KubeCon | CloudNativeCon

Europe 2018

Deep Dive into NATS

Thursday, May 3 16:35 - 17:10

Waldemar Quevedo / *wally@synadia.com*
Colin Sullivan / *colin@synadia.com*

Agenda

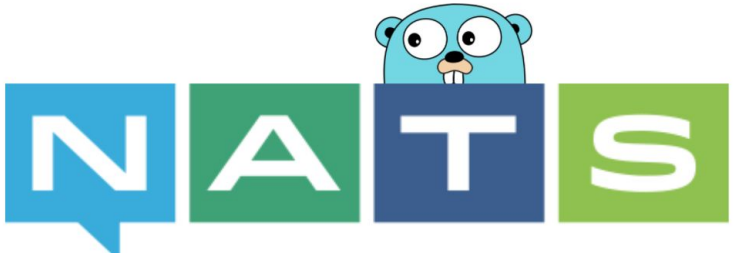
KubeCon | CloudNativeCon Europe 2018

- Internal workings of NATS & NATS Streaming
- Highlights from the NATS Ecosystem
- Demos


KubeCon
CloudNativeCon
Europe 2018
NATS

NATS

- High Performance Messaging Server written in Go
  - originally started in Ruby with Eventmachine
- Pure PubSub on top of TCP/IP
- Simple Plain Text Protocol
  - makes writing clients for it much easier and fun
- Binary name is *gnatsd*
  - *http://github.com/nats-io/gnatsd*



The NATS Protocol


KubeCon | CloudNativeCon Europe 2018

Straightforward use with `telnet` to interact with the NATS server

```
telnet demo.nats.io 4222
Connected to demo.nats.io.
INFO {"server_id":"T8NVSIWciEMgVSyFLHiDEs",...,"max_payload":1048576}
SUB hello.copenhagen 1
+OK
PUB hello.copenhagen 5
world
+OK
MSG hello.copenhagen 1 5
world
```

The NATS Protocol

KubeCon | CloudNativeCon Europe 2018

Complete NATS protocol is only 10 commands

```
Client → Server :=   | PUB | SUB | UNSUB | CONNECT |
Client ← Server :=   | INFO | MSG | -ERR | +OK |
Client ⇆ Server :=   | PING | PONG |
```

The NATS Clients

The latest features are in the Go client but there are clients available for most platforms and are fairly simple

```
nc, _:= nats.Connect("nats://demo.nats.io:4222")
done := make(chan struct{})
nc.Subscribe("hello", func(m *nats.Msg){
    log.Printf("[Received] %s", string(m.Data))
    done <- struct{}{}
})
nc.Publish("hello", []byte("world"))
<-done
```

The NATS Server

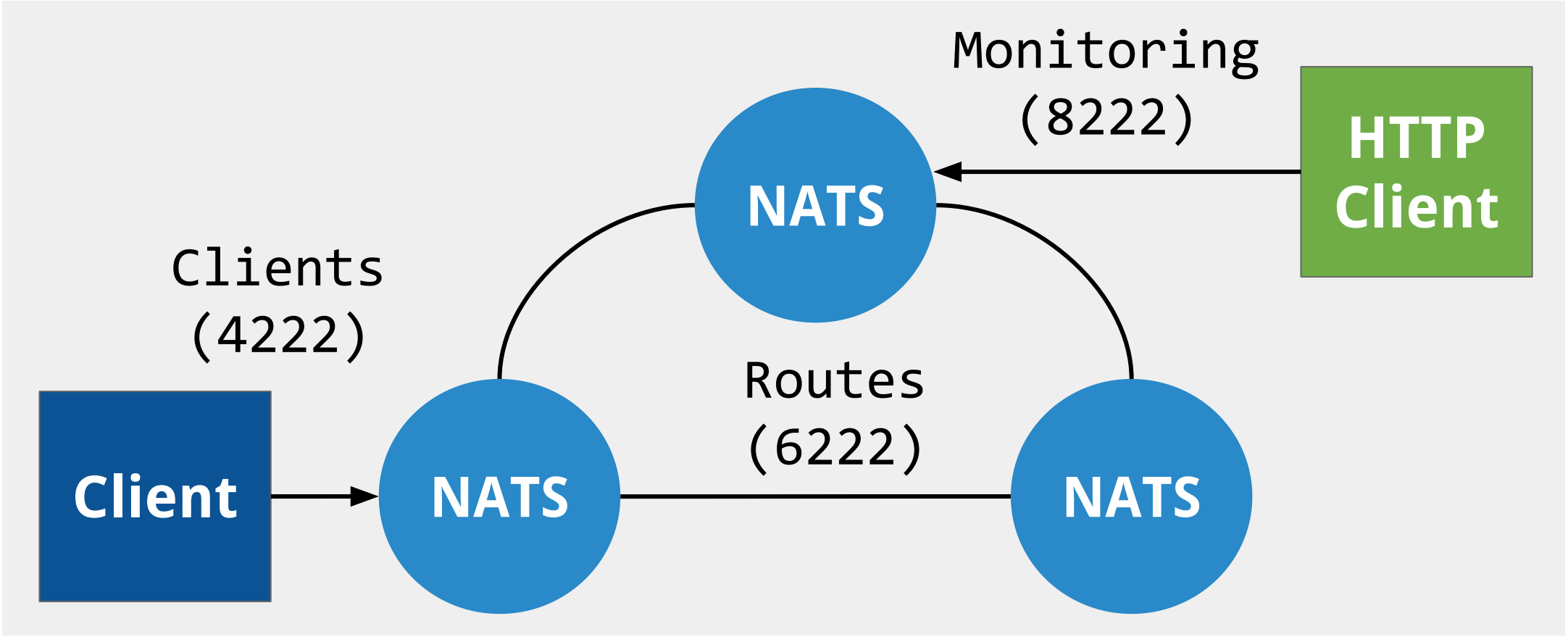
KubeCon | CloudNativeCon Europe 2018

Server is a 7MB binary, no dependencies
Releases are in Github: https://github.com/nats-io/gnatsd/releases

```
$ gnatsd -m 8222 --logtime=false
[28110] [INF] Starting nats-server version 0.1.1
[28110] [INF] Starting http monitor on 0.0.0.0:8222
[28110] [INF] Listening for client connections on 0.0.0.0:4222
[28110] [INF] Server is ready
```

