A New Way of Thinking
NATS 2.0 and Connectivity

Derek Collison / @derekcollison

Founder and CEO at Synadia

- Creator of NATS
- Founder and Former CEO at Apcera
- CTO, Chief Architect at VMware
- Architected CloudFoundry
- Technical Director at Google
- SVP and Chief Architect at TIBCO

Think Different

The Beginning of the **Connected Economy**

1. Have moved on from the compute economy
2. Many say we are in the data economy
3. How are data and services securely accessed?
4. That is where the largest opportunity lives
5. We are moving into the connected economy
6. For all the world's digital systems, services and devices

Distributed Systems History

1. **We** have created this problem, this complexity
2. There used to be just **one** computer
3. Wanted to do more, bought a **faster** computer
4. Systems are now global
5. Systems are hundred and thousands of components
6. Systems are **complex**, **expensive**, and **slow...**
7. And they are getting worse

Connectivity

This is the **KEY**

Connectivity - What we Need

1. Needs to be a ubiquitous technology
2. Needs to be a true utility technology, not a silo
3. Needs to be secure and isolated by default
4. Securely Shareable
5. Scales effortlessly
6. Multiple patterns, Services and Data and Event Streams
7. Self healing, agile, Multi-Cloud, On-Prem, Edge, and IoT

Connectivity

1. Location transparent
2. Access to services, event streams, data (K/V, RDB)
3. Not confined to deployment platforms, clouds, etc
4. Observable from anywhere
5. Open Source Technology, Open Ecosystem
6. Decentralized and Federated Global SaaS

Why is messaging important?

Developing and deploying applications that communicate in distributed systems can be **complex**, **difficult** and **costly**. A communication infrastructure should provide features to make this **easier** including:

- Multiple messaging patterns bundled into one technology
  - Streams and Services
- Location transparency
- Transparent scalability and disaster recovery
- Decoupled producers and consumers
- Synchronous & Asynchronous communications
- Extensible and Open by default

What is NATS?

NATS is a simple and secure, production-proven messaging technology for modern distributed systems.

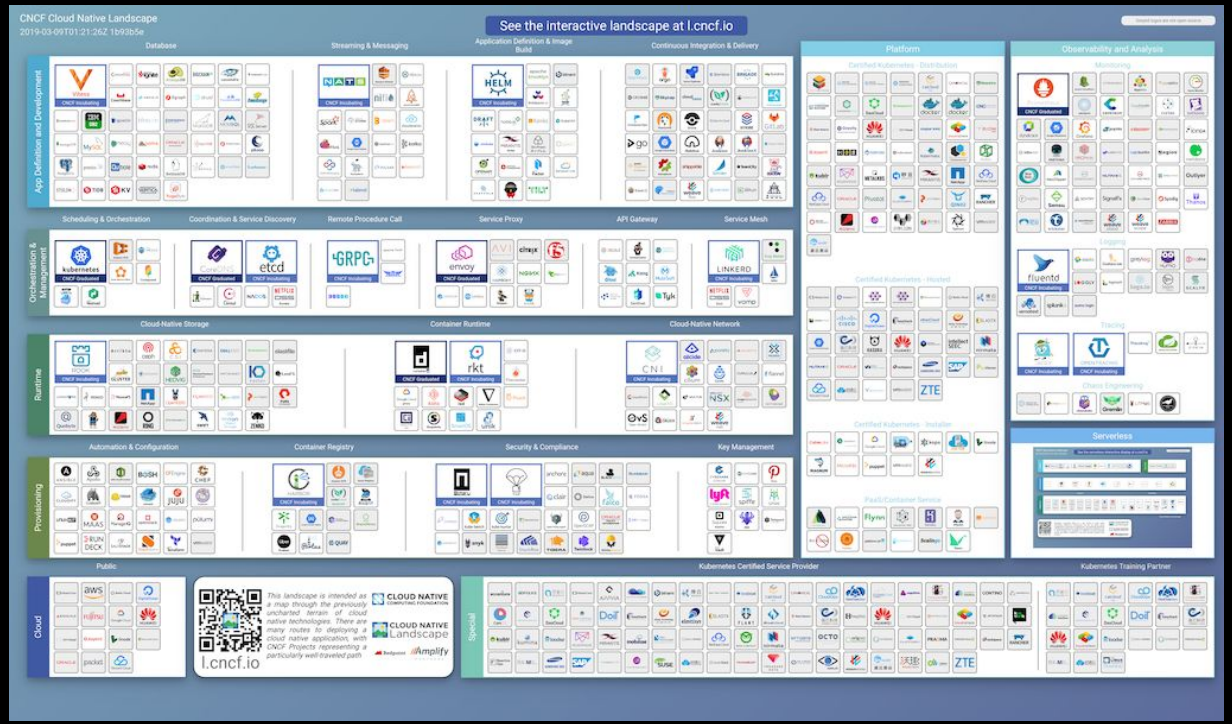
- **Scalable** Services and Streams
- **Easy** to use for developers and operators
- **Highly** Resilient
- **Highly** Secure
- **Extremely** lightweight and performant
- Client support for over 32+ different programming languages
- A CNCF project with Kubernetes and Prometheus integrations
- Originally built to power CloudFoundry. Mature ~10yrs old.
- Applicable in the Cloud, On-Premise, Edge and IoT.

What is NATS?

