Introduction To NATS

Imre Nagi
@imrenagi
Google Developer Expert - Google Cloud Platform

Google Developers Experts 2019
Google Cloud Platform GDE

Agenda

- What is NATS?
- NATS Messaging Model
- Running NATS in Docker and Google Kubernetes Engine

What is NATS?

NATS is an open-source, cloud-native messaging system.

- Highly performant (fast)
- Extremely lightweight (small footprint)
- Always on and available (dial tone)

kafka

RabbitMQ

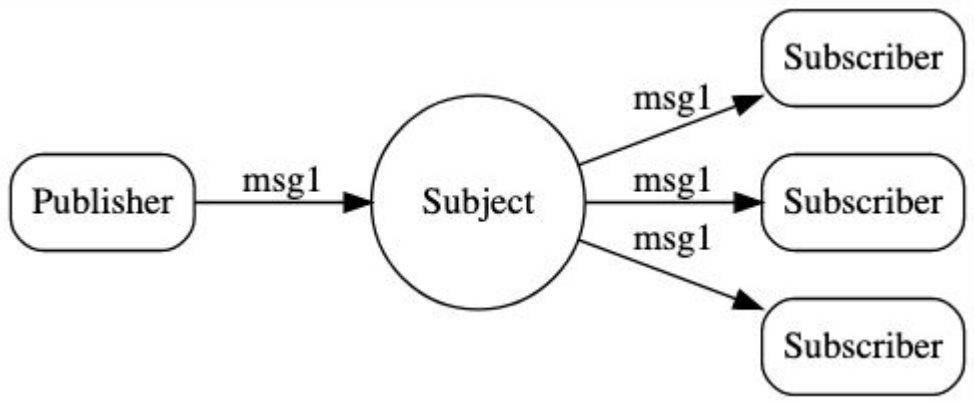
NATS

NATS Use Cases

- **High throughput message fanout** - a small number of data producers (publishers) need to frequently send data to a much larger group of consumers (subscribers), many of whom share common interest in specific data sets or categories (subjects)
- **Command and control (control plane)** - sending commands to running applications/devices and receiving status back from applications/devices, e.g. SCADA, satellite telemetry, IOT.
- **Load balancing** - your application(s) produces a large volume of work items or requests and you would like to use a dynamically scalable pool of worker application instances to ensure you’re meeting SLAs or other performance targets.
- **Fault tolerance** - your application needs to be highly resilient to network or other outages that may be beyond your control, and you need the underlying application data communication to seamlessly recover from connectivity outages

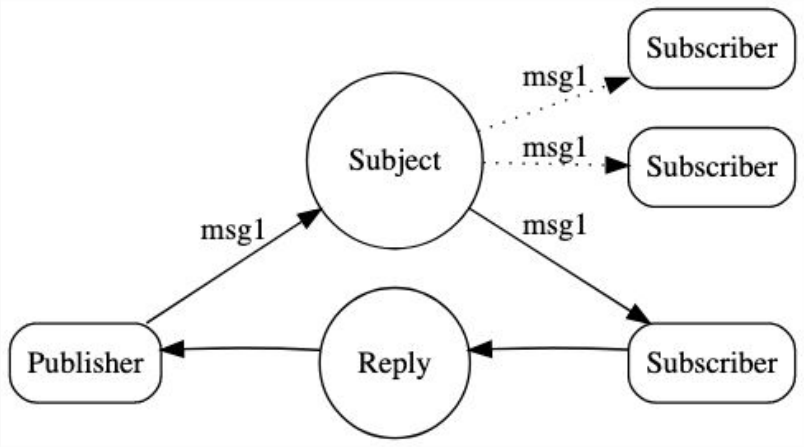
NATS Messaging Model

Publish Subscribe



A publisher sends a message on a subject and any active subscriber listening on that subject receives the message

