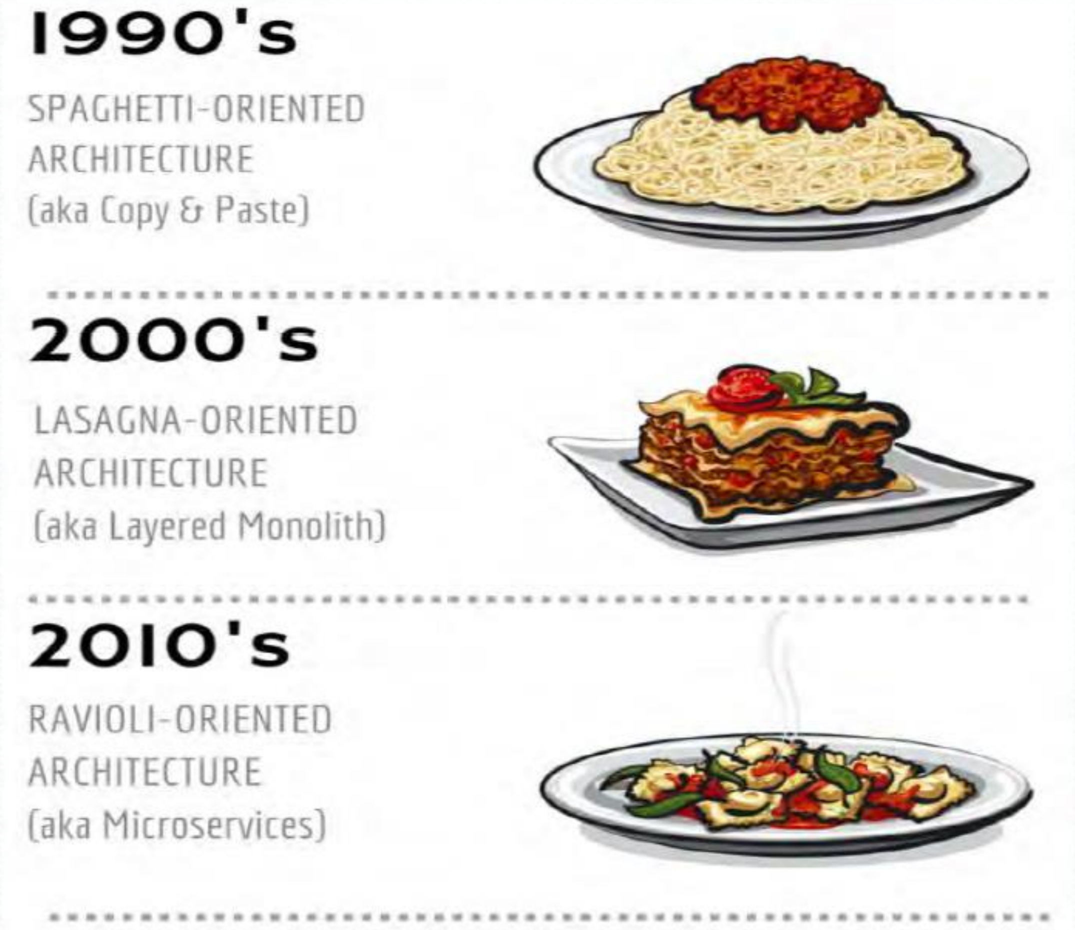
РАЗВОЈ СОФТВЕРА 2

Реализација Микросервиса

ОПИС МИКРОСЕРВИСА

Software architecture evolution



1990's

SPAGHETTI-ORIENTED ARCHITECTURE
(aka Copy & Paste)

2000's

LASAGNA-ORIENTED ARCHITECTURE
(aka Layered Monolith)

2010's

RAVIOLI-ORIENTED ARCHITECTURE
(aka Microservices)

2020's

Pizza-oriented
Architecture

What are microservices

Key Features

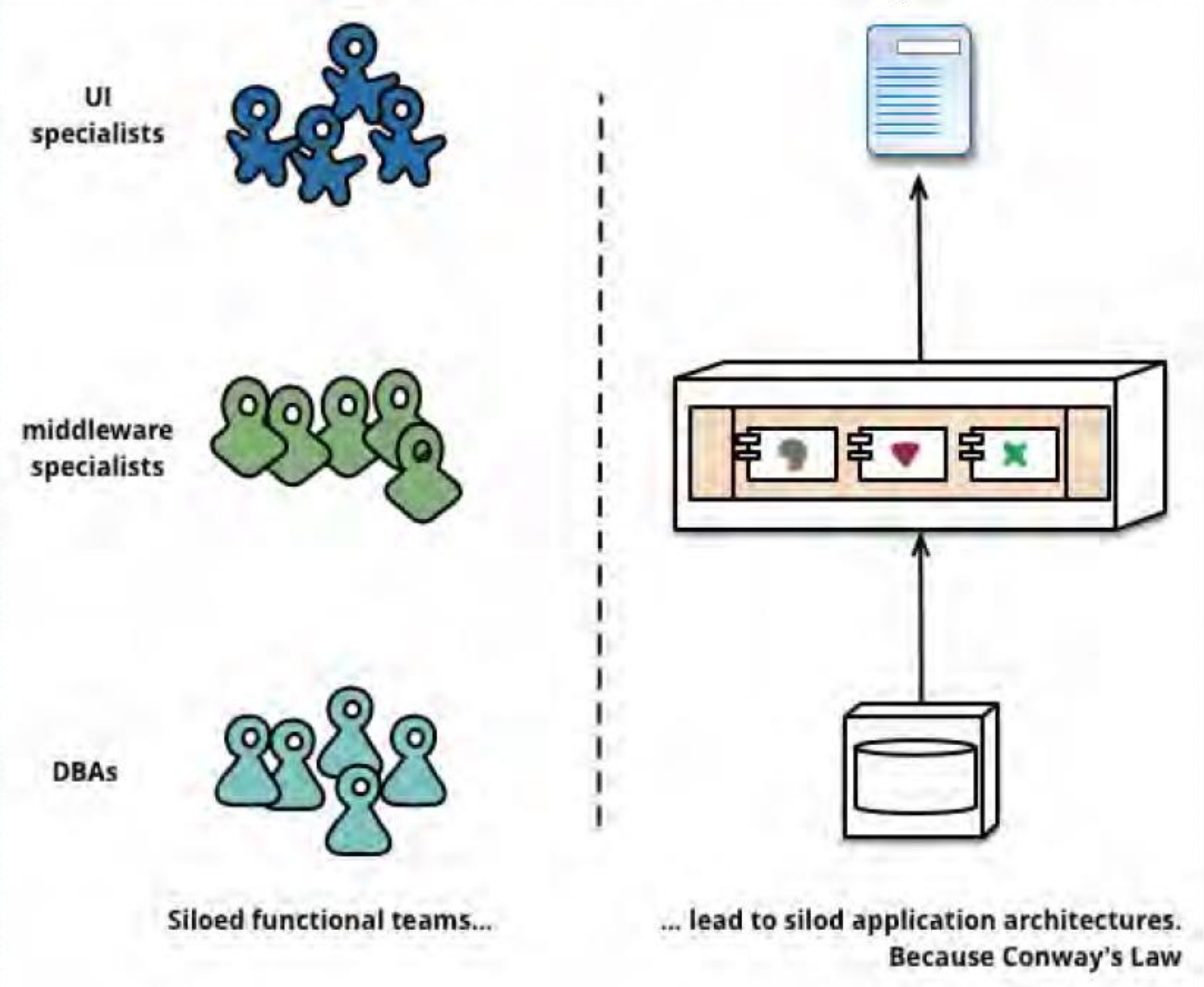
- **Small** and **Focused** on doing **one** thing well
- **Autonomous**
  - Can you make a change to a service and deploy it by itself without changing anything else ?

