safespring

Using NATS and Huma to Enhance Open Source Infrastructure

Empowering B2B and European Research Communities with Self-Service Access

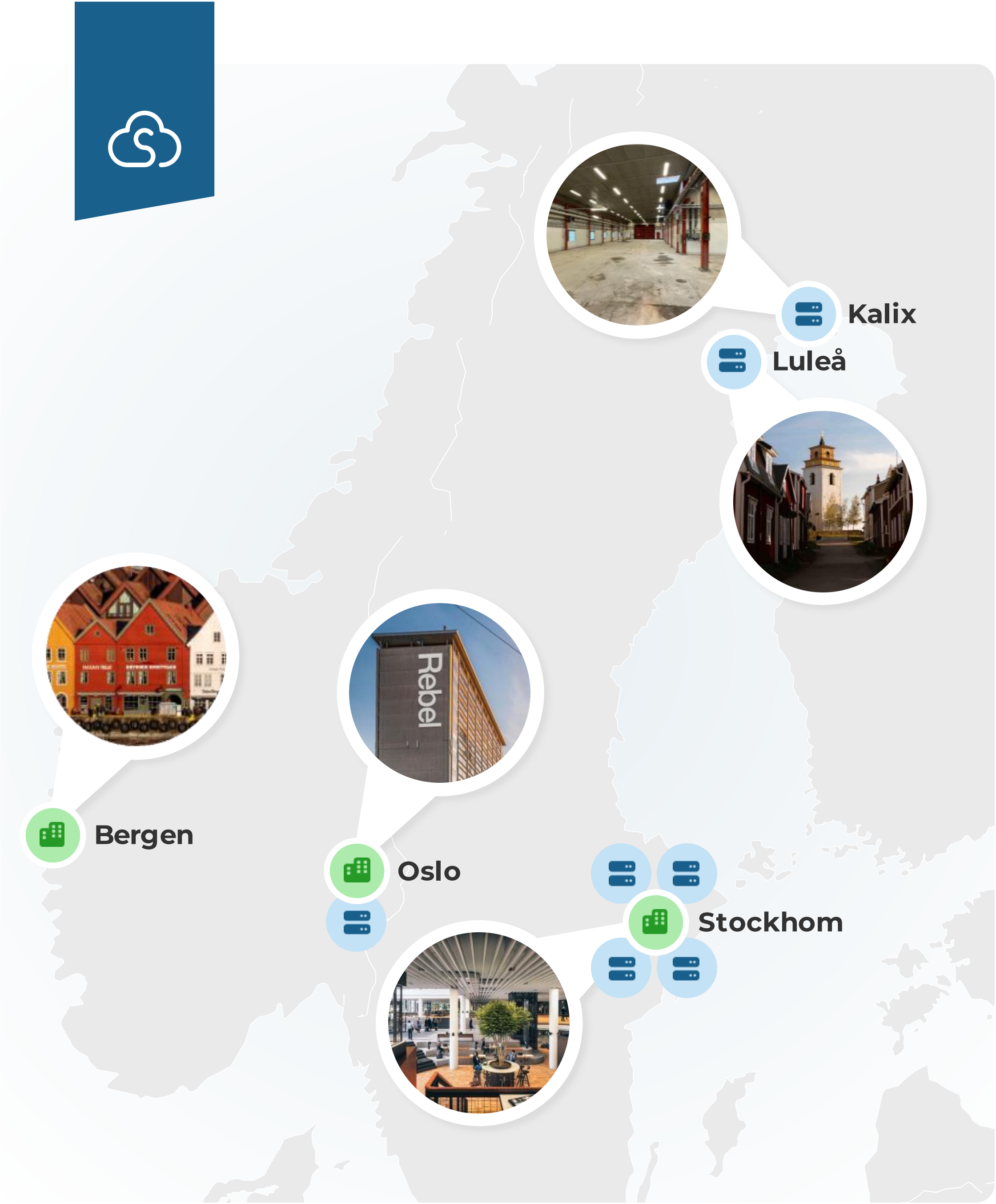
NAME
Jon Ander Novella de Miguel

E-MAIL
jon.novella@safespring.com

DATE
2024-05-07

Agenda

1. Safespring's mission
2. Goal of the self-service API
3. Huma framework
4. NATS microservices
5. Architecture
6. Messaging patterns
7. Challenges



Safespring

Scandinavian Company offering nordic IaaS and PaaS.

Compliance

Regulatory compliance for GDPR. No data transfer to 3:rd countries

Datacenter-security

Secure data center with 100% renewable energy and focus on sustainability

Open Standard

flexibility, control over data, interoperability, cost savings, data portability

Safespring's Vision

The platform of choice for European Cloud Computing

Safespring's mission

Through expertise, modern infrastructure services, and flexibility, Safespring is the foundation of digital development. We enable rapid innovation through reliable and scalable services without lock-in effects.