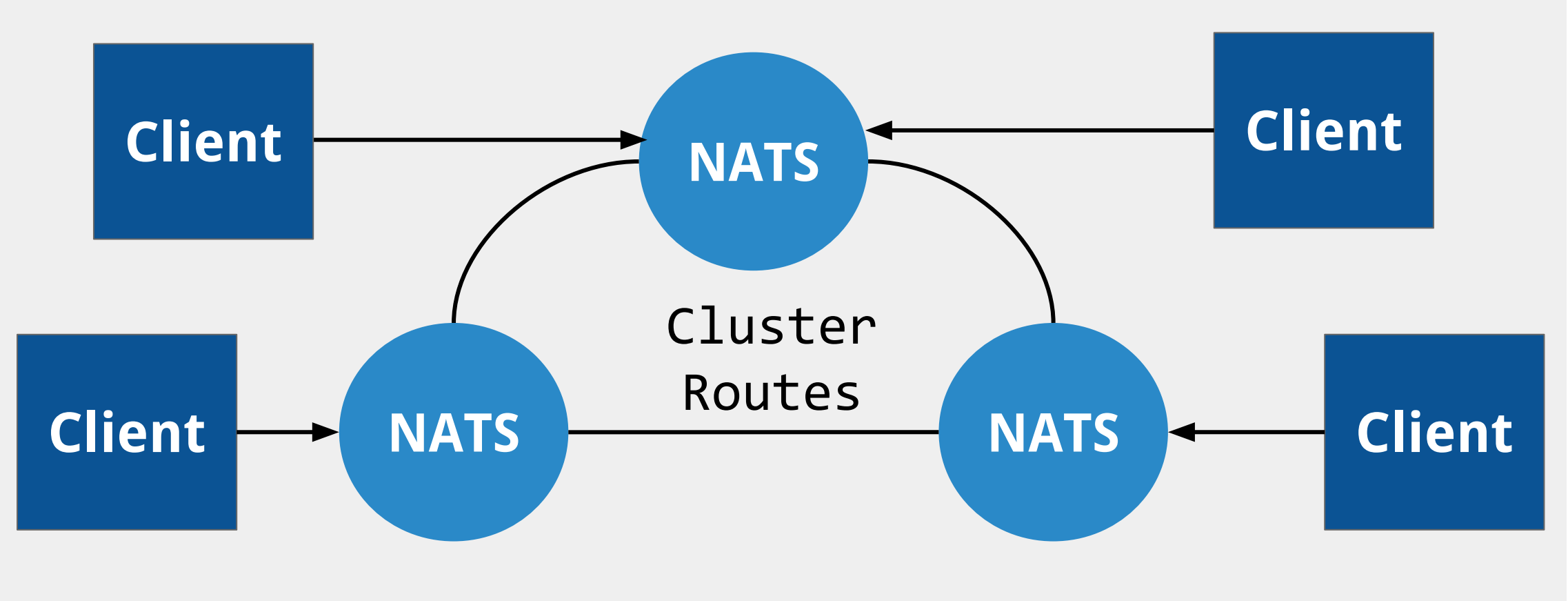
Used Ports

Production setup typically has the following 3 ports



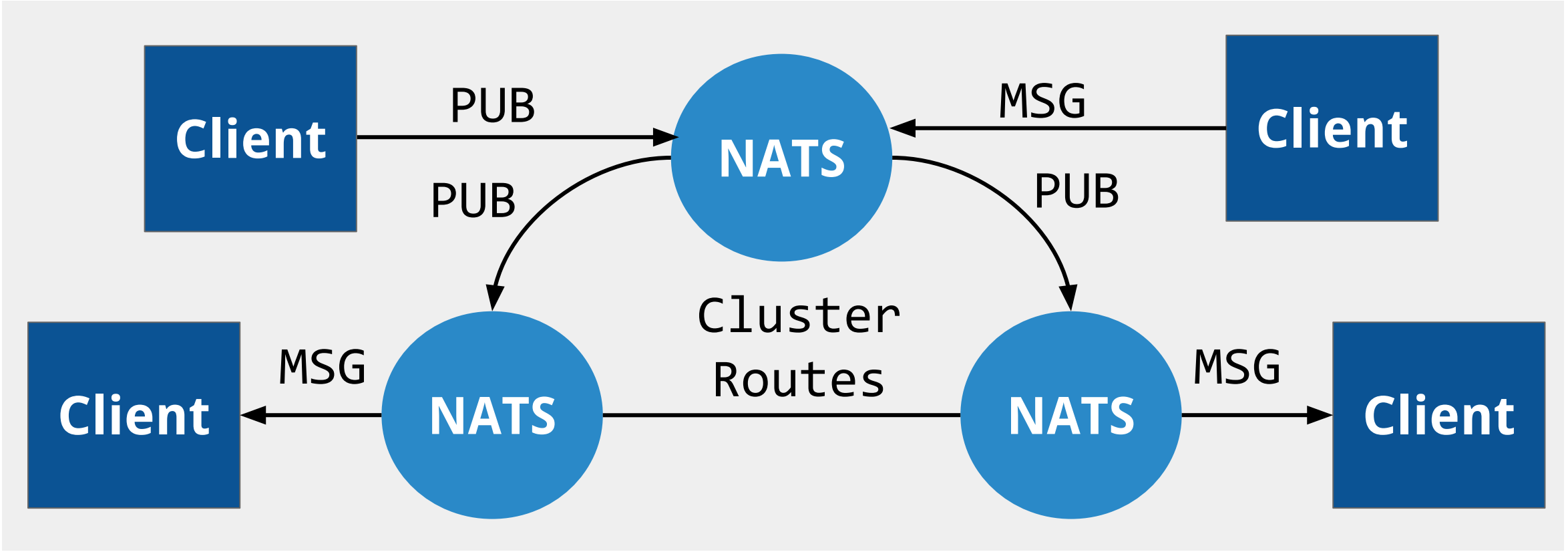
NATS Clustering

For high availability, a full mesh of NATS servers can be setup



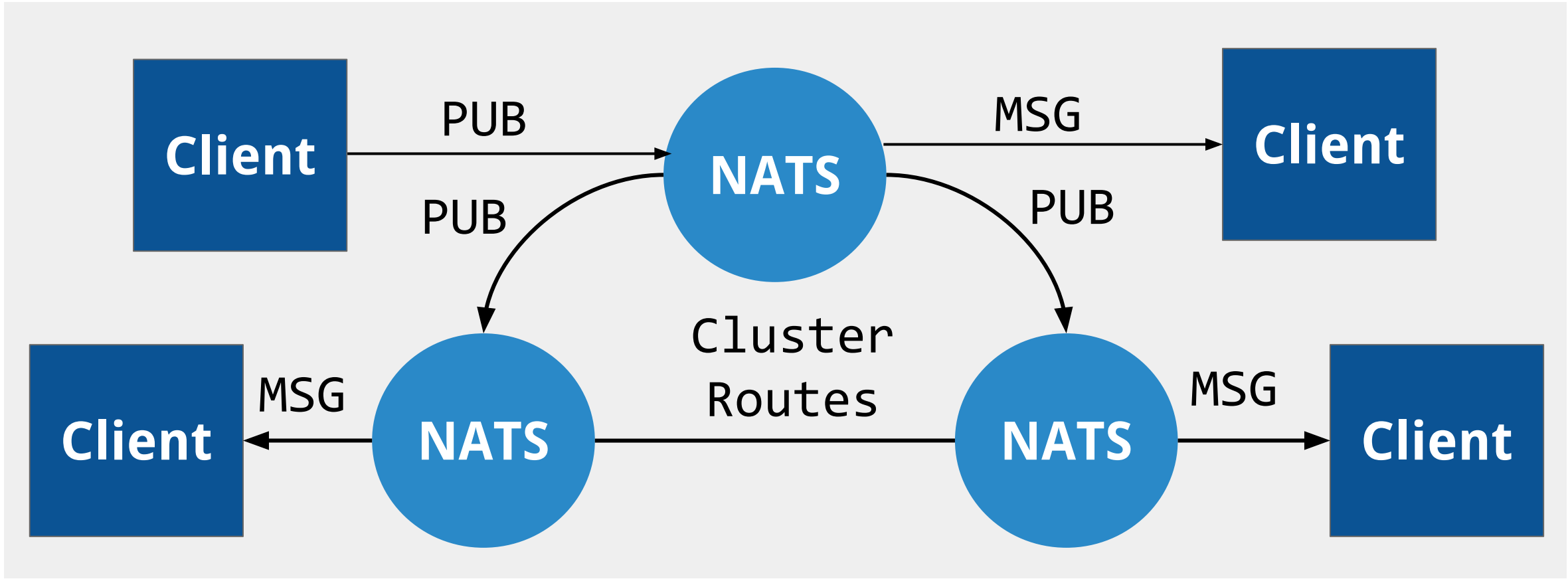
NATS Clustering

Clients can connect to any of the nodes to communicate with other clients, the NATS cluster would then route the messages.



NATS Clustering

Routing would only be done if clients showed interest in subject



Setting up a NATS cluster

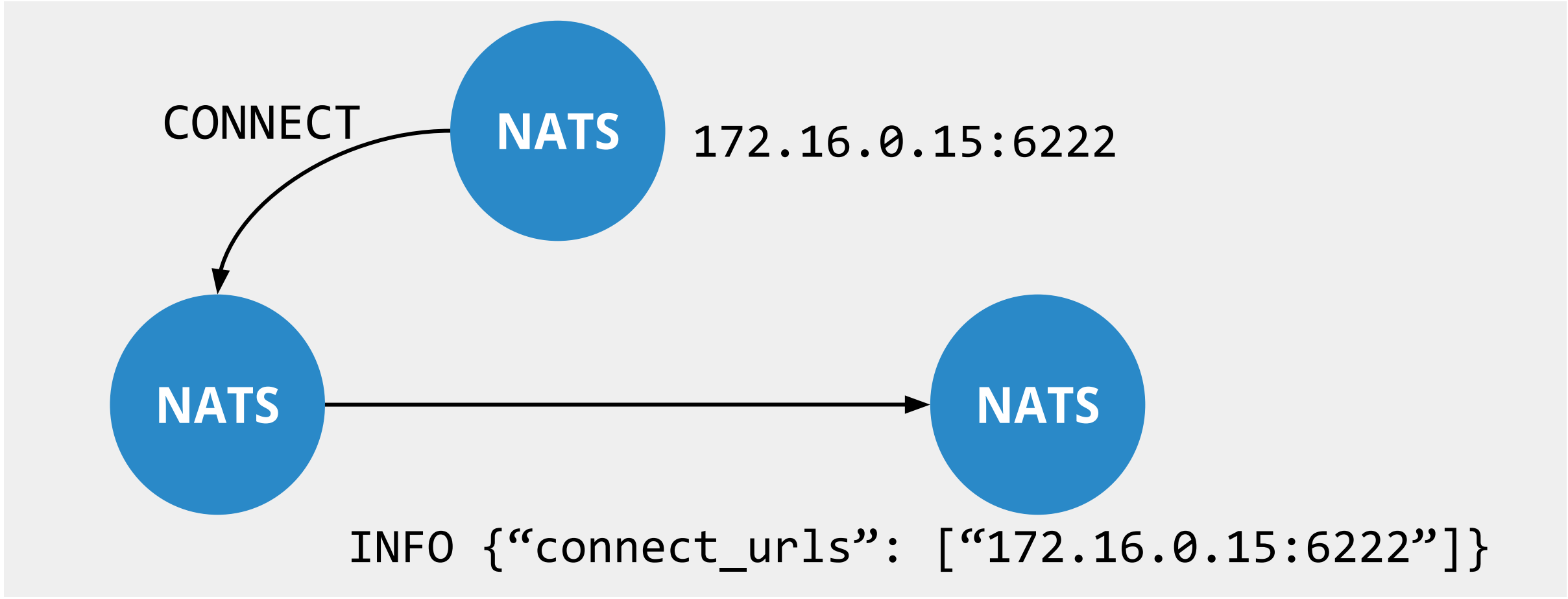
KubeCon | CloudNativeCon Europe 2018

A simple way to setup clustering is via the auto discovery support

```
$ gnatsd -m 8222 -p 4222 --cluster "nats://127.0.0.1:6222"

$  gnatsd -m 8223 -p 4223 --cluster "nats://127.0.0.1:6223" \

   --routes "nats://127.0.0.1:6222"

$ gnatsd -m 8224 -p 4224  --cluster "nats://127.0.0.1:6224" \
   --routes "nats://127.0.0.1:6222"
```