What are microservices

Key Benefits

- Technology Heterogeneity
- Resilience
- Scaling
- Ease of Deployment
- Composability & Replaceability
- **Enforcing different security**

Organization and business capabilities

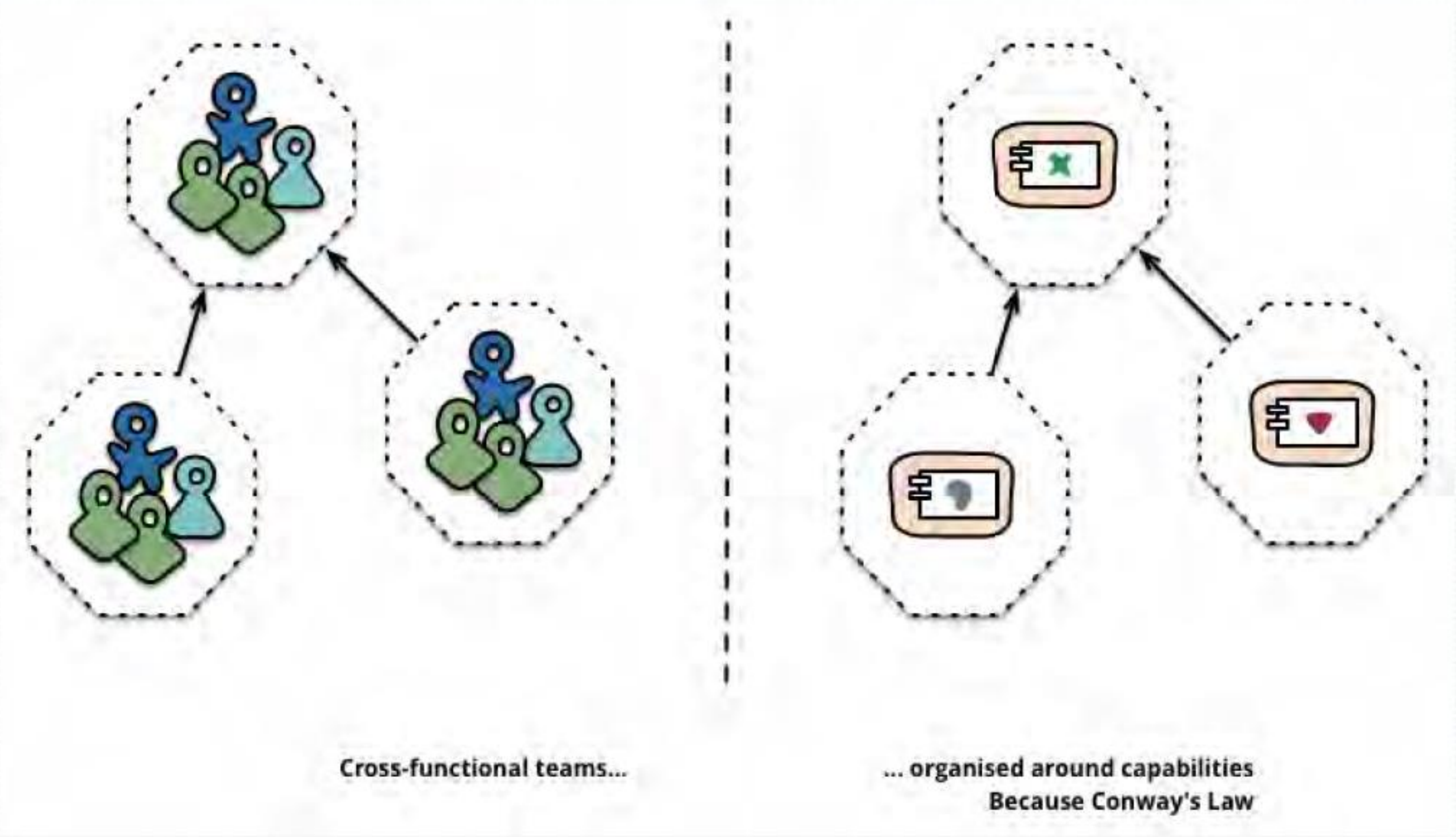

UI specialists
middleware specialists
DBAs
Siloed functional teams...
... lead to silod application architectures.
Because Conway's Law

"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

Melvyn Conway, 1967

Source: http://martinfowler.com

Organization and business capabilities



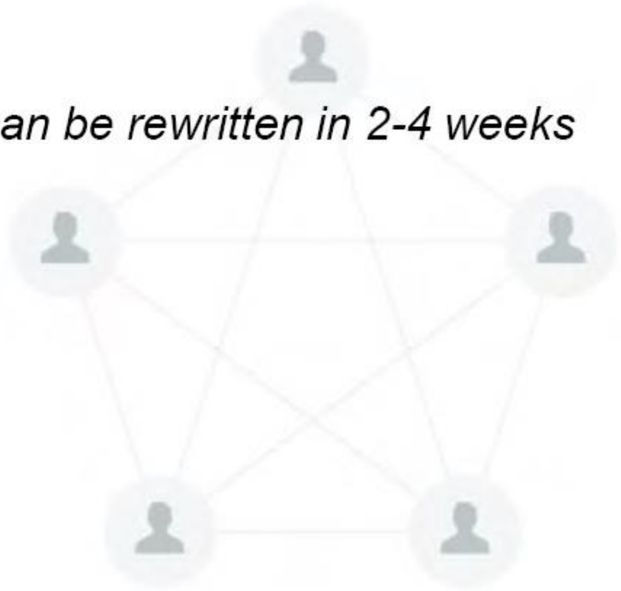
Source: http://martinfowler.com

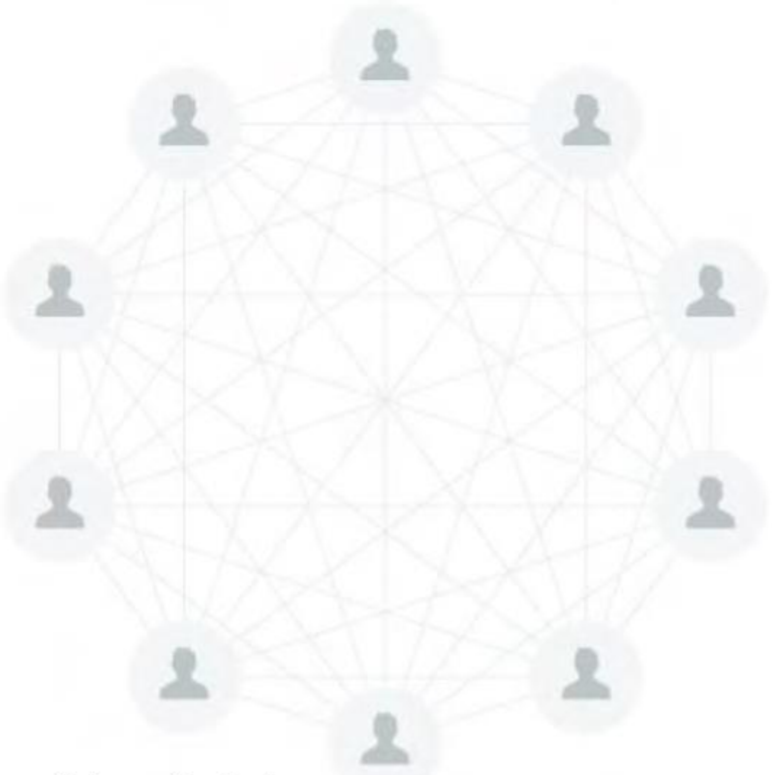
Microservice size

- *If you can’t feed a team with two pizzas, it’s too large. That limits a task force to five to seven people, depending on their appetites.*

Jeff Bezos (Amazon)