- We strive for the position to be the leading provider of innovative and secure cloud solutions.
- Our mission to deliver a robust and flexible public and private cloud platform.
- We are committed to ensuring data sovereignty, security, and privacy, whilst promoting cost-efficiency and scalability through open standards and cutting-edge technology.

Goals of the selfservice API

Distributed management of customers

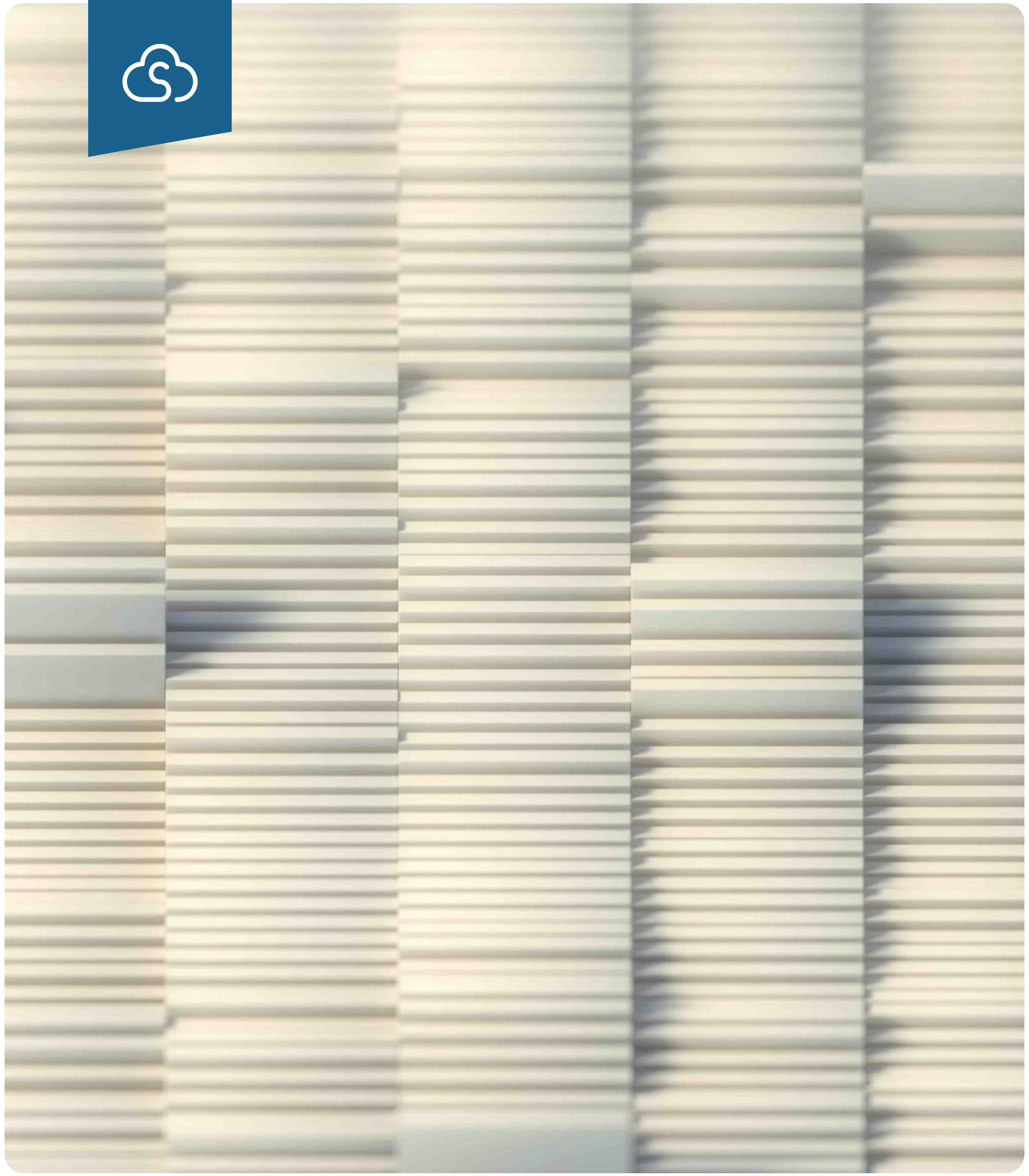
- Easier on-boarding for customers across multiple data centres
- Cut down support management costs when existing customers grow

Infrastructure federation

- Federation across businesses and research communities.
- The European Open Science Cloud project (EOSC).

