KAFKA CHEAT SHEET

Display Topic Information

```
$ kafka-topics.sh --describe --zookeeper localhost:2181 --topic
beacon
Topic:beacon  PartitionCount:6  ReplicationFactor:1 Configs:
  Topic: beacon Partition: 0  Leader: 1 Replicas: 1 Isr: 1
  Topic: beacon Partition: 1  Leader: 1 Replicas: 1 Isr: 1
```

Add Partitions to a Topic

```
$ kafka-topics.sh --alter --zookeeper localhost:2181 --topic
beacon --partitions 3
WARNING: If partitions are increased for a topic that has a
key, the partition logic or ordering of the messages will be
affected
Adding partitions succeeded!
```

Change topic retention i.e set SLA

```
bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic
mytopic --config retention.ms=28800000*
```

This set retention of 8-hours on messages coming to topic *mytopic*. After 8 hours message will be deleted.

Delete Topic

```
$ kafka-run-class.sh kafka.admin.DeleteTopicCommand --zookeeper
localhost:2181 --topic test
```

```
kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic
file_acquire_complete
kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic job_result
kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic trigger_match
kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic event_result
```

```
$ kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic job_result
Created topic "job_result".
```

```
$ kafka-topics.sh --list --zookeeper localhost:2181
event_result
file_acquire_complete
job_result
trigger_match
```

```
$ kafka-console-producer.sh --broker-list localhost:9092 --
topic test
```

```
$ kafka-console-consumer.sh --zookeeper localhost:2181 --topic
test --from-beginning
```

List existing topics

`bin/kafka-topics.sh --zookeeper localhost:2181 --list`

Purge a topic

`bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic mytopic --config retention.ms=1000`

... wait a minute ...

```
bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic mytopic --delete-config retention.ms
```

Delete a topic

```
bin/kafka-topics.sh --zookeeper localhost:2181 --delete --topic mytopic
```

Get the earliest offset still in a topic

```
bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic mytopic --time -2
```

Get the latest offset still in a topic

```
bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic mytopic --time -1
```

Consume messages with the console consumer

```
bin/kafka-console-consumer.sh --new-consumer --bootstrap-server localhost:9092 --topic mytopic --from-beginning
```

GET THE CONSUMER OFFSETS FOR A TOPIC

```
bin/kafka-consumer-offset-checker.sh --zookeeper=localhost:2181 --topic=mytopic --group=my_consumer_group
```

Read from __consumer_offsets

Add the following property to `config/consumer.properties`:

```
exclude.internal.topics=false
```

```
bin/kafka-console-consumer.sh --consumer.config config/consumer.properties --from-beginning --topic __consumer_offsets --zookeeper localhost:2181 --formatter "kafka.coordinator.GroupMetadataManager\$OffsetsMessageFormatter"
```

KAFKA CONSUMER GROUPS

List the consumer groups known to Kafka

`bin/kafka-consumer-groups.sh --zookeeper localhost:2181 --list` (old api)

`bin/kafka-consumer-groups.sh --new-consumer --bootstrap-server localhost:9092 --list` (new api)

View the details of a consumer group

```
bin/kafka-consumer-groups.sh --zookeeper localhost:2181 --describe --group <group name>
```

KAFKACAT

Getting the last five message of a topic

```
kafkacat -C -b localhost:9092 -t mytopic -p 0 -o -5 -e
```

ZOOKEEPER

Starting the Zookeeper Shell

```
bin/zookeeper-shell.sh localhost:2181
```