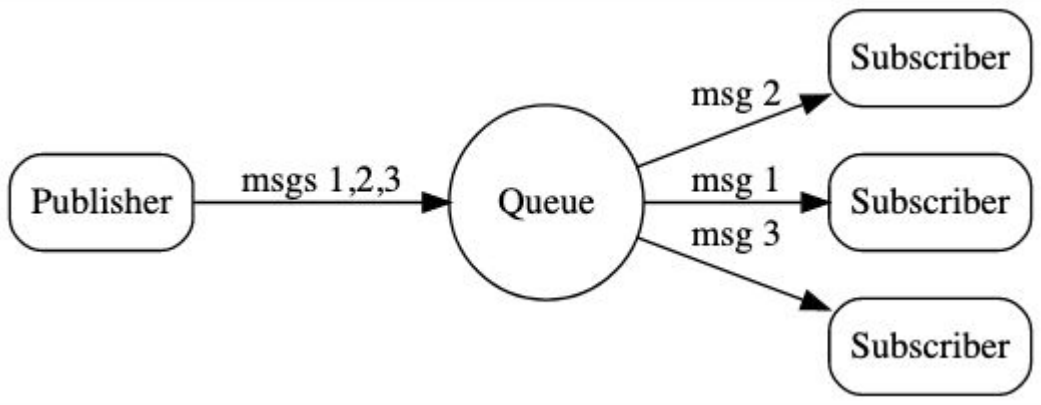
Request Reply



Point to Point: The Fastest to reply

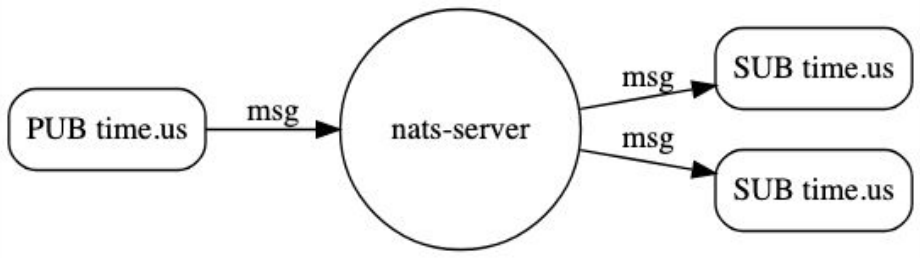
One to Many: Needs some replies

Queue Subscribers



Using queue subscribers will load balance message delivery across a group of subscribers which can be used to provide application fault tolerance and scale workload processing.

Subject Based Messaging



Subject Hierarchies (separated by **.**)

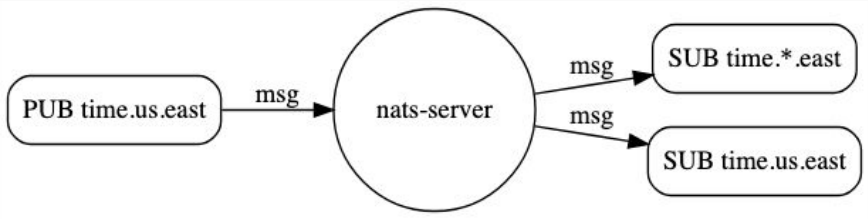
time.us
time.us.east
time.us.east.atlanta
time.eu.east
time.eu.east.warsaw

Subject Wildcard

Matching a Single Token with ***asterix***

time.*.east

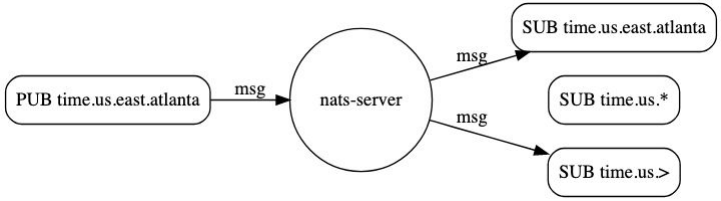
- **time.us.east**
- **time.eu.east**



Matching Multiple Token with **>**

time.us.>

- **time.us.east**
- **time.us.east.atlanta**



NATS Programming Model

Running NATS in DOCKER

```
docker run -p 4222:4222 -p 8222:8222 -p
6222:6222 -ti nats:latest
```

```
helm install --name nats-demo-release stable/nats
```

Connecting To NATS

```go
nc, err := nats.Connect("demo.nats.io", nats.MaxReconnects(10))
if err != nil {
    log.Fatal(err)
}
 defer nc.Close()
```

Connecting To NATS

```go
nc, err := nats.Connect("localhost", nats.UserInfo("myname", "password"))
if err != nil {
    log.Fatal(err)
}
defer nc.Close()
```

Publishing Message To NATS

```go
nc, err := nats.Connect("nats://nats-demo:4222")
if err != nil {
    log.Fatal(err)
}
defer nc.Close()

if err := nc.Publish("time.us.east", []byte("All is Well")); err != nil {
    log.Fatal(err)
}
nc.Flush()
```

Publishing Message With Optional Reply-To To NATS

```go
// Create a unique subject name
uniqueReplyTo := nats.NewInbox()

// Listen for a single response
sub, err := nc.SubscribeSync(uniqueReplyTo)
if err != nil {
    log.Fatal(err)
}

// Send the request
if err := nc.PublishRequest("time", uniqueReplyTo, nil); err != nil {
    log.Fatal(err)
}

// Read the reply
msg, err := sub.NextMsg(time.Second)
if err != nil {
    log.Fatal(err)
}
```