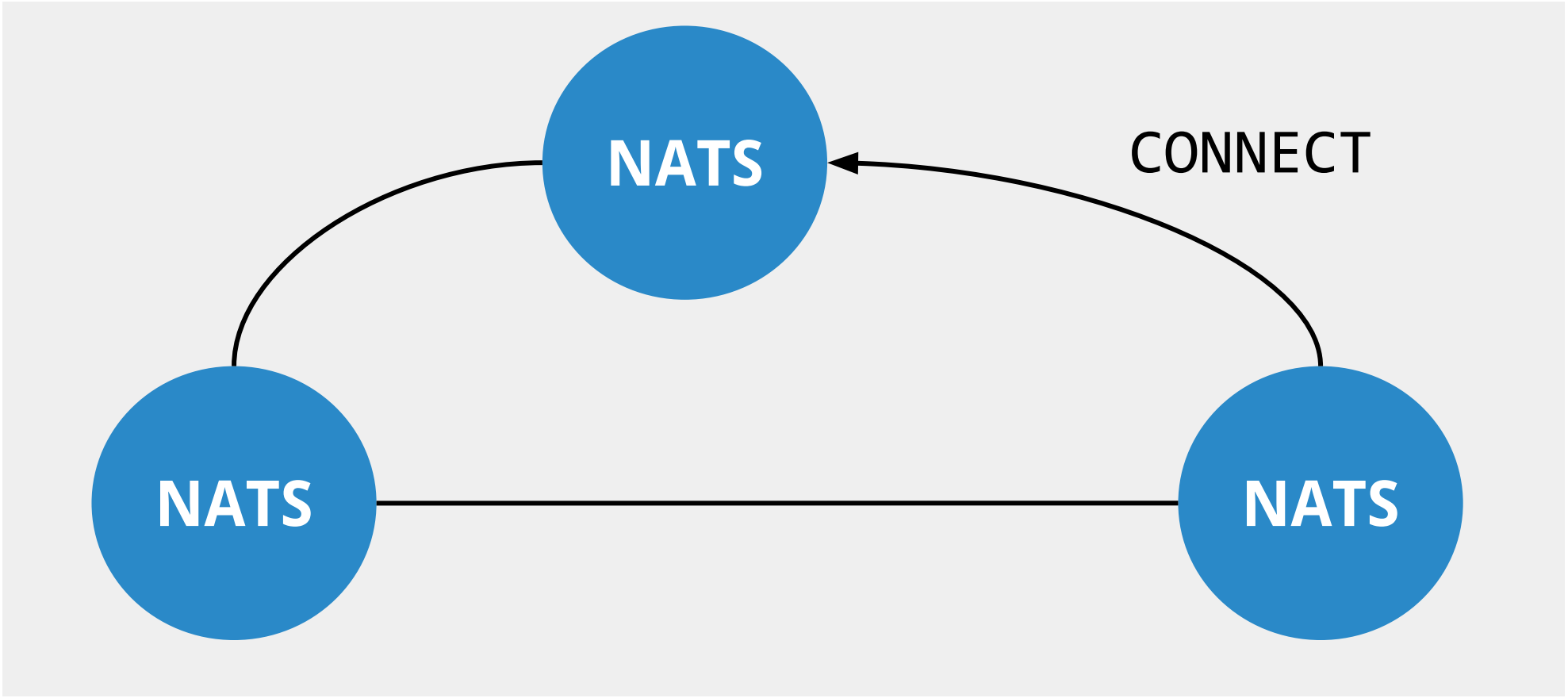
Cluster Auto Discovery

Whenever a new NATS servers joins, its network location is announced...



