Nothing but NATS

Going Beyond Cloud-Native

KubeCon North America 2024

Kevin Hoffman

- **Director of Cloud Engineering** at Synadia
- Creator of **Nex** and **wasmCloud**
- Co-founder of **Cosmonic**
- Author of **“Practical Event Sourcing”** and **“Programming WebAssembly with Rust”**
- Distributed Systems **nerd**

Byron Ruth

- **VP of Product & Engineering** at Synadia
- A NATS project **maintainer**
- **Release team** for the NATS server
- Previously 14 years in **pediatric biomedical research**

NATS 101

Current **mainstream tech** used for building and operating distributed systems is **limiting** and overly **complex** for engineering **teams**.

A “modern” OSS stack

What components do we need?

- **gRPC** - 1:1 request-reply and streaming
- **RabbitMQ** - M:N messaging and queues
- **Kafka** - scalable data streaming
- **Redis** - key-value
- **Minio** - object storage
- **Envoy** - proxy, routing, load balancing
- **Istio** - security, policy, observability

What do we have

Complexity stems from inherent limitations.

- ✗ Cumbersome discovery with HTTP/DNS
- ✗ Limited 1:1 communication patterns
- ✗ Perimeter-based security models
- ✗ Routing via gateways and load balancers
- ✗ Centralized and location dependent
- ✗ Architectural and operational complexity
- ✗ Multiple technologies to learn

What do we want

Rethinking the fundamentals to simplify.

- ✓ Services that are implicitly discoverable
- ✓ Flexible M:N communication patterns
- ✓ Decentralized, zero trust security
- ✓ Intelligent routing without additional infra
- ✓ Localized data for decision making
- ✓ Single platform to architect and operate
- ✓ Single technology to learn

NATS enables **coherent** and **secure** application **connectivity** and **communications** of **services** and **data** spanning **clouds**, **geographies**, and **edges**.

Teams ❤ NATS

Designed with each role in mind.

Developers

Build progressive distributed applications with location transparent **messaging**, **streaming**, **key-value**, and **object storage** APIs using a single client SDK, supported in all major languages.

Architects

Design and dynamically adapt topologies spanning **multiple clouds**, **geographies**, and extending to the **edge** without interrupting existing workloads.

Operators

Leverage built-in server **multi-tenancy**, **security**, and **monitoring** enabling **complete visibility** and **control** over the entire system.

What is NATS?

Optimized for simplicity, adaptability, and portability.

- 18MB static Go binary
- Client-server architecture
- No external dependencies
- 4 OSes, 7 arches
- 11 official client libraries

1000+
GitHub contributors

300M+
Docker pulls

30+
Community clients

9500+
Slack members

NATS Year in review

What has been going on?

2024 launches

- NATS Execution Engine (Nex)
- Swift client
- .NET v2 client
- JavaScript client rewrite
- Initiated “Orbit” project

NATS 2.11 release coming soon

- Multi-key direct gets
- Distributed debug tracing
- Consumer pause/unpause
- Sparkplug B compatibility
- Consumer priority groups

Nothing but NATS

Introducing Nex

The OSS NATS Execution Engine

- Store your artifacts anywhere: JetStream, OCI, etc
- Deploy apps anywhere you have NATS connectivity
  - Native
  - JavaScript
  - MicroVM (Firecracker)
  - WebAssembly
  - OCI (Docker)
- All with a single binary, nex

Dev / prod disparity

The hard way.

- Build locally, hope it works in prod
- Install a full prod environment locally
- Simulate so much of prod in local dev that we lose confidence
  - “Test in prod”
- Sacrifice ideal architecture to accommodate easier dev-prod loop
- Point to point comms is brittle
  - Need service discovery and client-side load balancing libraries
- **Spend most of your time debugging your dev environment, not your app**

Dev / prod parity

The NATS way.