Controlled provisioning of resources

- B2B and large customers
- Total quota and service specific quotas
- Code driven customer resource definitions
- Self-service users



Huma

Back your HTTP API by OpenAPI 3
and JSON Schema

Huma

Back your HTTP API by OpenAPI 3 and JSON Schema

- Generic HTTP handler signature
  - *Operations based on generic target function signature*
  - *Composable HTTP handlers*
  - *Input and output must be structs*
  - *Open API spec generation*
- Annotated Go types for I/O models
  - *JSON schema generation from Go types*
  - *Static typing for parameters, bodies, headers, etc.*
  - *Documentation generation using Stoplight Elements*
- Compatibility
  - *Huma implements the http.Handler interface*
  - *Uses standard context.Context*
  - *Standard streaming support via io.Reader and io.Writer interfaces*
  - *Compatible with most popular routers*

Huma: input/output models and operations

Annotated models

```
type ProjectInput struct {
  Body    Project `json:"body"`
  RawBody []byte  `json:"-"`
}
type ProjectDetInput struct {
  Name     string   `path:"name" json:"name,omitempty"`
  Services []string `query:"services" json:"services"`
}
type ProjectOutput struct {
  Body Project `json:"project"`
}
type User struct {
  Username     string `json:"username" minLength:"1"`
  Email        string `json:"email" format:"email"`
}
```

Generic Operation handler signature

```
func[I,O any](ctx.Context,*I)(*O, error))
```

Huma: JSON schemas and API spec

JSON schema

```
user:
 additionalProperties: false
  additionalProperties: false
  properties:
   email:
    description: Email address
    format: email
    minLength: 1
    type: string
   username:
    description: Username
    minLength: 1
    type: string
  required:
   - username
   - email
  type: object
```

OpenAPI specification

```
/users:
 get:
  operationId: listUsers
  responses:
   '200':
   content:
    application/json:
     schema:
      description: List of users
      items:
       $ref: '#/User'
      type: array
(...)
```

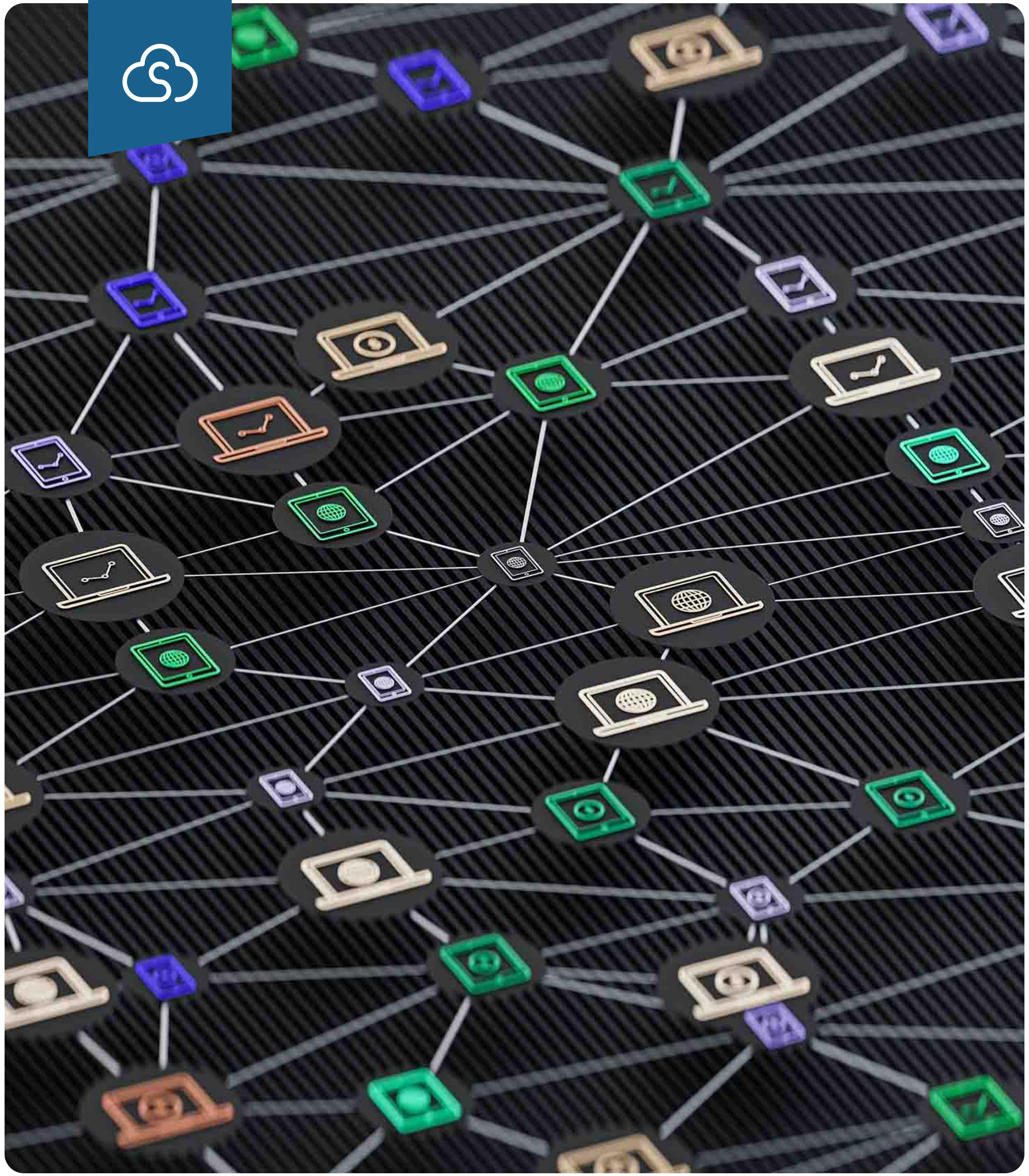
Huma

Back your HTTP API by OpenAPI 3 and JSON Schema

We want something that can discover multiple services seamlessly and scales to many data-centers

