Trois heures à l'assaut d'une application réactive

Xavier Bucchiotty
Nicolas Jozwiak
Vincent Spiewak

#3ReactApp

@njozwiak @xbucchiotty @vspiewak

DEVOXX™ France

Speakers

@vspiewak

@xbucchiotty

@njozwiak

Xebia

http://blog.xebia.fr

Disclaimer

DEVOXX™ France

Agenda

- Contexte
- Comment concevoir ?
- Comment développer ?
- Comment déployer ?
- Comment monitorer ?

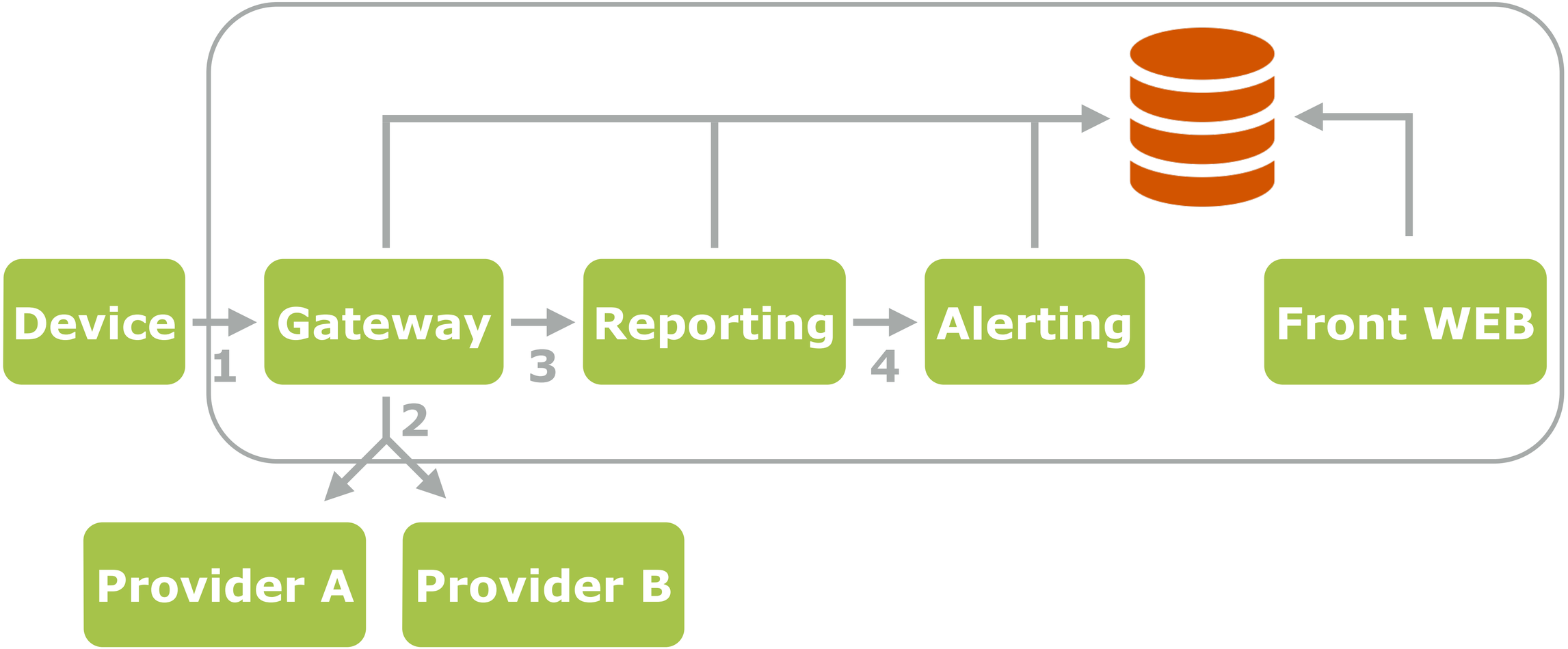
DEVOXX™ France

Agenda

- **Contexte**
- Comment concevoir ?
- Comment développer ?
- Comment déployer ?
- Comment monitorer ?