- Build locally, know it works the same in prod
- Rely on NATS to communicate with external services
  - Easy mocks for testing and local simulators
  - Hard work is in API definitions and maintaining boundaries
  - You don’t need to install any “real” prod services
  - https://12factor.net/backing-services
- Just Use NATS for persistence
  - Durable streams, server-side consumers, key value buckets, Object stores
- Design the architecture you want, not the one you’re forced to use
- **Leverage emergent behavior**

Nex Host Services

```
(_, payload) => {
  try {
    const js = String.fromCharCode.apply(null, payload);
    const todo = JSON.parse(js);

    this.hostServices.kv.set(todo.id, payload);
    return {
      id: todo.id,
      status: "success"
    }
  } catch (error) {
    return {
      status: "failed",
      error: error
    }
  }
};
```

```
$ nex run ./create.js –trigger-subject todo.create
$ nats req todo.create ‘{“foo”: “bar”}’
```

Counter App

- Illustrate using the 1.0-bound branch of `nex`
  - Start/stop a Nex node
  - Start/query workloads
- Deploy services from public OCI registries
  - Multiple CPU/OS targets
- Runs a nats `micro` service
- Embedded web server

Demo

All-In NATS poker

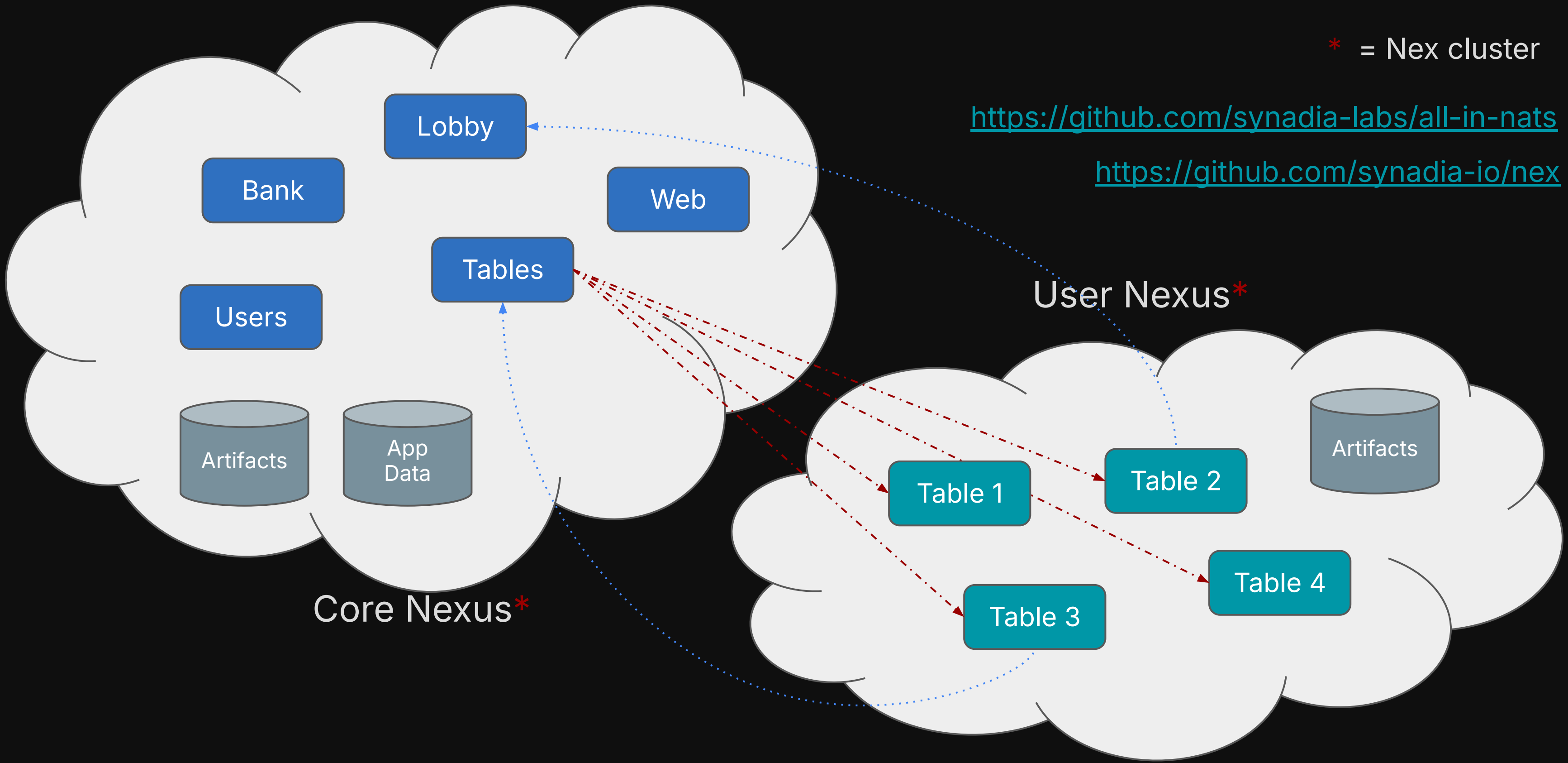
- Suite of small services
- Single Go binary for the web server
- Each poker “table” managed by a single server process
- Uses Nex to start and stop poker table processes dynamically
  - Classic “lobby and shard” pattern
- NATS for everything
  - Persistence
  - Messaging
  - Streams