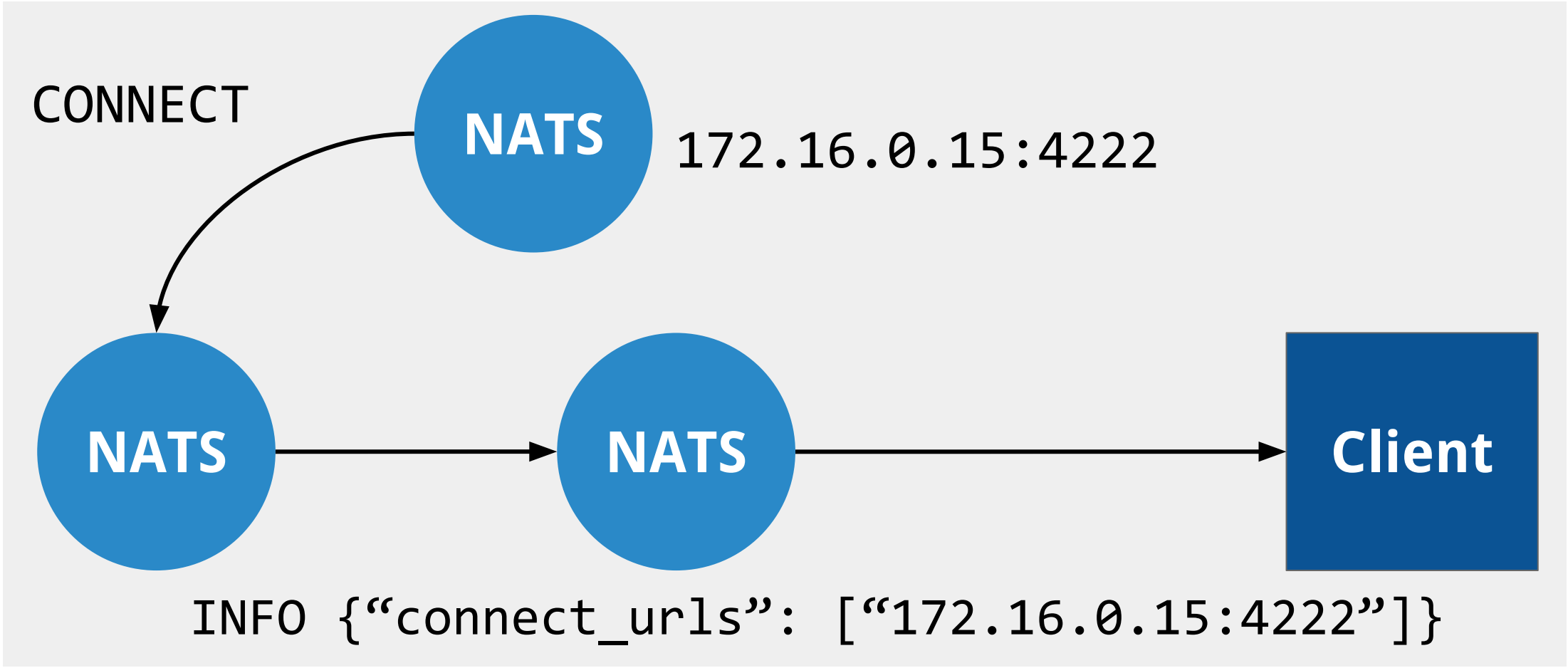
Cluster Auto Discovery

...then members already in cluster connect as well to form the full mesh



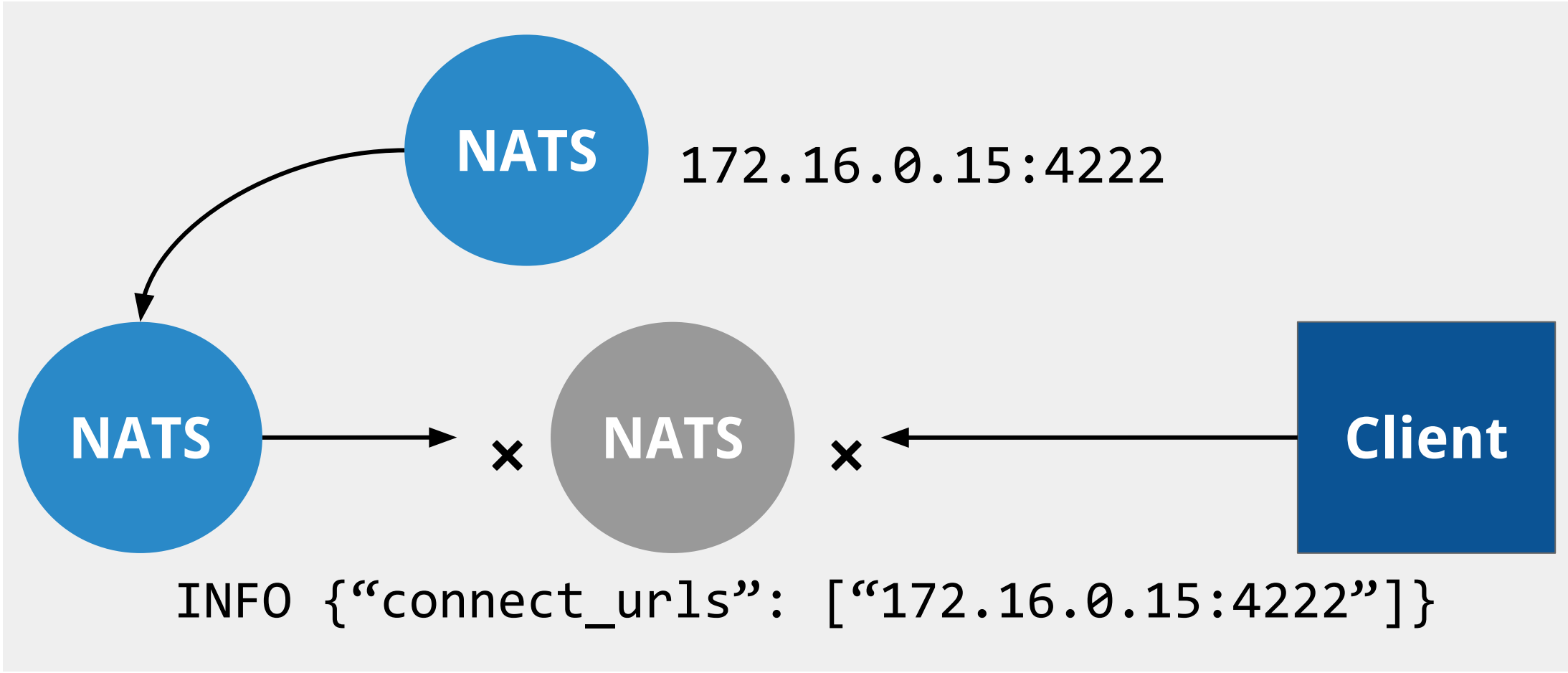
Cluster Auto Discovery

Its network endpoint is also gossiped to the clients already connected



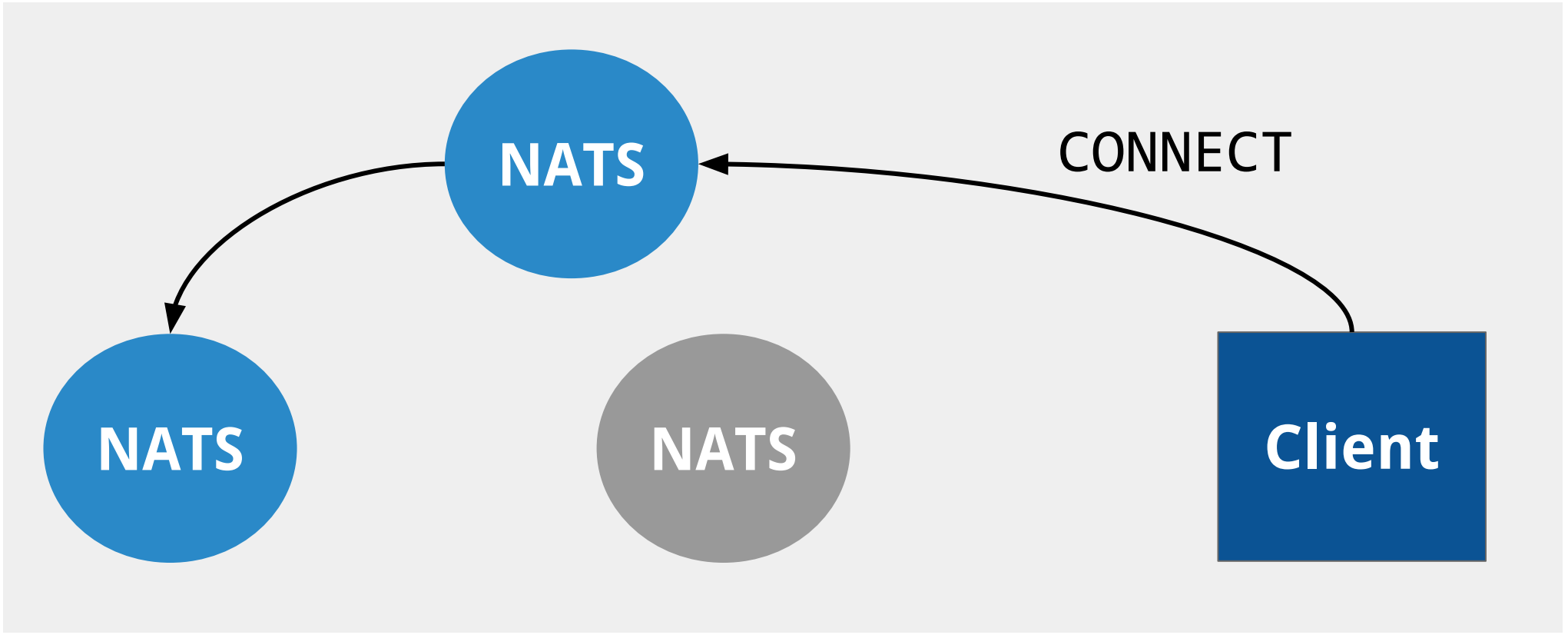
Cluster Auto Discovery

On failure, clients that are able to...



