NATS Cheatsheet

Server (NATS Server or NATS Core)

- **Single executable** that manages message routing between clients.
- **Self clustering**, no zookeeper etc.
- Written in **Go** for simplicity, performance and scalability.
- Supports **publish/subscribe, requests/reply and point-to-point (queueing) messaging patterns.**

Client/Server architecture

- Applications use NATS client libraries to connect to the server.
- Can **publish messages, subscribe to subjects**, or **both**.
- Supports multiple programming languages, including Go, Java, Python and JavaScript.

Subject addressing

- Subjects are topic/channels for message exchange.
- Support **hierarchical namespaces** and **wildcard subscriptions** for flexible message routing.

Message

- Unit of data transmission. Everything is a message in NATS.
- Composed of:
  - **Subject:** The channel to which the Message is published.
  - **Payload:** The binary content of the Message.
  - **Headers:** Map with metadata.
  - **Reply-Subject:** Reply channel for RPC style communication

Queue Groups

- Used for **load balancing** among multiple subscribers.
- **Each message is delivered to only one subscriber** in the queue group.

JetStream (Persistence)

- **Built-in persistence layer in NATS**. Replicated and resilient.
- Adds advanced capabilities like:
  - **Streaming:** Store and replay Messages.
  - **Queues:** Manages message delivery to multiple consumers.
  - **Delivery Guarantees:** Supports at least once and exactly-once delivery.
  - **Flow Control:** Decouples message production and consumption.
  - **Key/Value Store:** Provides a simple distributed key/value storage system.
  - **Per Message Acks:** Ensures message processing and redelivery.

Consistent Replication

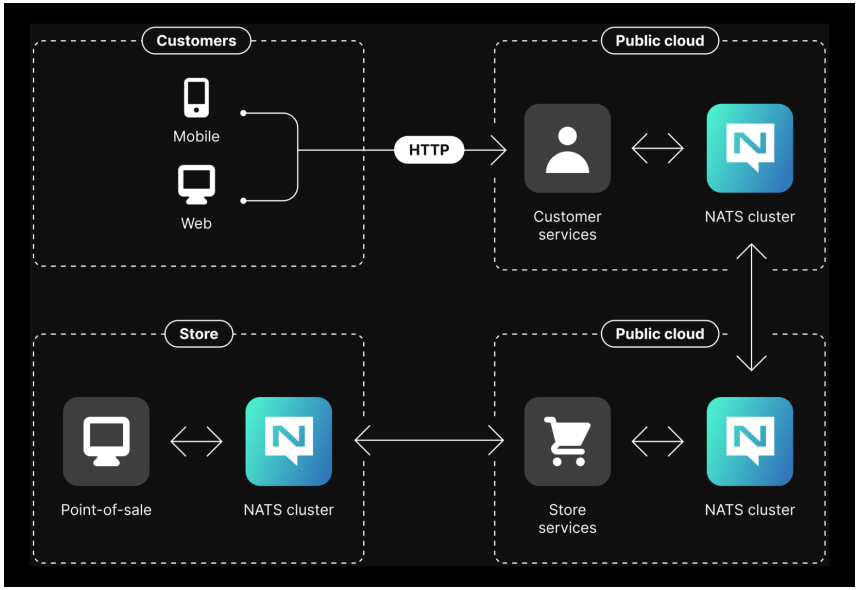
- **Clustered NATS servers** replicate messages for **high availability**.
- JetStream enables **persistent storage with fault tolerance.**

Wildcard Subscriptions

- **Single-level (*):** Matches one token in the subject hierarchy.
- **Multi-level (>):** Matches one or more tokens at the end of the subject hierarchy.
- Used for **subscription, filtering, security, and transformation.**

Key Features

- **Lightweight:** Minimal resource consumption, ideal for microservices and edge computing.
- **Low Latency:** Optimized for high performance low-latency messaging.
- **Persistence:** JetStream enables reliable message storage and delivery.
- **Scalability:** Supports horizontal scaling via clustering for high availability.
- **Security:** Offers **TLS encryption, token-based authentication** and **subject based permissions.**



Use Cases

- **Microservices:** Communication: Lightweight and fast inter-service messaging.
- **IoT and Edge Computing:** Low latency messaging for distributed Systems.
- **Real-time Analytics:** Stream processing and event-driven architectures.
- **Event Streaming:** Reliable message delivery with JetStream.
- Financial Services: High-Performance messaging for real-time trading systems.

A Brief History of NATS

- **2010:** Created by Derek Collison as a lightweight messaging system for Cloud Foundry.
- **2012:** Open-sourced under the Apache 2.0 license.
- **2017:** Founded Synadia to drive enterprise support, and a long-term roadmap. Synadia also manages the nats.io ecosystem, including the client libraries and NATS Connect.
- **2021:** Introduction of NATS JetStream for message persistence and streaming.
- **2025 and Beyond:** Continued adoption in edge computing, IoT, and microservices. Architectures, with ongoing enhancements to Jetstream and clustering capabilities.
- **Open-Source Community** — NATS is developed openly under the **nats-io organization** on GitHub, with contributions from companies like **Siemens, VMware, and Cisco**, as well as independent developers worldwide.
- **Widespread Deployment:** NATS is trusted by leading enterprises and platforms, including Mastercard, VMware, Cloud Foundry, Siemens, Schaeffler, Walmart, Alibaba and GE.