All-In NATS poker

- Deploy NATS server(s)
- Start Nex node process(es)
- Deploy services:
  - user
  - bank
  - lobby
  - Web
- Code deploys the `table` services

```
$ nats-server -js
$ nex node up
$ nex run ...
```

All-In NATS poker



* = Nex cluster

https://github.com/synadia-labs/all-in-nats

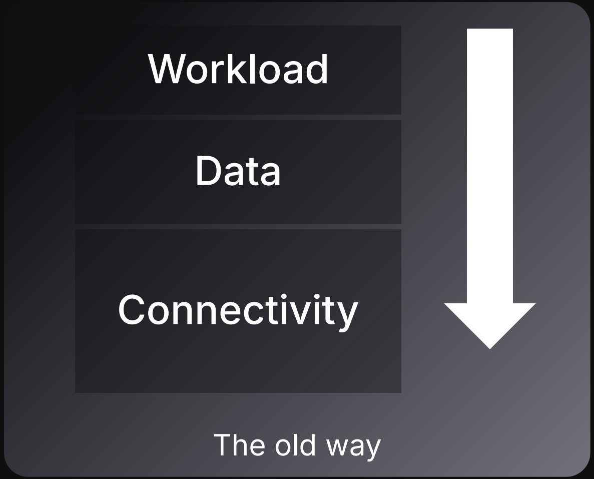
https://github.com/synadia-io/nex

Demo

Going Beyond Cloud-Native

Rethinking application design

Solve the hard part first.