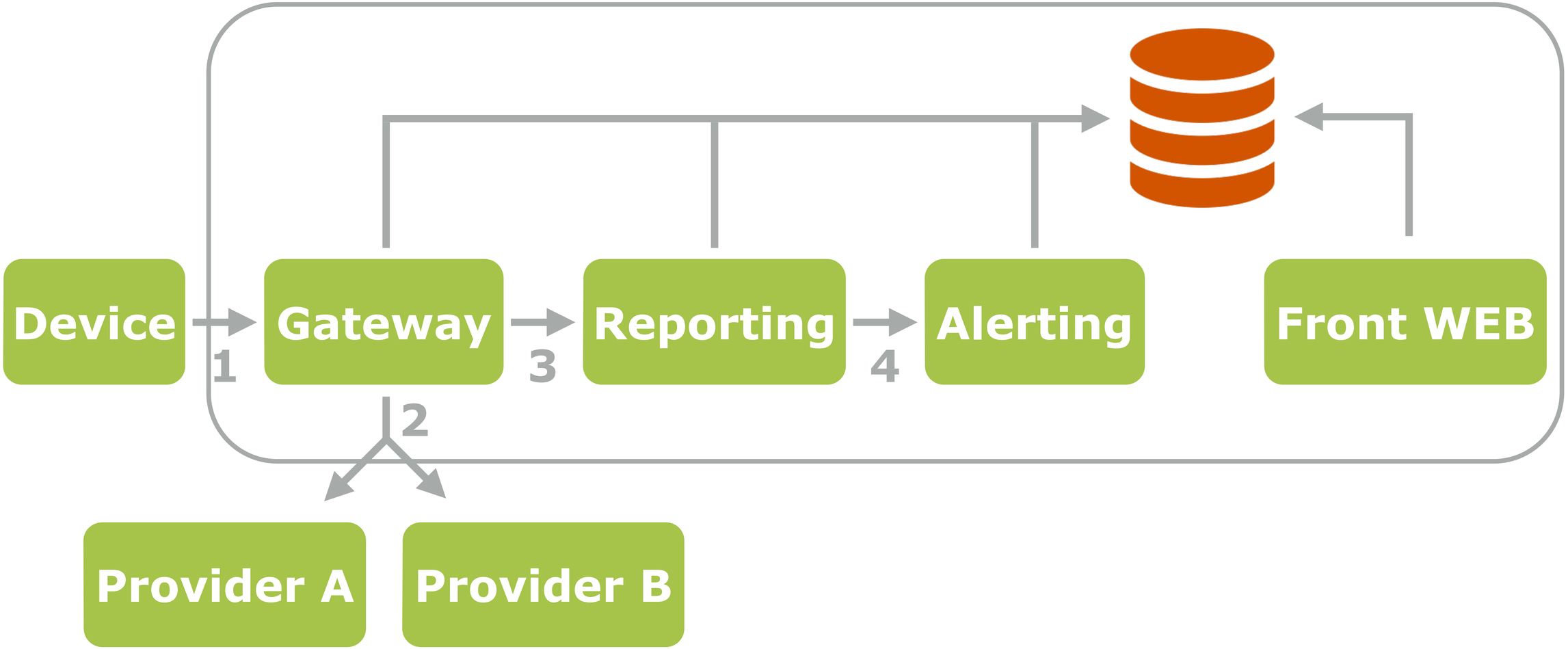
Contexte

Architecture monolithique

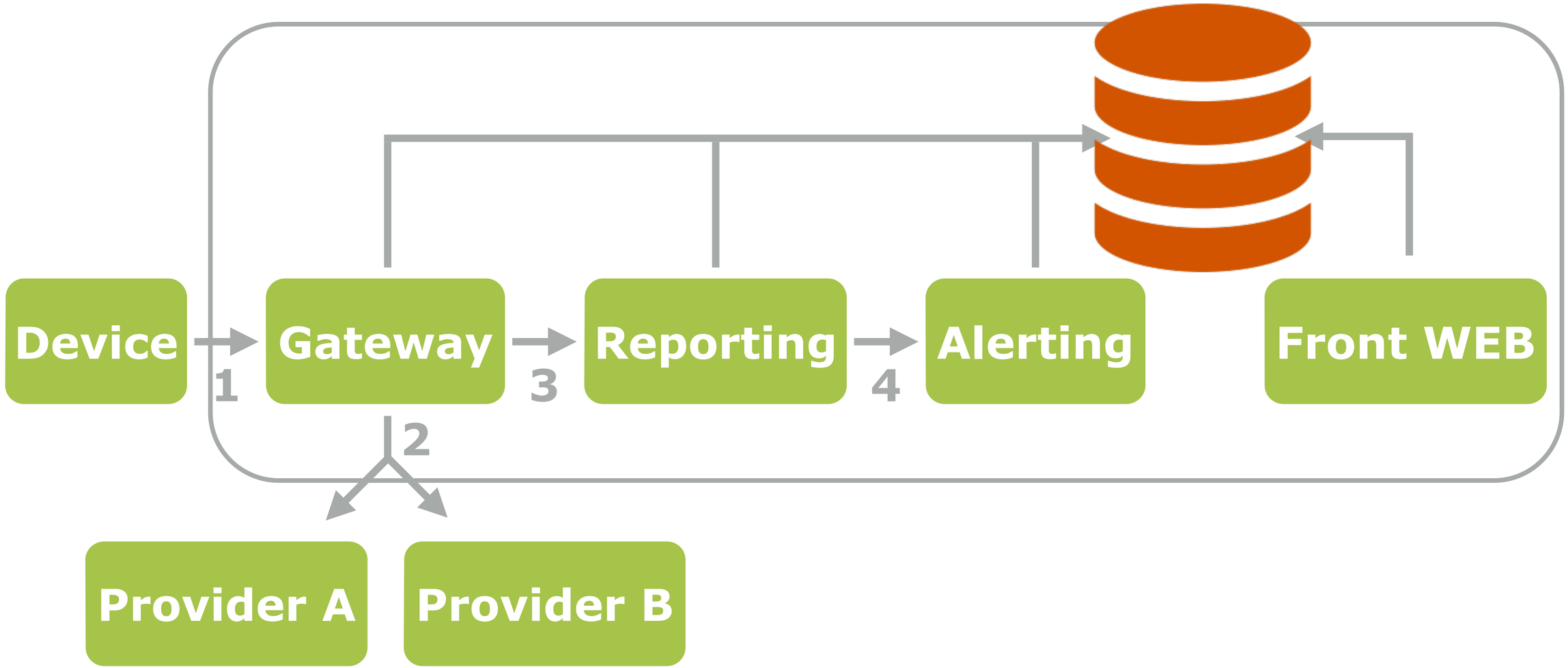


DEVOXX France

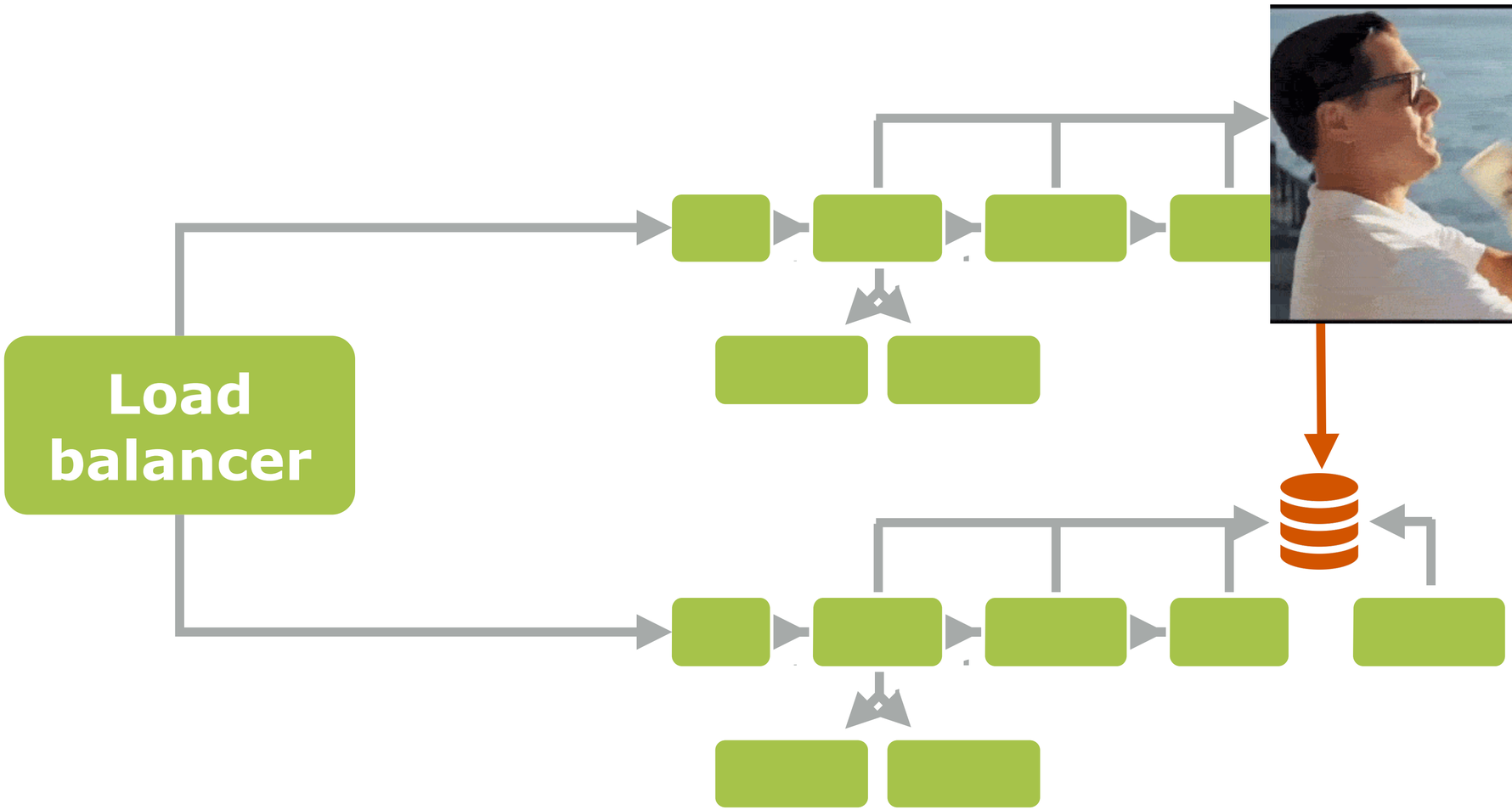
Architecture monolithique



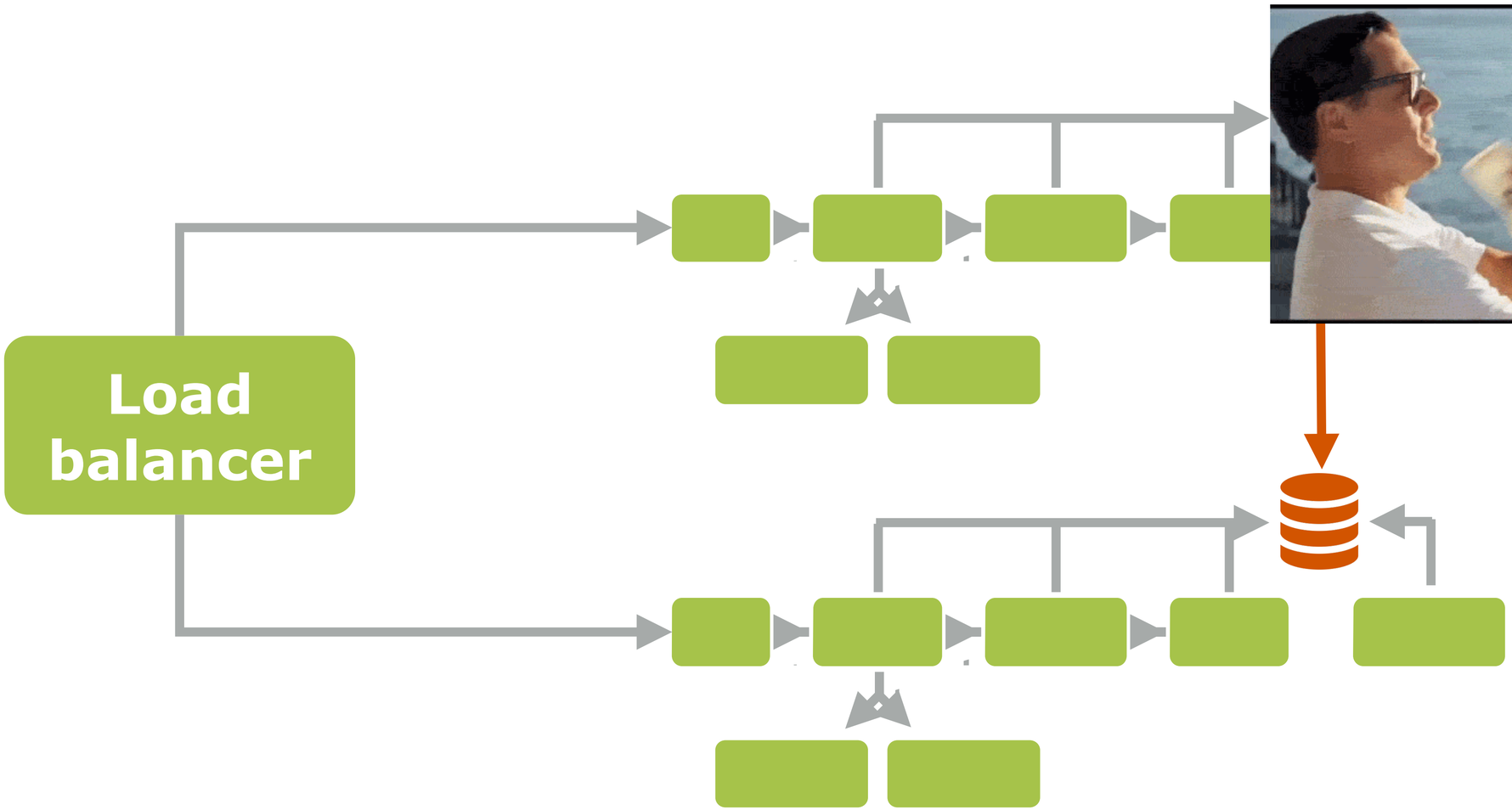
Architecture monolithique



Architecture monolithique

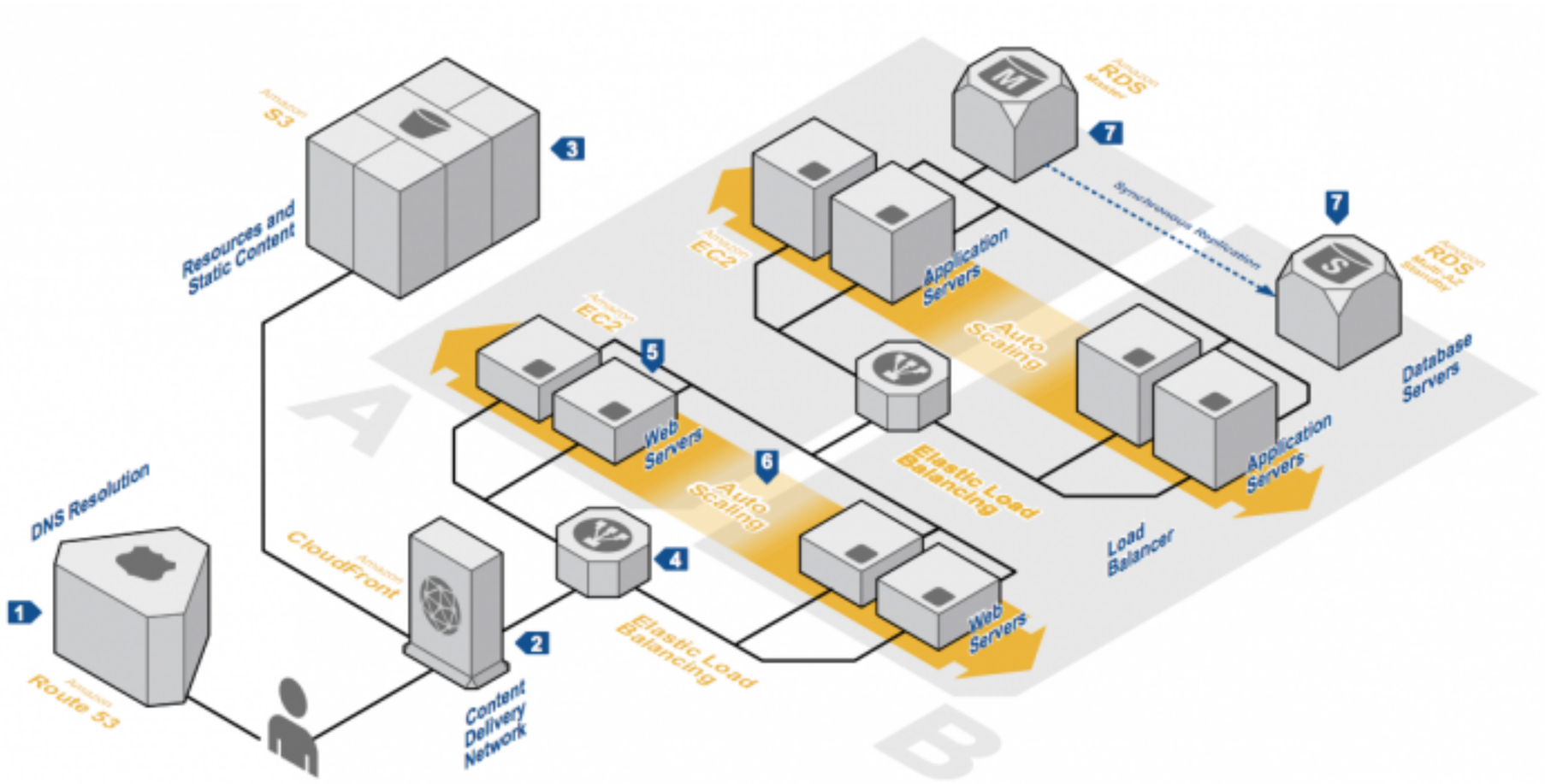


Architecture monolithique



DEVOXX France