- **Subject-Based** routing, not IP
- **Message** based, not packet or byte streams
- **Scalable** Services and **Streams,** same technology
- Natively does **N:M** with queue delivery. Not just 1:1
- **Secure** and **Isolated,** decentralized by design
- **Secure Sharing:** Exports and Imports of Services and Streams
- **Self Healing** and **Observable**
- Capable of being a federated global utility - a **Dial Tone**


NATS is an always available
dial tone to connect everything

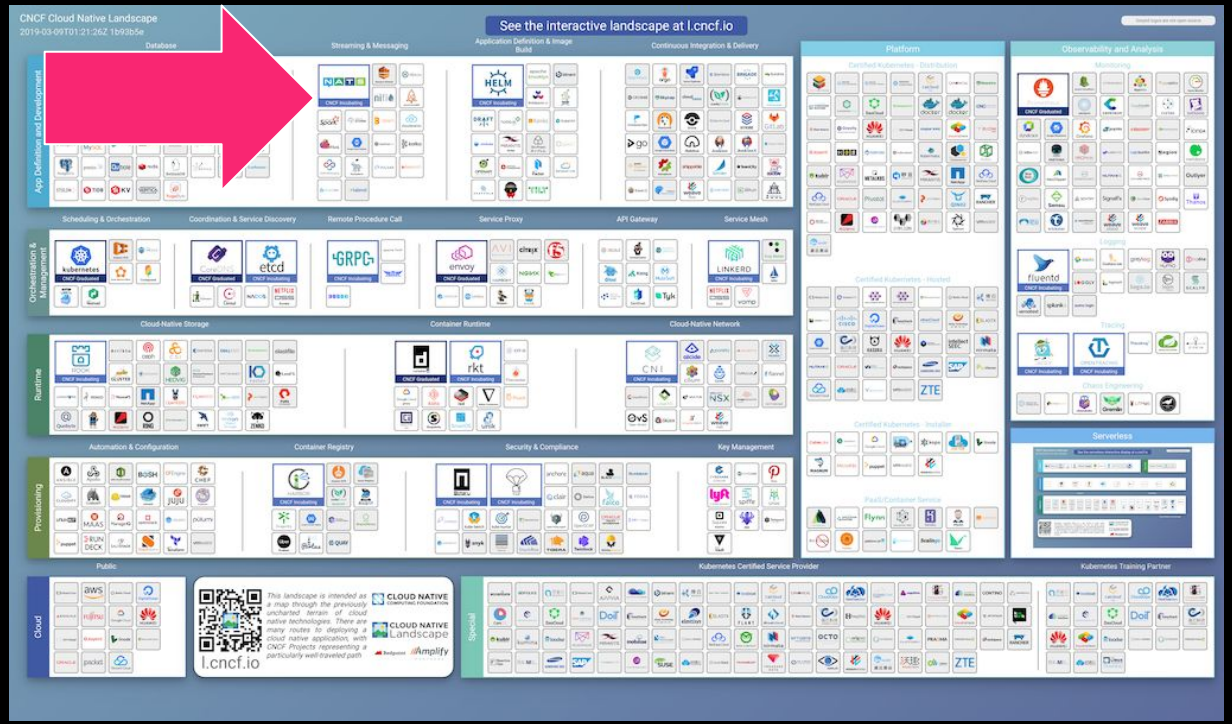
CNCF Landscape



Joined CNCF as an incubation project in 2018

https://landscape.cncf.io

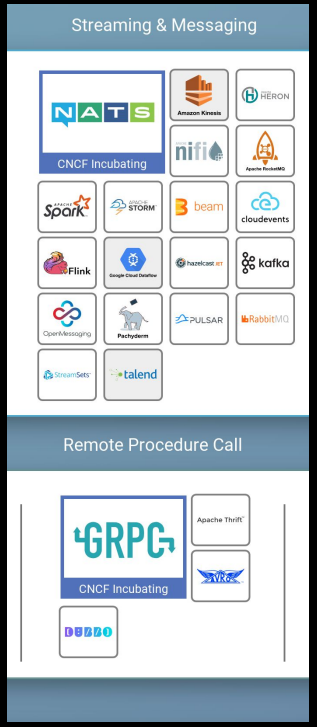
CNCF Landscape



Joined CNCF as an incubation project in 2018

https://landscape.cncf.io

CNCF Landscape



8. STREAMING & MESSAGING

When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues.

GRPC CNCF Incubating

NATS CNCF Incubating

https://landscape.cncf.io

Growing Community: NATS End Users

NATS 2.0

NATS 2.0 Release

https://nats-io.github.io/docs/whats_new/whats_new_20.html

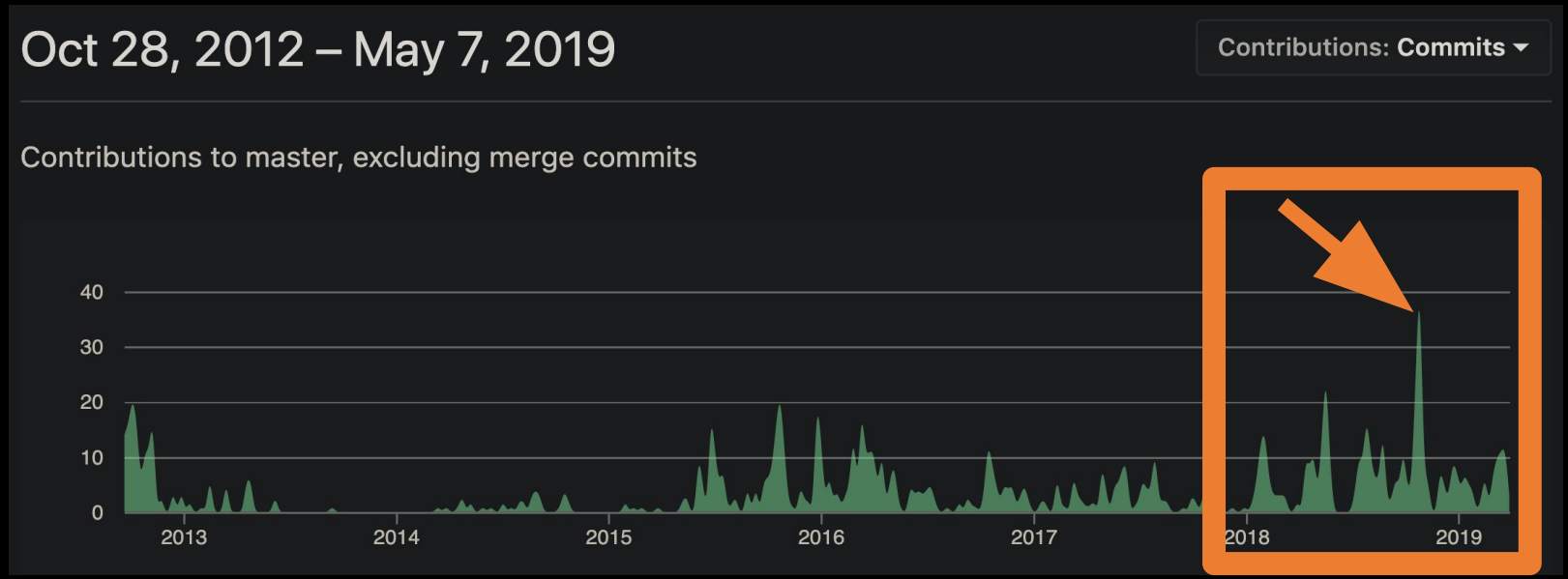
- NATS 2.0 is the largest feature release since the original code.
  - Backward Client compatibility
- Created to allow a new way of thinking about NATS as a secure shared utility, globally through NGS, within an enterprise, or both.
- NATS now solves problems at scale through distributed security, multi-tenancy, larger networks, and secure sharing of data.