Limitations of HTTP:

- DNS/hostnames/IP based discovery
- Use of pull based request/reply semantics
- HTTP calls generally act on location-dependent backends



NATS

Connect Your Services
with High-speed Messaging

Core NATS

- Fire and forget fast message publishing
- Flexible subject based addressing using wildcards
- Accepts any type of payload
- Patterns:
  - *Request and reply*
  - *Publish and subscribe*
  - *Fan in and fan out*
  - *Scatter and gather*
  - *Load balancing using queue groups*

NATS micro

Why?

- Discoverable, observable and nomadic
- Dead simple load balancing
- Observe:
  - *Service instances*
  - *Subject names per svc*
  - *Total requests / errors per svc*

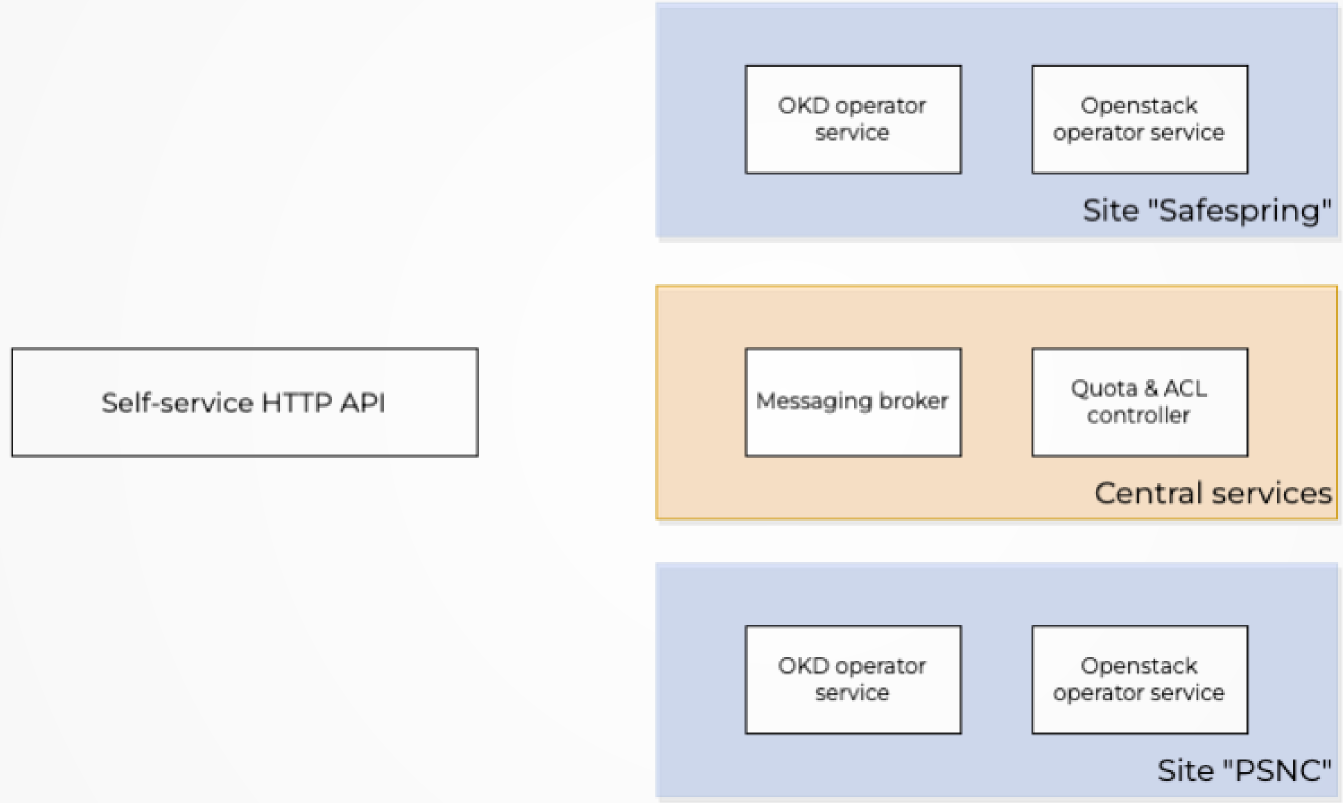
```
nats micro list / info svc
nats micro stats svc
```