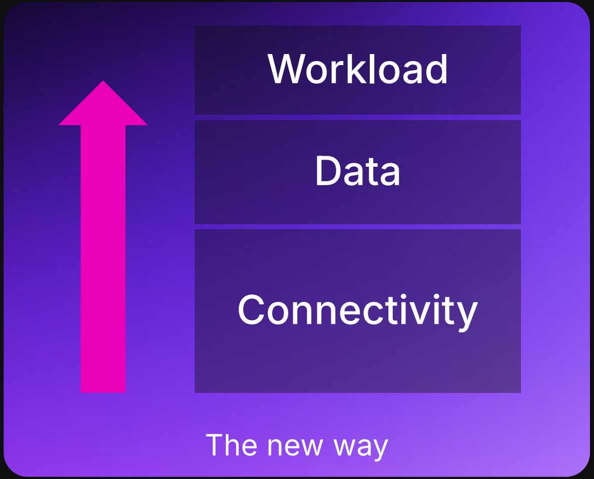


Workload

Data

Connectivity

The old way



Workload

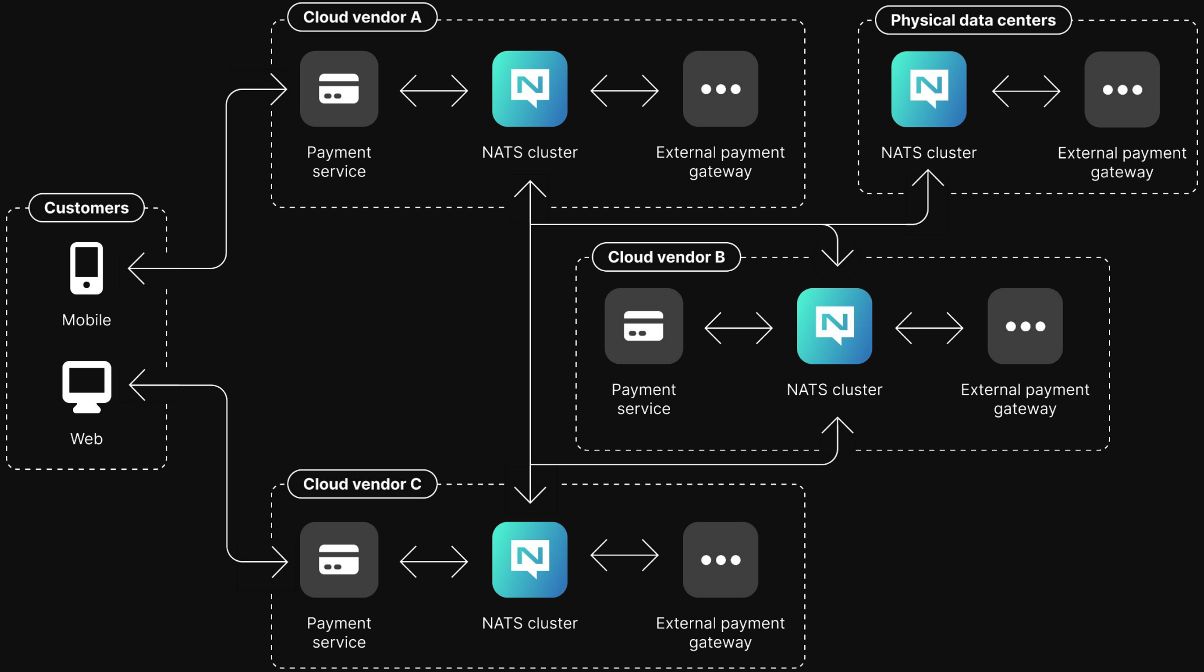
Data

Connectivity

The new way

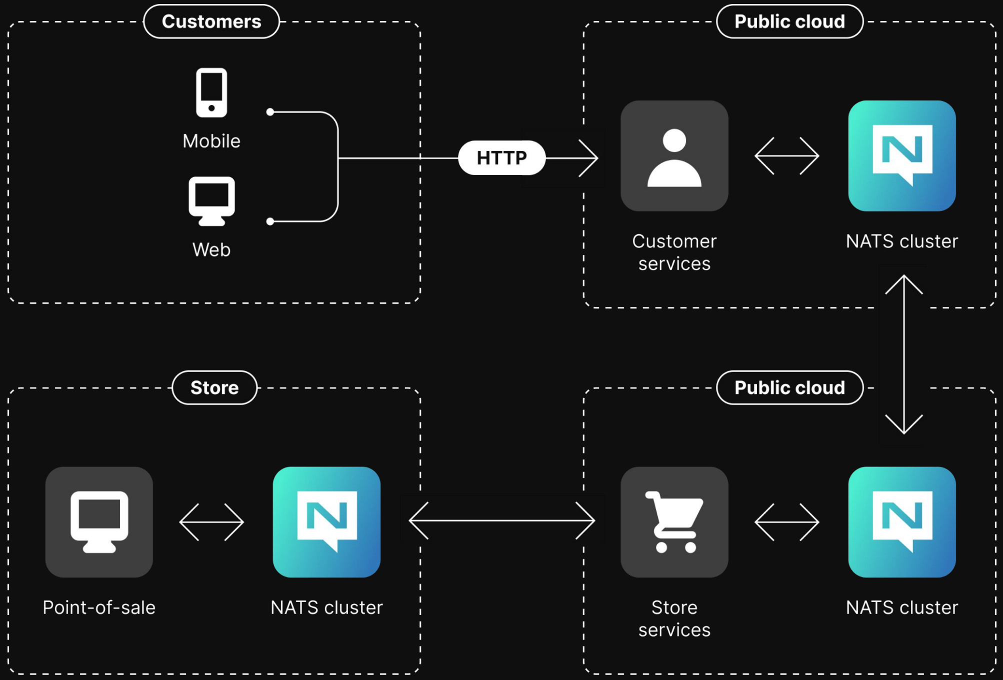
Multi-cloud/geo distribution

Global presence with cluster-aware serving.