Solution: Cloud



En résumé

Elastic

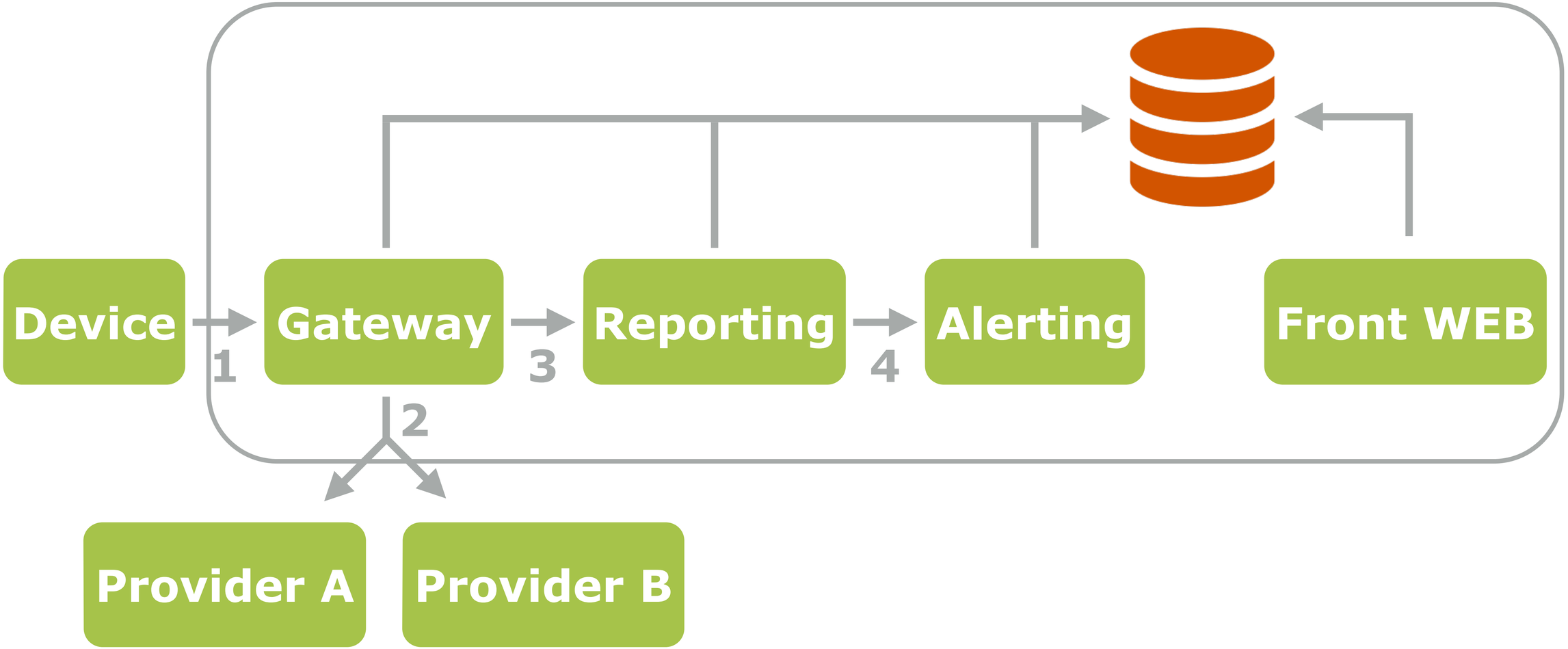
Resilient

DEVOXX™ France

Agenda

- Contexte
- **Comment concevoir ?**
- Comment développer ?
- Comment déployer ?
- Comment monitorer ?

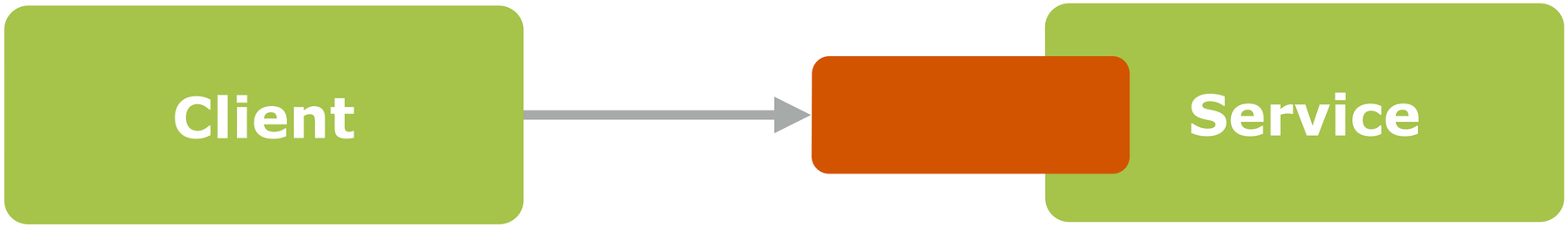
Architecture monolithique



Message Driven

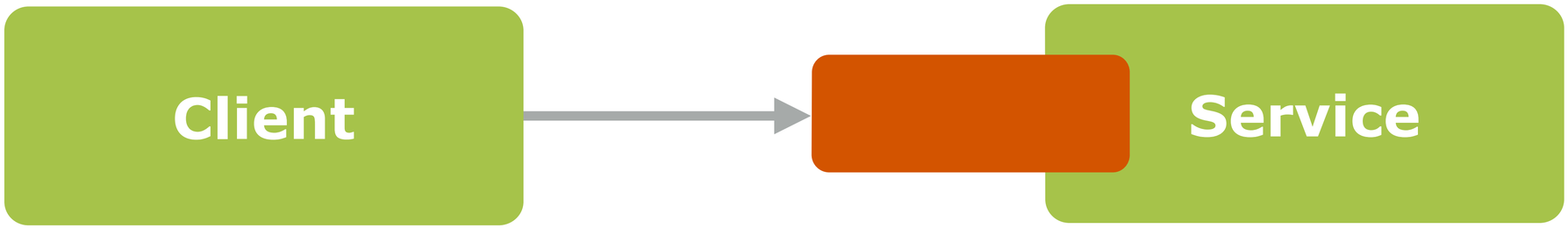
- **Couplage faible**
- **Asynchrone**
- **Non bloquant**