NATS 2.0 allows **new ways** of architecting distributed systems, service meshes and event/data streams.

NATS v2

Released on Monday (June 10th, 2019)

Biggest release of the project since it started.



Secure Multi-Tenancy (Accounts)

Accounts - Containers for Messaging

- Accounts are securely isolated communication contexts that allow multi-tenancy.
- Bifurcate technology from business driven use cases
  - Data silos are created by design, not software limitations
- Securely share data between accounts
  - Secure Streams and Services
  - Mutual agreement permits data flow across account boundaries
- Easy, Secure and Cost Effective
  - One NATS deployment for operators to manage
  - Decentralized - Organizations can self-manage

Services and Streams

Service definitions are a secure RPC endpoint

- Export a service to allow other accounts to import
- Import a service to allow requests to be sent and securely and seamlessly to another account
- Use cases include most business applications, certificate generation service, secure vault, location aware services

Stream definitions allow data flow between accounts

- Export a stream to allow egress
- Import a stream to allow ingress
- Use cases include observability, metrics, data analytics, etc.

...All with zero client configuration or API changes

System Account

The system account publishes system messages under established subject patterns.

Server initiated events:

- `$SYS.ACCOUNT.<id>.CONNECT` (client connects)
- `$SYS.ACCOUNT.<id>.DISCONNECT` (client disconnects)
- `$SYS.SERVER.ACCOUNT.<id>.CONNS` (account connections status)
- `$SYS.SERVER.<id>.CLIENT.AUTH.ERR` (authentication error)
- `$SYS.ACCOUNT.<id>.LEAFNODE.CONNECT` (leaf node connects)
- `$SYS.ACCOUNT.<id>.LEAFNODE.DISCONNECT` (leaf node disconnects)
- `$SYS.SERVER.<id>.STATSZ` (stats summary)

System Account - Continued

In addition other tools with system account privileges can initiate requests:

- `$SYS.REQ.SERVER.<id>.STATSZ (request server stat summary)`
- `$SYS.REQ.SERVER.PING (discover all servers and metrics)`

Secure account servers publish messages when a claim is updated:

- `$SYS.ACCOUNT.<id>.CLAIMS.UPDATE`

With these few messages you can build useful monitoring and anomaly detection tools.

Global Deployments

Low Touch Operations - NATS Self-Healing

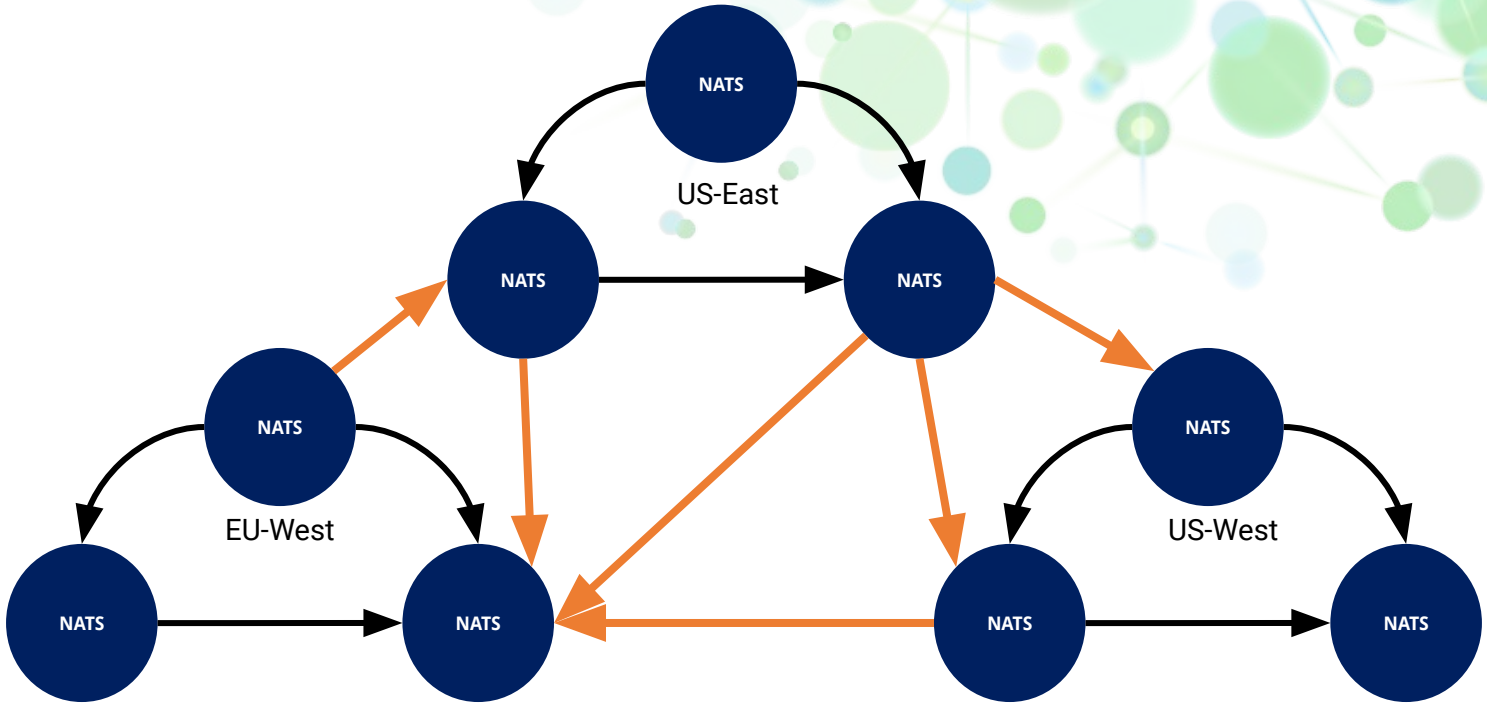
- Client and server connections reconnect automatically
- Auto-Discovery automatically exchanges server topology
  - Server ⇆ Server
  - Server ⟶ Client
  - Zero configuration changes, **zero downtime**
  - Dynamic and entirely transparent to applications
- Clients can failover to new servers that weren't originally configured
- NATS server clusters dynamically adjust to new or removed servers
  - Used for **rolling upgrades** and **scaling** (up or down).
- NATS is **deployment**, **cloud** and **geo** agnostic.