- *Something that can be rewritten in 2-4 weeks*



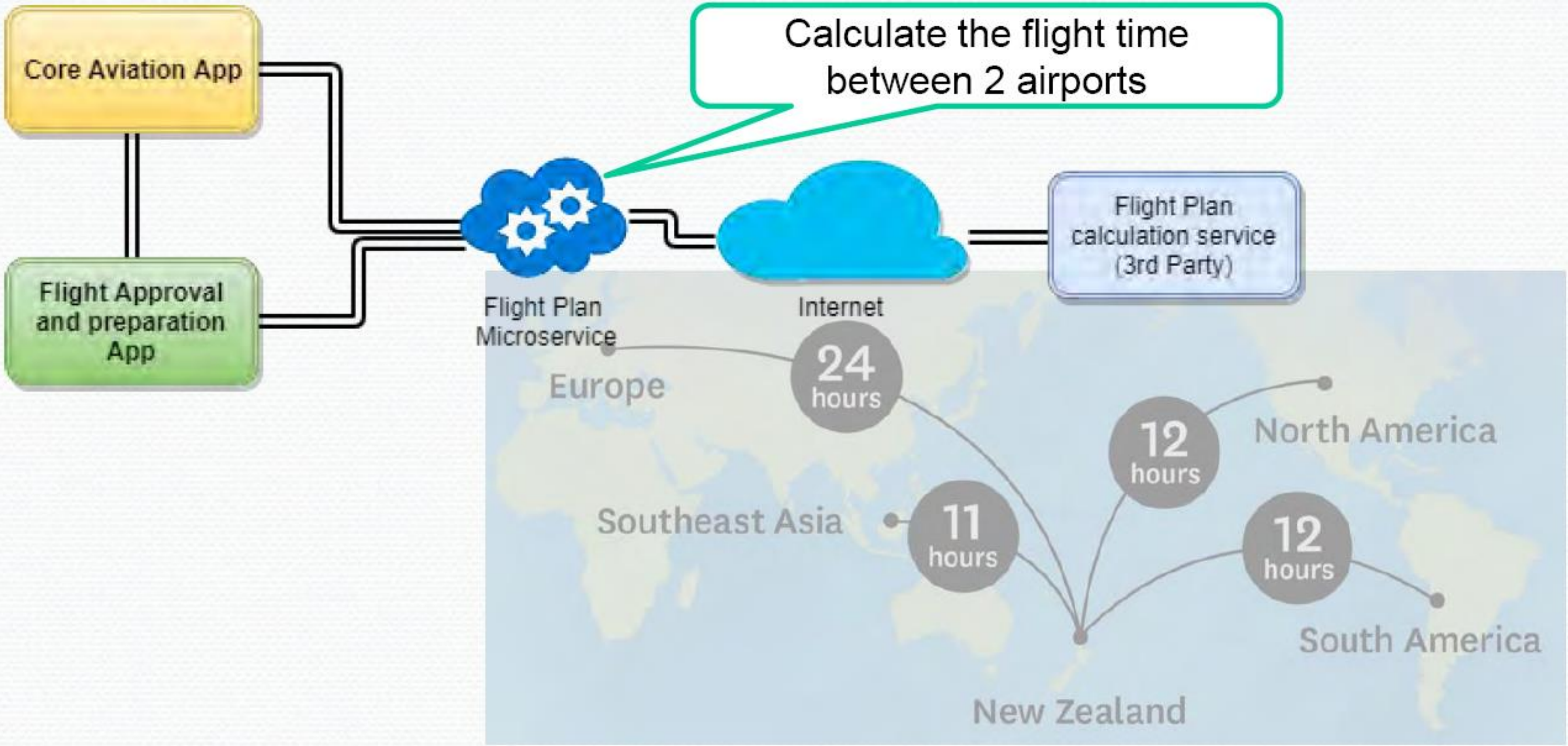


“Gather together those things that change for the same reason, and separate those things that change for different reasons.”

SRP – Robert Martins

РЕАЛИЗАЦИЈА МИКРОСЕРВИСА

The initial requirements



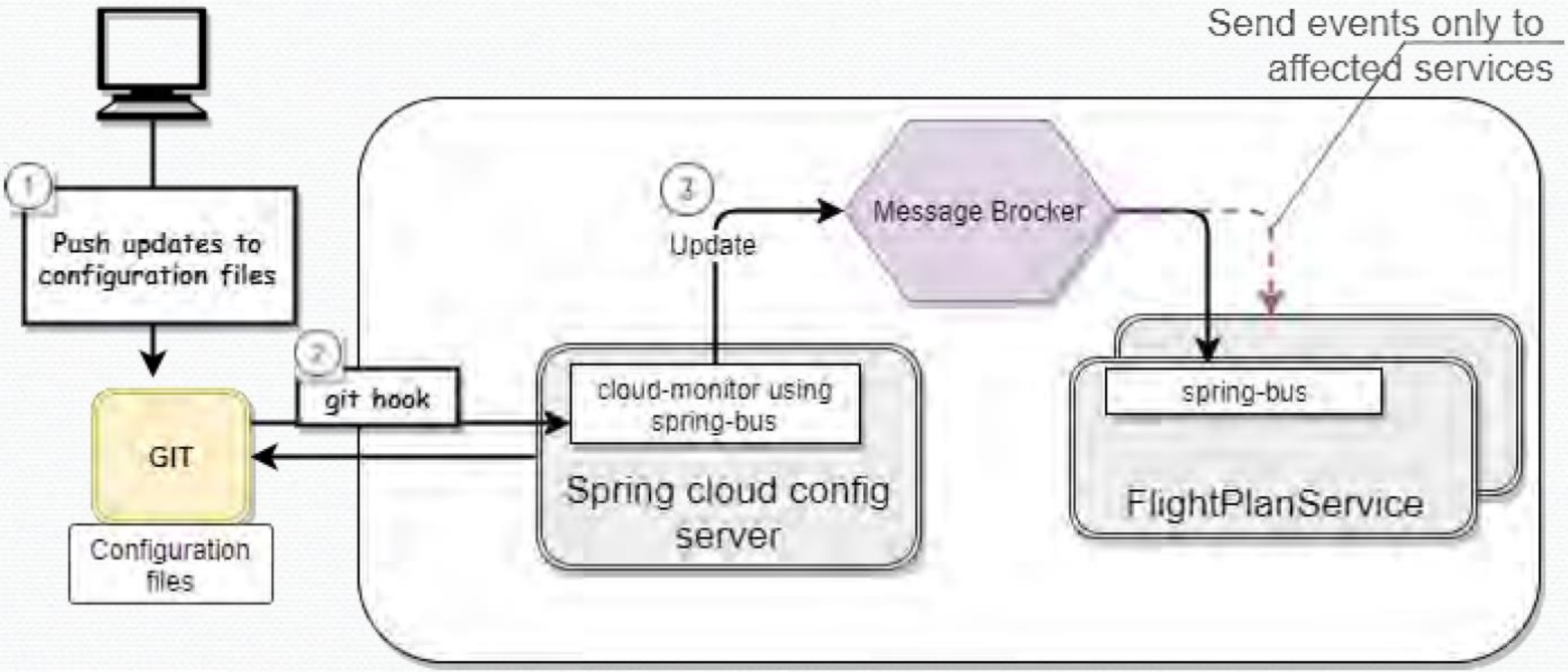
Where is my service?