Queues



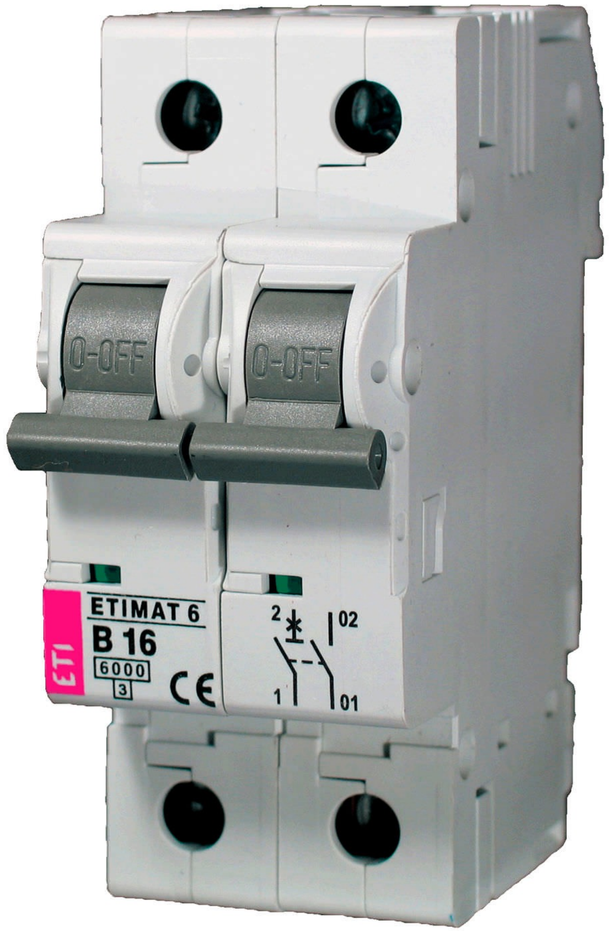
Queues

Bounded Queues



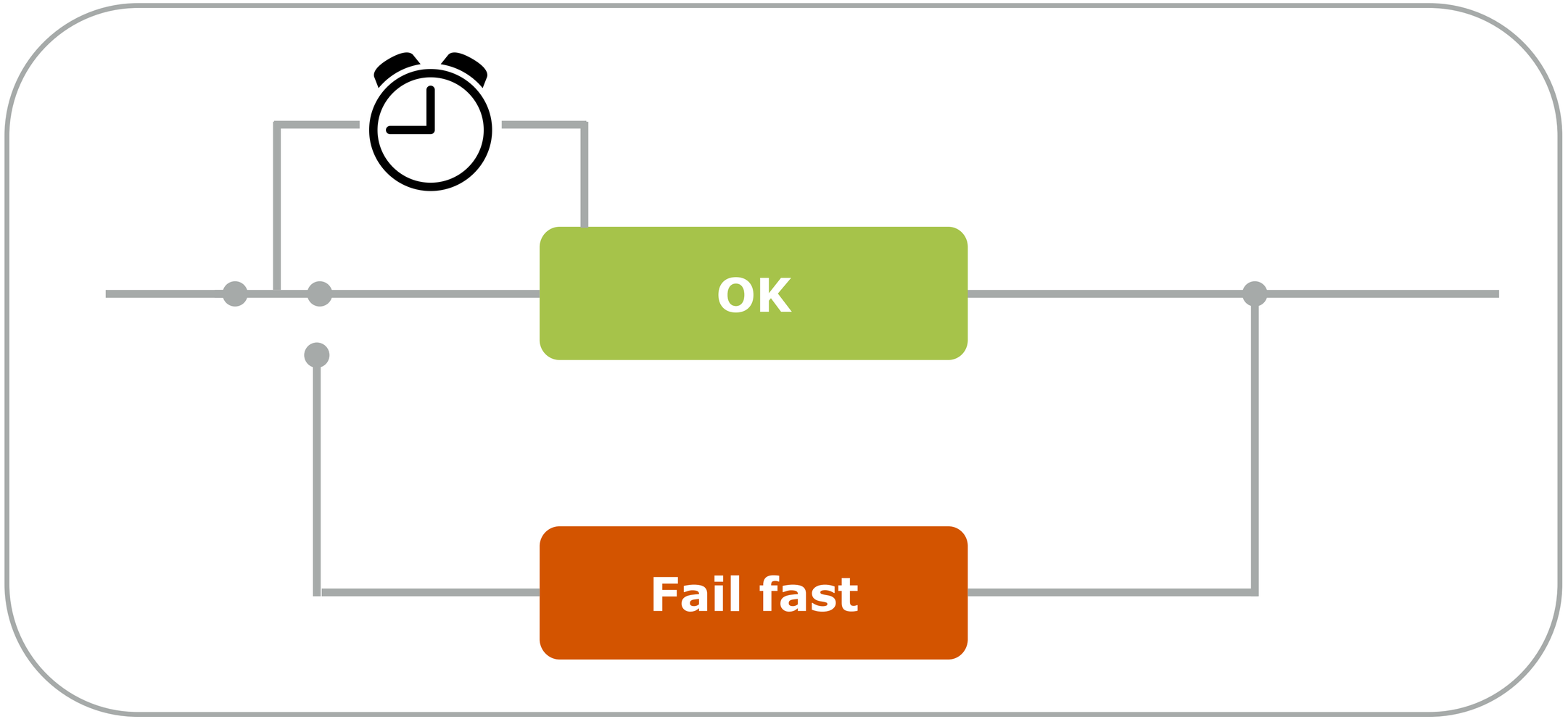
Bounded Queues

Circuit Breaker

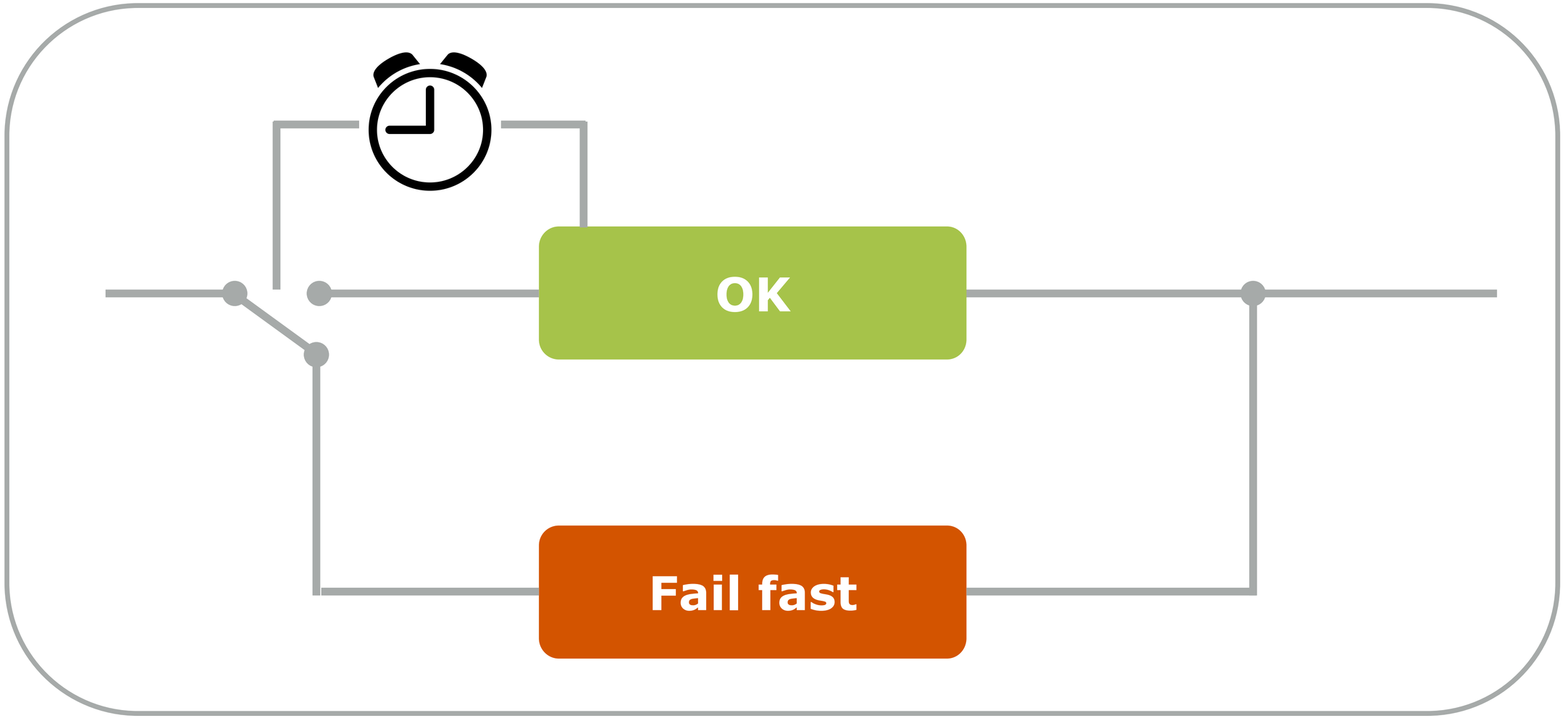