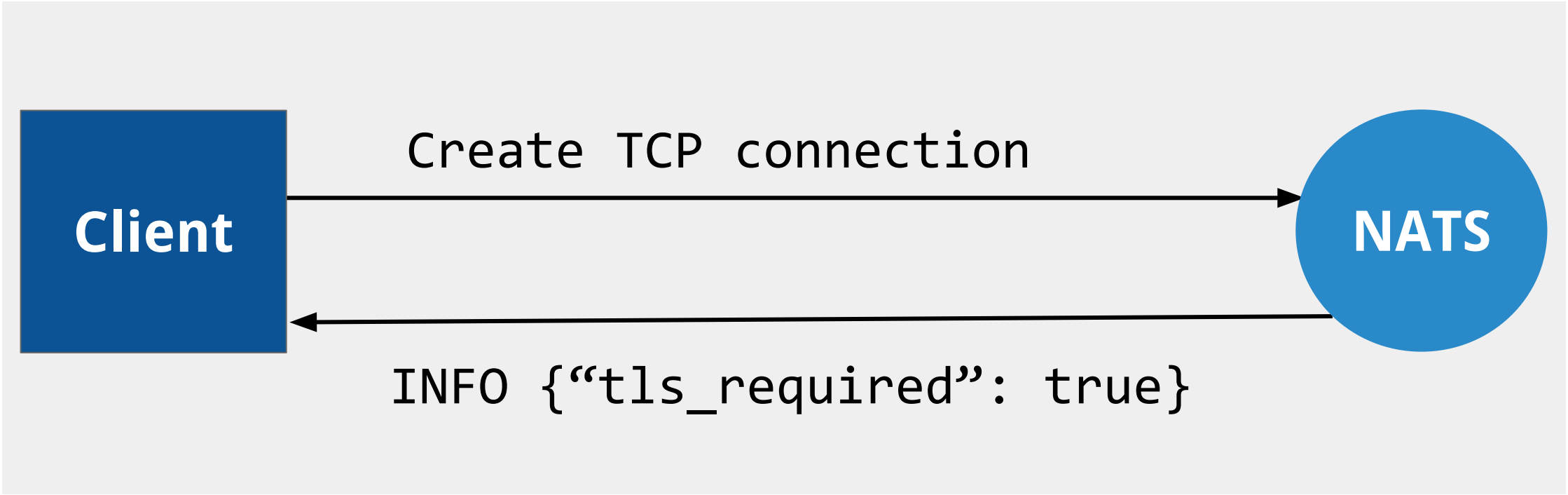
Cluster Auto Discovery

… failover to an available node



TLS support

Secure connection to NATS can be setup to be enforced by the server

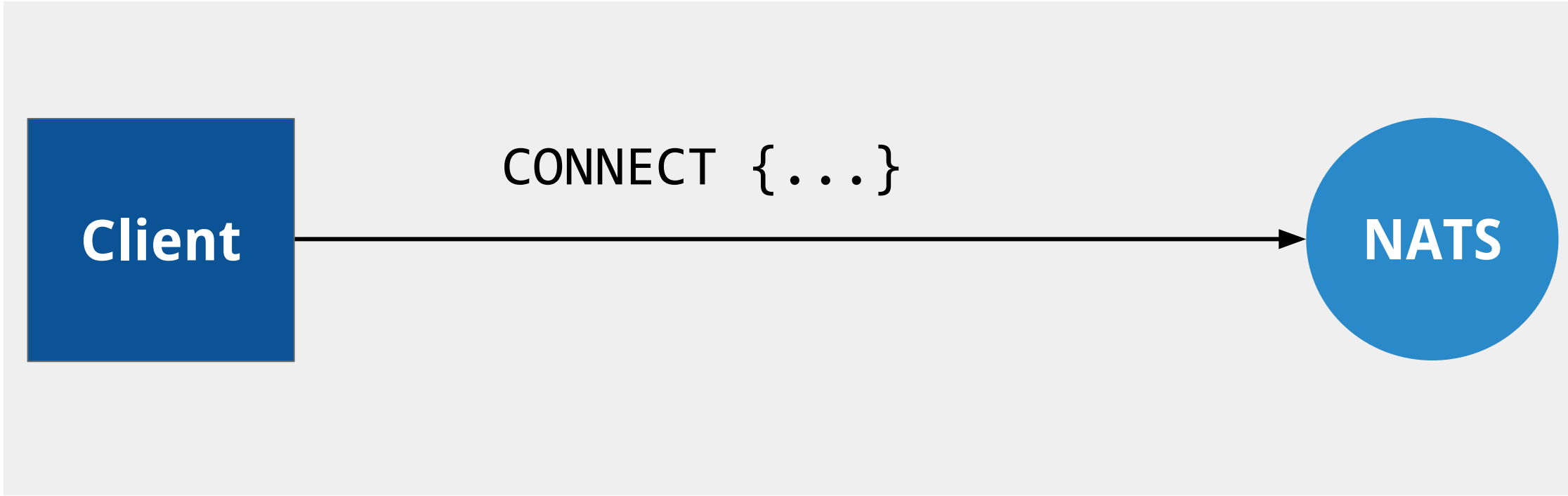


TLS support

Secure connection to NATS can be setup to be enforced by the server

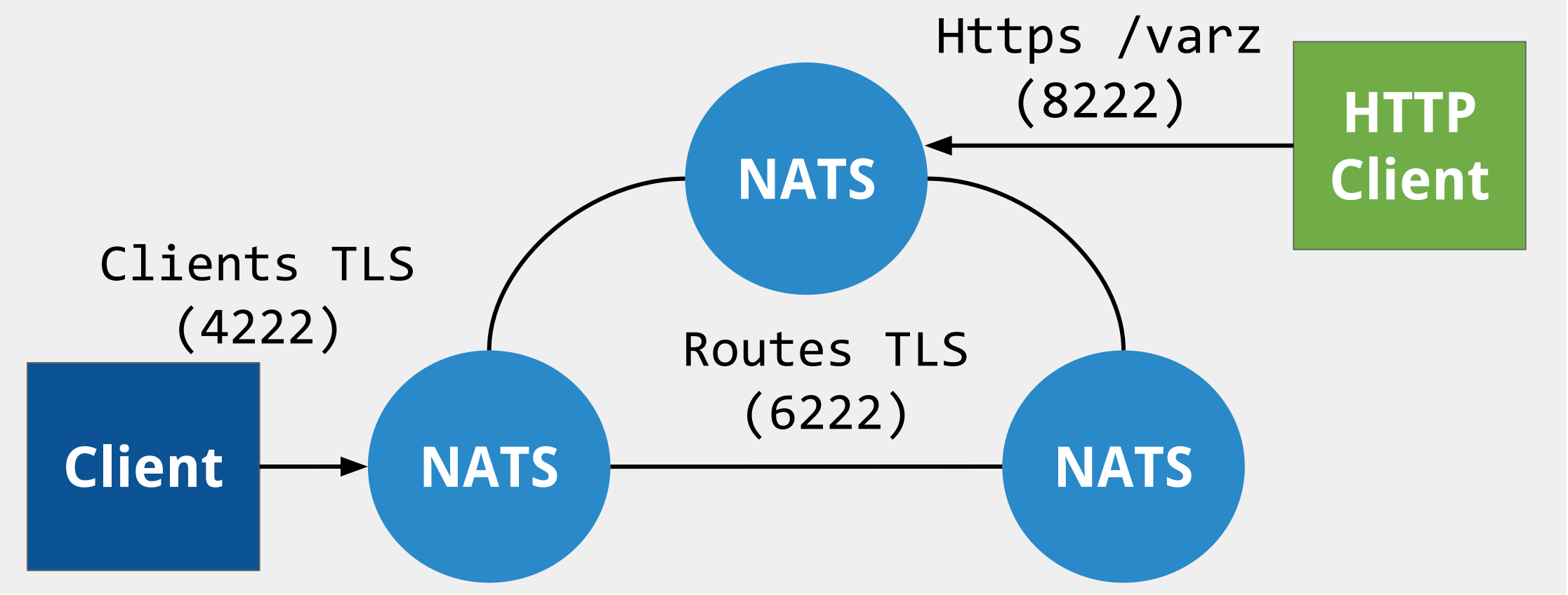
TLS support

Secure connection to NATS can be setup to be enforced by the server



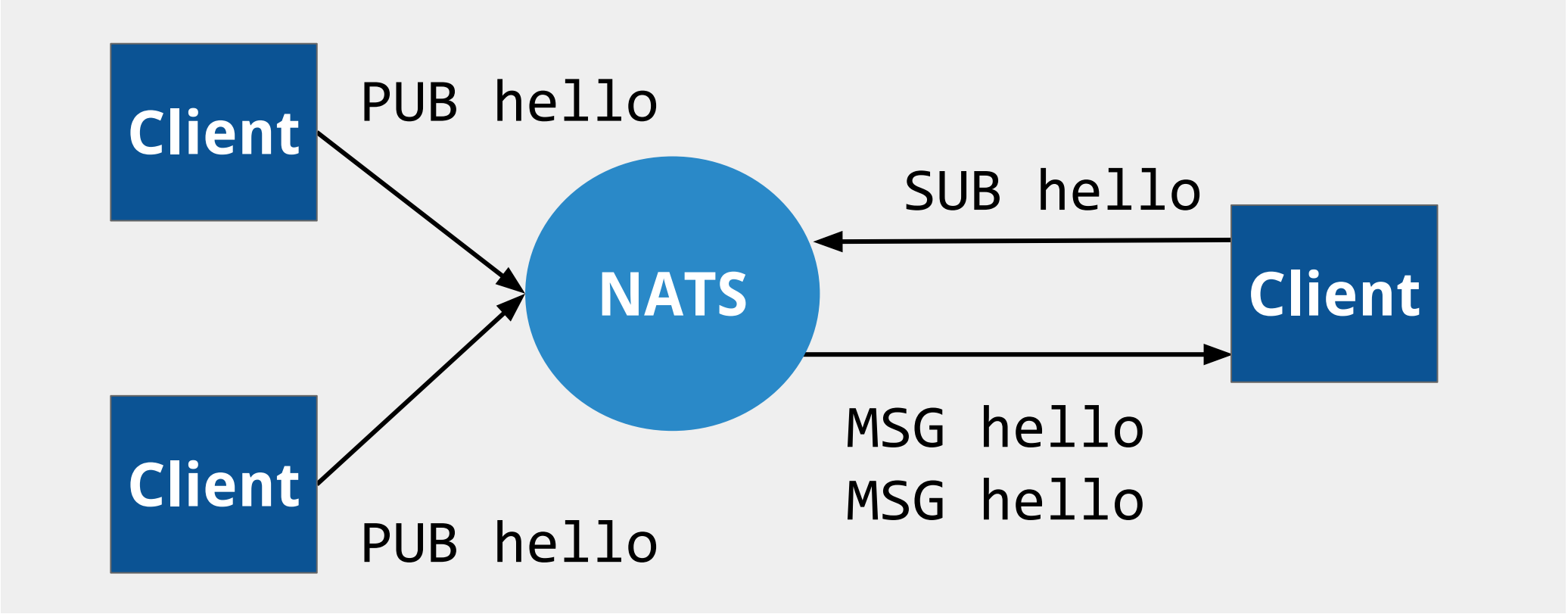
Secure NATS setup

For a secure setup, TLS can be enabled for clients, routes, and the monitoring endpoint as well



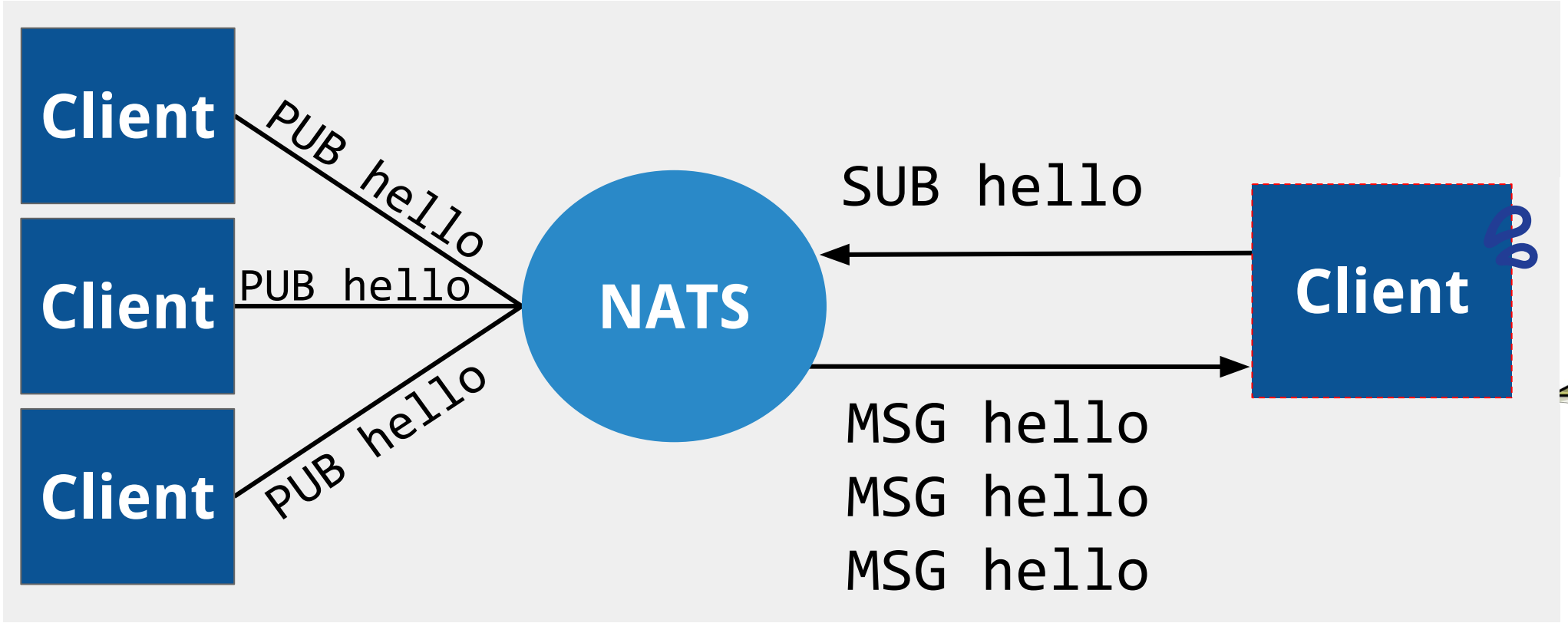
Resiliency

NATS was developed with resiliency in mind. Keeping balance when communicating is very important.