- The problems:
  - Many small services in the cloud
  - Constantly destroying and deploying new instances
  - Different environments
- How to specify the URL of the service?
- Dynamic Service Registries applications
      - Zookeeper
      - Netflix Eureka
      - Consul

Multiple configuration files

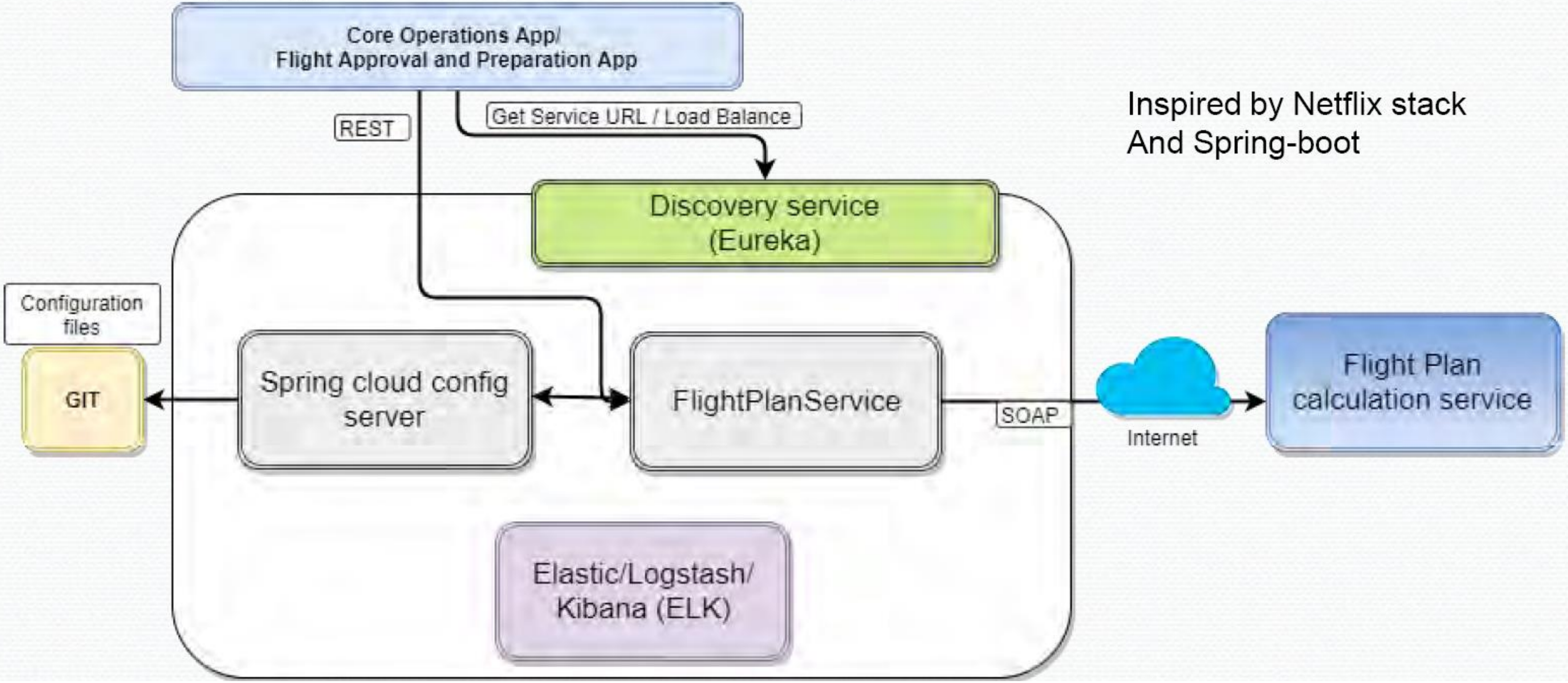
- Multiple services = many configuration files
- Multiple services * several environments = too many configuration files

Centralized configuration



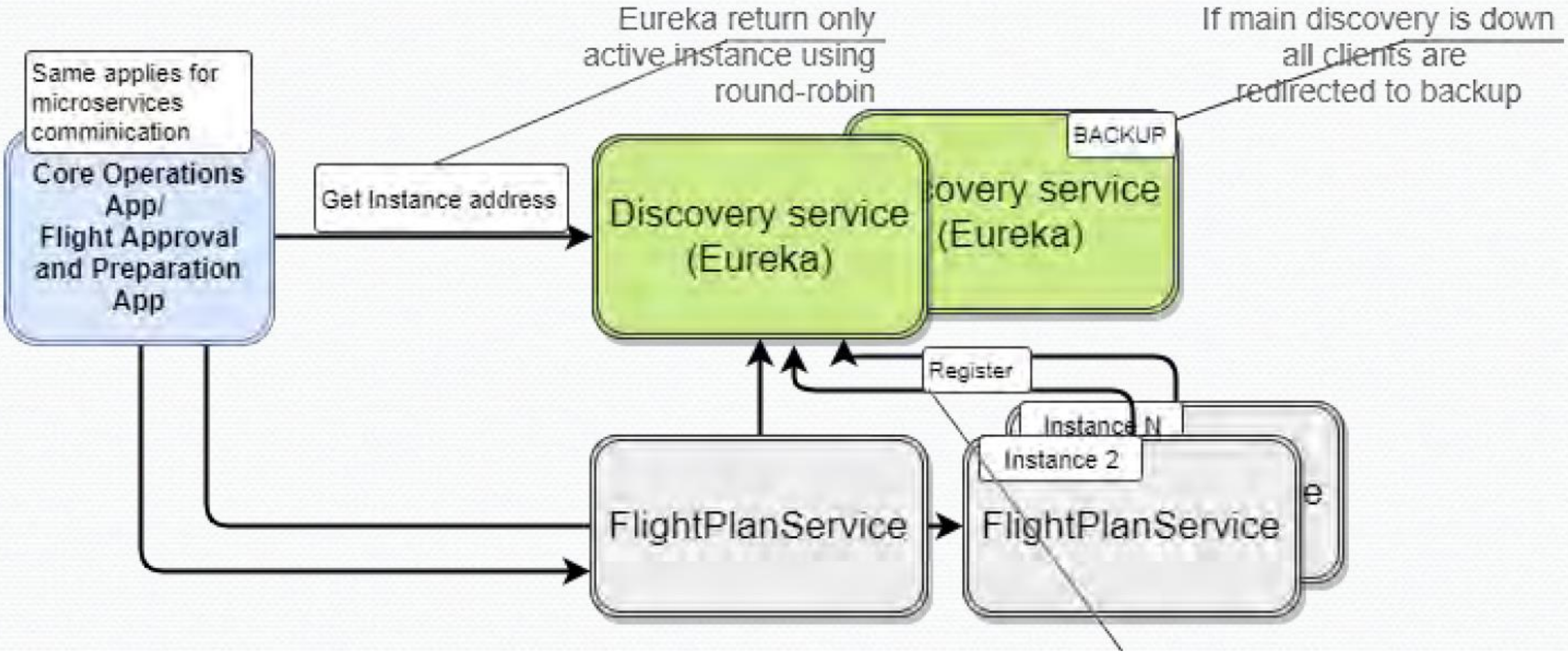
Send events only to affected services
1
Push updates to configuration files
2
git hook
GIT
Configuration files
3
Update
Message Brocker
cloud-monitor using spring-bus
Spring cloud config server
spring-bus
FlightPlanService