Circuit Breaker

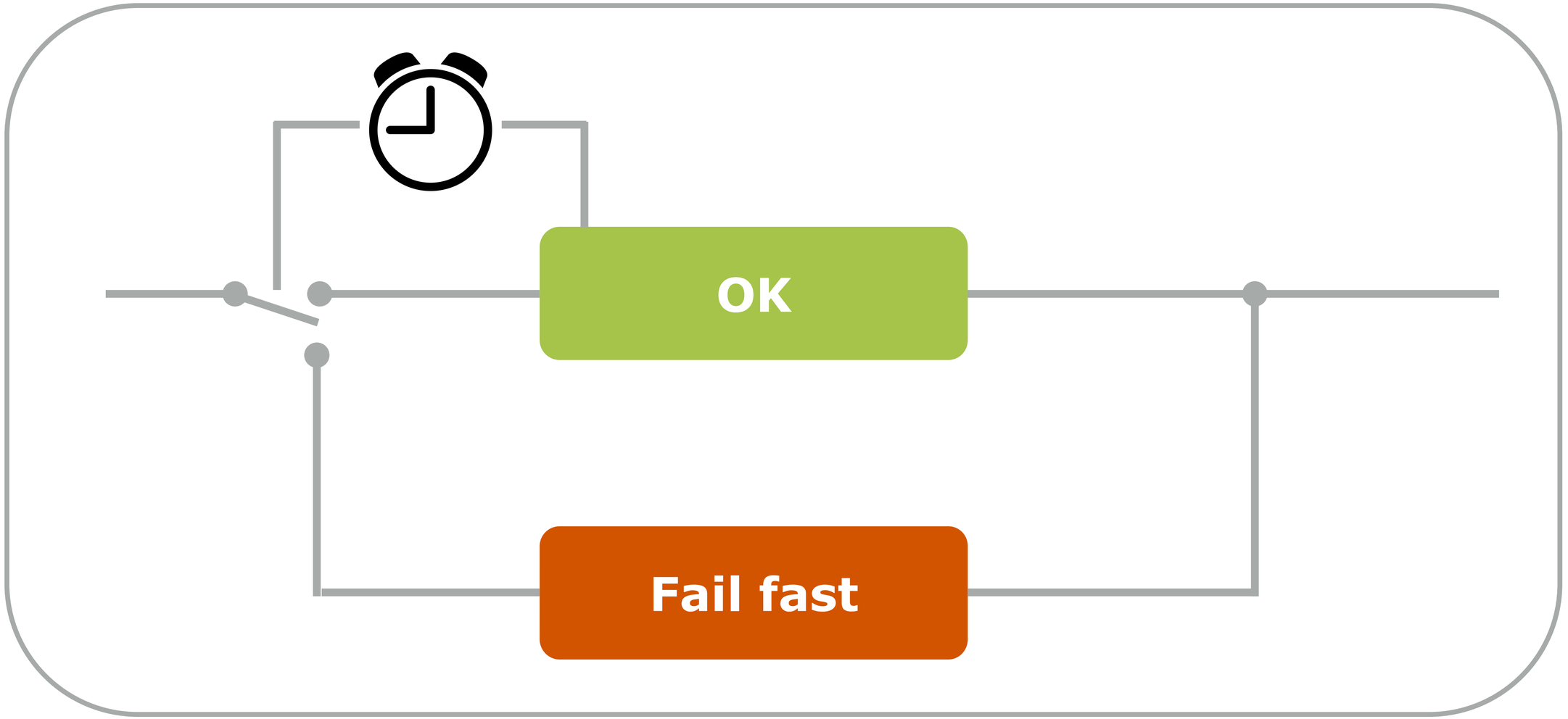
Circuit Breaker



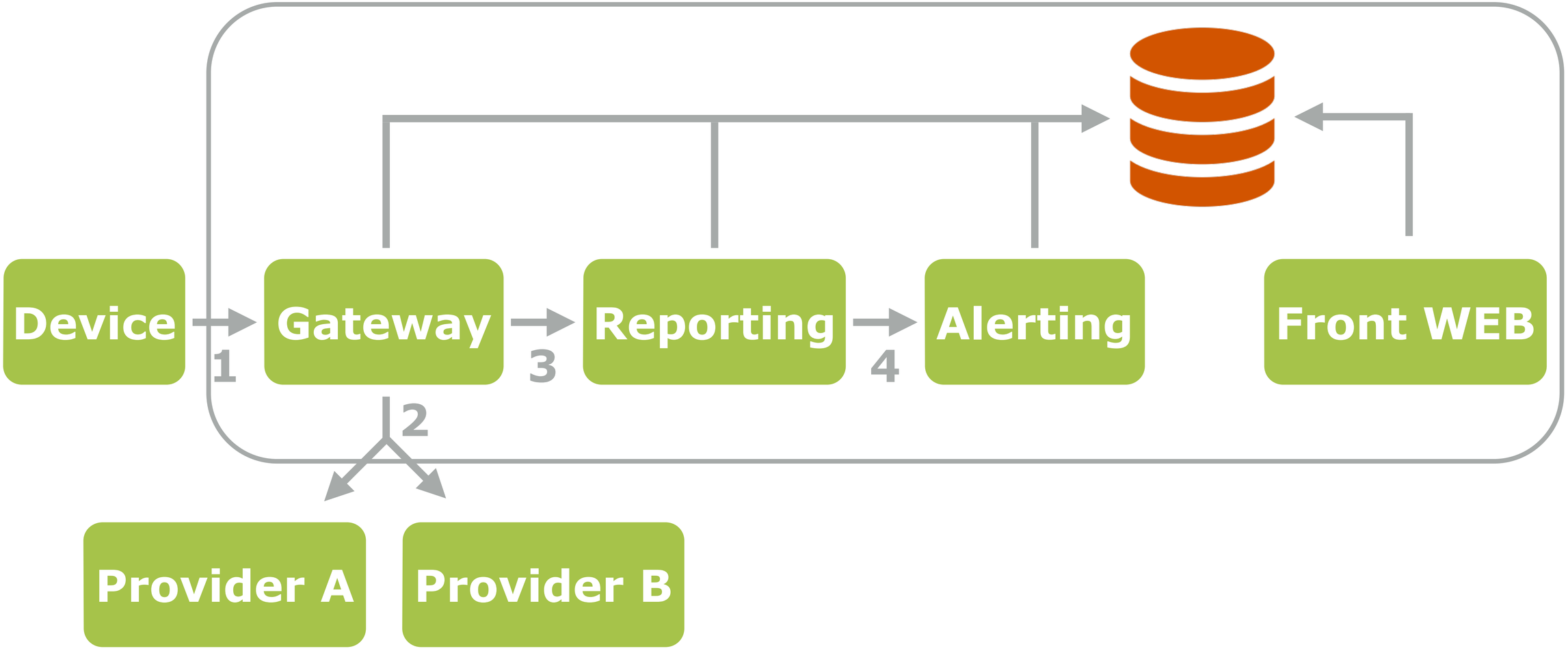
Circuit Breaker



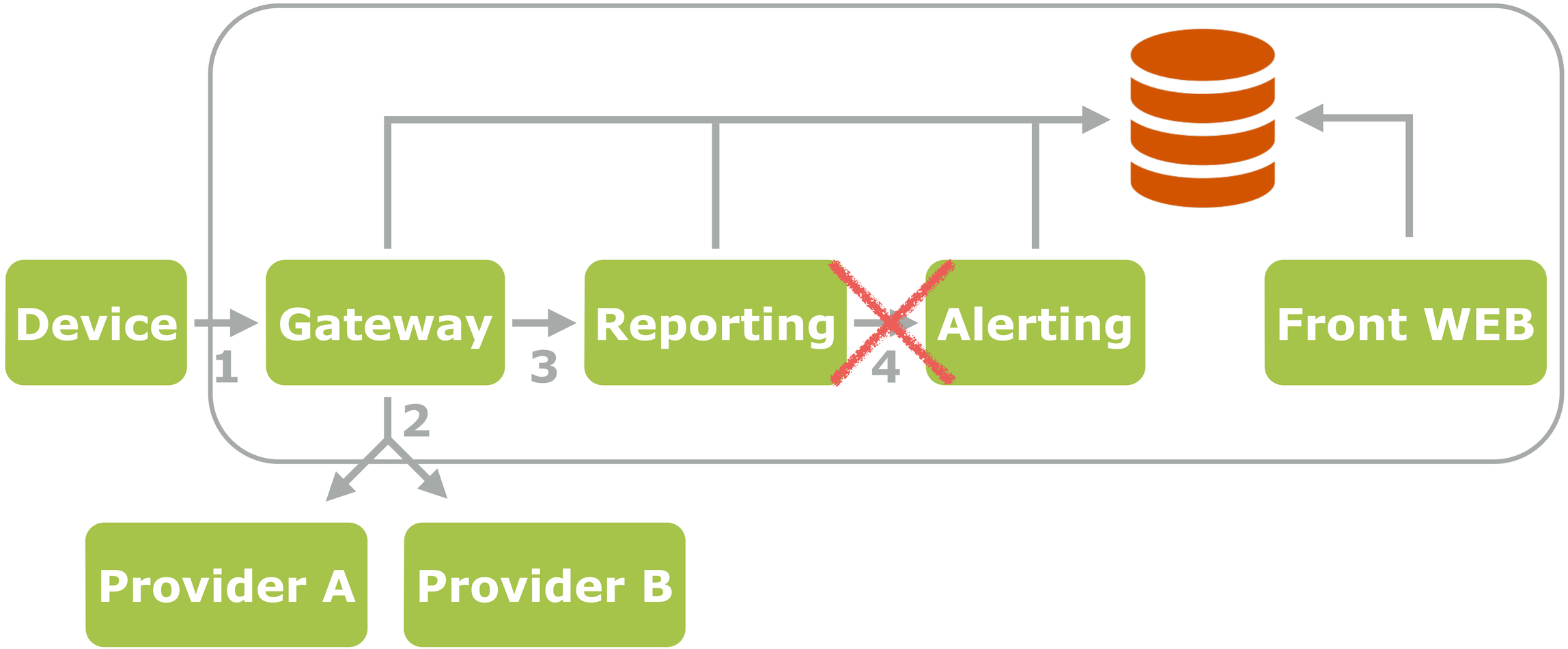