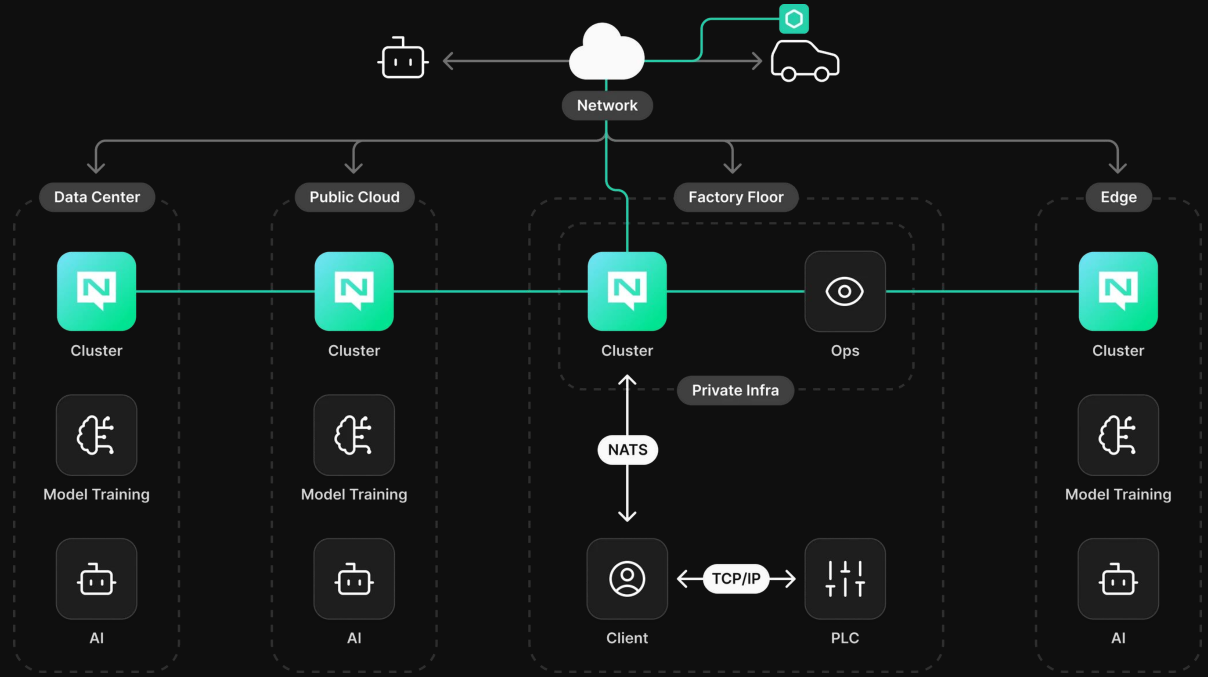
Retail/site/device edge

Offline connectivity and data access.



AI at the edge

Data collection, model push-down, inference serving.



NATS Tech Stack

Connectivity, data, workloads.

- Orchestration
- Runtimes
- Serverless

Nex

- Streaming
- Key value buckets
- Object stores

JetStream

- Message queues
- IoT connectivity
- Load Balancing
- Service discovery
- API gateways
- Observability

NATS

Thank you

Resources and questions!

- Website - nats.io
- Slack - slack.nats.io
- Docs - docs.nats.io
- Examples - natsbyexample.com
- Podcast - nats.fm
- Newsletter - synadia.com/newsletter
- Screencast - synadia.com/screencast
- Demos - synadia.com/demos

Meet the team at **booth P4!**

Share your feedback!