Security

NATS has a lot of security features:

- Connections can be *encrypted* with TLS
- Client connections can be *authenticated* in many ways:
  - Token Authentication
  - Username/Password credentials
  - TLS Certificate
  - NKEY with Challenge
  - Decentralized JWT Authentication/Authorization
  - You can also integrate NATS with your existing authentication/authorization system or create your own custom authentication using the Auth callout
- Authenticated clients are identified as users and have a set of *authorizations*

You can use accounts for multi-tenancy: each account has its own independent 'subject namespace' and you control the import/export of both streams of messages and services between accounts, and any number of users that client applications can be authenticated as. The subjects or subject wildcards that a user is allowed to publish and/or subscribe to can be controlled either through server configuration or as part of signed JWTs.

JWT authentication/authorization administration is decentralized because each account private key holder can manage their users and their authorizations on their own, without the need for any configuration change on the NATS servers by minting their own JWTs and distributing them to the users. There is no need for the NATS server to ever store any user private keys as they only need to validate the signature chain of trust contained in the user JWT presented by the client application to validate that they have the proper public key for that user.

The JetStream persistence layer of NATS also provides encryption at rest.

□□□ #3
□ cocobook-sync □□□ 2026-10-28 11:10:39 UTC
□ cocobook-sync □□□ 2026-10-29 01:42:34 UTC