Coming up with the first design

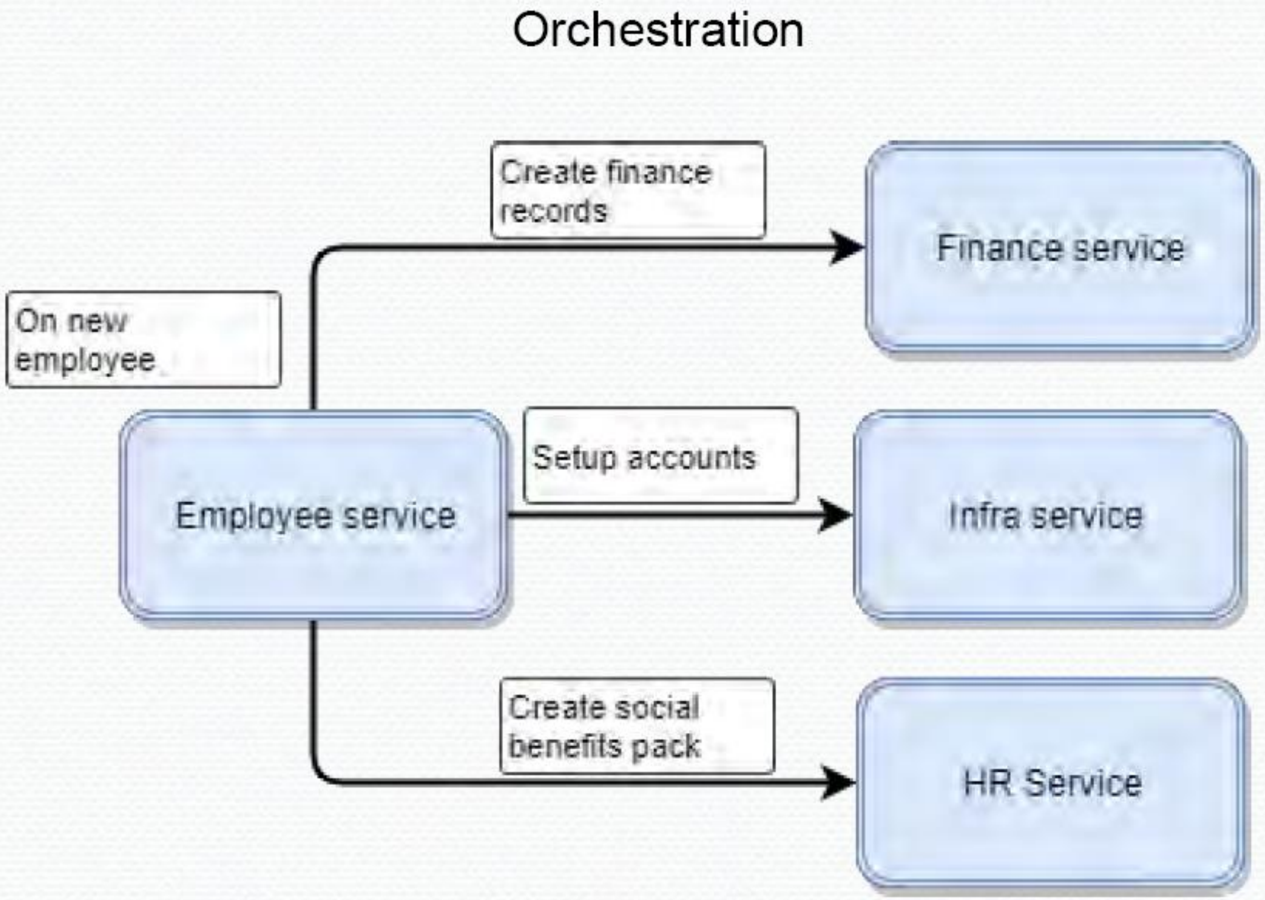


Inspired by Netflix stack
And Spring-boot

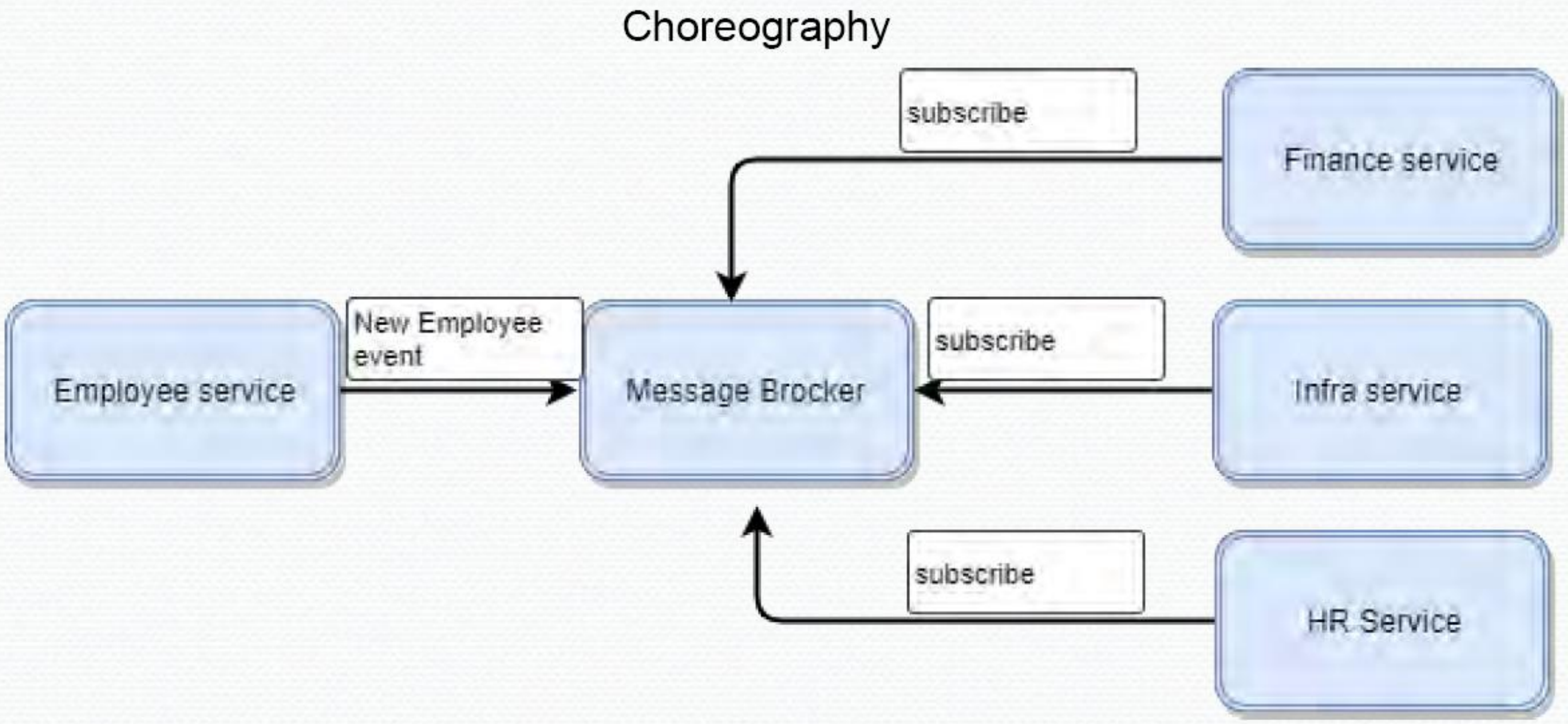
Getting ready for High Availability



Orchestration vs. Choreography

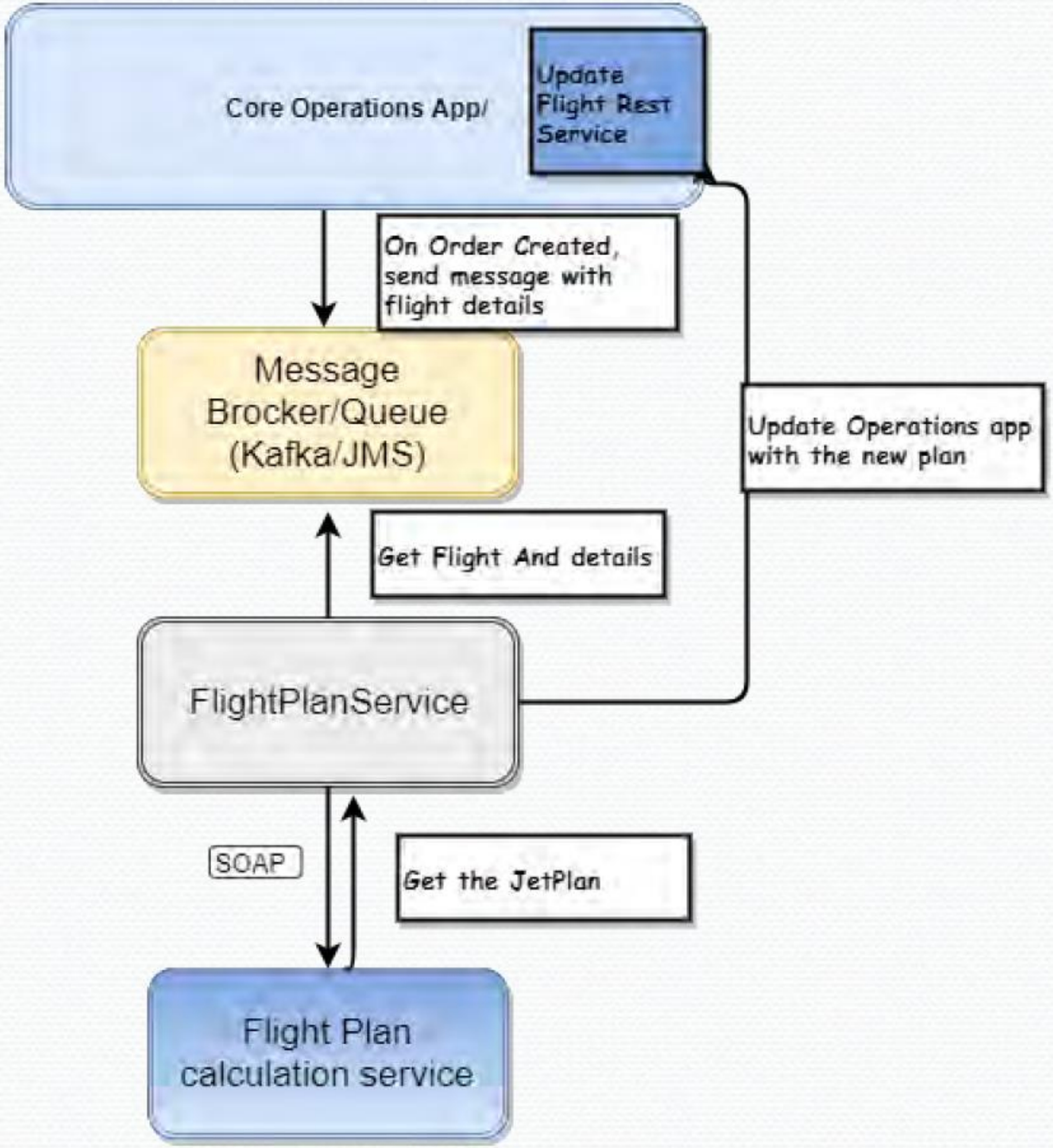


Orchestration vs. Choreography



Event driven approach

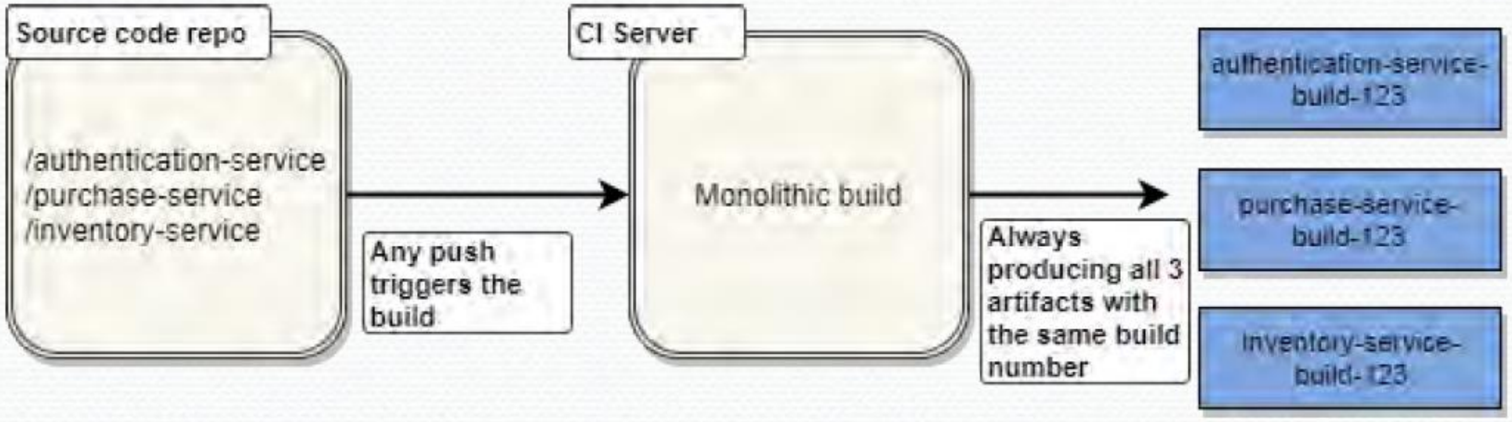
- Fire events on certain triggers
- Use message brocker
- Callbacks


Core Operations App/
Update Flight Rest Service
On Order Created, send message with flight details
Message Brocker/Queue (Kafka/JMS)
Update Operations app with the new plan
Get Flight And details
FlightPlanService
SOAP
Get the JetPlan