Circuit Breaker



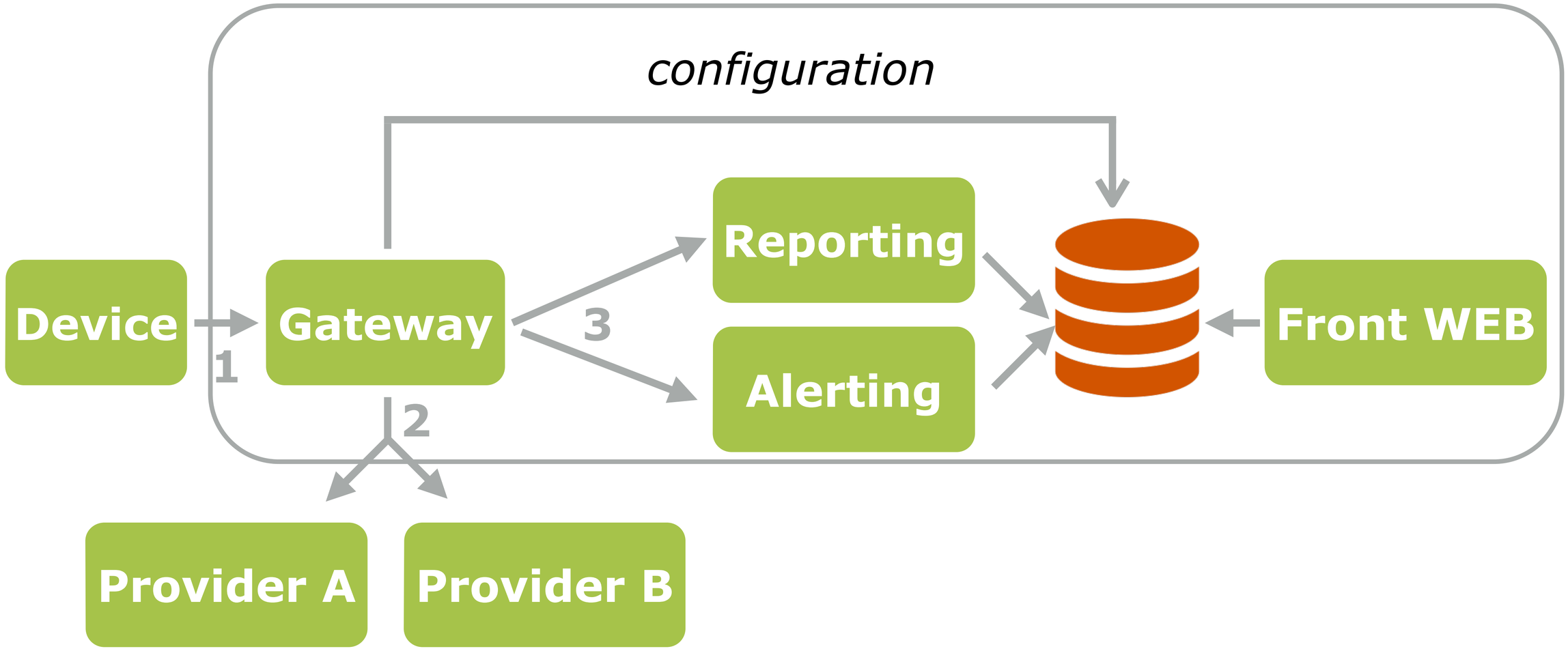
Architecture



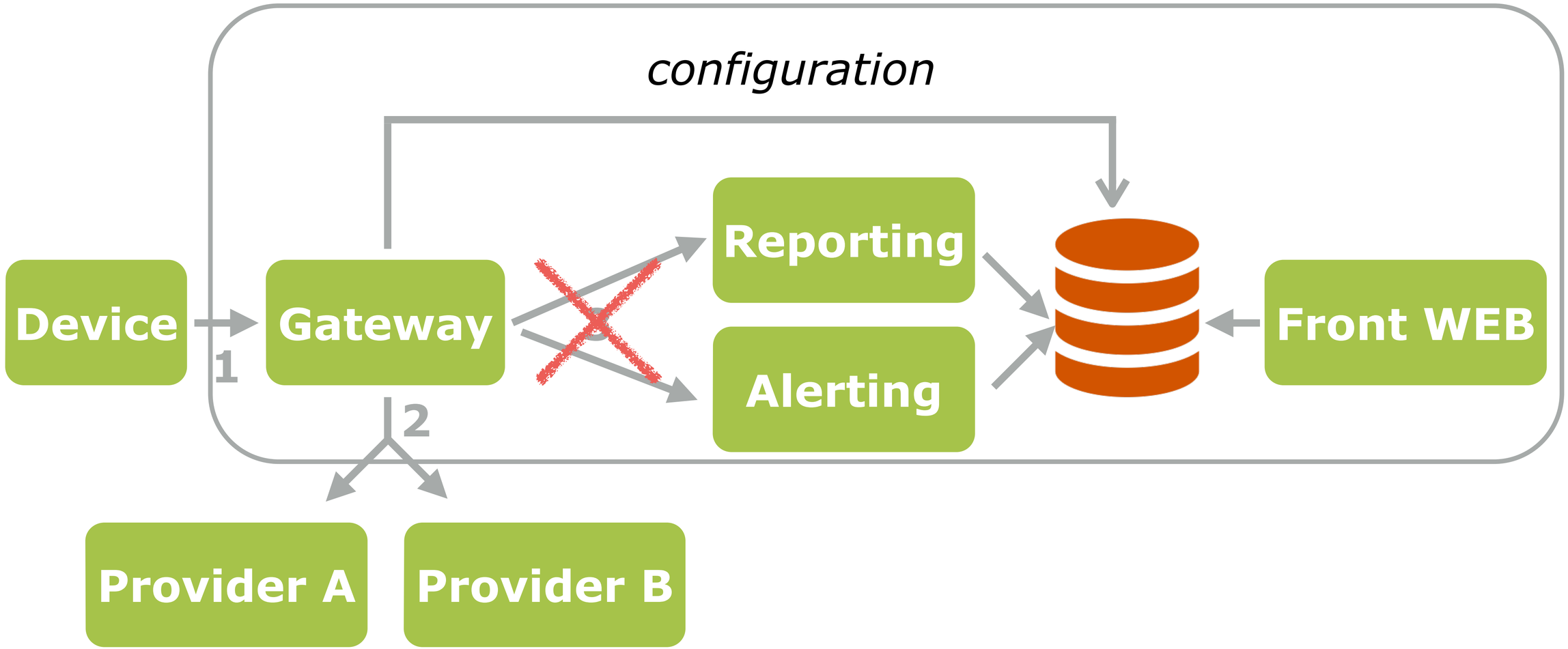
Architecture



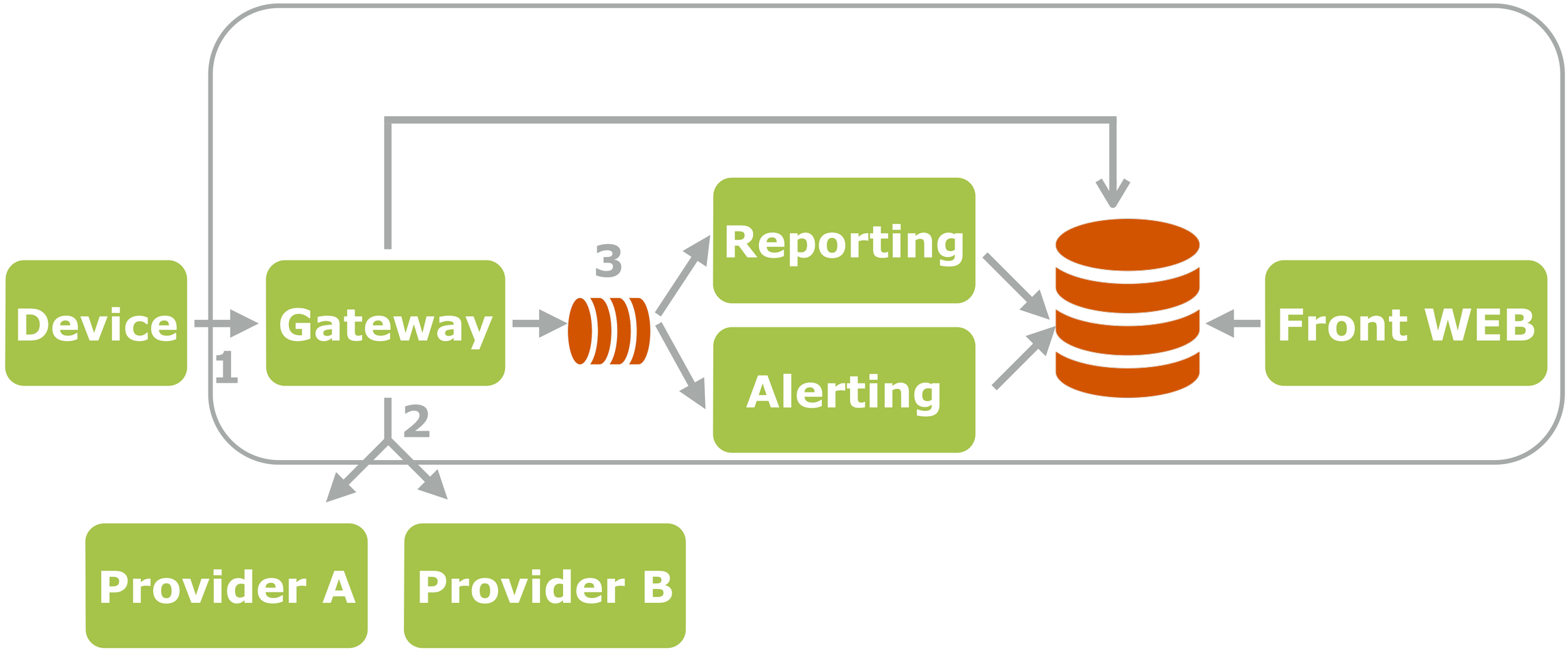
