Apache Kafka CLI Event streaming tools

START HERE

`kafka-topics.sh --bootstrap-server <broker> --list`
What topics exist?

`kafka-topics.sh --bootstrap-server <broker> --create --topic <topic>`
Make a topic

`kafka-topics.sh --bootstrap-server <broker> --describe --topic <topic>`
Partitions and leaders

`kafka-console-producer.sh --bootstrap-server <broker> --topic <topic>`
Type messages in

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --from-beginning`
Read it all

`kafka-consumer-groups.sh --bootstrap-server <broker> --list`
Who is consuming?

`kafka-consumer-groups.sh --bootstrap-server <broker> --describe --group <group>`
Check consumer lag

`kafka-configs.sh --bootstrap-server <broker> --describe --entity-type topics --entity-name <topic>`
Show topic settings

Producer writes records → **`key picks partition`** → **Topic** split in partitions → **`offset per partition`** → **Consumer group** one member per partition → **`commits offsets`** → **`__consumer_offsets`** saved position

CONCEPTS

On a tarball install the tools are bin/*.sh; package and container installs often drop the .sh suffix.

Order is guaranteed only within a partition; give related records the same key.

Every tool needs --bootstrap-server; the old --zookeeper flag is gone in KRaft-era Kafka.

TOPICS

`kafka-topics.sh --bootstrap-server <broker> --list`
List topics

`kafka-topics.sh --bootstrap-server <broker> --create --topic <topic>`
Create a topic
Uses broker defaults for partitions and replication.

`kafka-topics.sh --bootstrap-server <broker> --create --topic <topic> --partitions <n> --replication-factor <n>`
Create with sizing

`kafka-topics.sh --bootstrap-server <broker> --describe --topic <topic>`
Show partitions, leaders, replicas

`kafka-topics.sh --bootstrap-server <broker> --alter --topic <topic> --partitions <n>`
Add partitions
Partitions can only grow, never shrink; key-to-partition mapping changes.

`kafka-topics.sh --bootstrap-server <broker> --delete --topic <topic>`
Delete a topic

`kafka-topics.sh --bootstrap-server <broker> --describe --under-replicated-partitions`
Find unhealthy partitions

PRODUCE

`kafka-console-producer.sh --bootstrap-server <broker> --topic <topic>`
Send lines as messages
One record per line; Ctrl+D or Ctrl+C ends.

`kafka-console-producer.sh --bootstrap-server <broker> --topic <topic> --property parse.key=true --property key.separator=:`
Send key:value records

`kafka-console-producer.sh --bootstrap-server <broker> --topic <topic> < <file>`
Send a file line by line

`kafka-console-producer.sh --bootstrap-server <broker> --topic <topic> --producer-property acks=all`
Set a producer setting

CONSUME

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic>`
Read new messages

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --from-beginning`
Read from the start
Only applies to a group with no committed offsets.

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --group <group>`
Consume as a named group

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --property print.key=true`
Show keys too

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --property print.timestamp=true --property print.offset=true`
Show time and offset

CONSUME (cont.)

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --max-messages <n>`
Stop after n messages

`kafka-console-consumer.sh --bootstrap-server <broker> --topic <topic> --partition <n> --offset <n>`
Read from one offset

CONSUMER GROUPS

`kafka-consumer-groups.sh --bootstrap-server <broker> --list`
List groups

`kafka-consumer-groups.sh --bootstrap-server <broker> --describe --group <group>`
Show offsets and lag
LAG is log-end offset minus committed offset.

`kafka-consumer-groups.sh --bootstrap-server <broker> --describe --group <group> --members --verbose`
Show members and assignments

`kafka-consumer-groups.sh --bootstrap-server <broker> --describe --group <group> --state`
Show group state

`kafka-consumer-groups.sh --bootstrap-server <broker> --group <group> --reset-offsets --to-earliest --topic <topic> --dry-run`
Preview an offset reset
Default without --execute is a dry run; the group must be inactive.

`kafka-consumer-groups.sh --bootstrap-server <broker> --group <group> --reset-offsets --to-earliest --topic <topic> --execute`
Rewind to the start

`kafka-consumer-groups.sh --bootstrap-server <broker> --group <group> --reset-offsets --to-latest --all-topics --execute`
Skip to the end

CONFIGS AND ACLS

`kafka-configs.sh --bootstrap-server <broker> --describe --entity-type topics --entity-name <topic>`
Show topic overrides

`kafka-configs.sh --bootstrap-server <broker> --alter --entity-type topics --entity-name <topic> --add-config retention.ms=<ms>`
Set a topic config

`kafka-configs.sh --bootstrap-server <broker> --alter --entity-type topics --entity-name <topic> --delete-config retention.ms`
Revert to default

INSPECT AND OPERATE

`kafka-get-offsets.sh --bootstrap-server <broker> --topic <topic>`
Show end offset per partition

`kafka-metadata-quorum.sh --bootstrap-server <broker> describe --status`
Show KRaft quorum status

RECIPES

Create, send and read back

```
kafka-topics.sh --bootstrap-server <broker> --create --topic orders --partitions 3
echo 'k1:hello' | kafka-console-producer.sh --bootstrap-server <broker> --topic orders --property parse.key=true --property key.separator=:
kafka-console-consumer.sh --bootstrap-server <broker> --topic orders --from-beginning --property print.key=true --max-messages 1
```

Replay a group from a time

```
kafka-consumer-groups.sh --bootstrap-server <broker> --group app --reset-offsets --to-datetime 2026-01-01T00:00:00.000 --topic orders --dry-run
# stop the consumers, then repeat with --execute
```