Flight Plan calculation service

АУТОМАТИЗОВАНА ИЗГРАДЊА И ИСПОРУКА

Source repositories and CI

- The monolithic approach



Source code repo

/authentication-service
/purchase-service
/inventory-service

Any push triggers the build

CI Server

Monolithic build

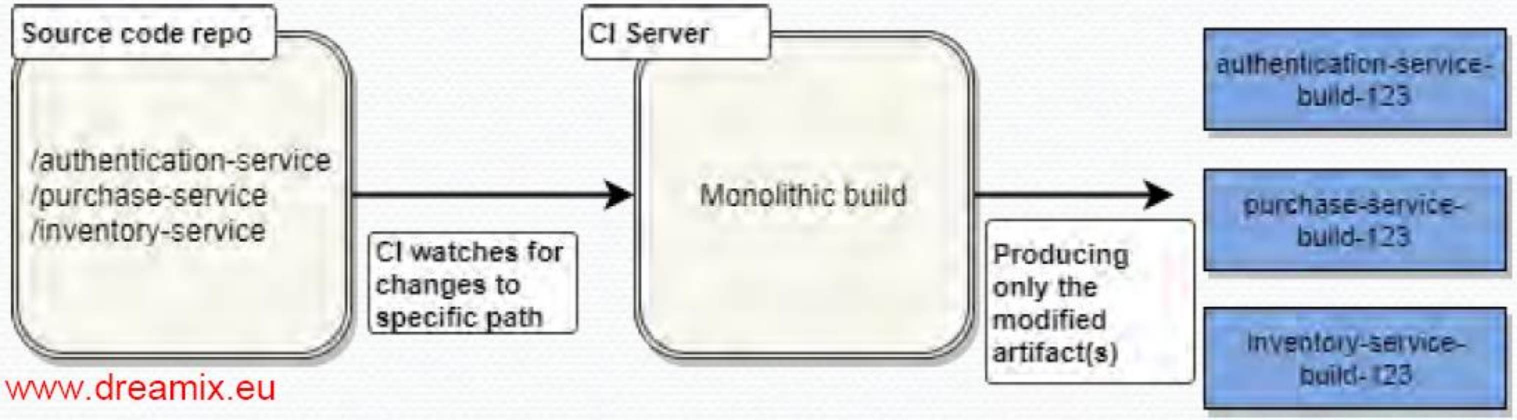
Always producing all 3 artifacts with the same build number

authentication-service-build-123

purchase-service-build-123

inventory-service-build-123

- The improved monolithic approach



Source code repo

/authentication-service
/purchase-service
/inventory-service

CI watches for changes to specific path

CI Server

Monolithic build

Producing only the modified artifact(s)

