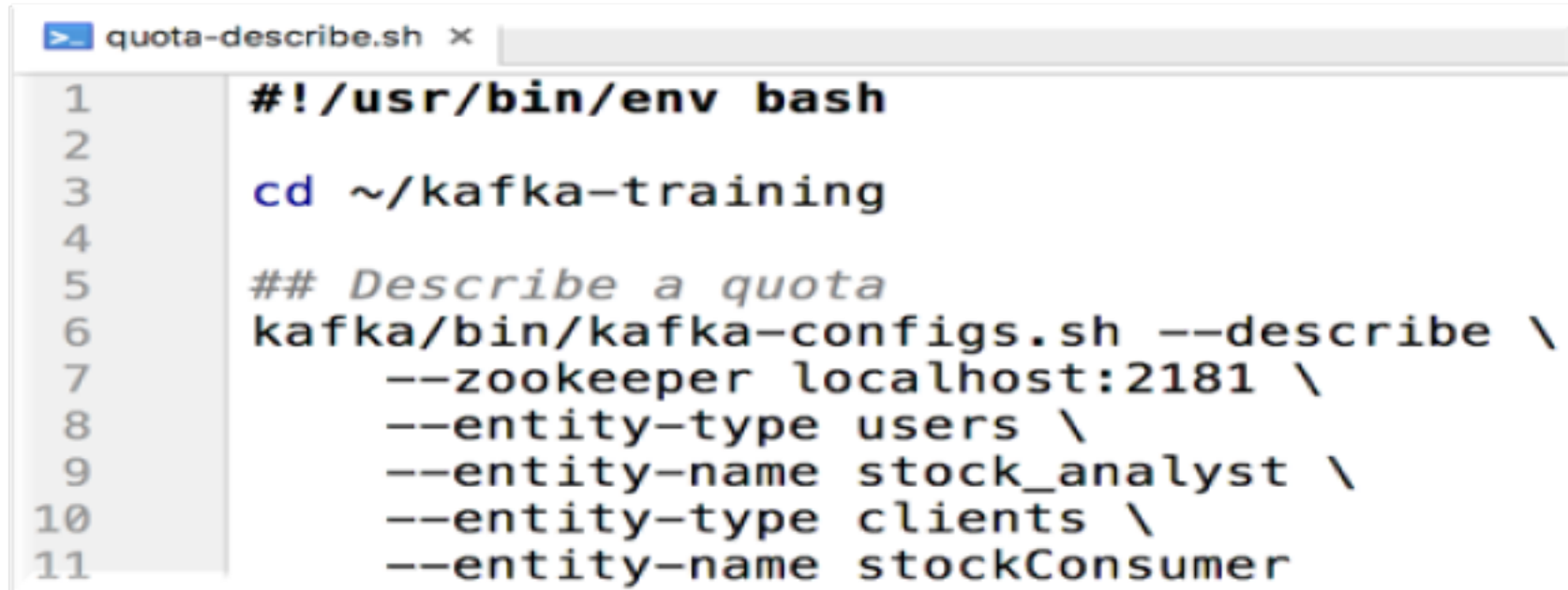
❖ User stock_analyst

❖ client id stockConsumer

```
quota-default-user.sh  ×

1   #!/usr/bin/env bash
2
3   cd ~/kafka-training
4
5   ## Add limit to default user
6   kafka/bin/kafka-configs.sh --alter \
7       --zookeeper localhost:2181 \
8       --add-config 'producer_byte_rate=512,consumer_byte_rate=512' \
9       --entity-type users --entity-default
```

❖ Sets default quota for users

Default Quota for Clients

```
quota-default-client.sh ×

1    #!/usr/bin/env bash
2
3    cd ~/kafka-training
4
5    ## Add limit to default client
6    kafka/bin/kafka-configs.sh --alter \
7         --zookeeper localhost:2181 \
8         --add-config 'producer_byte_rate=512,consumer_byte_rate=512' \
9         --entity-type clients --entity-default
```

❖ Sets default quota for clients

Describe a Quota

```
quota-describe.sh  ×
1    #!/usr/bin/env bash
2
3    cd ~/kafka-training
4
5    ## Describe a quota
6    kafka/bin/kafka-configs.sh --describe \
7         --zookeeper localhost:2181 \
8         --entity-type users \
9         --entity-name stock_analyst \
10        --entity-type clients \
11        --entity-name stockConsumer
```

❖ You can see what quotas are set for a user

Describe a Quota Output

```
$ bin/quota-describe.sh

Configs for user-principal 'stock_analyst', client-id
'stockConsumer'  are
producer_byte_rate=1024,consumer_byte_rate=2048
```

❖ Output from describe quota

❖ Kafka batches and compresses records

  ❖ Both producer and consumer can achieve high-throughput even over a high-latency connection

  ❖ If needed increase the TCP socket buffer sizes for the producer, consumer, and broker

    ❖ *socket.send.buffer.bytes* and *socket.receive.buffer.bytes*

❖ Not a good idea to span DCs or regions

  ❖ Really bad for ZooKeeper

  ❖ More outages due to latency

Important Client Configurations

- Producer configurations control

  - acks

  - compression

  - batch size

- Consumer Configuration

  - fetch size

```
server-0.properties ×

1    ## Increment by 1 for each broker
2    broker.id=0
3
4    # Kafka should have its own dedicated disk(s) or use SSD(s)
5    # To increase reads and writes, add more disks/log dirs JBOD.
6    log.dirs=./logs/kafka-0
7
8    ## Log config
9    default.replication.factor=3
10   num.partitions=8
11
12   ## Data must be replicated to at least two brokers
13   min.insync.replicas=2
14
15   ## Don't allow un-managed topics for production
16   auto.create.topics.enable=false
17
18   ## Run brokers spread over AZs or Racks
19   broker.rack=us-west2-a
20
21   ## Number of concurrent requests allowed
22   queued.max.requests=1000
23
24   ## Allow leaders to auto rebalance
25   auto.leader.rebalance.enable=true
```

Java GC config

```
Xms6g    Xmx6g

-XX:MetaspaceSize=96m

-XX:+UseG1GC

-XX:MaxGCPauseMillis=20

-XX:InitiatingHeapOccupancyPercent=35

-XX:G1HeapRegionSize=16M

-XX:MinMetaspaceFreeRatio=50
```

- ❖ Heap Space should be 25% to 35% of available space for server

- ❖ Leave 50% for OS, Remember Kafka uses OS page cache

- ❖ Other tweaks for GC to limit overhead

Lab : Administration