Superclusters

Create clusters of clusters to create a truly global NATS network

- Novel spline based technology
- Optimistic sends with interest graph pruning
- Transparent, intelligent support for geo-distributed queue subscribers

Clusters of Clusters

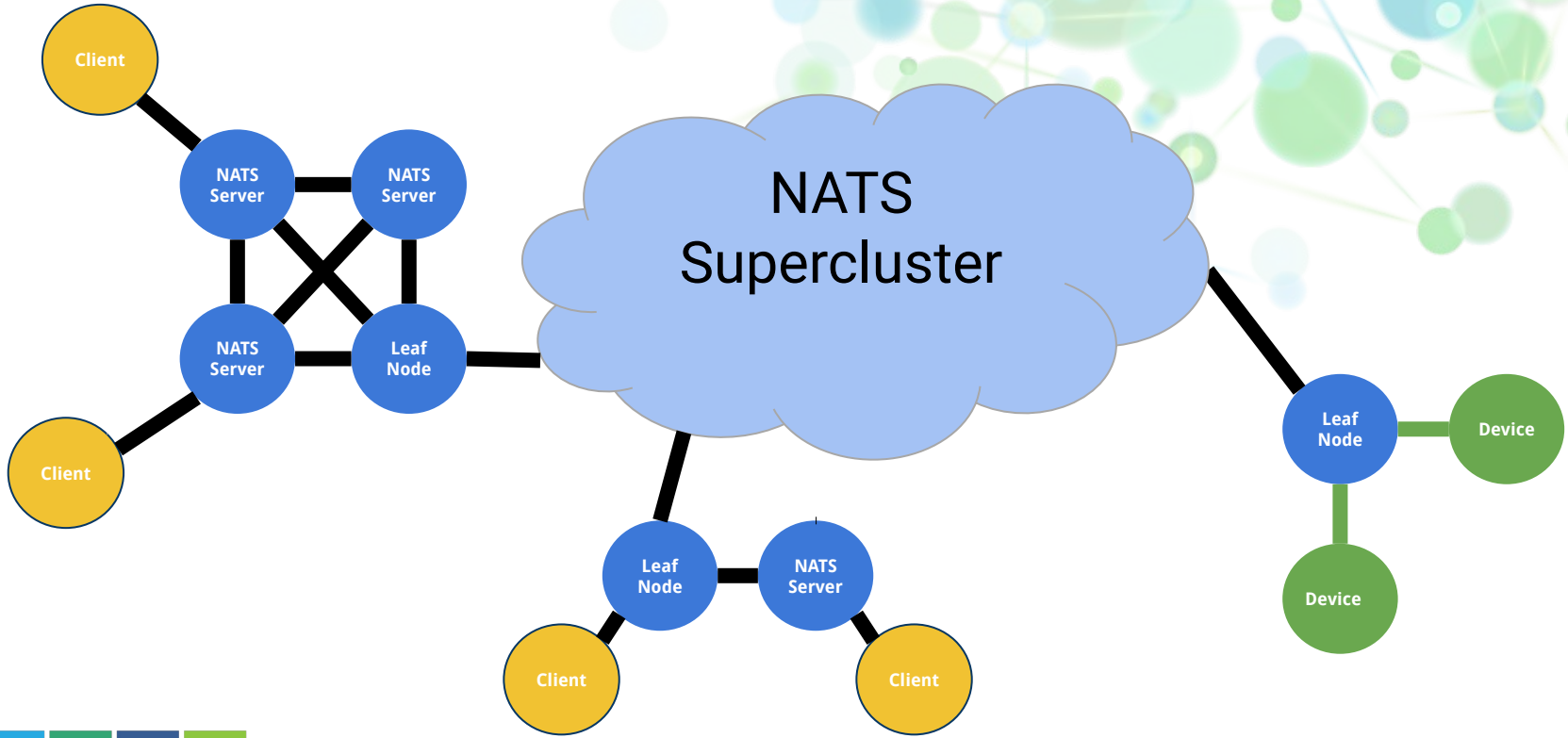


Leaf Nodes

Leaf Nodes

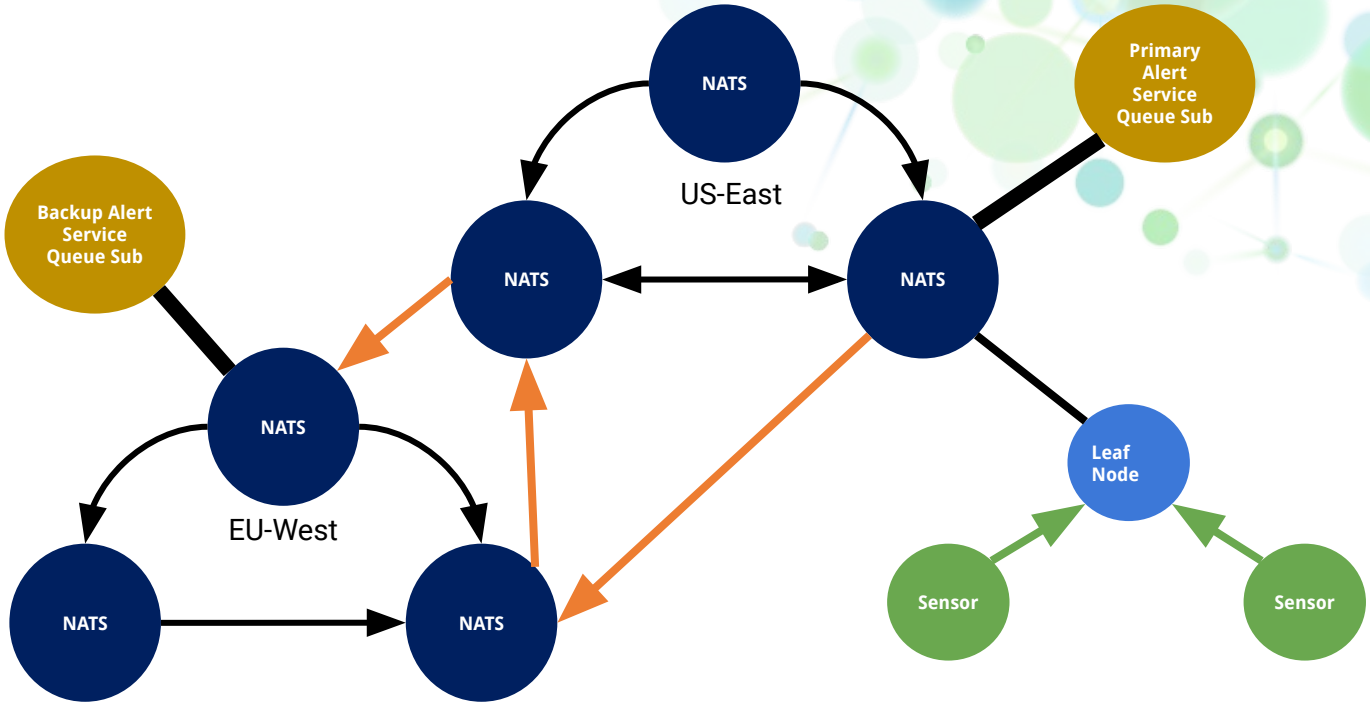
- Leaf nodes allow hub and spoke topologies to extend superclusters.