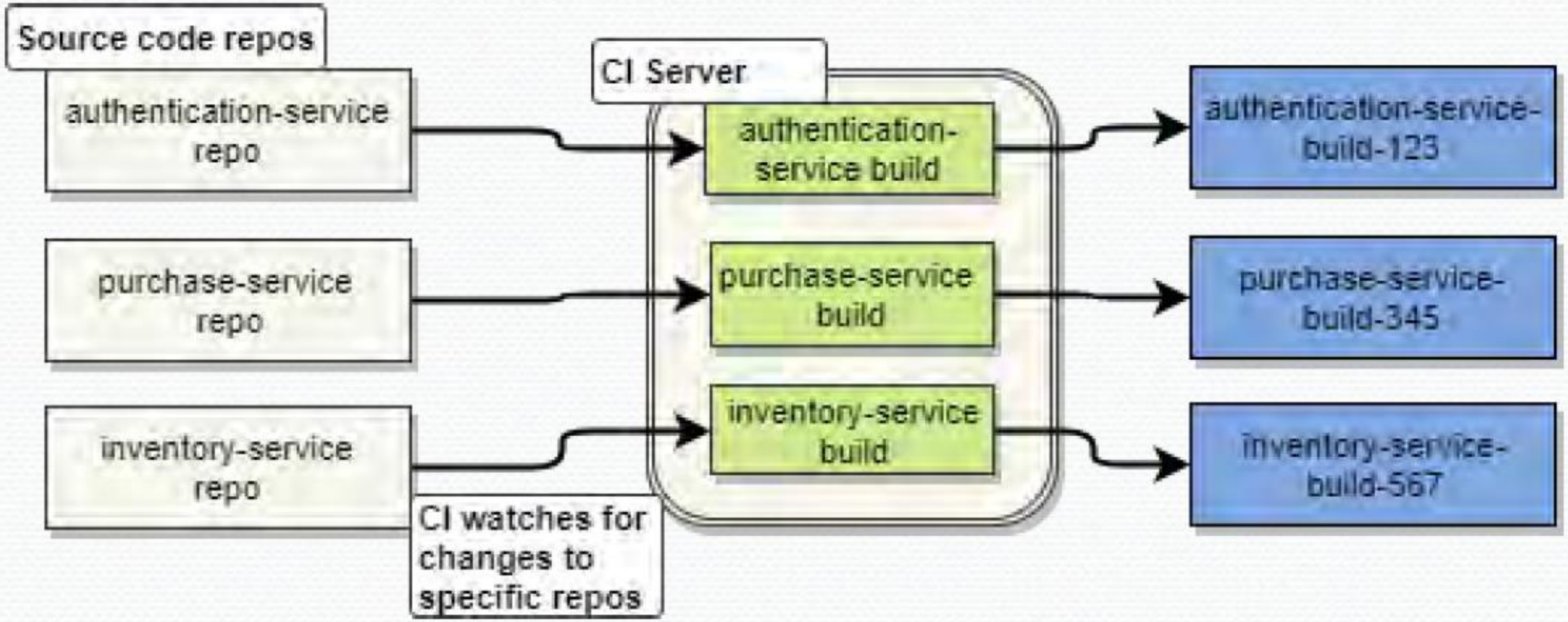
authentication-service-build-123

purchase-service-build-123

inventory-service-build-123

Source repositories and CI

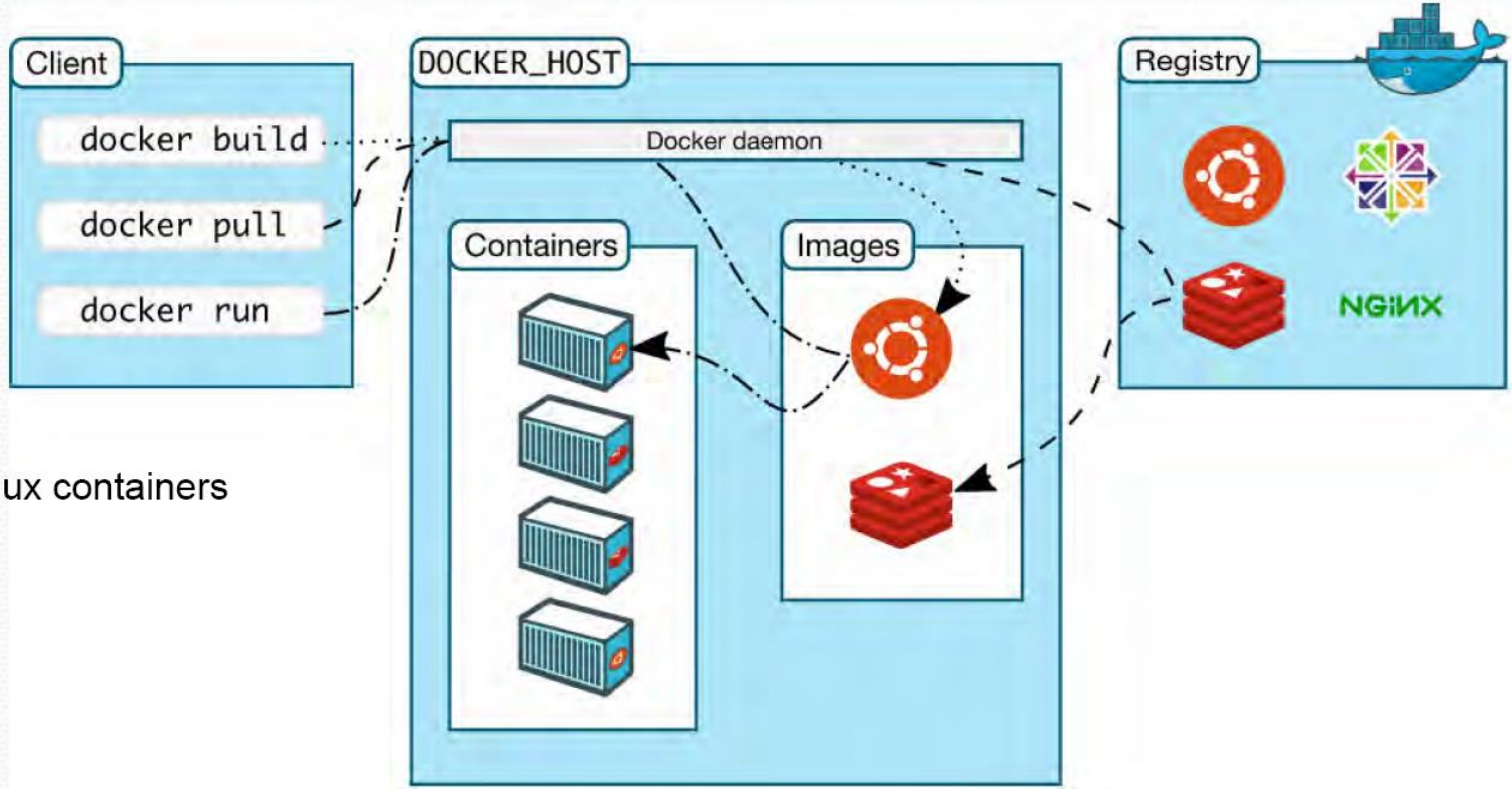
- The microservice approach


Source code repos
authentication-service repo
purchase-service repo
inventory-service repo
CI Server
authentication-service build
purchase-service build
inventory-service build
authentication-service-build-123
purchase-service-build-345
inventory-service-build-567
CI watches for changes to specific repos

How many services per machine ?

