Streams & Services: *From Zero to Hero* 🚀

Waldemar Quevedo, Synadia Communications
Jaime Piña, Synadia Communications

Agenda

- Intro to NATS
- Demo!
  - Decentralized Auth
  - Websockets
  - Kubernetes.yaml

About NATS

NATS is a ten year old, production proven, cloud-native messaging system made for developers and operators who want to spend more time doing their work and less time worrying about how to do messaging.

- ✓ DNA: **Performance**, **simplicity**, **security**, and **availability**
- ✓ Built from the ground up to be cloud native
- ✓ Multiple qualities of service
- ✓ Support for multiple communication patterns
- ✓ Over 30 client languages

About NATS

- The server is a single binary deployable anywhere
- 10.2 MB docker image with no external dependencies
- “Text-based” protocol with a handful of verbs

  PUB | HPUB | SUB | UNSUB | CONNECT | INFO | MSG | HMSG | -ERR | +OK | PING | PONG

- Low Configuration
  - ✓ Clients only need a url and credentials
  - ✓ Servers auto-discover
  - ✓ You can share configuration files amongst servers
- Simple and Straightforward API

NATS Clients

KubeCon CloudNativeCon North America 2020 Virtual

nats.go

Golang client for NATS, the cloud native messaging system.

go golang microservices nats cloud-native

Go ★ 2,265 303 Apache-2.0 3 issues need help Updated a day ago

nats.rb

Ruby client for NATS, the cloud native messaging system.

ruby client messaging cncf pubsub nats eventmachine

Ruby ★ 823 131 Apache-2.0 Updated a day ago

nats.java

Java client for NATS

java client middleware messaging nats messaging-library

Java ★ 194 68 Apache-2.0 Updated a day ago

nats.ex

Elixir client for NATS, the cloud native messaging system. https://nats.io

client elixir nats nats-io

Elixir ★ 33 11 MIT 1 issue needs help Updated 6 days ago

nats.js

Node.js client for NATS, the cloud native messaging system.

JavaScript ★ 672 96 Apache-2.0 Updated 8 days ago

nats.net

The official C# Client for NATS

client visual-studio csharp messaging message-bus pubsub

C# ★ 232 63 Apache-2.0 3 issues need help Updated 2 days ago

nats.c

A C client for NATS

c messaging message-bus message-queue messaging-library

C ★ 139 45 Apache-2.0 Updated 7 days ago

nats.py

An asyncio based Python 3 client for NATS

aio nats python3 asyncio cloud-native aio-nats

Python ★ 187 34 Apache-2.0 Updated 4 days ago

Streams & Services

NATS in a Nutshell

- **Streams**
  - A *flow* of data
  - Fan out
- **Services**
  - Do some work and return a result
  - Load balanced

NATS Subjects

A subject is simply a string representing an interest in data.

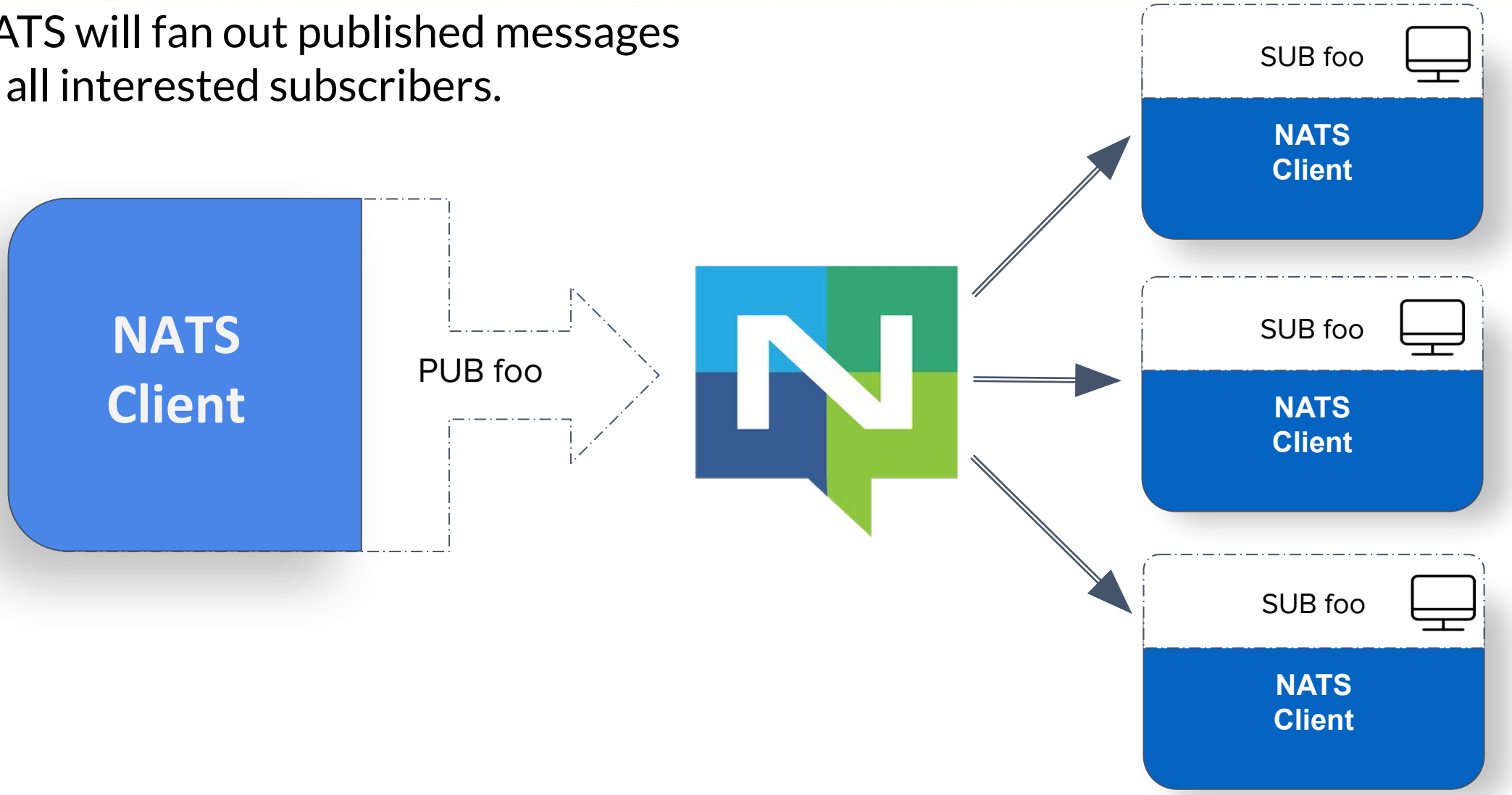
- Simple subject: **foo** or **weather**
- Hierarchically Tokenized: **foo.bar, weather.us.co.denver**
- Wildcard subscriptions
  - ✓ **foo.*** matches **foo.bar** and **foo.baz**.
  - ✓ **foo.*.bar** matches **foo.a.bar** and **foo.b.bar**.
  - ✓ **foo.>** matches any of the above
  - ✓ **>** matches everything in NATS
- Unique subjects for 1:1 addressability