Architecture



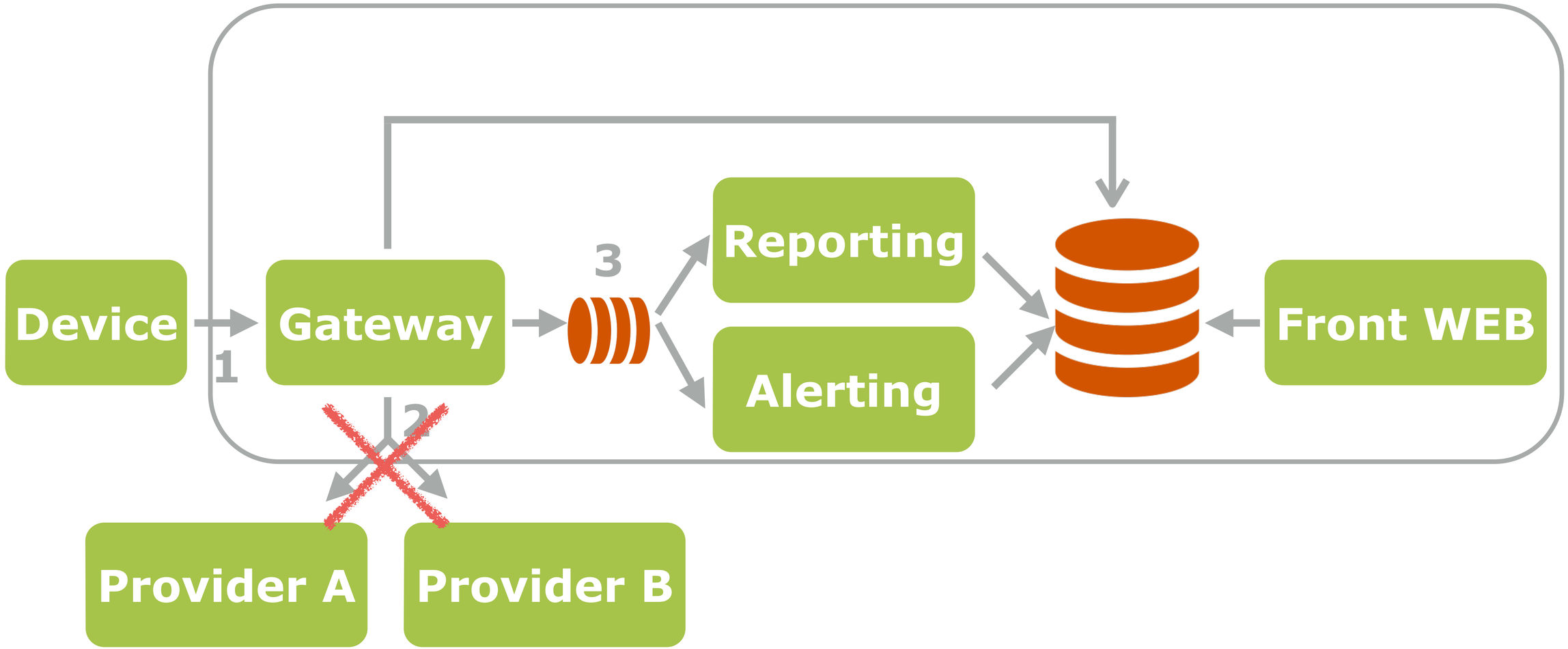
Architecture



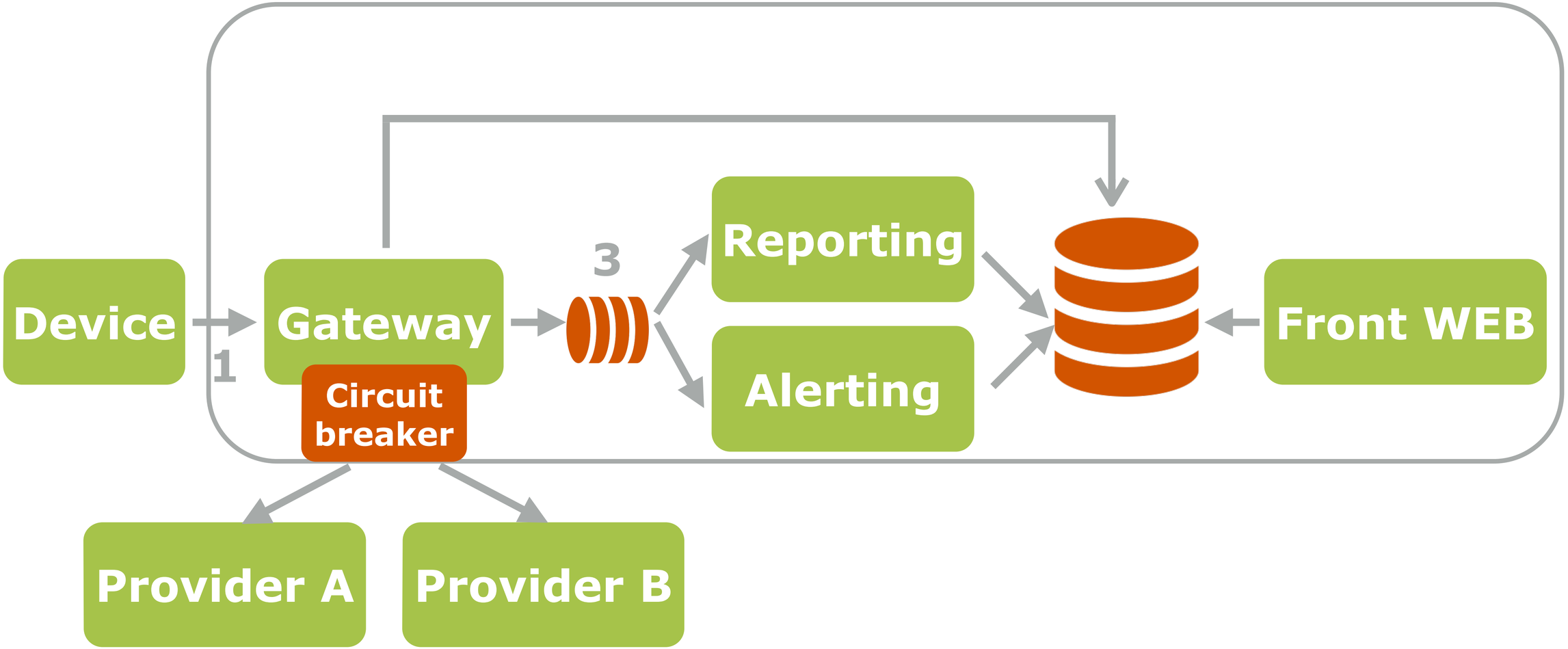
Architecture



Architecture

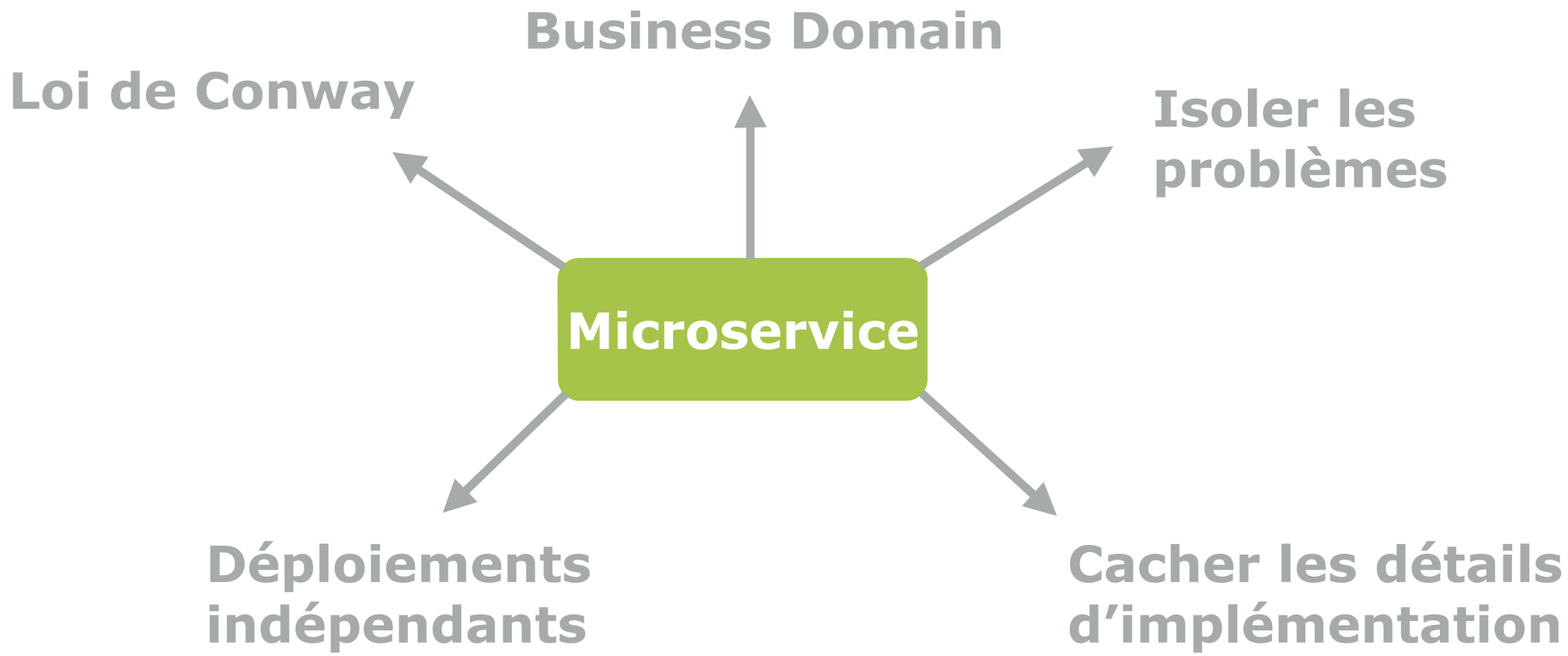


