

NATS 2.2



- [Next Generation Streaming](#)
 - [Security and Simplified Account Management](#)
 - [CIDR Block Account Restrictions](#)
 - [Time-Based Account Restrictions](#)
 - [Default User Permissions](#)
 - [WebSockets](#)
 - [Native MQTT Support](#)
 - [Build Better Systems](#)
 - [Message Headers](#)
 - [Seamless Maintenance with Lam Duck Notifications](#)
 - [React Quicker with No-Responder Notifications](#)
 - [Subject Mapping and Traffic Shaping](#)
 - [Account Monitoring - More Meaningful Metrics](#)
-

NATS 2.2 is the largest feature release since version 2.0. The 2.2 release provides highly scalable, highly performant, secure and easy-to-use next generation streaming in the form of JetStream, allows remote access via websockets, has simplified NATS account management, native MQTT support, and further enables NATS toward our goal of securely democratizing streams and services for the hyperconnected world we live in.

Next Generation Streaming

JetStream is the next generation streaming platform for NATS, highly resilient, highly available, and easy to use. We've spent a long time listening to our community, learning from our experiences, looking at the needs of today, and thinking deeply about the needs of tomorrow. We built JetStream to address these needs.

JetStream:

- is easy to deploy and manage, built into the NATS server
- simplifies and accelerates development

- supports wildcard subjects
- supports at least once delivery and exactly once within a window
- is horizontally scalable at runtime with no interruptions
- persists data via streams and delivers or replays via consumers
- supports multiple patterns to consume data on the same stream
- supports push and pull modes when consuming messages
- is account aware
- allows for detailed granularity of security, by stream, by consumer, by function

Get started with [JetStream](#).

Security and Simplified Account Management

Account management just became much easier. This version of NATS has a built-in account management system, eliminating the need to set up an account manager when not using the memory account resolver. With automated default system account generation, and the ability to preload accounts, simply enable a set of servers in your deployment to be account resolvers or account resolver caches, and they will handle public account information provided to the NATS system through the NATS nsc tooling. Have enterprise-scale account management up and running in minutes.

CIDR Block Account Restrictions

By specifying a CIDR block restriction for a user, policy can be applied to limit connections from clients within a certain range or set of IP addresses. Use this as another layer of security atop user credentials to better secure your distributed system. Ensure your applications can only connect from within a specific cloud, enterprise, geographic location, virtual or physical network.

Time-Based Account Restrictions

Scoped to the user, you can now [specify a specific block of time](#) during the day when applications can connect. For example, permit certain users or applications to access the system during specified business hours, or protect business operations during the busiest parts of the day from batch driven back-office applications that could adversely impact the system when run at the wrong time.

Default User Permissions

Now you can specify [default user permissions](#) within an account. This significantly reduces efforts around policy, reduces chances for error in permissioning, and simplifies the provisioning of user credentials.

WebSockets

Connect mobile and web applications to any NATS server using [WebSockets](#). Built to more easily traverse firewalls and load balancers, NATS WebSocket support provides even more flexibility to NATS deployments and makes it easier to communicate to the edge and endpoints. This is currently supported in NATS server leaf nodes, nats.ts, nats.deno, and the nats.js clients.

Native MQTT Support

With the [Adaptive Edge architecture](#) and the ease with which NATS can extend a cloud deployment to the edge, it makes perfect sense to leverage existing investments in IoT deployments. It's expensive to update devices and large edge deployments. Our goal is to enable the hyperconnected world, so we added first-class support for [MQTT 3.1.1](#) directly into the NATS Server.

Seamlessly integrate existing IoT deployments using MQTT 3.1.1 with a cloud-native NATS deployment. Add a leaf node that is MQTT enabled and instantly send and receive messages to your MQTT applications and devices from a NATS deployment whether it be edge, single-cloud, multi-cloud, on-premise, or any combination thereof.

Build Better Systems

We've added a variety of features to allow you to build a more resilient, secure, and simply better system at scale.

Message Headers

We've added the ability to optionally use headers, following the HTTP semantics familiar to developers. Headers naturally apply overhead, which was why we resisted adding them for so long.

By creating new internal protocol messages transparent to developers, we maintain the extremely fast processing of simple NATS messages that we have always had while supporting headers for those who would like to leverage them. Adding headers to messages allows you to provide application-specific metadata, such as compression or encryption-related information, without touching the payload. We also provide some NATS specific headers for use in JetStream and other features.

Seamless Maintenance with Lame Duck Notifications

When taking down a server for maintenance, servers can be signaled to enter [Lame Duck Mode](#) where they do not accept new connections and evict existing connections over a period of time. Maintainer supported clients will notify applications that a server has entered this state and will be shutting down, allowing a client to smoothly transition to another server or cluster and better maintain business continuity during scheduled maintenance periods.

React Quicker with No-Responder Notifications

Why wait for timeouts when services aren't available? When a request is made to a service (request-reply) and the NATS Server knows there are no services available the server will short circuit the request. A "no-responders" protocol message will be sent back to the requesting client which will break from blocking API calls. This allows applications to immediately react which further enables building a highly responsive system at scale, even in the face of application failures and network partitions.

Subject Mapping and Traffic Shaping

Reduce risk when onboarding new services. Canary deployments, A/B testing, and transparent teeing of data streams are now fully supported in NATS. The NATS Server allows accounts to form subject mappings from one subject to another for both client inbound and service import invocations and allows weighted sets for the destinations. Map any percentage - 1 to 100 percent of your traffic - to other subjects, and change this at runtime with a server configuration reload. You can even artificially drop a percentage of traffic to introduce chaos testing into your system. See [Configuring Subject Mapping and Traffic Shaping](#) in NATS Server configuration for more details.

Account Monitoring - More Meaningful Metrics

NATS now allows for [fine-grained monitoring](#) to identify usage metrics tied to a particular account. Inspect messages and bytes sent or received and various connection statistics for a particular account. Accounts can represent anything - a group of applications, a team or organization, a geographic location, or even roles. If NATS is enabling your SaaS solution you could use NATS account scoped metrics to bill users.

📄 #3

📄 cocobook-sync 📄 2026-10-28 11:09:58 UTC

📄 cocobook-sync 📄 2026-10-29 01:42:32 UTC