- Leaf nodes allow you to bridge separate security domains. e.g. IoT, mobile, POS, Web.
- Ideal for edge computing, IoT hubs, or data centers that need to be connected to a global, regional, or national NATS deployment.
- Transparently bridge on-premise and cloud or edge deployments.
- Have your cake and eat it too!

Leaf Nodes

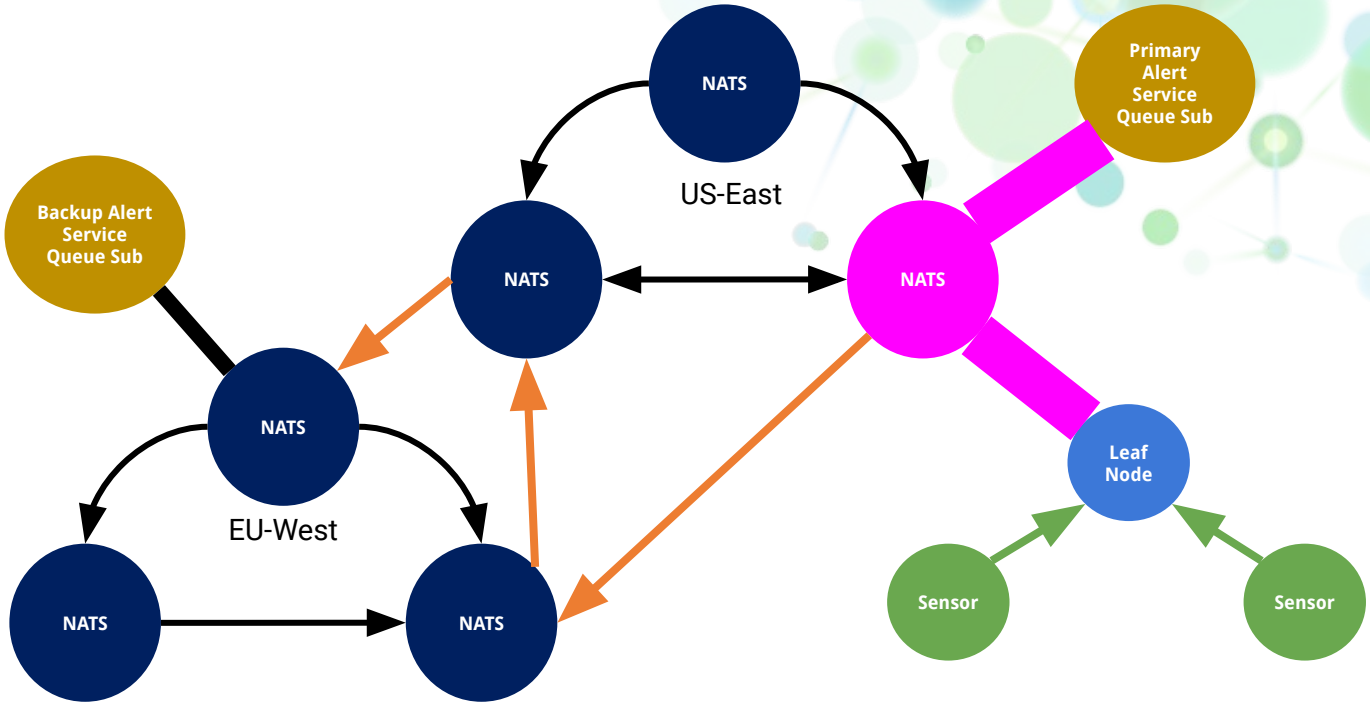

Client
NATS Server
NATS Server
NATS Server
Leaf Node
Client
NATS Supercluster
Leaf Node
NATS Server
Client
Client
Leaf Node
Device
Device