NATS Client API

- Request/Reply (**Services**)
- Publish/Subscribe (**Streams**)
- Load Balanced Queue Subscribers (**Streams** or **Services**)

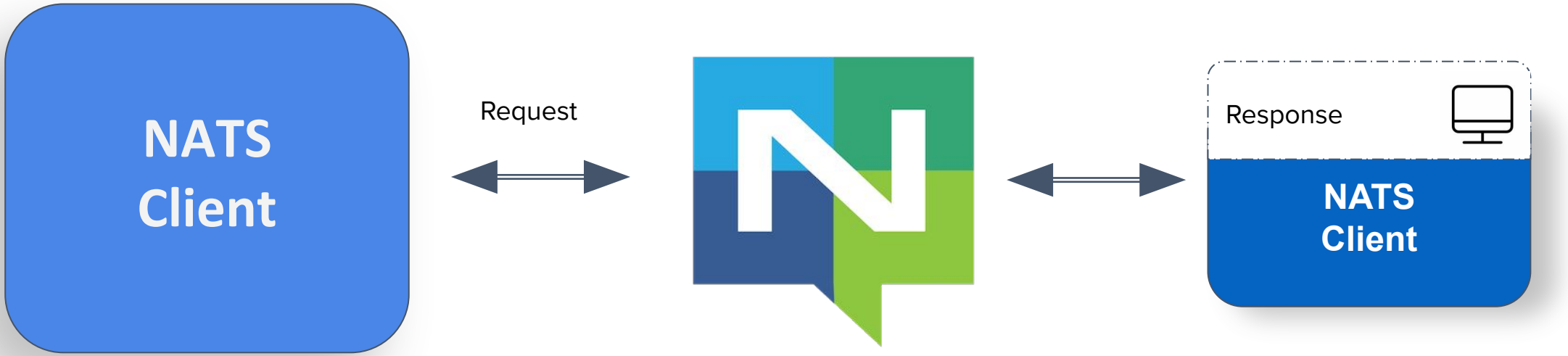
Streams: Publish/Subscribe

NATS will fan out published messages to all interested subscribers.