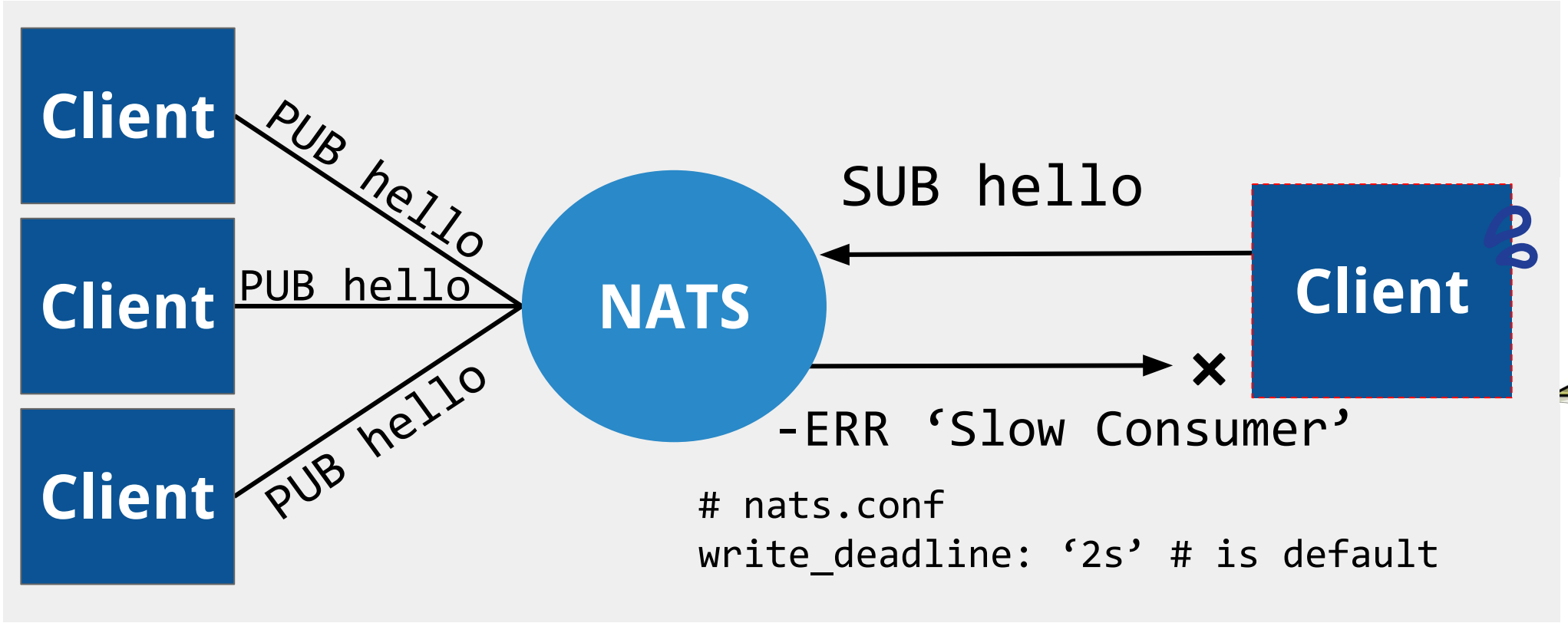
Resiliency: Slow Consumers

If a client is not able to consume messages the server is sending, then it becomes a *slow consumer*.



Resiliency: Slow Consumers

The slow consumer is determined when the client has not been able to consume new messages for more than 2 seconds.



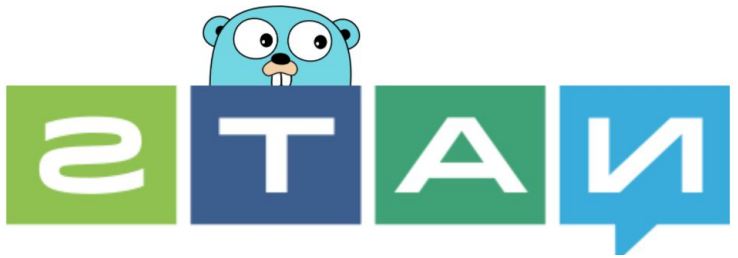
KubeCon | CloudNativeCon

Europe 2018

NATS Streaming

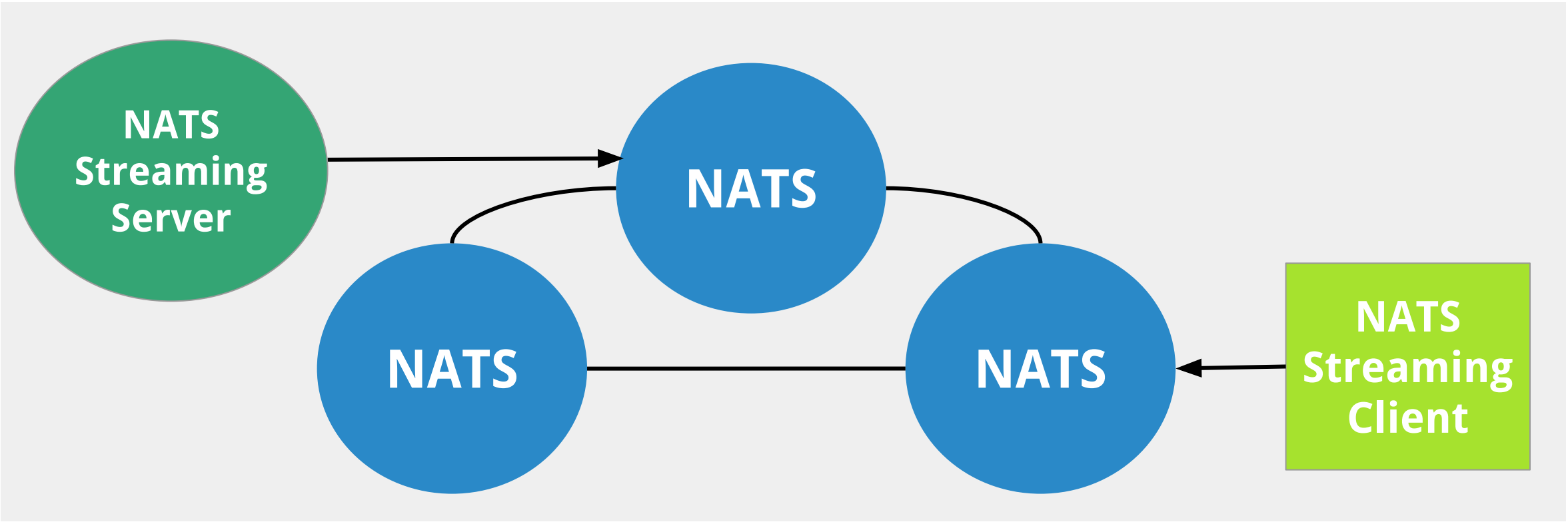
NATS Streaming

- Layer on top of NATS which enables at-least-once delivery
  - Codename: *STAN*
- It is the complete opposite of NATS
  - Stateful; it has built-in persistence of messages enabling message replay features
- Protocol is Request/Response based using Protobufs
- Binary name is *nats-streaming-server*
  - https://github.com/nats-io/nats-streaming-server



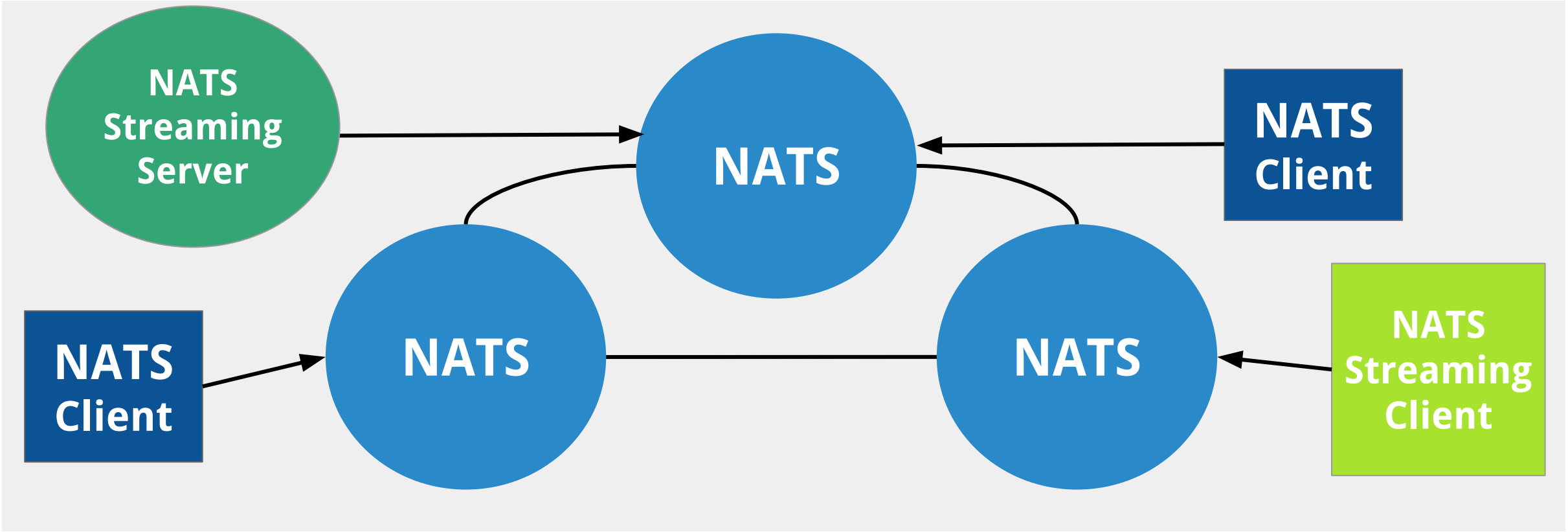
NATS ➡ NATS Streaming

NATS Streaming connects to a NATS server/cluster and uses it as it's transport to communicate with NATS Streaming clients.