Disaster Recovery with Geo-Aware and Scalable and Observable Services

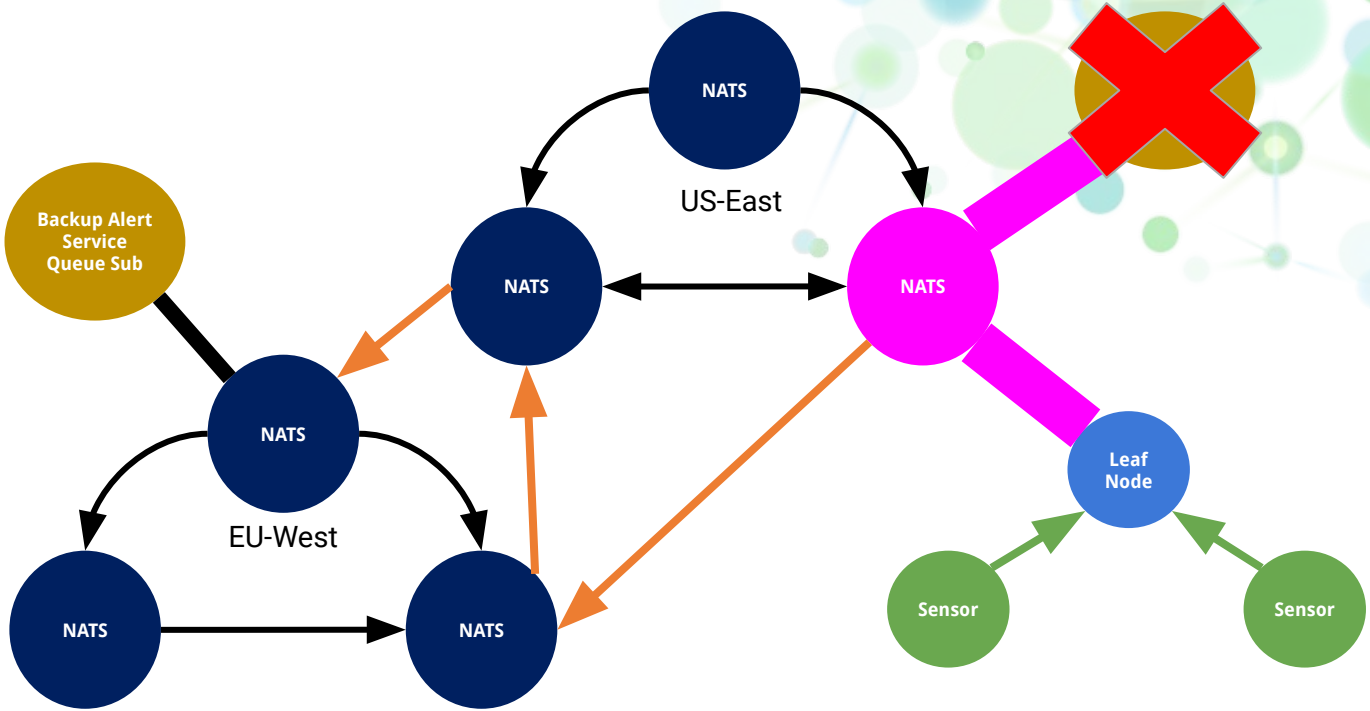
Geo-Aware Services



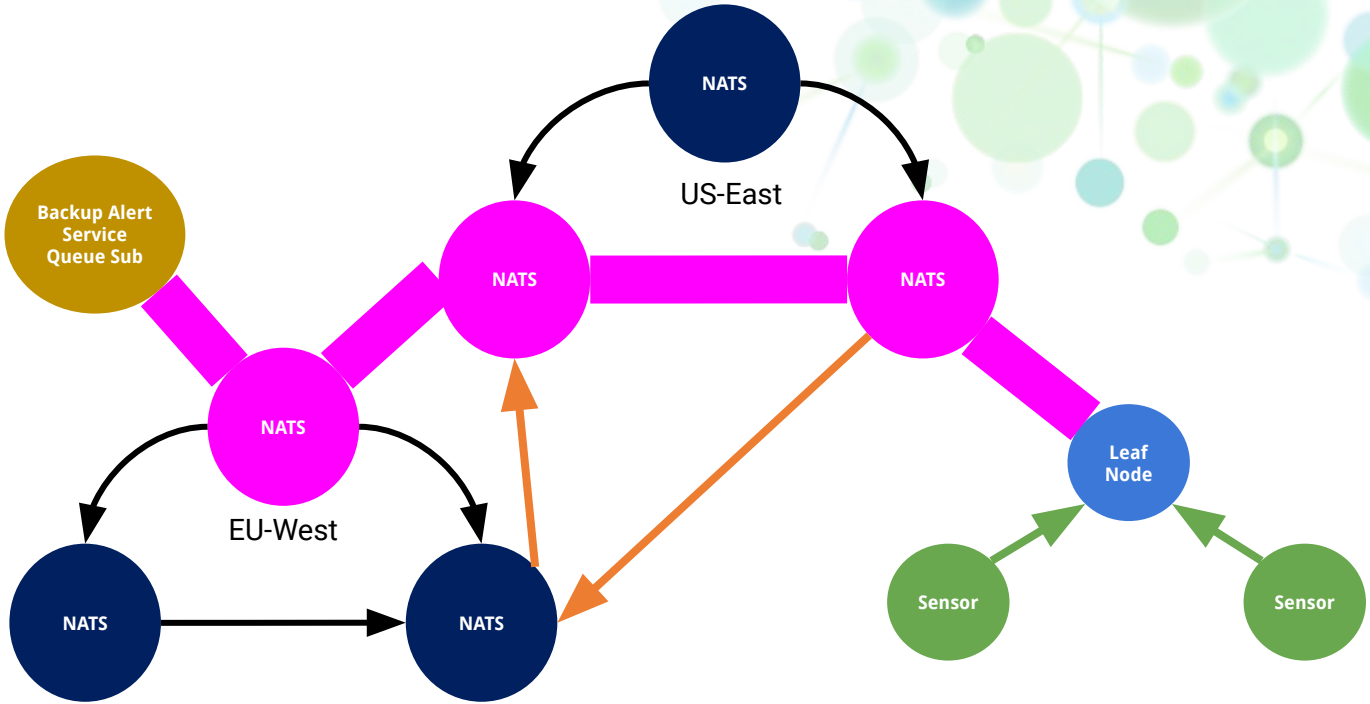
Geo-Aware Services - Flow



Geo-Aware Services - Failure



Geo-Aware Services - Remote Routing



Decentralized Security

NATS Security Today

- Full TLS Support: CA certificates, bidirectional support, default to most secure ciphers.
  - Support for DN or SAN in certificates for NATS user identity
- Support for standard user/password auth
- Permissions restrict who can send and receive on what subjects
- Change these through configuration reload at runtime with **zero downtime**.
- Operator Mode - NATS >= 2.0

