

NATS 2.0



- [Rationale](#)
 - [Accounts](#)
 - [Service and Streams](#)
 - [System Accounts](#)
 - [Global Deployments](#)
 - [Self Healing](#)
 - [Superclusters](#)
 - [Disaster Recovery](#)
 - [Leaf Nodes](#)
 - [Decentralized Security](#)
 - [Operators, Accounts, and Users](#)
 - [Trust Chain](#)
-

NATS 2.0 was the largest feature release since the original code base for the server was released. NATS 2.0 was created to allow a new way of thinking about NATS as a shared utility, solving problems at scale through distributed security, multi-tenancy, larger networks, and secure sharing of data.

Rationale

NATS 2.0 was created to address problems in large scale distributed computing.

It is difficult at best to combine identity management end-to-end (or end-to-edge), with data sharing, while adhering to policy and compliance. Current distributed systems increase significantly in operational complexity as they scale upward. Problems arise around service discovery, connectivity, scaling for volume, and application onboarding and updates. Disaster recovery is difficult, especially as systems have evolved to operate in silos defined by technology rather than business needs. As complexity increases, systems become expensive to operate in terms of time and money. They become fragile making it difficult to deploy services and applications hindering innovation, increasing time to value and total cost of ownership.

We decided to:

- **Reduce total cost of ownership:** Users want reduced TCO for their distributed systems. This is addressed by an easy to use technology that can operate at global scale with simple configuration and a resilient and cloud-native architecture.
- **Decrease Time to Value:** As systems scale, *time to value* increases. Operations resist change due to risk in touching a complex and fragile system. Providing isolation contexts can help mitigate this.
- **Support manageable large scale deployments:** No data silos defined by software, instead easily managed through software to provide exactly what the business needs. We wanted to provide easy to configure disaster recovery.
- **Decentralize security:** Provide security supporting one technology end-to-end where organizations may self-manage making it easier to support a massive number of endpoints.

To achieve this, we added a number of new features that are transparent to existing clients with 100% backward client compatibility.

Accounts

Accounts are securely isolated communication contexts that allow multi-tenancy spanning a NATS deployment. Accounts allow users to bifurcate technology from business driven use cases, where data silos are created by design, not software limitations. When a client connects, it specifies an account or will default to authentication with a global account.

At least some services need to share data outside of their account. Data can be securely shared between accounts with secure services and streams. Only mutual agreement between account owners permit data flow, and the import account has complete control over its own subject space.

This means within an account, limitations may be set and subjects can be used without worry of collisions with other groups or organizations. Development groups choose any subjects without affecting the rest of the system, and open up accounts to export or import only the services and streams they need.

Accounts are easy, secure, and cost effective. There is one NATS deployment to manage, but organizations and development teams can self manage with more autonomy reducing time to value with faster, more agile development practices.

Service and Streams

Services and streams are mechanisms to share messages between accounts.

Think of a service as an RPC endpoint into an account. Behind that account there might be many microservices working in concert to handle requests, but from outside the account there is simply one subject exposed.

Service definitions share an endpoint:

- Export a service to allow other accounts to import
- Import a service to allow requests to be sent securely and seamlessly to another account

Use cases include most applications - anything that accepts a request and returns a response.

Stream definitions allow continuous data flow between accounts:

- Export a stream to allow egress
- Import a stream to allow ingress

Use cases include Observability, Metrics, and Data analytics. Any application or endpoint reading a stream of data.

Note that services and streams operate with **zero** client configuration or API changes. Services may even move between accounts, entirely transparent to end clients.

System Accounts

The system account publishes system messages under established subject patterns. These are internal NATS system messages that may be useful to operators.

Server initiated events and data include:

- Client connection events
- Account connection status
- Authentication errors
- Leaf node connection events
- Server stats summary

Tools and clients with proper privileges can request:

- Service statistics
- Server discovery and metrics

Account servers will also publish messages when an account changes.

With this information and system metadata you can build useful monitoring and anomaly detection tools.

Global Deployments

NATS 2.0 supports global deployments, allowing for global topologies that optimize for WANs while extend to the edge or devices.