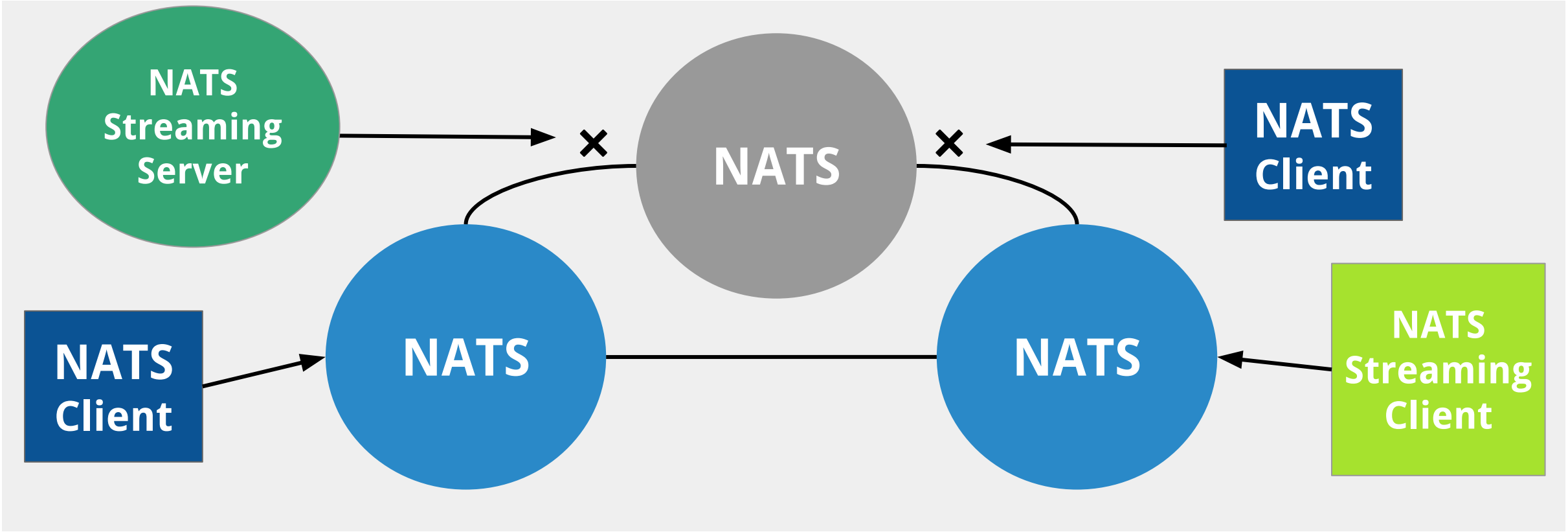
NATS ➡ NATS Streaming

NATS clients and NATS Streaming clients can all use the same cluster



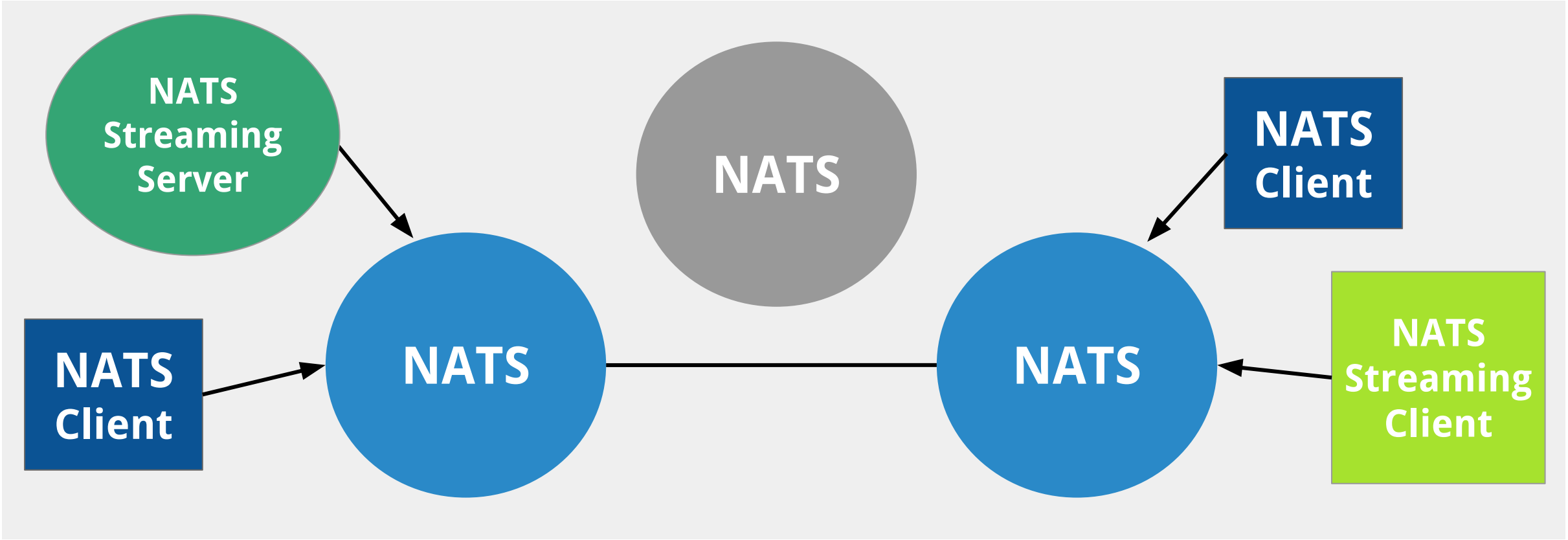
NATS ➡ NATS Streaming

Since NATS Streaming uses a NATS connection, it has the same reconnect features found in NATS.



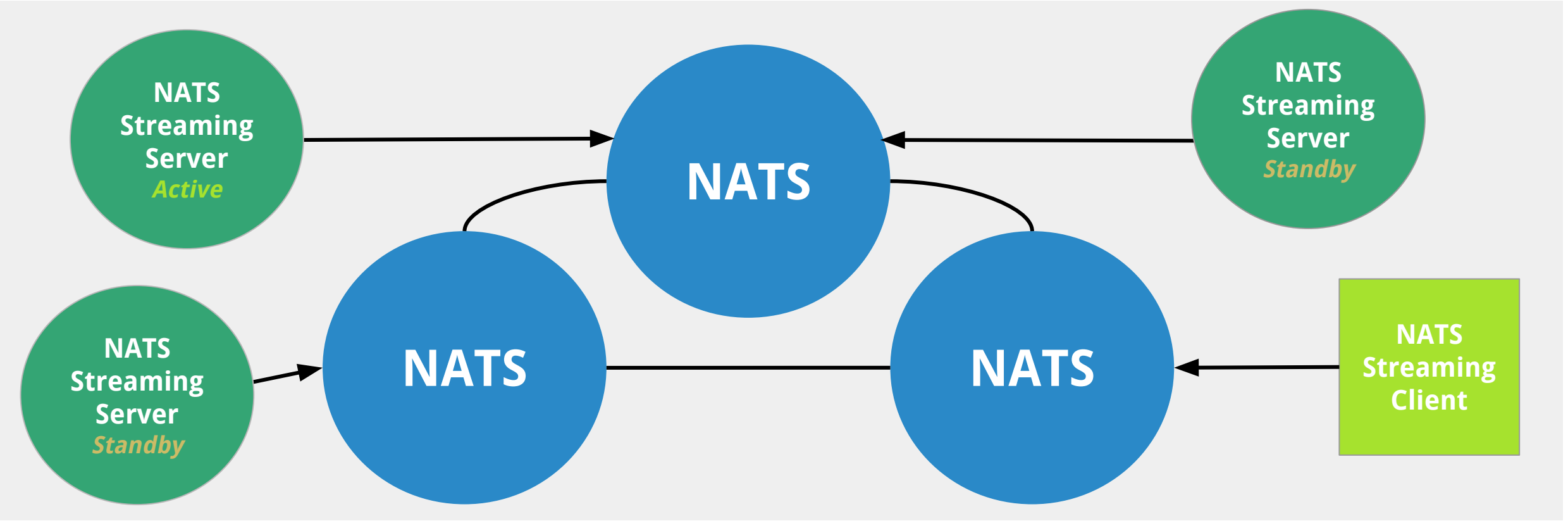
NATS ➡ NATS Streaming

Since NATS Streaming uses a NATS connection, it has the same reconnect features found in NATS.



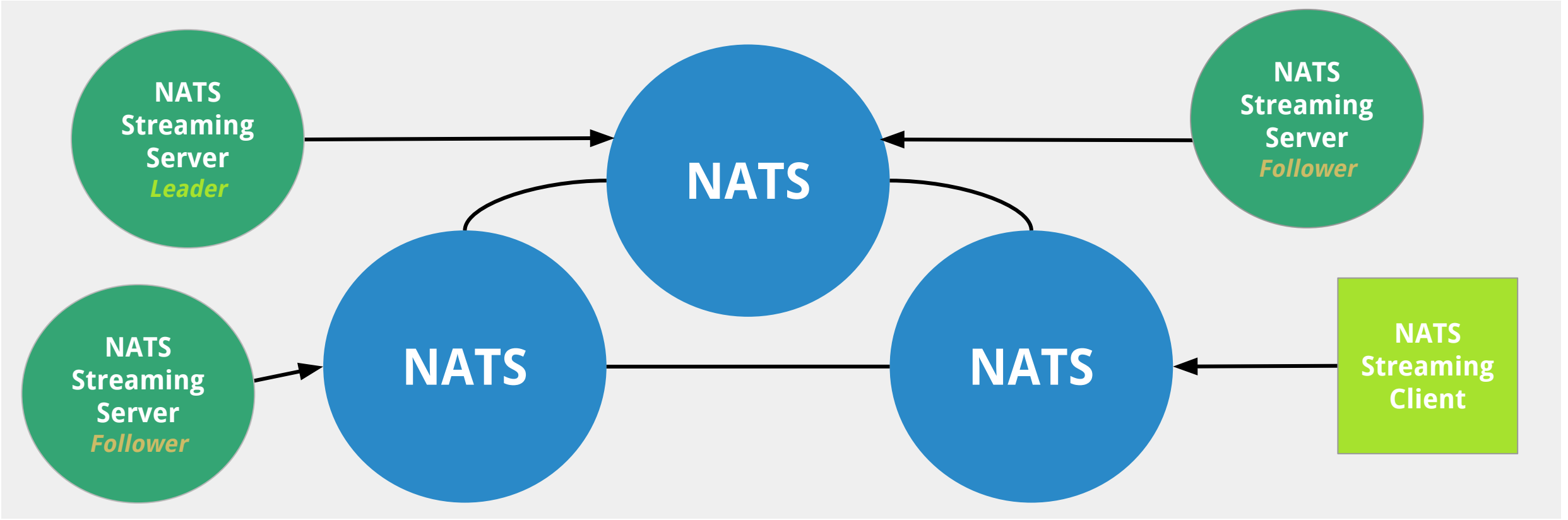
NATS Streaming Fault Tolerance

When using shared storage, the NATS Streaming server can be run in Fault Tolerance mode so that extra servers are in standby and became active only on active server failure.