Operator Mode - Decentralized

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

- **Operator**: Root of trust for the system, e.g. An Enterprise operator.
  - Create **Accounts** for account administrators. An account represents an organization with a secure context within the NATS deployment, for example a VAS system, an IT system monitoring group, a set of microservices, etc. Account creation would likely be managed by a central group.
- **Accounts** define limits and may securely export and import services and streams
  - Account managers create and manage **Users** with permissions
- **Users** have specific credentials and permissions.

Decentralized Security Management

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.
  - 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, limits, etc. to a secure multi-tenant NATS system.

Operator JWT

Operator Details	
Name	Synadia
Operator ID	OCDFJINL65FTXNLUB3GODXY3MTCOUFJWC23HONKXNGLZ4SCQRBSFWECQ
Issuer ID	OCDFJINL65FTXNLUB3GODXY3MTCOUFJWC23HONKXNGLZ4SCQRBSFWECQ
Issued	2019-05-13 16:17:29 UTC
Expires	

Account JWT

Account Details	
Name	Microservices-1
Account ID	AAHC5D6GVMRI753MOVEIEV2LVR3C7GUCYLAOHQH5DL5V7M6CXSYGWZRK
Issuer ID	OCDFJINL65FTXNLUB3GODXY3MTCOUFJWC23HONKXNGLZ4SCQRBSFWECQ
Issued	2019-05-14 01:21:19 UTC
Expires	
Max Connections	Unlimited
Max Leaf Node Connections	Unlimited
Max Data	Unlimited
Max Exports	Unlimited
Max Imports	Unlimited
Max Msg Payload	Unlimited
Max Subscriptions	Unlimited
Exports Allows Wildcards	True
Exports	None

Imports						
Name	Type	Remote	Local	Expires	From Account	Public
SvcControl	Service	svc.control	svc.control.1		ACOAYG5E4EWOE3VJ26RTQFD3QNOL2JA5EMEWHVM5RDKHYQAOEAIO35PS	Yes

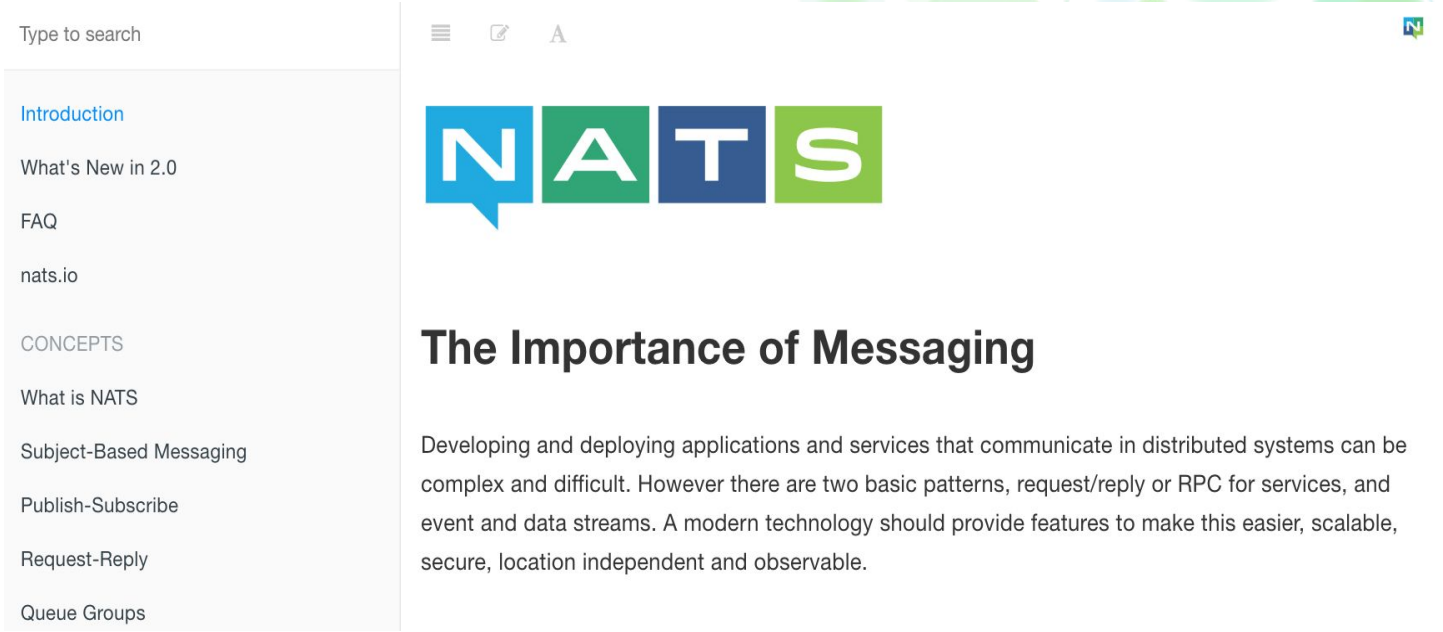
User JWT

User	
Name	ProvisioningService
User ID	UCLEREN55TYVHT3UC25YEC6XJR3TYZB35FJOGTFOACA5NRDRX3LTULUR
Issuer ID	AAHC5D6GVMRI753MOVEIEV2LVR3C7GUCYLAOHQH5DL5V7M6CXSYGWZRK
Issued	2019-05-13 16:21:33 UTC
Expires	
Max Messages	Unlimited
Max Msg Payload	Unlimited
Network Src	Any
Time	Any