Self Healing

While self healing features have been part of NATS 1.X releases, we ensured they continue to work in global deployments. These include:

- Client and server connections automatically reconnect
- Auto-Discovery where servers exchange server topology changes with each other and with clients, in real time with zero configuration changes and zero downtime while being entirely transparent to clients. Clients can failover to servers they were not originally configured with.
- NATS server clusters dynamically adjust to new or removed servers allowing for seamless rolling upgrades and scaling up or down.

Superclusters

Conceptually, superclusters are clusters of NATS clusters. Create superclusters to deploy a truly global NATS network. Superclusters use a novel spline based technology with a unique approach to topology, keeping one hop semantics and optimizing WAN traffic through optimistic sends with interest graph pruning. Superclusters provide transparent, intelligent support for geo-distributed queue subscribers.

Disaster Recovery

Superclusters inherently support disaster recovery. With geo-distributed queue subscribers, local clients are preferred, then an RTT is used to find the lowest latency NATS cluster containing a matching queue subscriber in the supercluster.

What does this mean?

Let's say you have a set of load balanced services in US East Coast (US-EAST), another set in the EU (EU-WEST), and a supercluster consisting of a NATS cluster in US-EAST connected to a NATS cluster in EU-WEST. Clients in the US would connect to a US-EAST, and services connected to that cluster would service those clients. Clients in Europe would automatically use services connected to EU-WEST. If the services in US-EAST disconnect, clients in US-EAST will begin using services in EU-WEST.

Once the Eastern US services have reconnected to US-EAST, those services will immediately begin servicing the Eastern US clients since they're local to the NATS cluster. This is automatic and entirely transparent to the client. There is no extra configuration in NATS servers.

This is **zero configuration disaster recovery**.

Leaf Nodes

Leaf nodes are NATS servers running in a special configuration, allowing hub and spoke topologies to extend superclusters.

Leaf nodes can also bridge separate security domains. e.g. IoT, mobile, web. They are ideal for edge computing, IoT hubs, or data centers that need to be connected to a global NATS deployment. Local applications that communicate using the loopback interface with physical VM or Container security can leverage leaf nodes as well.

Leaf nodes:

- Transparently and securely bind to a remote NATS account
- Securely bridge specific local data to a wider NATS deployment
- Are 100% transparent to clients which remain simple, lightweight, and easy to develop
- Allow for a local security scheme while using new NATS security features globally
- Can create a DMZ between a local NATS deployment and external NATS cluster or supercluster.

Decentralized Security

Operators, Accounts, and Users

NATS 2.0 Security consists of defining Operators, Accounts, and Users within a NATS deployment.

- An **Operator** provides the root of trust for the system, may represent a company or enterprise
 - Creates **Accounts** for account administrators. An account represents an organization, business unit, or service offering with a secure context within the NATS deployment, for example an IT system monitoring group, a set of microservices, or a regional IoT deployment. Account creation would likely be managed by a central group.
- **Accounts** define limits and may securely expose services and streams.
 - Account managers create **Users** with permissions
- **Users** have specific credentials and permissions.

Trust Chain

PKI (NKeys encoded [Ed25519](#)) and signed JWTs create a hierarchy of Operators, Accounts, and Users creating a scalable and flexible distributed security mechanism.

- **Operators** are represented by a self signed JWT and is the only thing that is required to be configured in the server. This JWT is usually signed by a master key that is kept offline. The JWT will contain valid signing keys that can be revoked with the master updating this JWT.
 - Operators will sign **Account** JWTs with various signing keys.
 - **Accounts** sign **User** JWTs, again with various signing keys.
- Clients or leaf nodes present **User** credentials and a signed nonce when connecting.
 - The server uses resolvers to obtain JWTs and verify the client trust chain.

This allows for rapid change of permissions, authentication and limits, to a secure multi-tenant NATS system.

```
[[[ #1
[] cocobook-sync [[[ 2026-10-28 11:09:59 UTC
[] cocobook-sync [[[ 2026-10-28 11:09:59 UTC
```