NATS CLI Messaging and JetStream

START HERE

`nats context add <name> --server <url> --select`
Save a server

`nats stream add <name> --subjects "<subject>.>"`
Persist messages

`nats sub <subject>`
Listen to a subject

`nats stream ls`
List streams

`nats pub <subject> "<msg>"`
Publish a message

`nats consumer next <stream> <consumer>`
Pull stored messages

`nats request <subject> "<msg>"`
Ask and wait for a reply

`nats kv put <bucket> <key> <value>`
Store a value

Publisher nats pub → `subject` → **Server** routes by subject → `captured by` → **Stream** JetStream, stored → `consumer reads` → **Consumer** cursor + acks

CONCEPTS

Subjects are dot-separated tokens. * matches one token, > matches all remaining tokens.

Plain pub/sub is fire and forget: a subscriber that is offline misses the message. Streams (JetStream) keep it.

JetStream must be enabled on the server (nats-server -js), otherwise stream, consumer, kv and obj fail with no responders.

CONNECT AND CONTEXTS

`nats context add <name> --server <url> --select`
Save a server as a context

Credentials, TLS and other connection flags can be stored in the context too.

`nats context ls`
List saved contexts

`nats context select <name>`
Switch context

`nats --server <url> <command>`
Use a server without a context

Short form -s. Also read from NATS_URL.

`nats --creds <file> <command>`
Authenticate with a .creds file

`NATS_URL=<url> nats <command>`
Set the server by environment

PUBLISH AND SUBSCRIBE

`nats pub <subject> "<msg>"`
Publish a message

`nats sub <subject>`
Print messages as they arrive

`nats sub "<prefix>.>"`
Subscribe to a whole subject tree

PUBLISH AND SUBSCRIBE (cont.)

`nats sub <subject> --queue <group>`
Share messages across a queue group

`echo "<msg>" | nats pub <subject>`
Publish from stdin

`nats pub <subject> "<msg>" --jetstream`
Publish and wait for a stream ack

Short form -J.

REQUEST AND REPLY

`nats request <subject> "<msg>"`
Send a request, print the reply

`nats reply "<subject>" "<body>"`
Answer requests with a fixed body

`nats reply "<subject>" --command "<cmd>"`
Answer with a command's output

{{N}} in the command is replaced by the Nth subject token.

SERVER AND ACCOUNT

`nats account info`
Show account limits and usage

`nats-server -js`
Start a local server with JetStream

STREAMS

`nats stream add <name> --subjects "<s>.>" --defaults`
Create a stream

--defaults skips the prompts. A subject can belong to only one stream.

`nats stream ls`
List streams

STREAMS (cont.)

`nats stream info <name>`
Show config and state

`nats stream view <name>`
Browse stored messages

Needs an interactive terminal.

`nats stream rm <name>`
Delete a stream

-f skips the confirmation.

`nats stream purge <name>`
Remove all messages

--keep <n> or --subject <subject> narrows it.

CONSUMERS

`nats consumer add <stream> <consumer> --pull --defaults`
Create a durable pull consumer

The consumer name is positional; there is no --durable flag.

`nats consumer next <stream> <consumer> --count <n>`
Fetch and ack n messages

`nats consumer ls <stream>`
List a stream's consumers

`nats consumer info <stream> <consumer>`
Show pending and ack state

KEY-VALUE STORE

`nats kv add <bucket>`
Create a bucket

--history <n> keeps up to 64 revisions per key (default 1).

`nats kv put <bucket> <key> <value>`
Set a key

Key names reject some characters such as a colon; use - _ . / instead.

`nats kv get <bucket> <key>`
Read a key

Add --raw to print only the value.

`nats kv ls`
List buckets

KEY-VALUE STORE (cont.)

`nats kv watch <bucket>`
Stream changes live

Add a key to watch just that one.

`nats kv ls <bucket>`
List keys in a bucket

`nats kv del <bucket> <key>`
Delete a key (leaves a marker)

Without a key it deletes the whole bucket.

`nats kv history <bucket> <key>`
Show all revisions of a key

OBJECT STORE

`nats object get <bucket> <name>`
Download a file

--output <path> picks the target path.

`nats object put <bucket> <file>`
Upload a file

obj is an alias for object, so nats obj put works too.

BENCHMARK

`nats bench pub <subject> --msgs <n> --clients <n>`
Measure publish throughput

Add --no-progress for clean output.

RECIPES

Local JetStream playground

```
nats-server -js &
nats context add local --server
nats://127.0.0.1:4222 --select
nats stream add ORDERS --
subjects "orders.>" --defaults
nats pub orders.new "order 1"
```

Work through a pull consumer

```
nats consumer add ORDERS
shipping --pull --defaults
nats consumer next ORDERS
shipping --count 5
```

Watch a config key

```
nats kv add CONFIG --history 5
nats kv watch CONFIG &
nats kv put CONFIG feature.x on
```