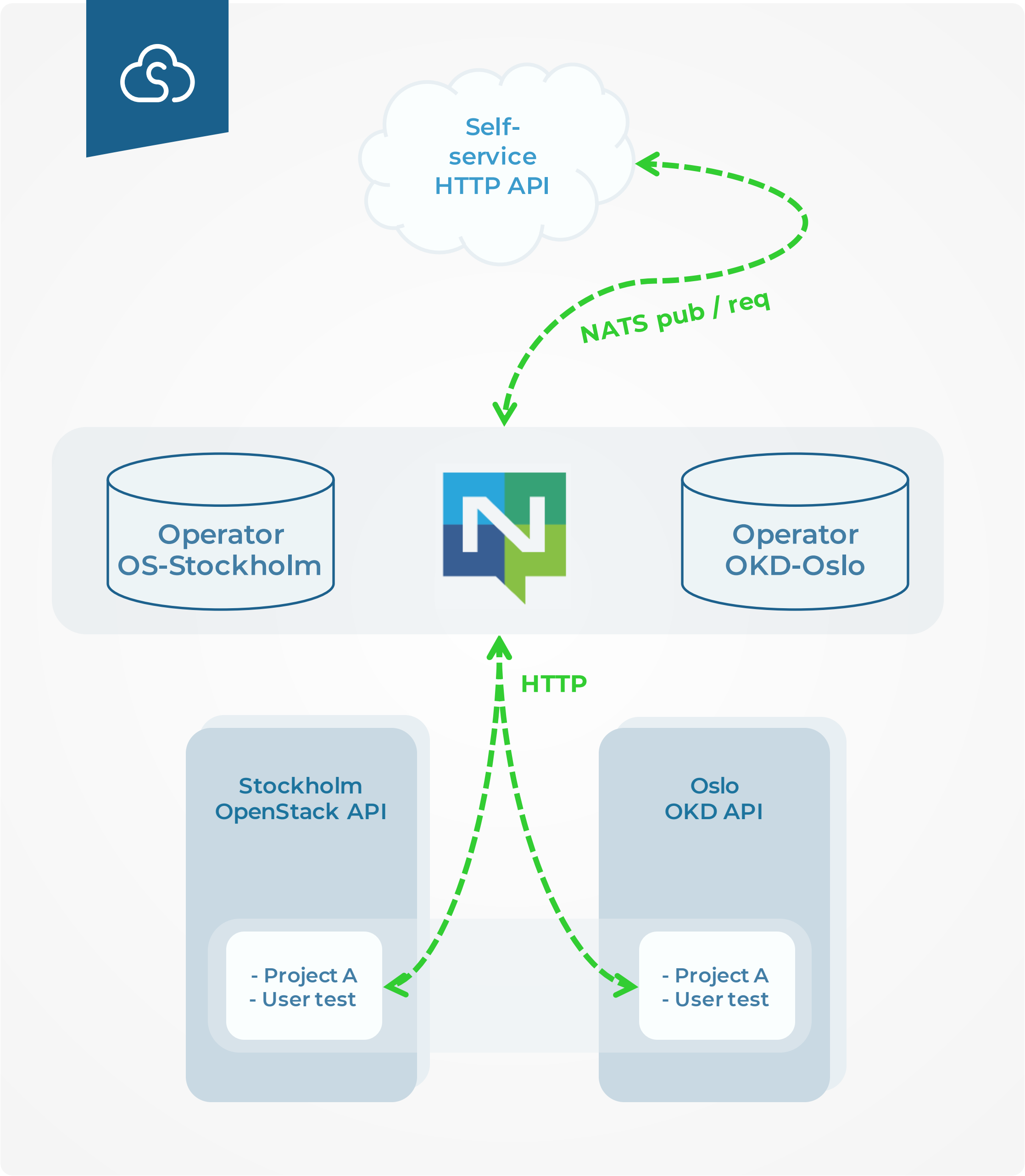
Service definition

type **HandlerFunc** func(Request)

- Service = set of endpoints or groups
- Group = common subject prefix used by all endpoints
- Endpoints = subject subscription + function handler
- Messaging patterns based on endpoint or service level queue groups