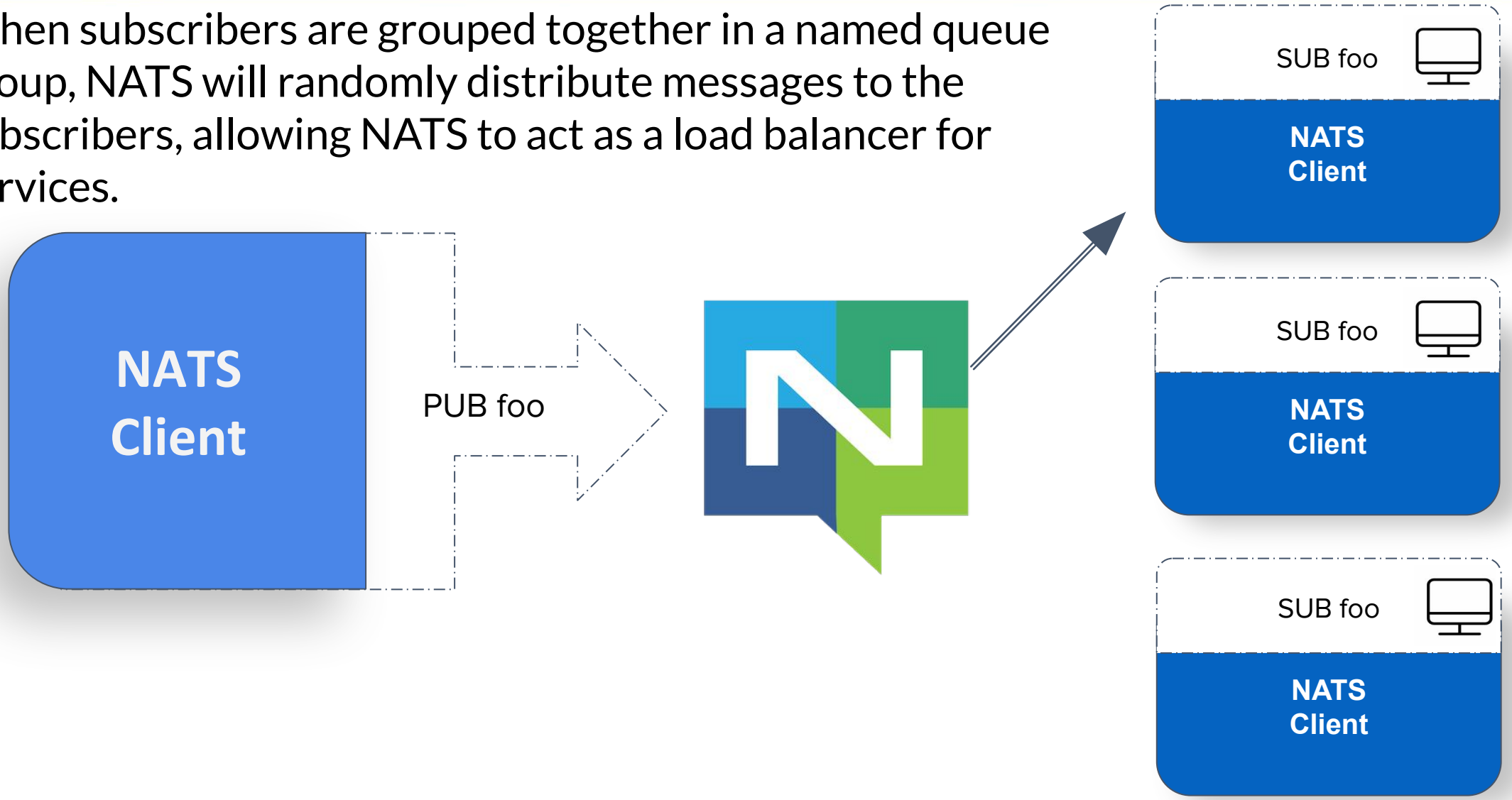
Services: Request/Response

Using unique reply subjects, clients can make requests to services that respond only to the request, creating a 1 to 1 relationship.



Load Balancing

When subscribers are grouped together in a named queue group, NATS will randomly distribute messages to the subscribers, allowing NATS to act as a load balancer for services.



Cloud + Edge: Multi-tenancy, Multi-region

Accounts (Multi-tenancy)

Streams

```
accounts {

  acme {
    users [
      { user: alice, pass: foo }
    ]

    exports = [
      { stream: "acme.>" }
    ]
  }

  cncf {
    users [
      { user: bob, pass: bar }
    ]

    imports = [
      { stream: {
          account: "acme",
          subject: "acme.>" },
        prefix: "imports" }
    ]
  }
}
```

Export

```
for range time.NewTicker(1 * time.Second).C {
        nc.Publish("acme.hello", []byte("Hello world"))
}
```

Import

```
// Receives message at: imports.acme.hello
nc.Subscribe("imports.>", func(m *nats.Msg) {
        log.Printf("[%s]: %s", id, string(m.Data))
})
```

Accounts (Multi-tenancy)

Services

```
accounts {
  acme {
    users [
      { user: alice, pass: foo }
    ]

    exports = [
      { service: "hello" }
    ]
  }

  cncf {
    users [
      { user: bob, pass: bar }
    ]

    imports = [
      { service: {
          account: "acme",
          subject: "hello" },
        to: "cncf.hello" }
    ]
  }
}
```