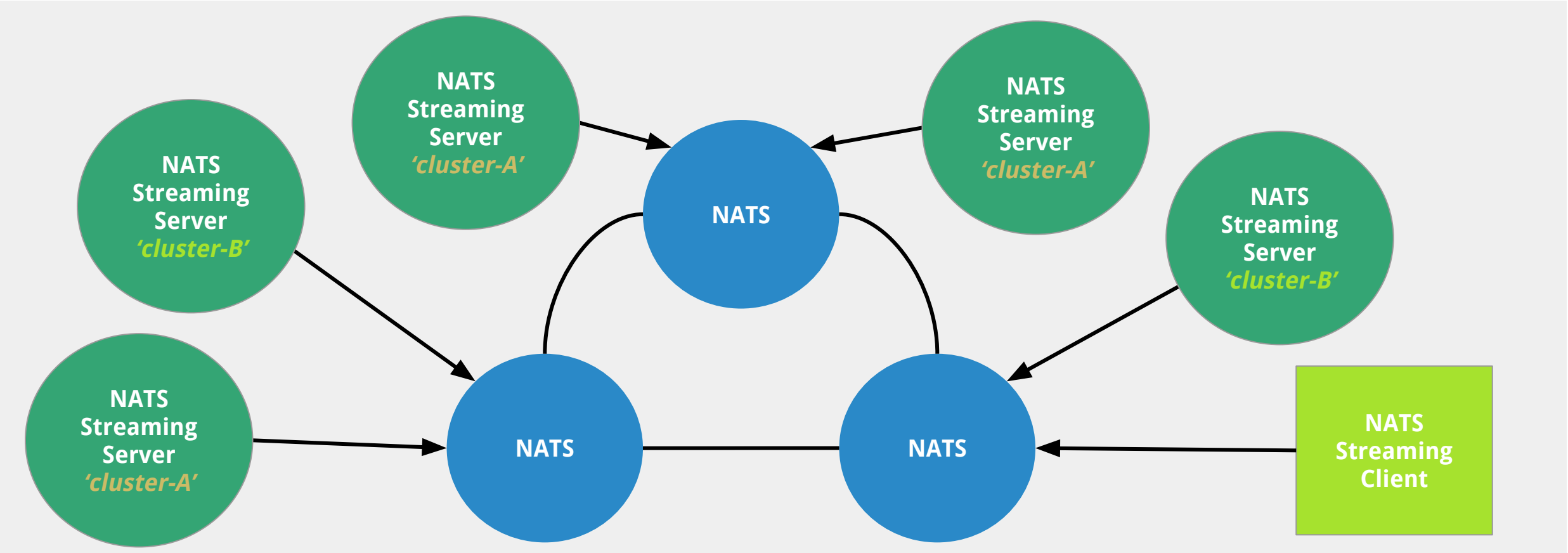
NATS Streaming Clustering

For replication and high availability, NATS Streaming can also use the Raft consensus algorithm.



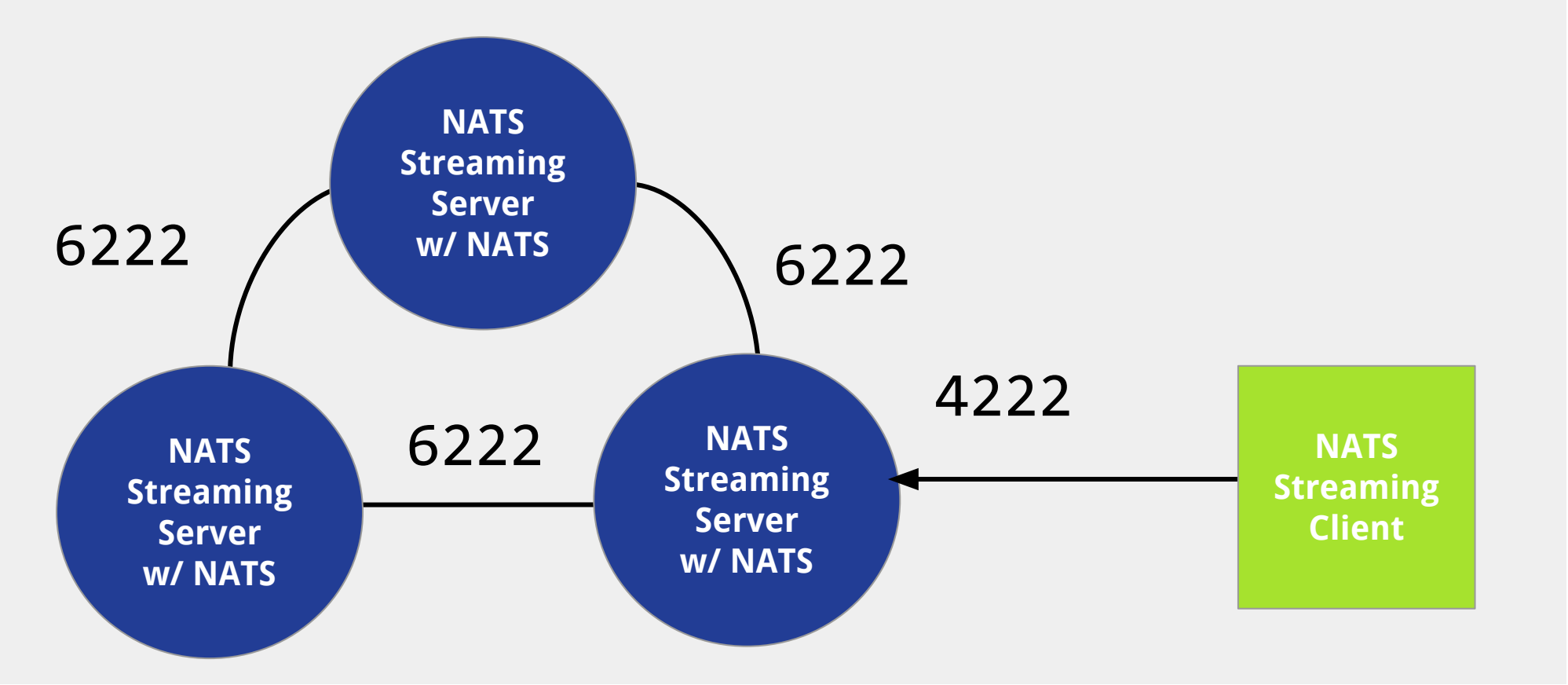
NATS Streaming Clustering

Multiple NATS Streaming clusters can coexist using the same NATS cluster by using different names



NATS Streaming using Embedded NATS Server

For convenience, the NATS Streaming server embeds a NATS server by default which can be used as its transport.



KubeCon

CloudNativeCon

Europe 2018

NATS Ecosystem

Official Docker Images

- The NATS Team maintains official Docker images
  - NATS: https://hub.docker.com/_/nats/
  - NATS Streaming: https://hub.docker.com/_/nats-streaming/

- They are very lightweight, using the FROM scratch to keep few layers

Prometheus NATS Exporter

KubeCon | CloudNativeCon Europe 2018

- Officially supported by NATS team
  - https://github.com/nats-io/prometheus-nats-exporter
- Can be used as a side car to monitor the /varz metrics from a single NATS server.

github.com/prometheus

Gloo + Envoy NATS Streaming

KubeCon | CloudNativeCon Europe 2018

- Developed by the solo-io team as part of the Gloo project; a function gateway
  - https://github.com/solo-io/gloo
- Gloo developed C++ clients for NATS & NATS Streaming
  - https://github.com/solo-io/envoy-nats-streaming
- Demo: https://www.youtube.com/watch?v=6mtvPrHfX1Q

solo.io

github.com/solo-io

nRPC: like gRPC but over NATS

KubeCon | CloudNativeCon Europe 2018

- Interesting project developed by the rapidloop team
  - https://github.com/rapidloop/nrpc

It can generate a Go client and server from the same .proto file that you'd use to generate gRPC clients and servers. The server is generated as a NATS MsgHandler.

Why NATS?

Doing RPC over NATS' request-response model has some advantages over a gRPC model:

- **Minimal service discovery**: The clients and servers only need to know the endpoints of a NATS cluster. The clients do not need to discover the endpoints of individual services they depend on.
- **Load balancing without load balancers**: Stateless microservices can be hosted redundantly and connected to the same NATS cluster. The incoming requests can then be random-routed among these using NATS queueing. There is no need to setup a (high availability) load balancer per microservice.

github.com/rapidloop

NATS Operator for Kubernetes

KubeCon | CloudNativeCon Europe 2018

- Original contribution from Paulo Pires
  - https://github.com/nats-io/nats-operator
- Creates a CRD in Kubernetes which could be used to help with the creation of NATS clusters
  - Current version: *nats.io/v1alpha2*
- Projects using it:
  - Kubeless project started using to add NATS support https://github.com/kubeless/kubeless/pull/689

github.com/pires

NATS Operator: Example


KubeCon | CloudNativeCon Europe 2018

```
apiVersion: "nats.io/v1alpha2"
kind: "NatsCluster"
metadata:
  name: "nats"
spec:
  # Number of nodes in the cluster
  size: 3
  version: "1.1.0"

  tls:
    # Certificates to secure the NATS client connections:
    serverSecret: "nats-clients-tls"

    # Certificates to secure the routes.
    routesSecret: "nats-routes-tls"
```

KubeCon

CloudNativeCon

Europe 2018

Demos

Thanks!

KubeCon CloudNativeCon Europe 2018

Be part of the NATS community :)
https://nats.io/community/