kcat Kafka on the shell

START HERE

`kcat -b <broker> -L`
List brokers and topics

`kcat -b <broker> -t <topic> -C -J`
Messages as JSON

`kcat -b <broker> -t <topic> -C`
Read a topic

`kcat -b <broker> -X list`
Show config properties

`kcat -b <broker> -t <topic> -C -e`
Read, then exit

`kcat -b <broker> -t <topic> -P`
Send lines from stdin

`kcat -b <broker> -G <group> <topic>`
Join a consumer group

`kcat -b <broker> -t <topic> -C -o -10 -e`
Last 10 per partition

stdin one message per line → `-P -t -p` → **Topic partition** ordered log → `-C -o -e` → **stdout** payload or -f / -J → `pipe to jq` → **Your tools** jq, grep, files

CONCEPTS

The mode flag picks the job: -P produce, -C consume, -G group consume, -L metadata, -Q offset query. Without one, kcat guesses from stdin and stdout.

Put -b, -X and credentials in a config file (-F or ~/.config/kcat.conf) so commands stay short.

METADATA

`kcat -b <broker> -L`
List brokers, topics and partitions

`kcat -b <broker> -L -t <topic>`
Show one topic only

`kcat -b <broker> -L -J`
Metadata as JSON

`kcat -b <broker> -Q -t <topic>:<p>:<ms>`
Offset at a timestamp

Timestamp is epoch milliseconds; returns the first offset at or after it.

CONSUME

`kcat -b <broker> -t <topic> -C`
Read all partitions and wait

`kcat -b <broker> -t <topic> -C -e`
Exit at end of partition

CONSUME (cont.)

`kcat -b <broker> -t <topic> -C -p <n>`
Read one partition

`kcat -b <broker> -t <topic> -C -o beginning`
Start from the oldest message

`kcat -b <broker> -t <topic> -C -o end`
Only new messages

`kcat -b <broker> -t <topic> -C -o -<n>`
Last n messages per partition

`kcat -b <broker> -t <topic> -C -o <offset> -p <n>`
Start at a given offset

`kcat -b <broker> -t <topic> -C -c <n>`
Stop after n messages

`kcat -b <broker> -t <topic> -C -o s@<ms> -o e@<ms>`
Read a time range

s@ and e@ take epoch milliseconds; start and stop of the range.

OUTPUT FORMAT

`kcat -b <broker> -t <topic> -C -J`
One JSON envelope per message

`kcat -b <broker> -t <topic> -C -K:`
Print key:value

OUTPUT FORMAT (cont.)

`kcat -b <broker> -t <topic> -C -f '<fmt>'`
Custom output format

`kcat -b <broker> -t <topic> -C -f '%t[%p]@%o %k=%s\n'`
Topic, partition, offset, key, value

`kcat -b <broker> -t <topic> -C -f '%h\n'`
Show message headers

Format tokens (-f)

Token	Prints
%s	Payload
%S	Payload length
%k	Key
%K	Key length
%t	Topic
%p	Partition
%o	Offset
%T	Timestamp (ms)
%h	Headers
%R	Payload length, then payload (binary)

PRODUCE

`kcat -b <broker> -t <topic> -P`
Send each stdin line as a message

`kcat -b <broker> -t <topic> -P -l <file>`
Send a file, one message per line

PRODUCE (cont.)

`kcat -b <broker> -t <topic> -P -K:`
Split key:value on the colon

`kcat -b <broker> -t <topic> -P -k <key>`
Use a fixed key

`kcat -b <broker> -t <topic> -P -p <n>`
Send to partition n

`kcat -b <broker> -t <topic> -P -H <name>=<value>`
Add a message header

`kcat -b <broker> -t <topic> -P -z <codec>`
Compress (gzip, snappy, lz4)

`kcat -b <broker> -t <topic> -P -Z -K:`
Empty value becomes a null tombstone

CONSUMER GROUPS

`kcat -b <broker> -G <group> <topic>`
Balanced consume with a group

`kcat -b <broker> -G <group> <topic1> <topic2>`
Group over several topics

`kcat -b <broker> -G <group> <topic> -X auto.offset.reset=earliest`
New group starts at the beginning

CONSUMER GROUPS (cont.)

`kcat -b <broker> -G <group> <topic> -e`
Drain the backlog, then exit

-G commits offsets for the group. Use -C when you only want to look without moving any group.

CONFIG AND SECURITY

`kcat -F <file>`
Read settings from a config file

`kcat -b <broker> -X <prop>=<value>`
Set any librdkafka property

`kcat -b <broker> -X list`
List all known properties

`kcat -b <broker> -X dump`
Print the effective config

`kcat -b <broker> -X security.protocol=SSL -X ssl.ca.location=<ca.pem>`
TLS with a CA file

`kcat -b <broker> -X security.protocol=SASL_SSL -X sasl.mechanism=PLAIN -X sasl.username=<user> -X sasl.password=<pass>`
SASL over TLS

CONFIG AND SECURITY (cont.)

~/.config/kcat.conf

```
bootstrap.servers=<broker>:9093
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.username=<user>
sasl.password=<pass>
```

SCHEMAS

`kcat -b <broker> -t <topic> -C -s avro -r <registry-url>`
Decode Avro via Schema Registry

RECIPES

Last 100 messages as pretty JSON

```
kcat -b <broker> -t orders -C -o -100 -e -J | jq .
```

Send a key and value

```
echo 'user42:{"status":"paid"}' | kcat -b <broker> -t orders -P -K:
kcat -b <broker> -t orders -C -e -K:
```

Replay a time window to a file

```
kcat -b <broker> -t orders -C -o s@1700000000000 -o e@1700003600000 -e > window.txt
```

Copy one topic to another

```
kcat -b <src> -t orders -C -e -K: -u | kcat -b <dst> -t orders-copy -P -K:
```