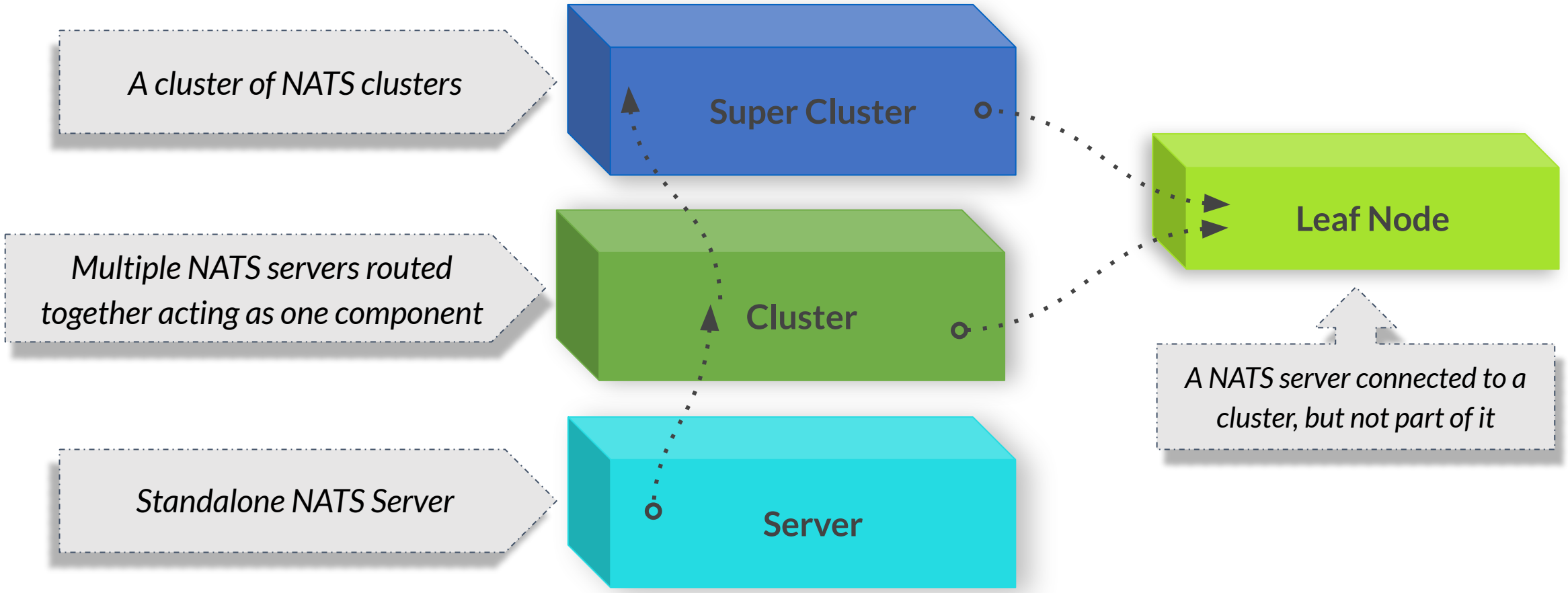
Export

```
nc.QueueSubscribe("hello", "worker", func(m *nats.Msg) {
        log.Printf("[%s]: %s", id, string(m.Data))
        nc.Publish(m.Reply, []byte("hi!"))
})
```

Import

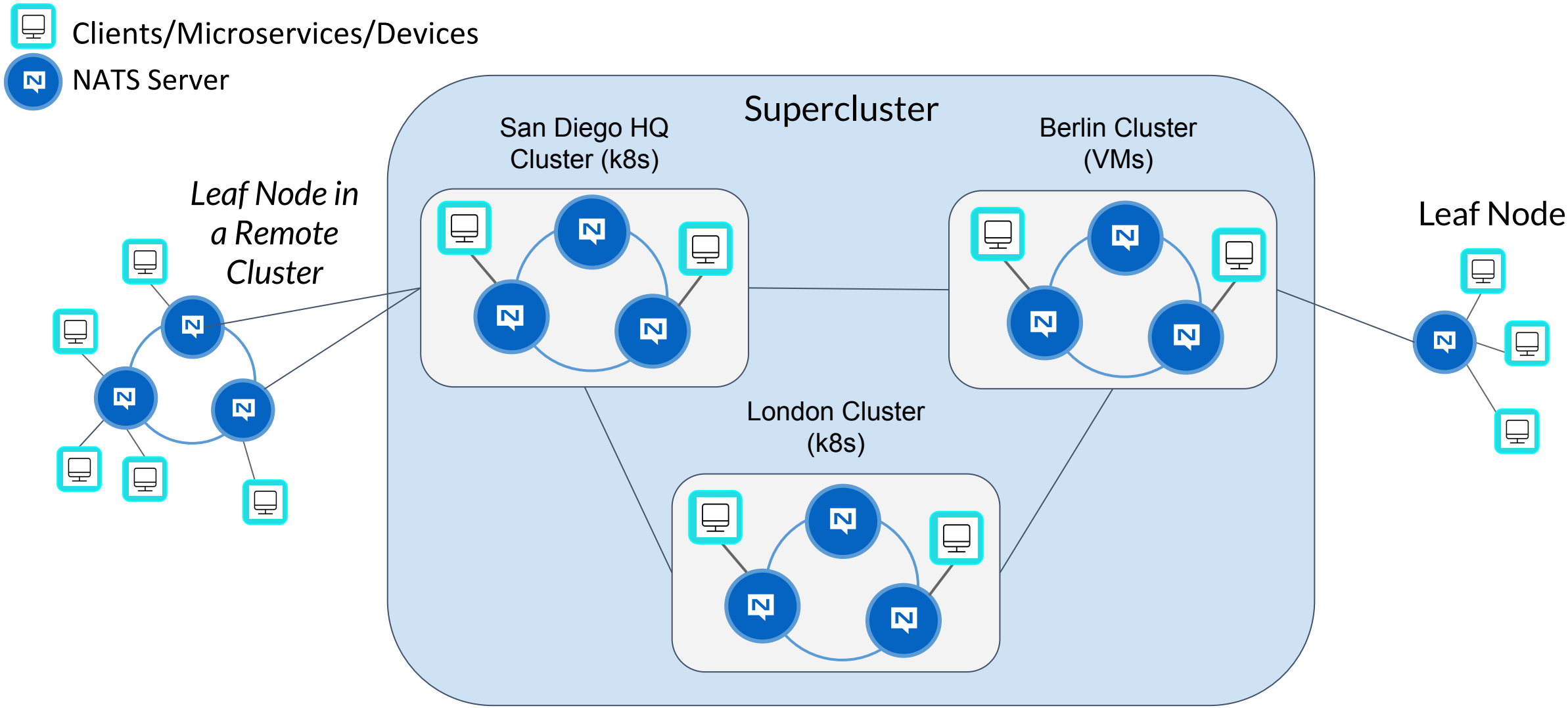
```
nc.Request("cncf.hello", []byte("world"),
```