Architecture

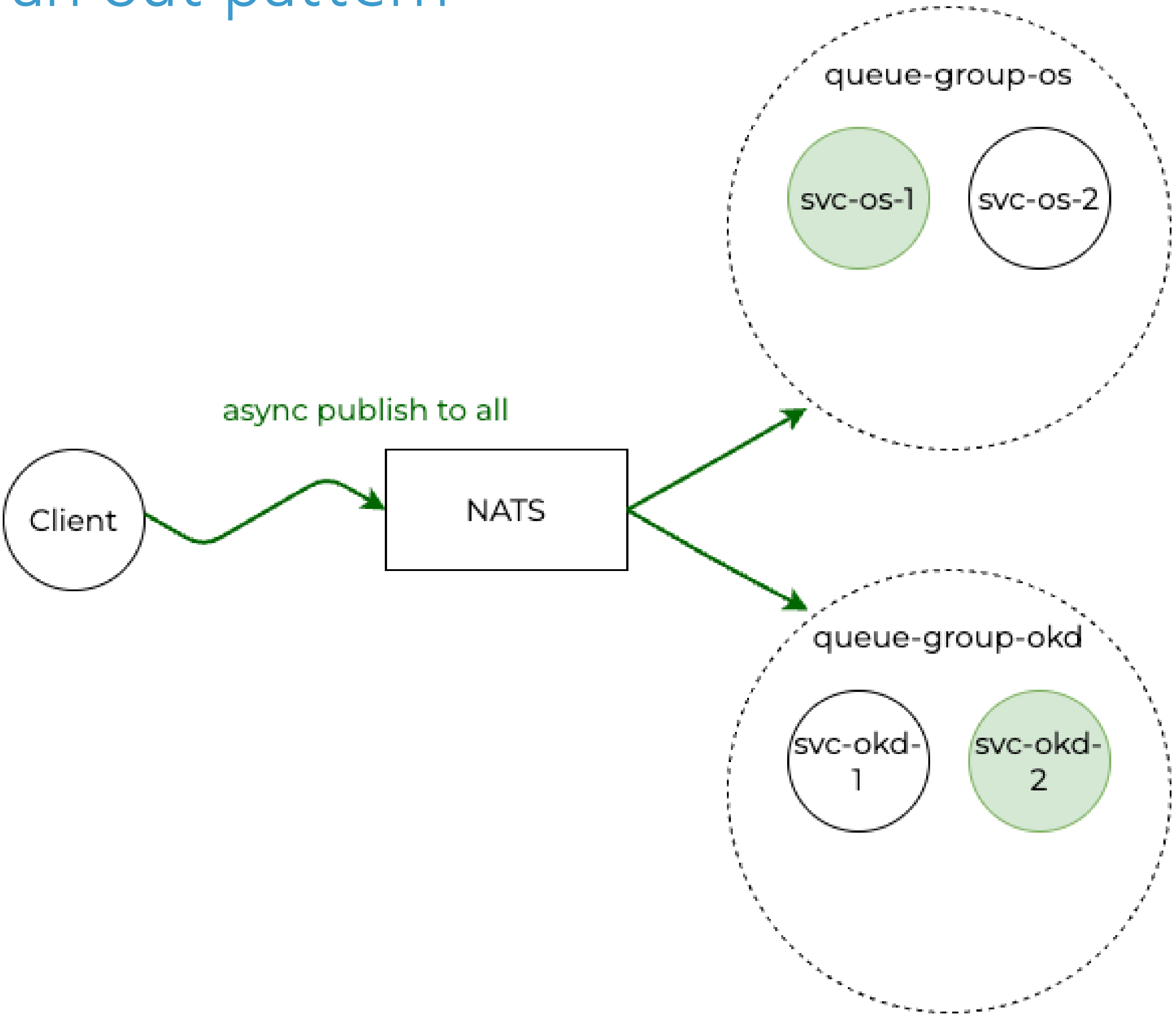




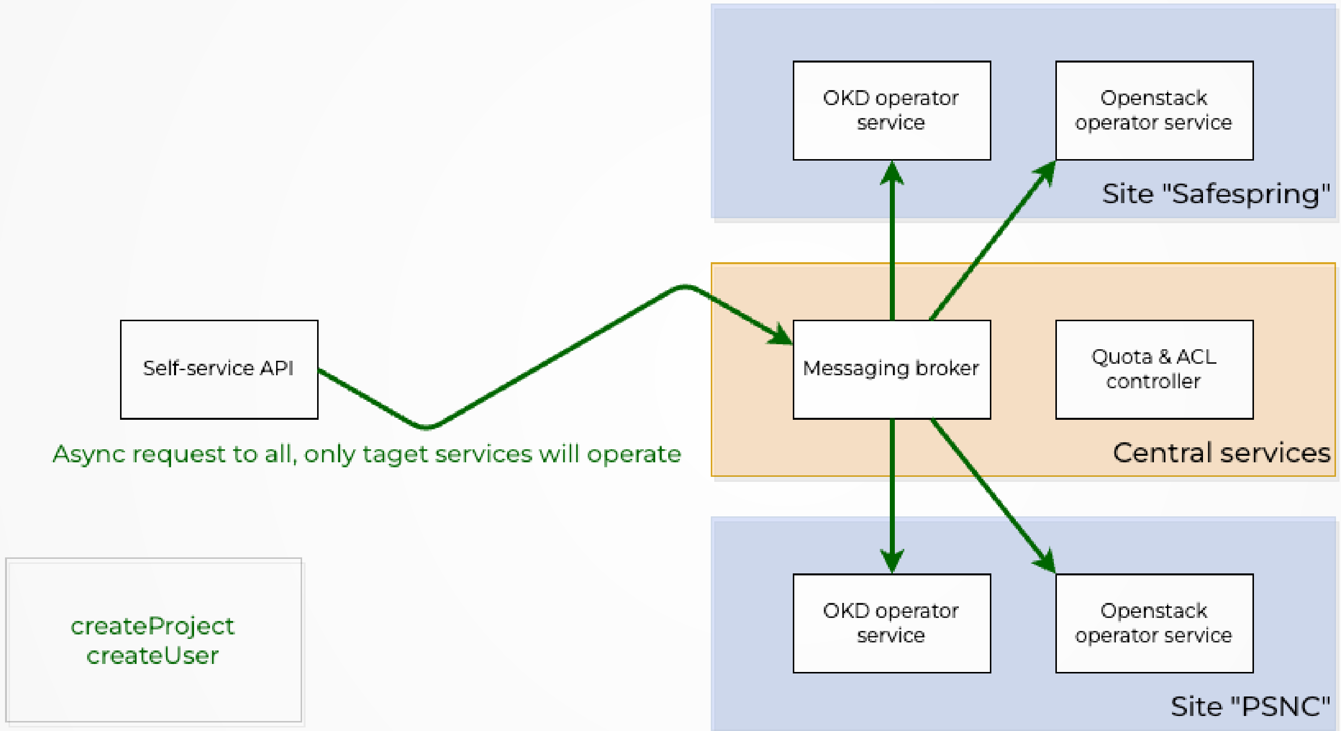
Orchestrating multiple HTTP APIs with NATS

