K8S async deploy with Nats

whoami

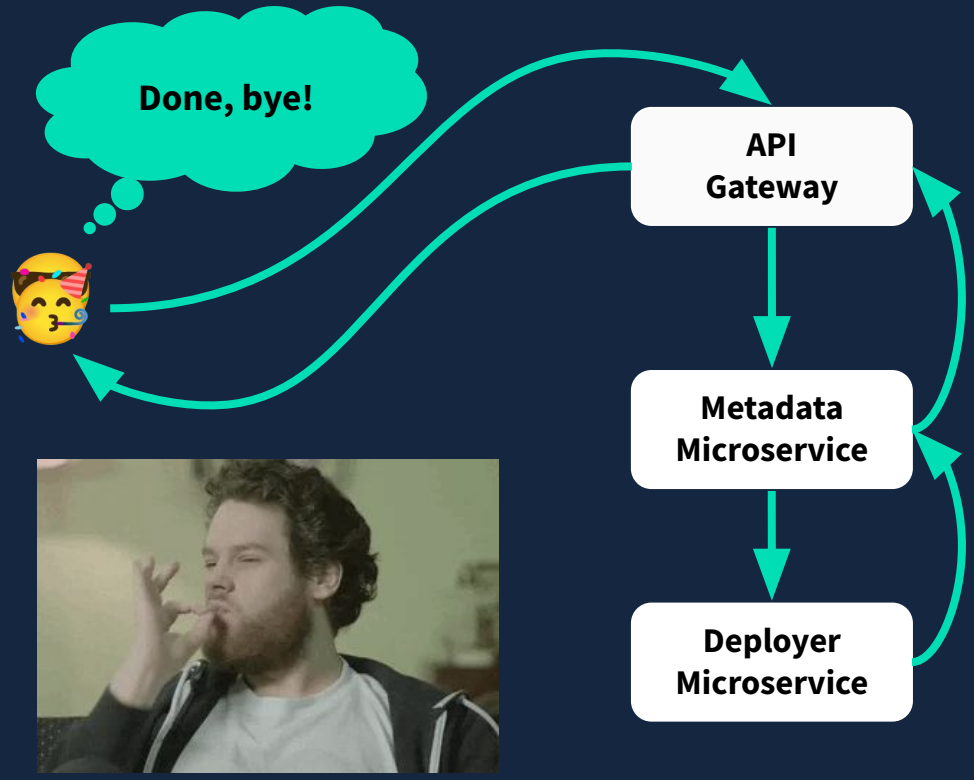
Francesco Donzello

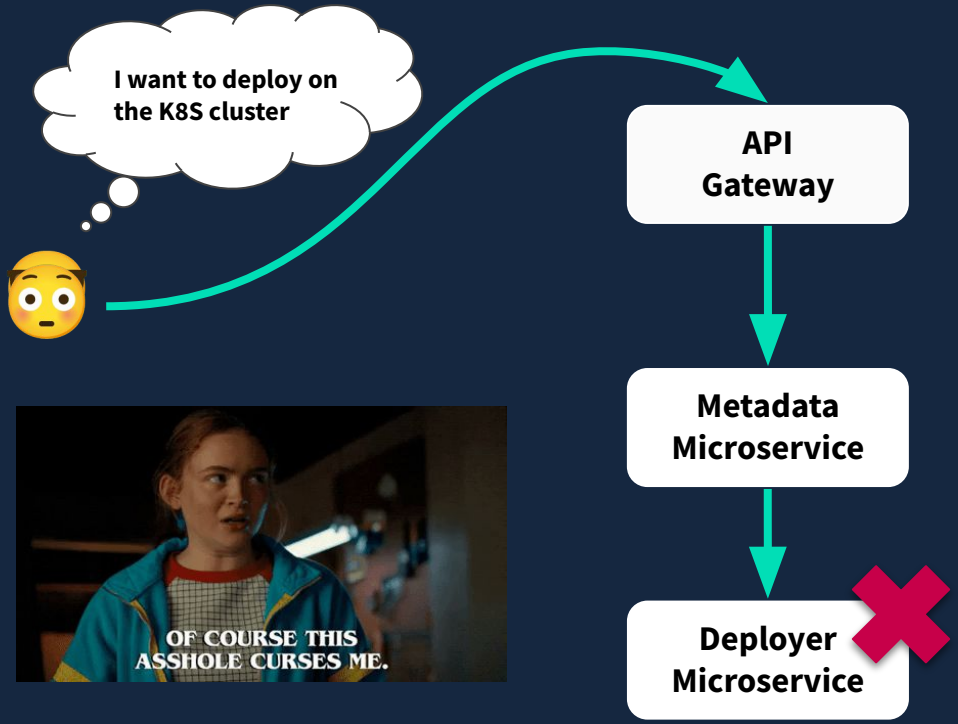
2014-2015	Android Developer @ immobiliare.it
2020-2020	CTO @ BadgeBox
2015–	Many Things @ Fraway