Asynchronous Subscription In NATS

```go
// Subscribe
if _, err := nc.Subscribe("updates", func(m *nats.Msg) {
    fmt.Println(m.Data)
}); err != nil {
    log.Fatal(err)
}
```

```go
if _, err := nc.Subscribe("time.*.east", func(m *nats.Msg) {
    log.Printf("%s: %s", m.Subject, m.Data)
}); err != nil {
    log.Fatal(err)
}
```

Asynchronous Subscription In NATS

```go
if _, err := nc.QueueSubscribe("updates", "worker", func(m *nats.Msg) {
    fmt.Println(m.Data)
}); err != nil {
    log.Fatal(err)
}
```

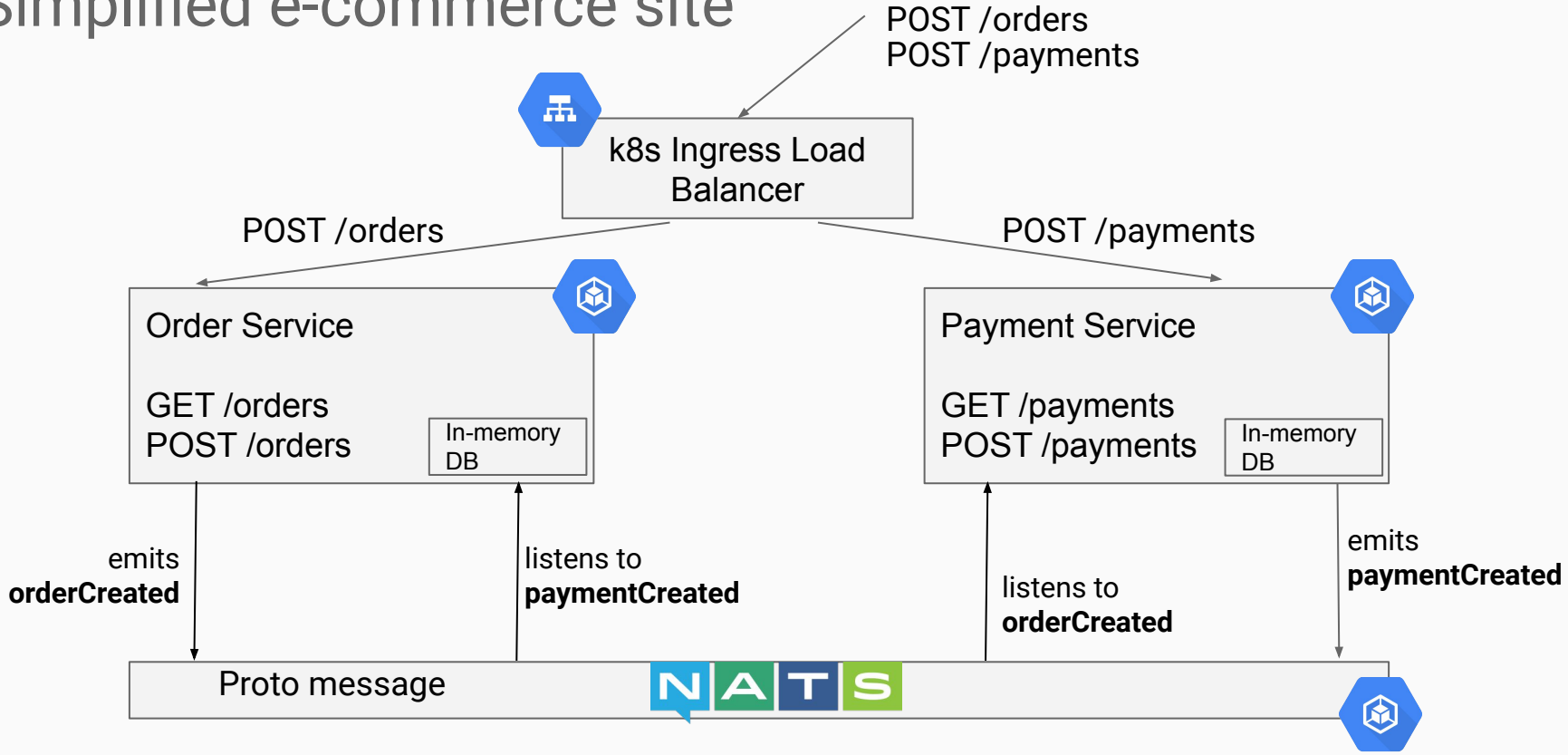
Request - Reply In NATS

```go
// Send the request
msg, err := nc.Request("time", nil, time.Second)
if err != nil {
    log.Fatal(err)
}

// Use the response
log.Printf("Reply: %s", msg.Data)
```

NATS for Microservices

Simplified e-commerce site



Thank you! Find me at

imrenagi/microservice-demo

Imre Nagi

https://medium.com/@imrenagi

Ngobrolin Startup & Teknologi