- Self-service HTTP rest API publishes or makes requests to NATS subjects.
- NATS micro operators suscribe to subjects, eg: **selfservice.project.create**
- Each operator has its own **queue group**, meaning all operators will receive a copy of the messages
- NATS does **load balancing** for operators sharing the same queue group

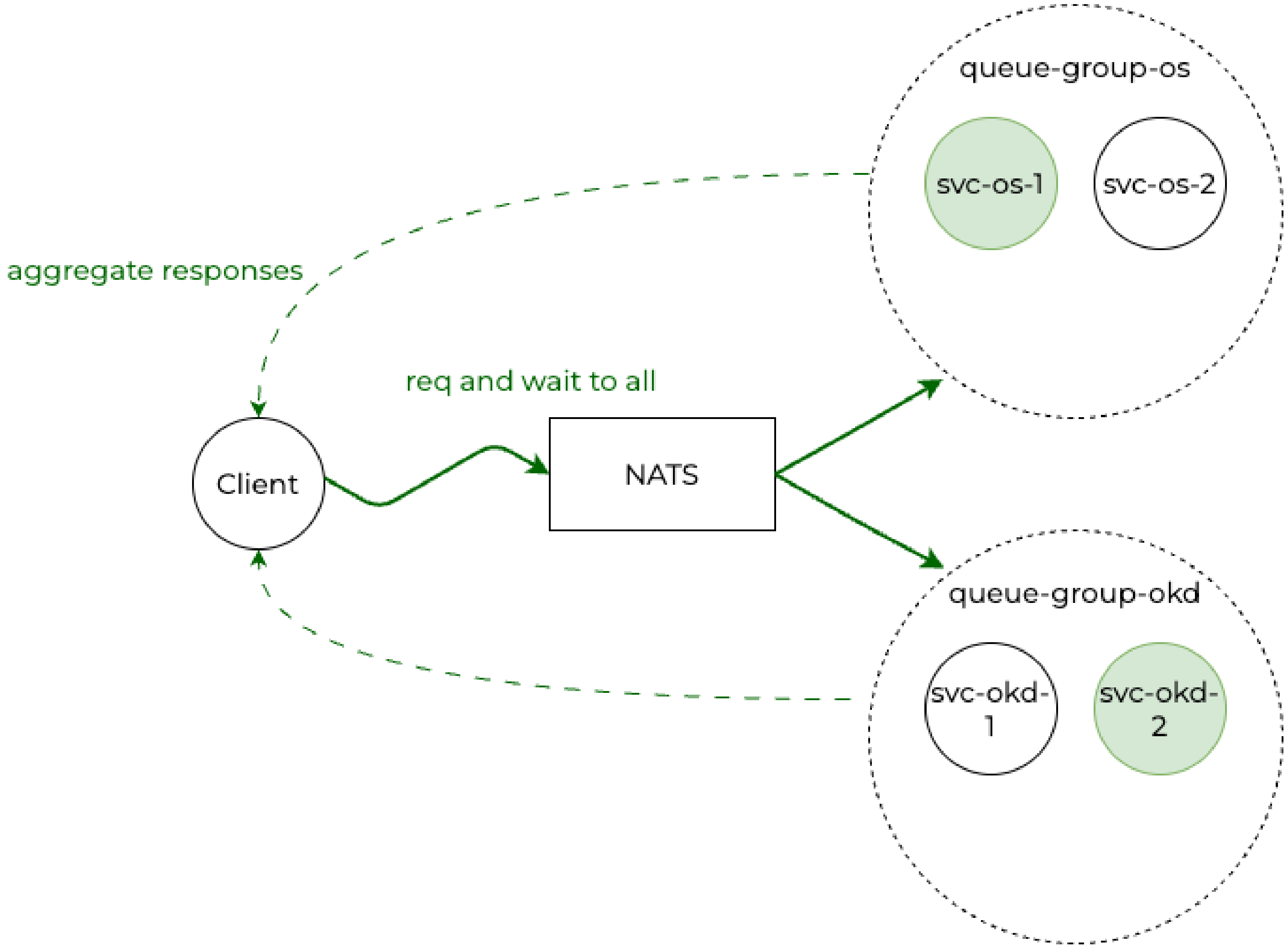
Fan in and fan out pattern



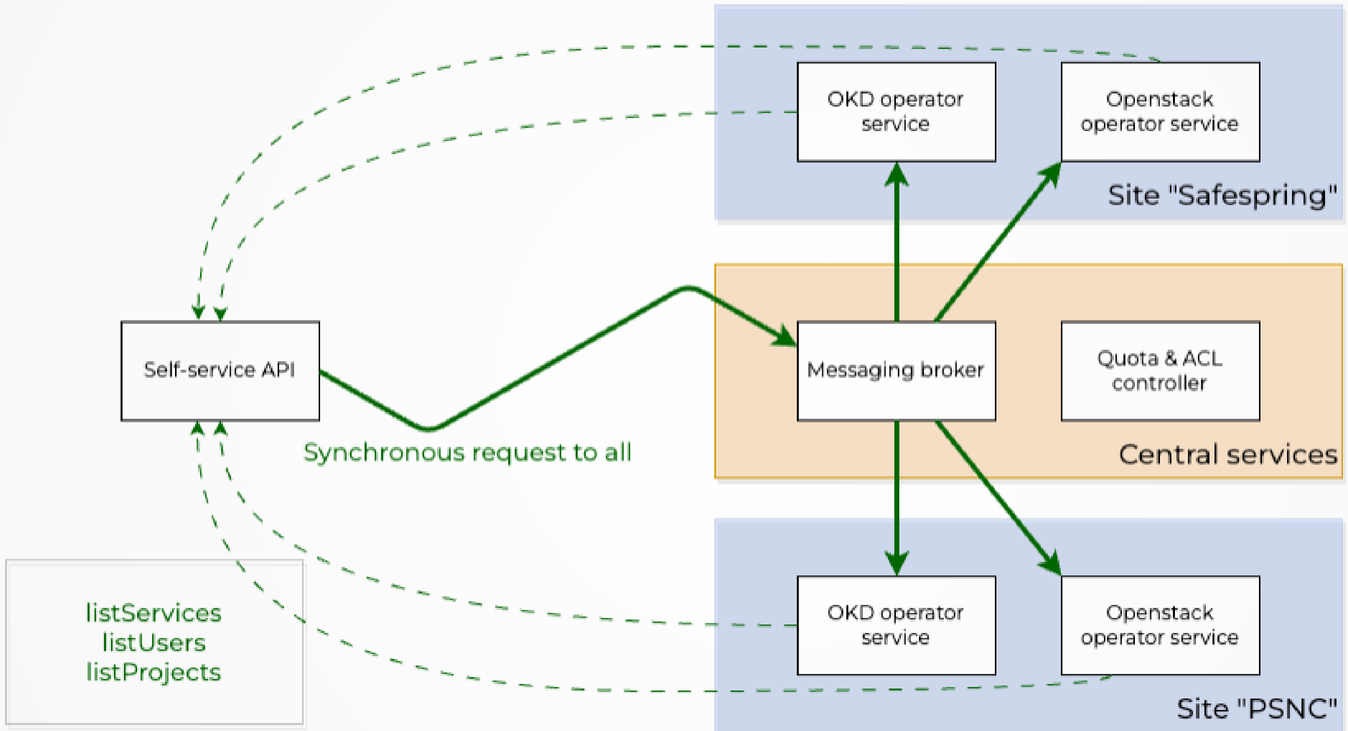
Fan in and fan out pattern



Scatter and gather pattern



Scatter and gather pattern



Challenges

Single API for OKD and Openstack

- Openstack projects and OKD Namespaces
- Users and groups
- Kubernetes annotations
- Quota synchronisation

Integration testing

- Recyclable Openstack and OKD environments
- Microstack and crc projects
- Error propagation
- Timeouts

safespring

We are hiring!

- We are building a platform team!
- Do you love open source network automation, BGP, SONiC?

Come talk to us, we are remote / hybrid! ❤

safespring

WEBSITE
www.safespring.com

LINKEDIN
@Safespring

CONTACT
contact@safespring.com