



# CLI Extended

## Login options

Login with username and password

```
nats --user <user> --password <pwd> sub foo
```

Login with token

```
nats --token <token> sub foo
```

Login with nkey file - must not contain line breaks

```
nats --nkey <file> sub foo
```

Login with creds file - operator mode - generated with nsc or Synadia Control Plane

```
nats --creds <file> sub foo
```

## Storing options in local context

Save connection options to context

```
nats context add LOCAL --server nats://127.0.0.1:4222 --user <user> --password <pwd>
```

Select context

```
nats context select LOCAL
```

Quick validate all contexts (connect servers)

```
nats context validate
```

## Server Configuration

Obtain current server configuration and statistics (SYS)

```
nats server request variables
```

Reload server config file (SYS) - server id can be obtained with the command above

```
nats server config reload <server id>
```

## Server and Stream Events

Connect/Disconnect and authentication errors

```
nats events
```

Including all Jetstream advisories and events

```
nats events --all
```

## Graphs - ASCII Dashboard

A number of commands have a **graph** subcommand or **--graph** option.

```
nats server graph
```

```
nats stream graph <stream>
```

```
nats consumer graph <stream> <consumer>
```

Graph subject traffic. E.g. Jetstream consumer management.

```
nats sub
```

```
'$JS.API.CONSUMER.CREATE.>'
```

```
'$JS.API.CONSUMER.INFO.>'
```

```
'$JS.API.CONSUMER.DELETE.>'
```

```
--graph
```

## Extended Health Check

Health probe (SYS) - used by Kubernetes et. al.

```
nats server report health
```

- Ensure you see all servers and all report 200
- Repeat multiple times if needed

Check clusters and routes (SYS)

```
nats server ls <server count>
```

- Ensure all servers respond quickly. This means all expected servers replied.
- Ensure there are 0 slow consumers
- Ensure routes/GW are consistent
- Look for consistent versions
- Look for consistent CPU usage
- Look for consistent uptime
- Look for consistent memory usage. Is one server using RAM excessively?
- Check the number of connections.

Message traffic (SYS) - Run as system user for extended info

```
nats traffic
```

- **Raft:** Ensure votes are all to 0 (non zero may indicate instability)
- **Raft:** Ensure append is half of reply
- **General:** Ensure there are less than 1000 JS API Requests per second

Jetstream health - Note that not all servers may have Jetstream enabled (SYS)

```
nats server report jetstream <JS server count>
```

- Ensure all servers respond quickly. This means all expected servers are there.
- Ensure the number of consumers is as expected
- Make sure there is a stable meta leader
- Compare the number of servers in the RAFT Meta Group with Jetstream summary. Ensure all server are current and show in the Jetstream summary.

Exhaustive Stream and consumer status (SYS)  
**WARNING - These reports can be large and impact server performance**

```
nats server stream-check
```

```
nats server consumer-check
```

- Check for out of sync streams and consumers. Use **--unsynced** to filter.
- Check for expected replicas count, stream size, message count
- Check for offline peers

Gather and analyze information for auditing

```
nats audit gather
```

- Keep audit data for post mortem.

And do a quick analysis

```
nats audit analyze
```

- This is not an exhaustive report. Details may change over time.
- See **nats audit checks** for a list of checks.

Jetstream queueing/bottlenecks

Detecting bottlenecks in Jetstream replications and request routing

Reports RAFT cluster wide when used in system account. (SYS)

```
nats server request ipqueue
```

- Check **Routed JS API Requests** for pending API requests.
- Check RAFT groups for pending operations. Reports account, streams and consumer.

## Account management

Current account information, connections overview and statistics

```
nats account info
```

```
nats account report connections
```

```
nats account report statistics
```

Backup and restore all streams in an account.

```
nats account backup <directory>
```

```
nats account restore <directory>
```

**Purge/Delete all streams in an account. (SYS)**

```
nats server account purge <account>
```

## Advanced Jetstream management

Change the cluster leader of a stream or consumer. Useful when stream or consumer are unresponsive

```
nats stream cluster step-down <stream>
```

```
nats consumer cluster step-down <stream> <consumer>
```

Change to a preferred leader

```
nats stream cluster step-down <stream> --preferred <server>
```

Rebalance **all** stream leaders in the account - Use **with caution**

```
nats stream cluster balance
```

Scaling a stream down and up

```
nats stream edit <stream> replicas=3
```

Catching a stream create config json with **--json** or **--trace**

```
nats stream add <stream> --defaults --subject 'foo.>' --json
```

```
nats stream add <stream> --defaults --subject 'foo.>' --trace
```

Creating a stream from json config

```
nats stream add --config <config file>
```

## Advanced cluster management

Rebalance client connections between servers (SYS) - **Use with caution** and check the extended options with --help

`nats server cluster balance`

Change meta raft leader (SYS) for repairing a cluster

`nats server cluster step-down`

Removing a node from a cluster after a cluster scale down (SYS). NATS cluster will wait for nodes to come back. Streams will not repair their replicas until nodes are removed - **Use with caution**

`nats server cluster peer-remove <server>`

If the server name is unknown find and use the server id (SYS)

`nats server report jetstream`

`nats server cluster peer-remove <server id>`

## Quick Server status reports

**Note:** some reports are accessible from the SYS account only or show a different details level.

The server health probe as exposed by the HTTP interface and used by K8S et.al. to report server readiness.

`nats server report health`

Jetstream status, including meta raft group status.

`nats server report jetstream`

Quick memory and cpu usage report

`nats server report cpu`

`nats server report mem`

## Detailed server status, profiling and connection kick

Most information is also available on the **http debug (port 8222)** interface. Most of this information is auto collected by `nats audit gather`. Except for **connections** all commands require a system account.

List all connections.

`nats server request connections`

List connections and all subscriptions

`nats server request connections --subscriptions`

Kick a connection. (SYS)

`nats server request kick <conn id> <server id>`

Current server configuration. (SYS)

`nats server request variables`

Detailed status report per server of all streams and replicas. (SYS)

`nats server request jetstream`

All subscriptions system wide (SYS)

`nats server request subscriptions --detail`

.. other reports are **accounts**, **gateways**, **leafnodes**, **routes** and **raft**

Various GoLang runtime profiles. (SYS) -

Supports: allocs, heap, goroutine, mutex, threadcreate, block, cpu

`nats server request profile <profile>`