Trainer for:

Unicredit, Swiss, Politecnico di Milano, Engineering, ARHS Development, Ericsson, Alten, Sisal

The problem


Done, bye!
API Gateway
Metadata Microservice
Deployer Microservice

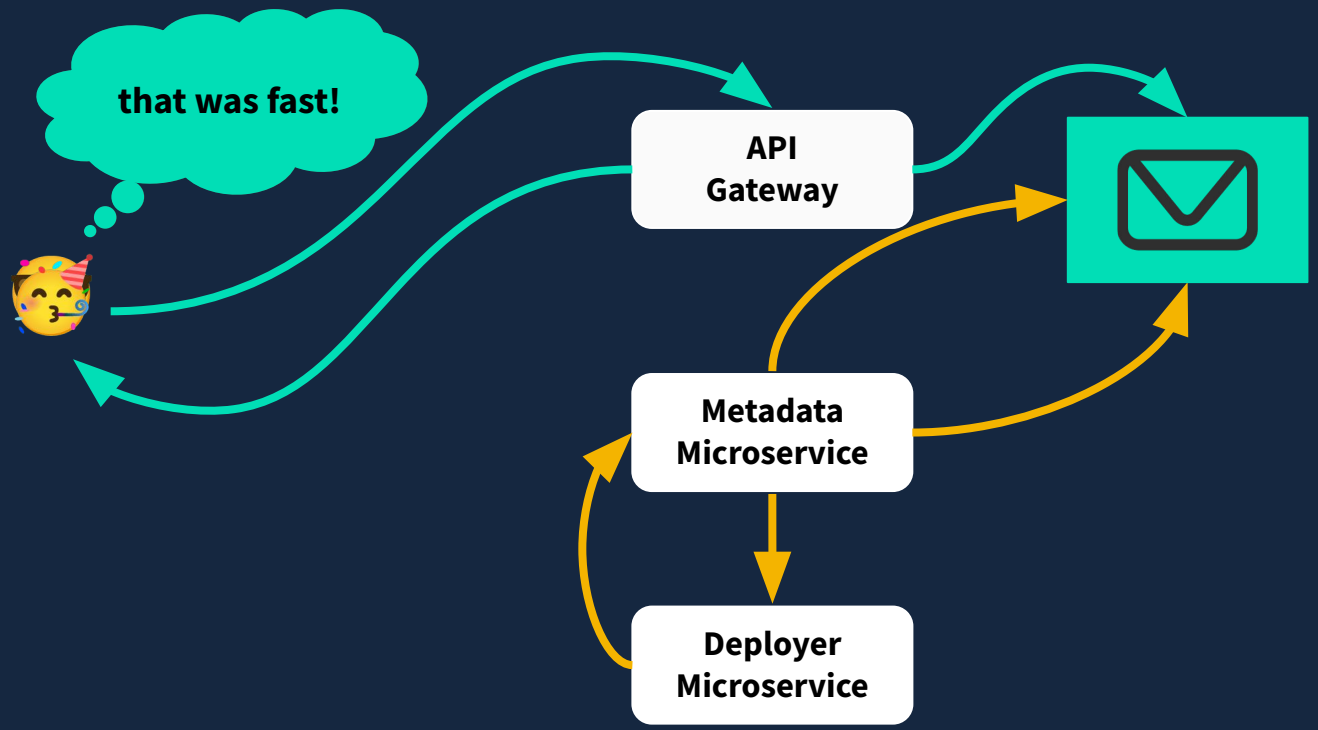


Francesco Donzello

Don’t be nuts

- synchronous communication leads to:
  - tasks never completed
  - bad performances
  - scalability limitations
  - reliability issues