Documentation

All New Documentation

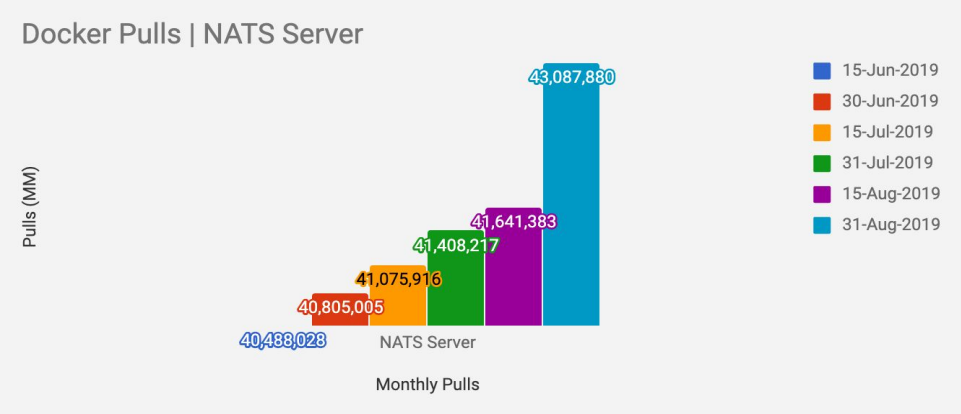
- https://nats-io.github.io/docs/

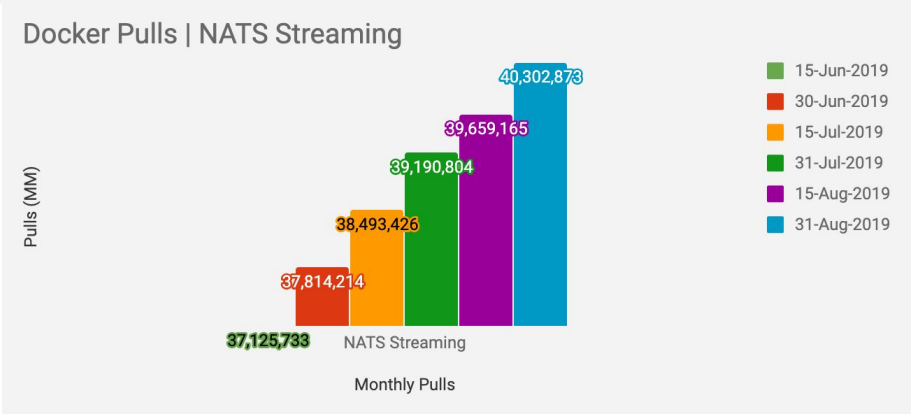


NATS

Ecosystem

NATS on Docker ~85M Combined Downloads!

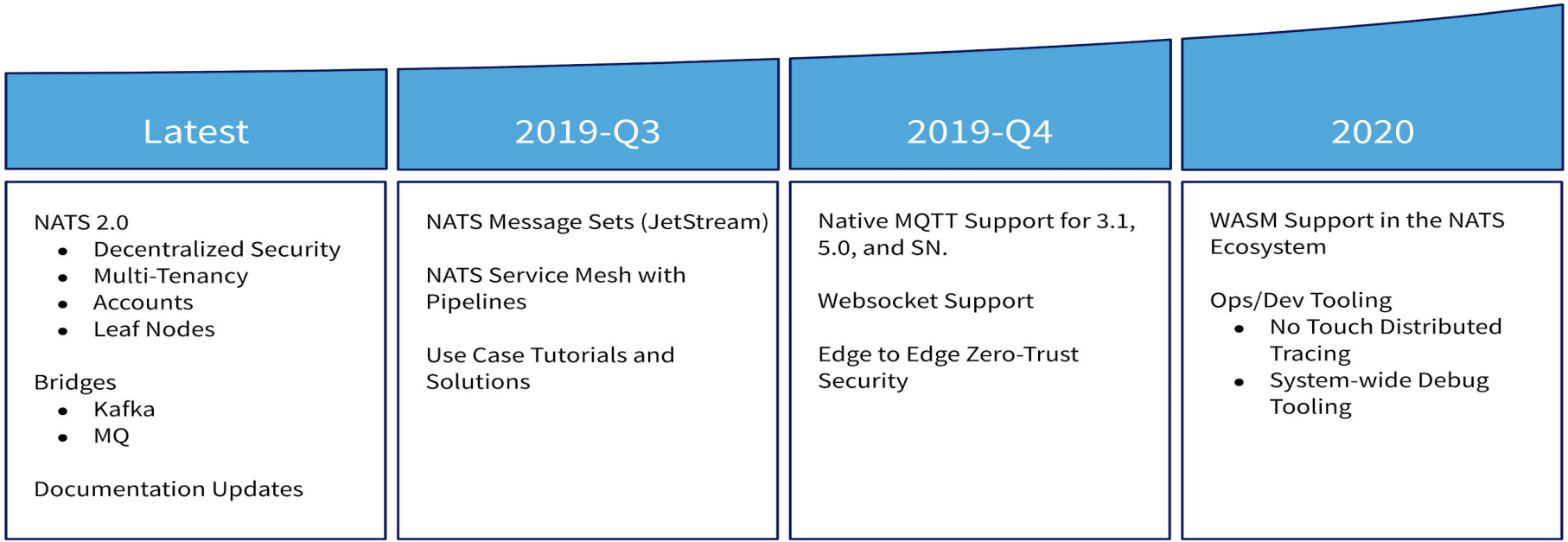




Roadmap

Roadmap 2019-2020

Our roadmap represents coming features lined up for the future of NATS. We are excited to bring these advances to the NATS community and look forward to your valuable input. Please contact us at info@nats.io or join our Slack channel with any questions, comments, or requests.



Think Different

Stop making things more complicated

Start making things Simpler!

We need our own Tesla Moment

Thanks!

github.com/nats-io / @nats_io

https://nats.io