Topologies



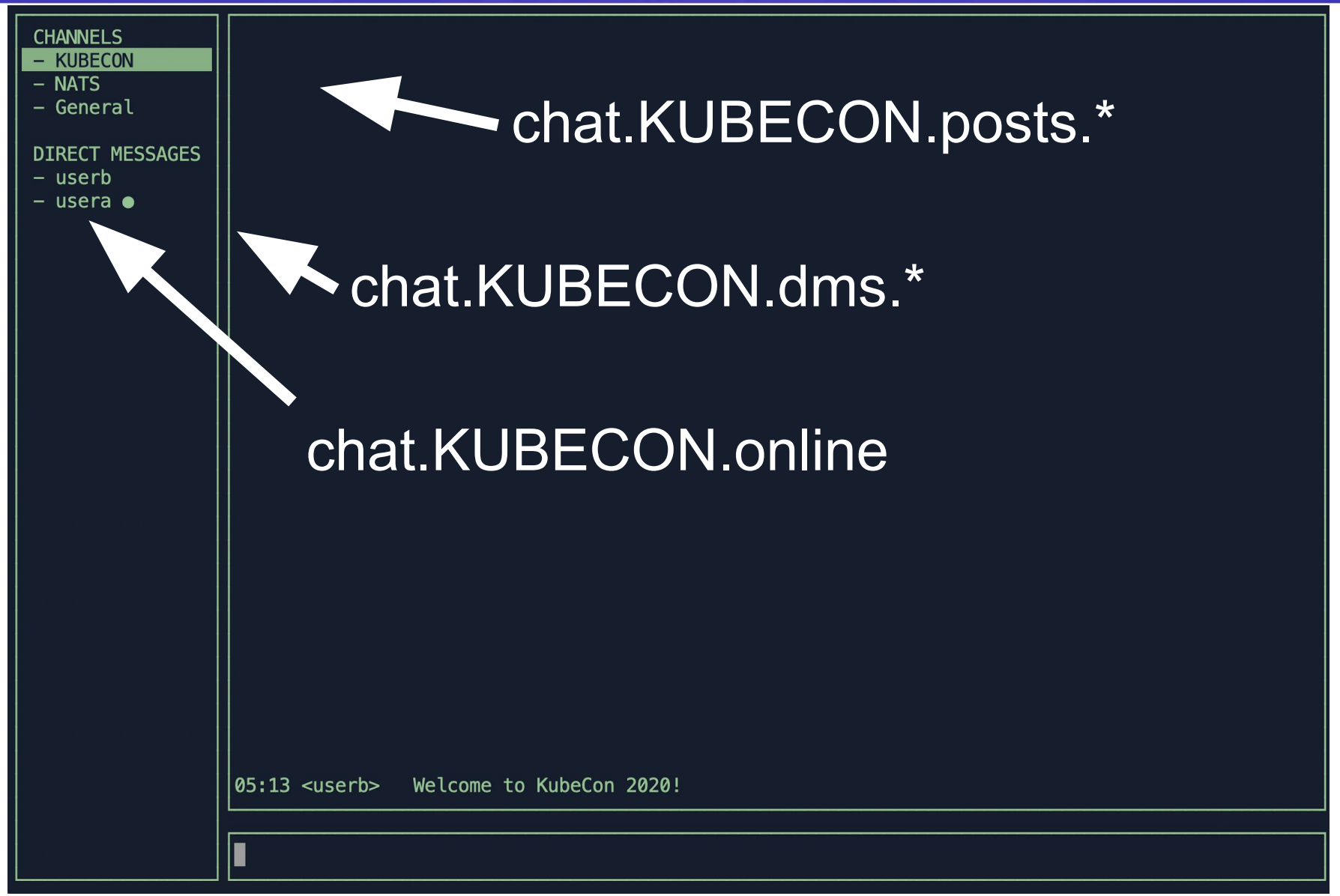
Clients require **no awareness** of server topology beyond a connection URL.