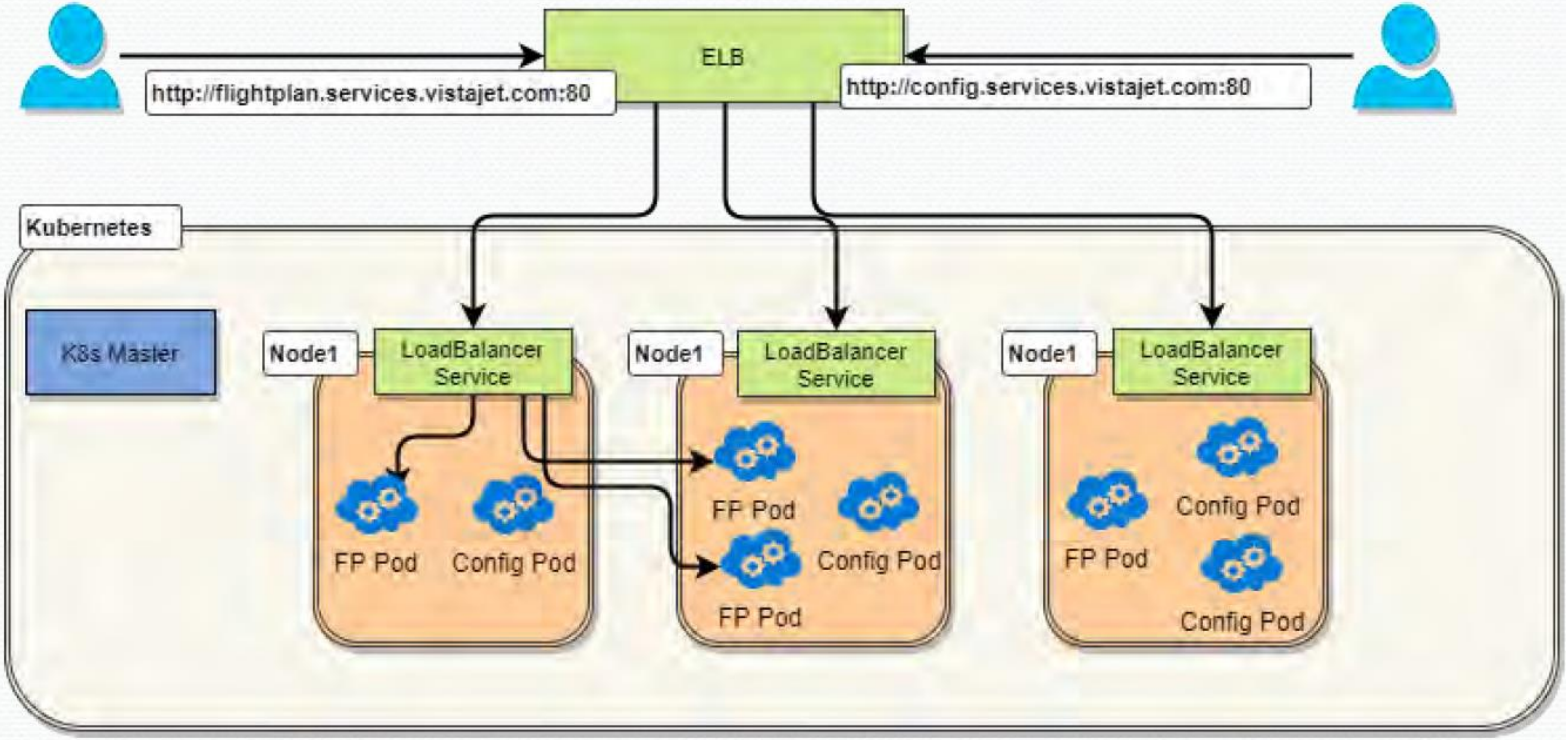
- Multiple services per one host
  - Hard to monitor and hard for problem determination
- Application containers
  - Example: One JVM hosting different service artifacts (wars)
- Single service per host
  - Easier to monitor and easier to scale
- Platform as a service
  - Automatically provisioning
  - Automatic scaling

Docker



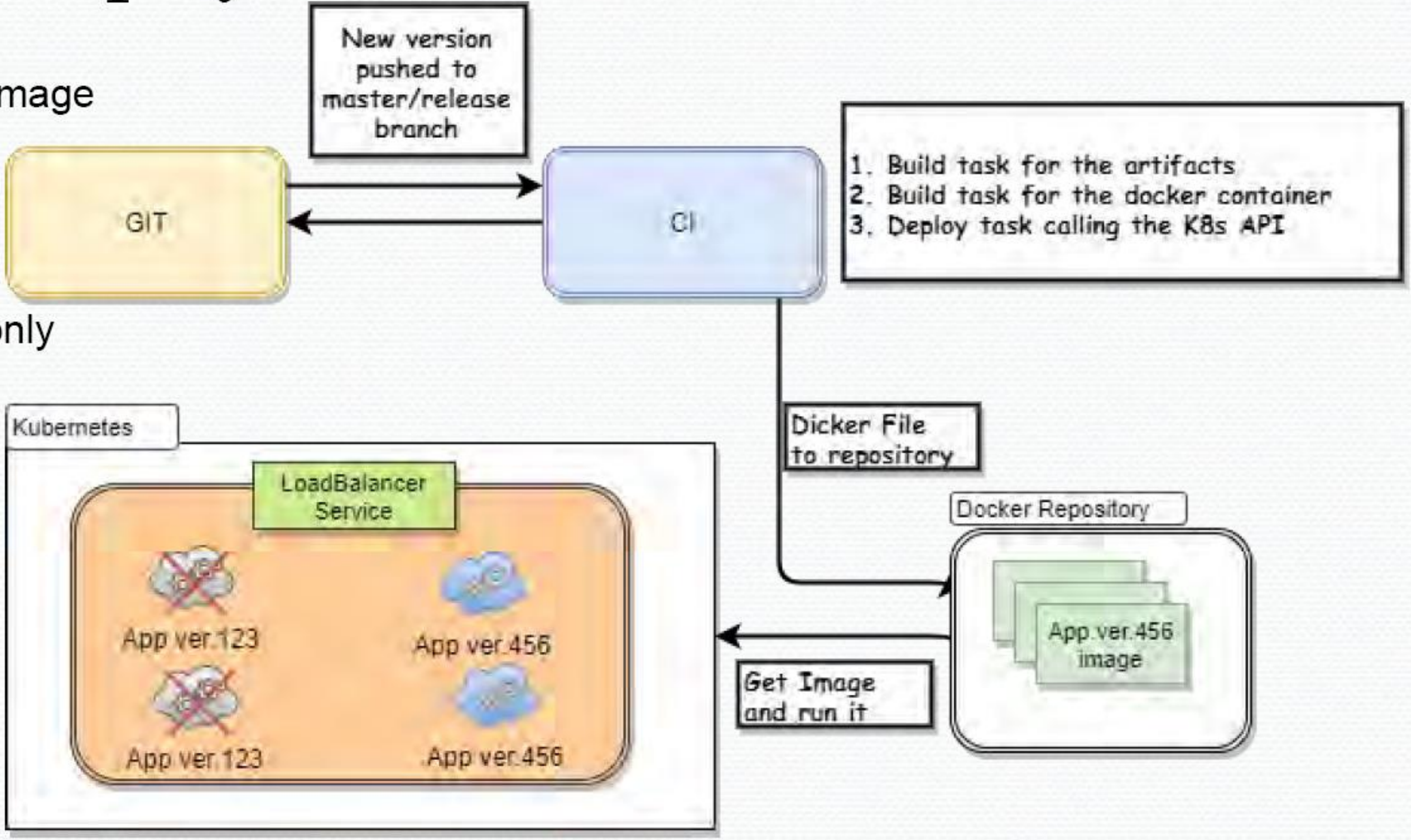
- Using existing Linux containers feature

Kubernetes & Docker



Zero-downtime deployments

1. Build new version docker image
2. Deploy to K8s
3. Health-check OK
4. LB routes to new version only
5. Wait for active request on the old version
6. Undeploy old version



New version pushed to master/release branch

1. Build task for the artifacts
2. Build task for the docker container
3. Deploy task calling the K8s API

Dicker File to repository

Get Image and run it

ЗАКЉУЧЦИ

No Silver Bullet

- Pay special attention to logging and problem determination
- Prepare for even more failures
  - Chaos Monkey
- Timeouts and Circuit Breakers
- Consider Initially go for monolith with well established modules and bounderies

НАПОМЕНА

Највећи део материјала ове презентације је преузет из презентације
Fly High With Microservices Architecture, аутора Angel Gruev, која је доступна на адреси:
https://dreamix.eu/insights/jax-london-speaker-angel-gruev-helps-you-fly-with-microservices/