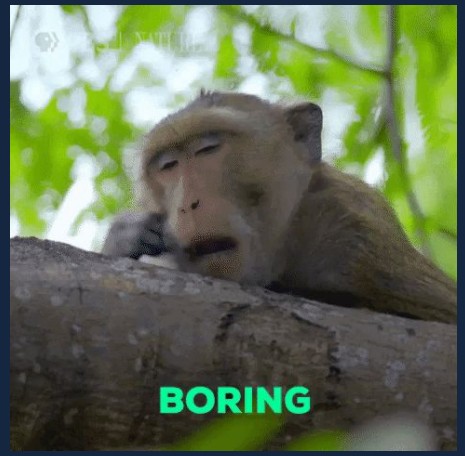
Let’s go async


that was fast!
API Gateway
Metadata Microservice
Deployer Microservice

Messaging options

- **KAFKA**
  - java based
  - most used one
  - since 2011
- **RabbitMQ**
  - erlang
  - since 2007
- **Redis**
  - made in Sicily
  - since 2009
- **Amazon SQS**
  - AWS Cloud
- **Google Cloud Pub/Sub**
  - GCP Cloud

Anything fresh?



Introducing Nats

- GO based and **open source**
- **cloud native** and part of the CNCF
- designed for being **performant, reliable** and **secure**
- **simple** by design
- great **DX**

Use cases

- **multi cloud** communication
- **IoT**
- **Big Data** thanks to JetStream
- **event sourcing**

Momentum

- **100M+** docker hub pulls for nats server
- **50M+** docker hub pulls for nats streaming server
- **67** releases
- used by Paypal, Tinder, Ericsson, Siemens and many other
- **48** supported client languages

NATS Core

- **single binary** (nats-server)
- **~12MB** docker image
- messaging patterns:
  - Publish-Subscribe
  - Request-Reply
  - Queue Grous
- powerful subjects
  - full and partial **wildcards**
- drain APIs for **graceful shutdown**
- **selfish optimization**
- **full mesh** clustering of servers
- **self-healing** connections

JetStream

- **built-in** into NATS Core (nats-server -js)
- allows messages **persistence**
- supports **message deduplication**
- introduces **streaming** where the stream is a regular subject
- streams can also be **mirrored**
- a **Source** can copy data from multiple streams
- **data encryption**
- **push (fastest)** and **pull (batch)** message consumption
- cluster sizing is recommended to be between 3 and 5 servers

Delivery Modes

- **at most once** with Nats Core
  - no delivery guarantee
  - if no one is listening, messages are lost
- **at least once** with JetStream
  - delivery is guaranteed
  - you may get duplicated messages
- **exactly once** with JetStream
  - combining Message Deduplication and double acks
  - less performant

Security

- encryption with TLS
- client connections may authenticate with
  - Tokens
  - JWTs
  - TLS certificates
  - username/password credentials
  - accounts for multi-tenancy
  - **subject based** policies

Is it easy?

Quick usage

- connect to the Server
- publish a message
- consume a message

```
nats.Connect("nats://127.0.0.1:4222")
```

nats-server can be run as a:

- Docker container
- Kubernetes Workload
- local process
- Cluster!

```go
nc.Publish("apps.internal.billing-ms.updated", []byte("v1.0.1"))
```

- token based subject
- the “.” creates a hierarchy
- message payload
- JSON encoding can be configured

```go
nc.Subscribe("apps.internal.billing-ms.updated", func(m *nats.Msg) {
    // TODO: payload in m.Data
})
```

- **subject of interest (exact match)**

```go
nc.Subscribe("apps.internal.*.updated", func(m *nats.Msg) {
    // TODO: payload in m.Data
})
```

- **“*” matches just one token**
- **hierarchy is taken into account**
- **can appear more than once**

```go
nc.Subscribe("apps.internal.>", func(m *nats.Msg) {
    // TODO: payload in m.Data
})
```

- **“>” matches the remaining tokens in the subject hierarchy**
- **can only be used at the end of the subject**

```go
nc.Subscribe("*.internal.>", func(m *nats.Msg) {
    // TODO: payload in m.Data
})
```

- **you can also mix!**

Demo Time!

Deploy on Kubernetes with Nats

Francesco Donzello

What’s next

- give it a try
- keep learning on the official docs (very well written)
- Ask questions!

Thank you

Stay in touch!

@frabird
@francescodonzello
@Francesco Donzello
francesco@fraway.io