NATS Adaptive Edge Architecture

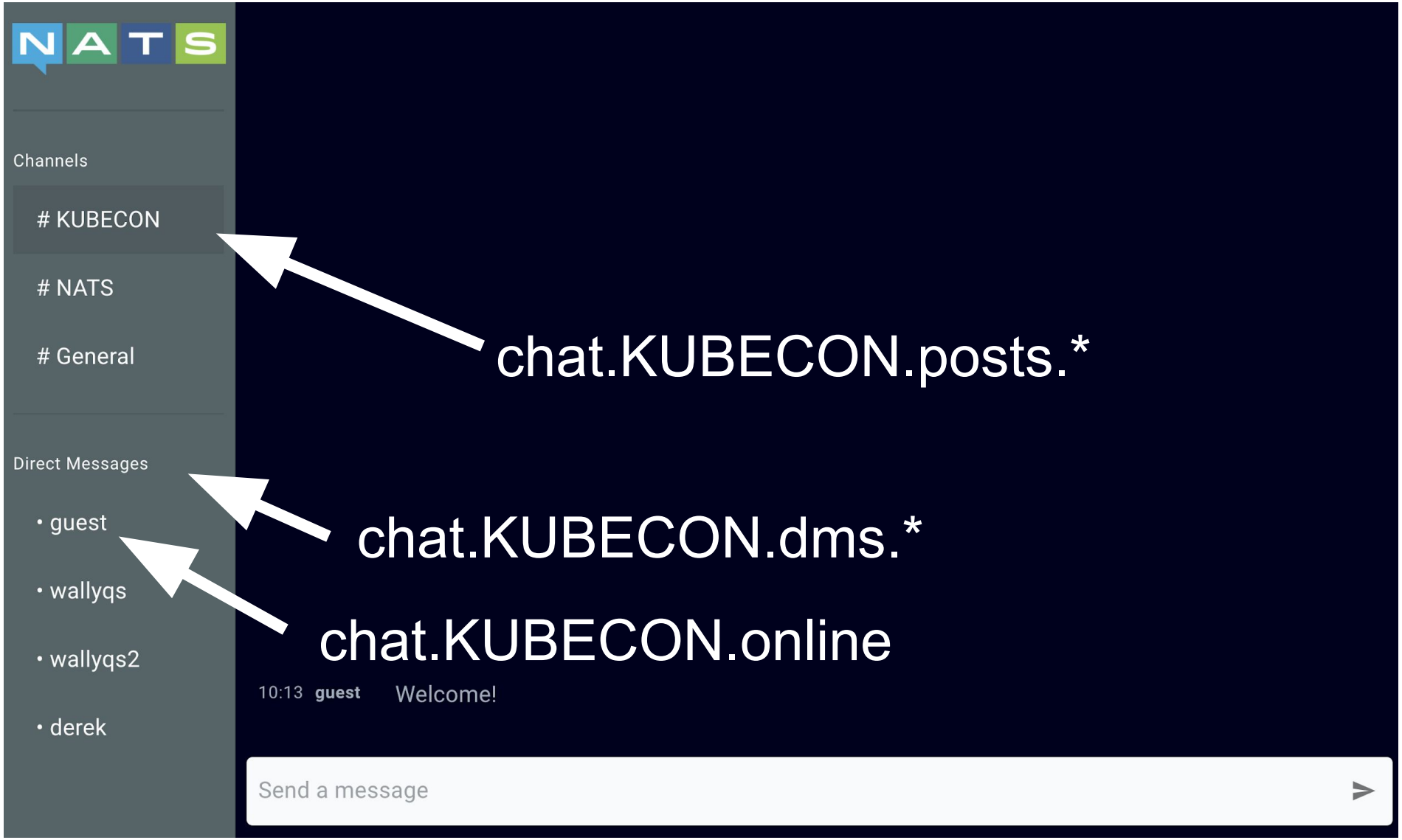


Demo: nats.chat

Demo: Streams



Demo: Streams over Websockets



Demo: Services

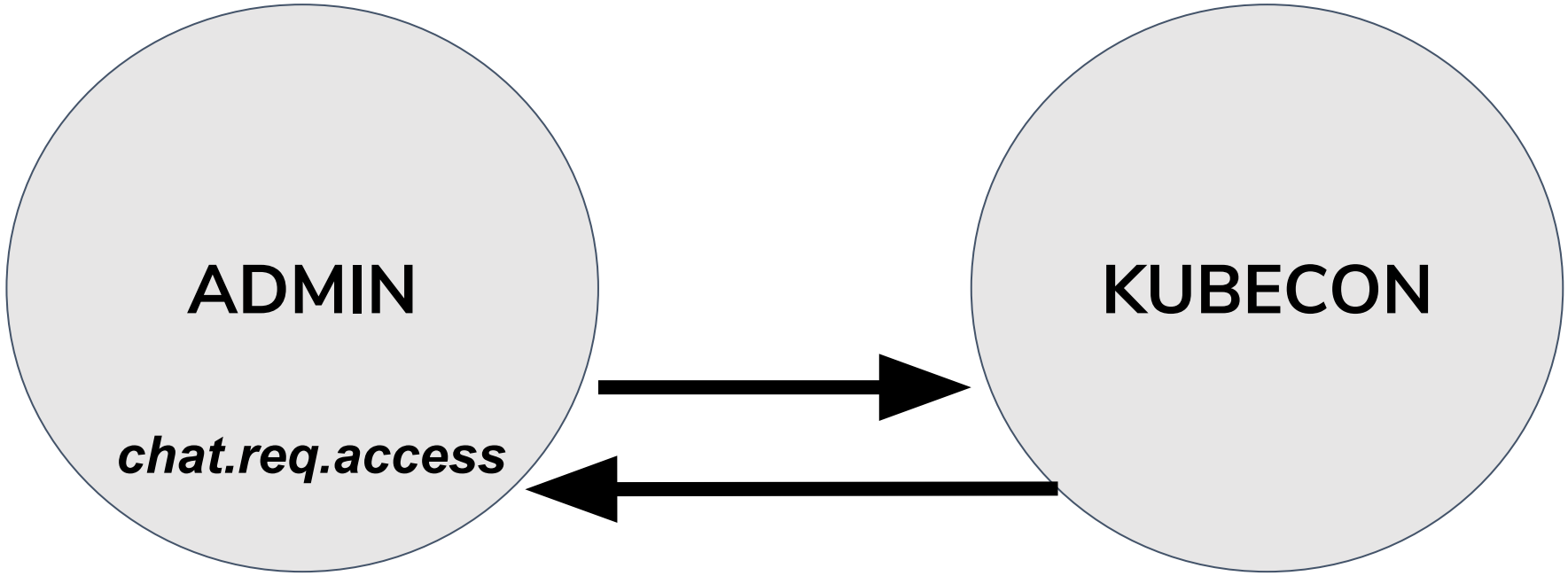
To get access to the Chat Streams, first need to request for credentials.

SERVICE: chat.req.access

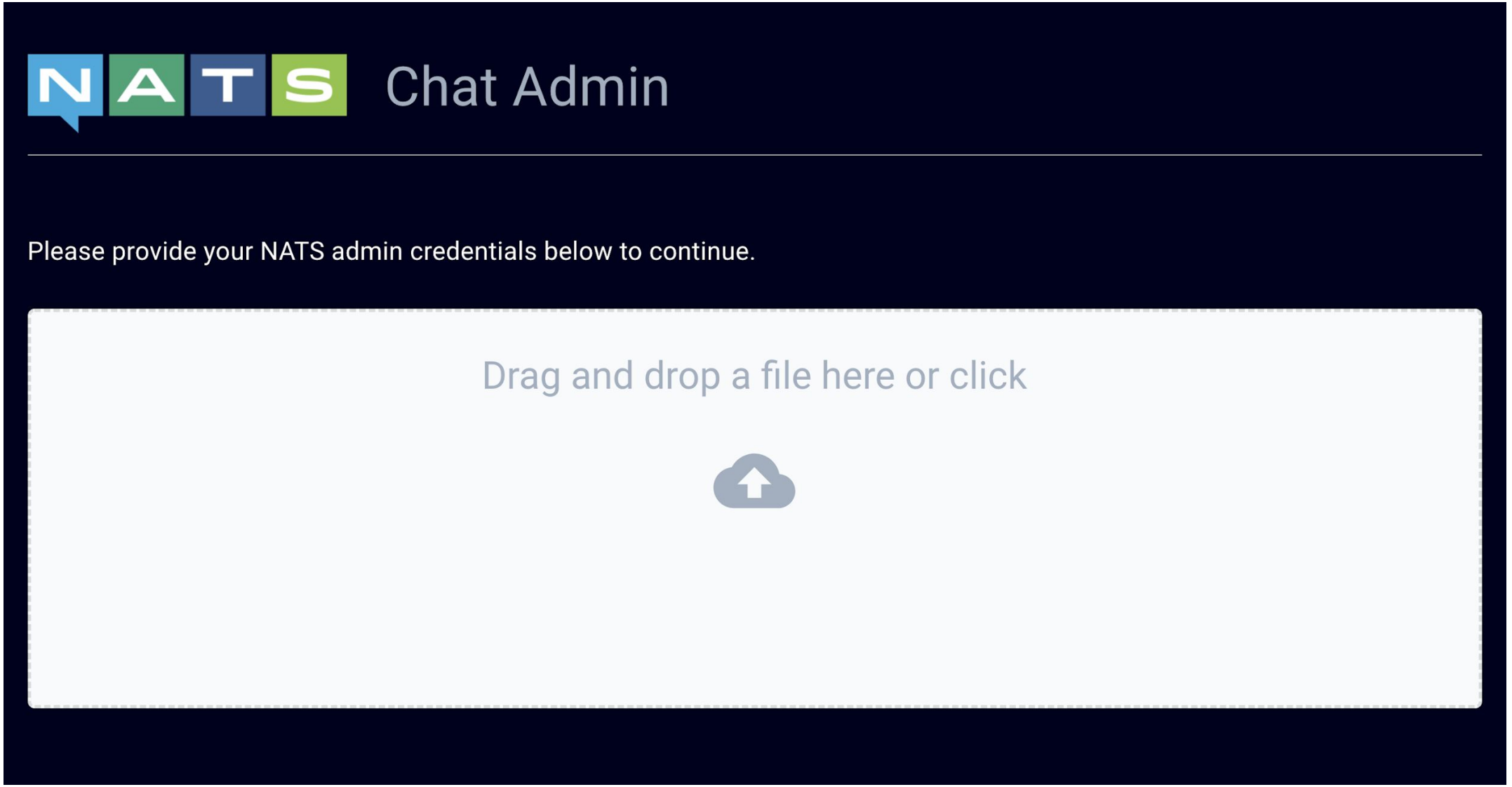
Demo: Accounts

chat.KUBECON.posts.*

chat.KUBECON.dms.*

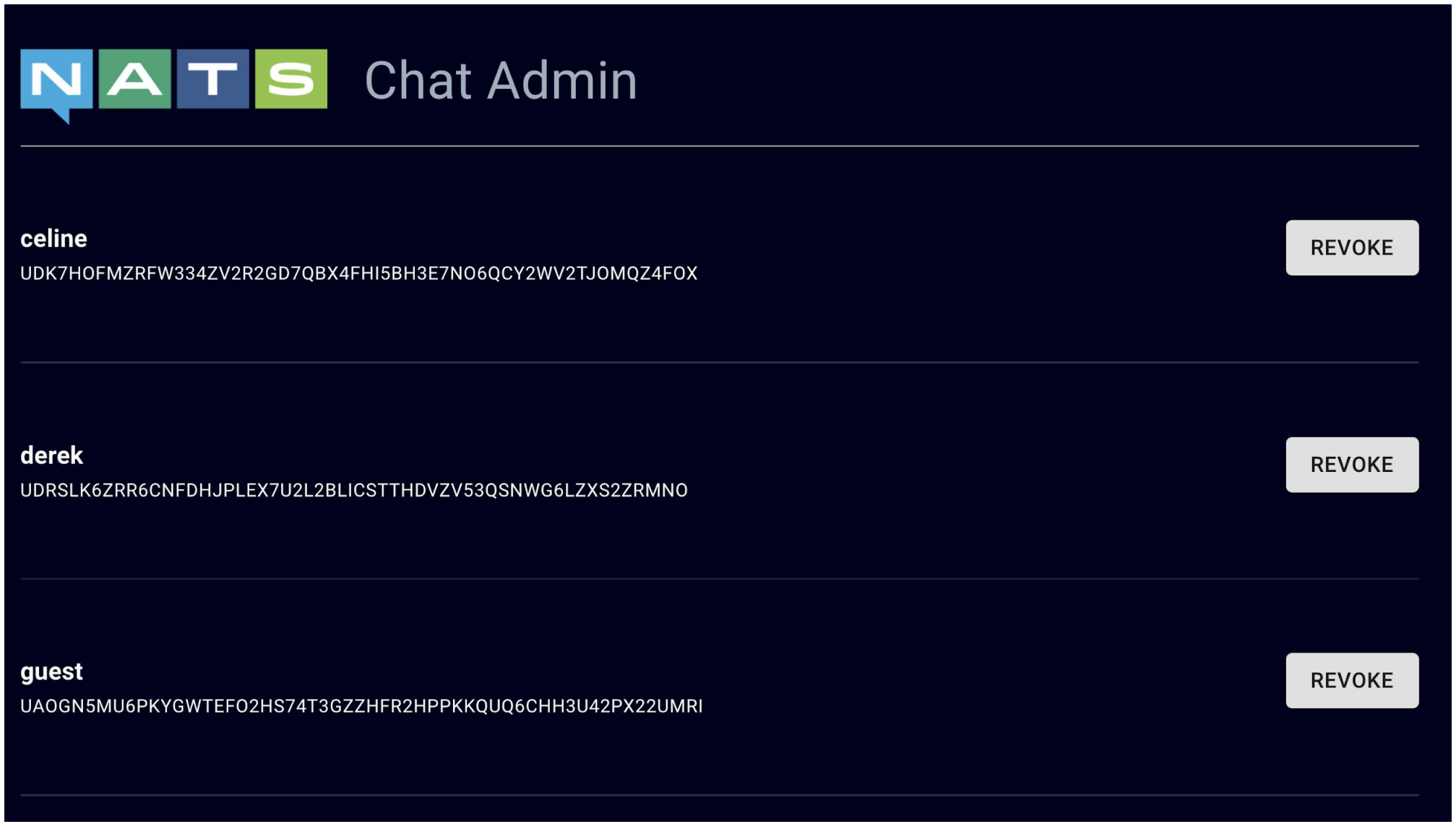


Demo: Services: Revoking Users



Demo: Services: Revoking Users

KubeCon CloudNativeCon North America 2020 Virtual



Questions?

https://natsio.slack.com ← *Join the NATS community :)*

github.com/nats-io / @nats_io

https://nats.io