Architecture

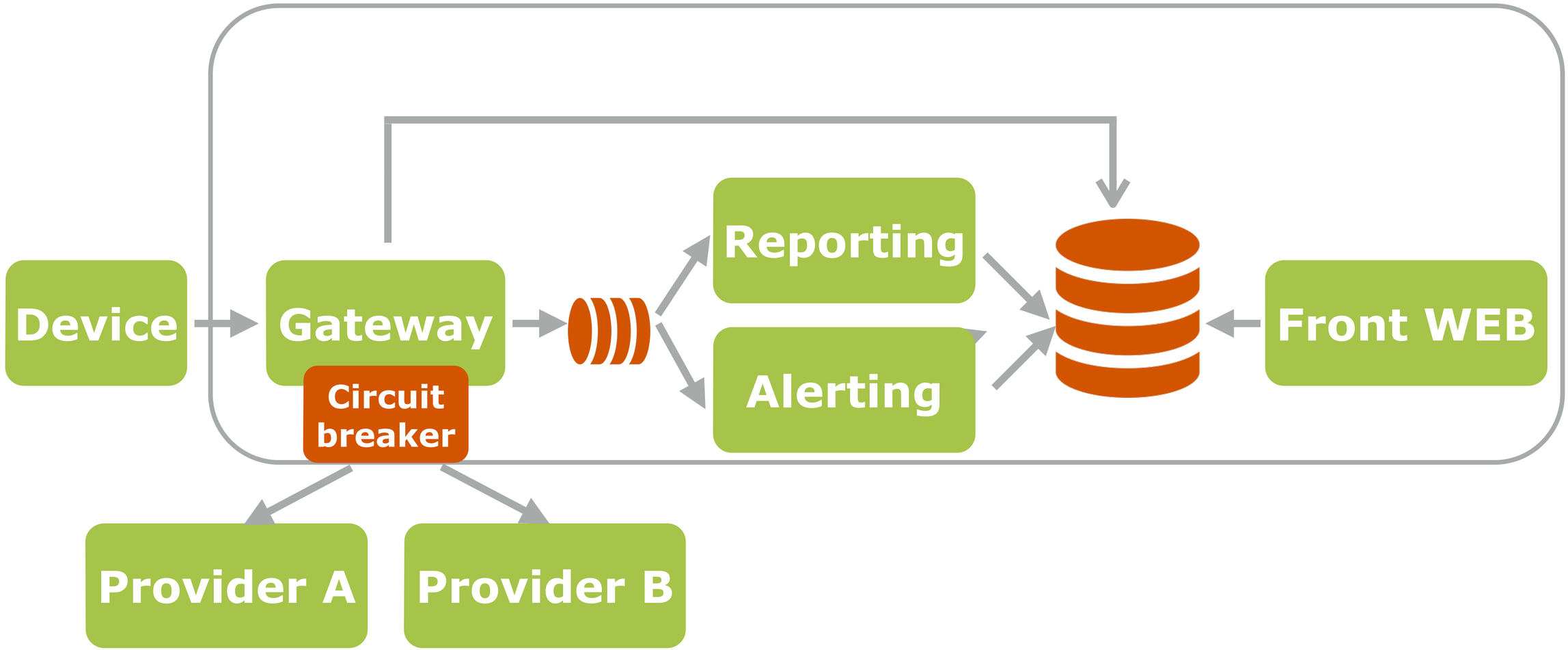


Architecture

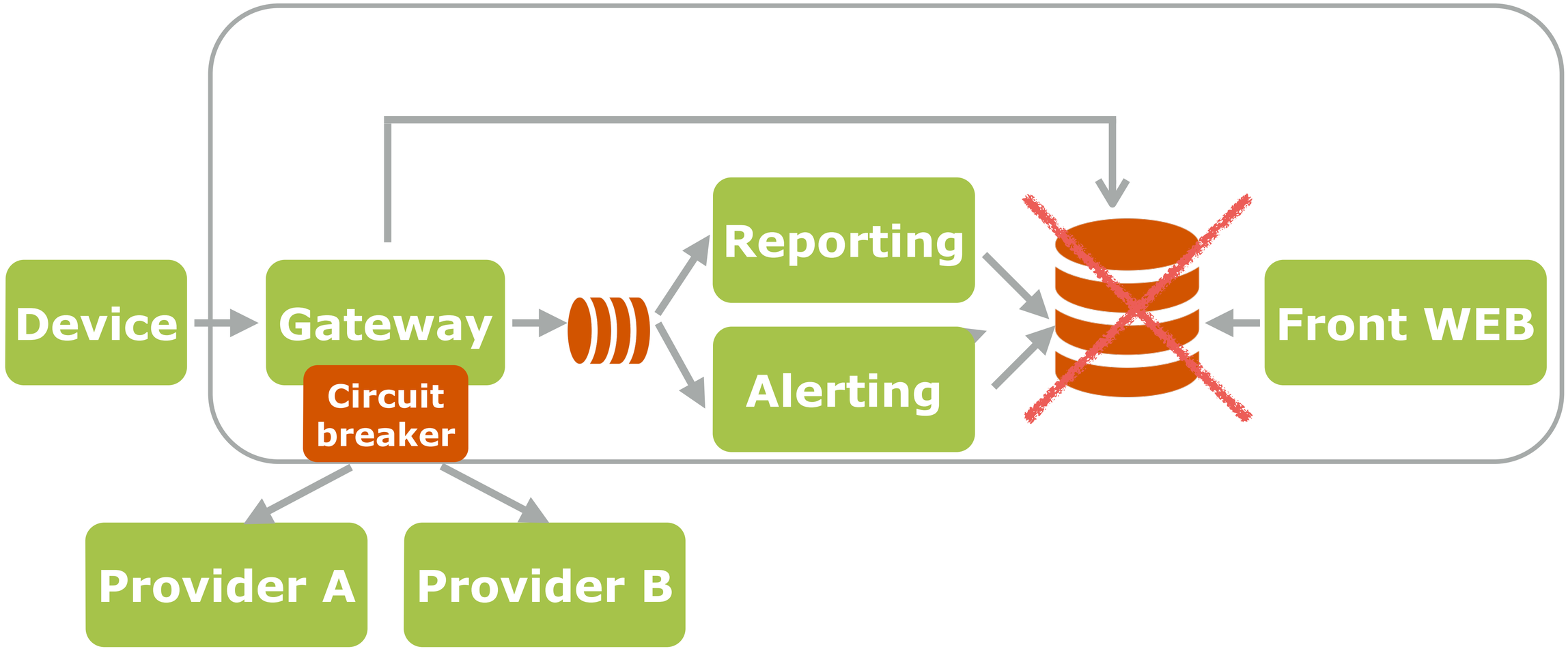
Microservices



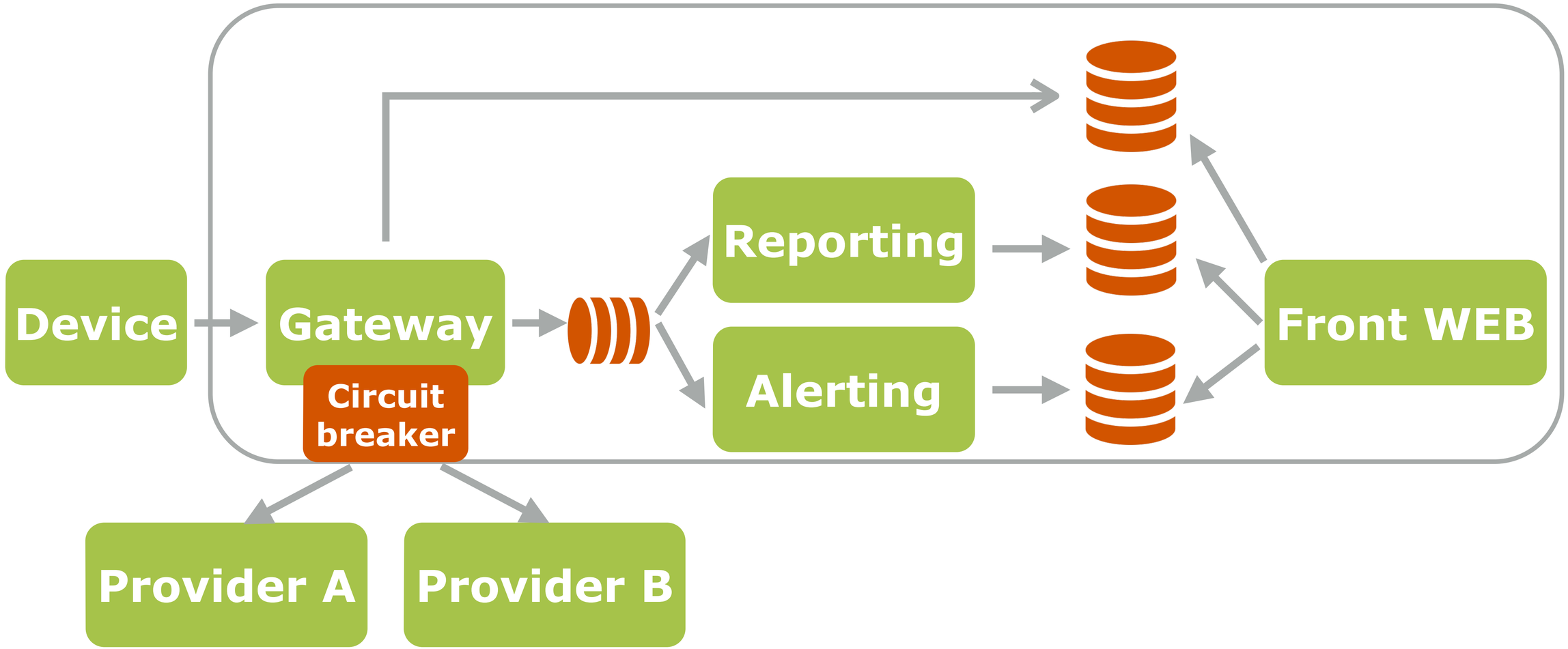
Architecture



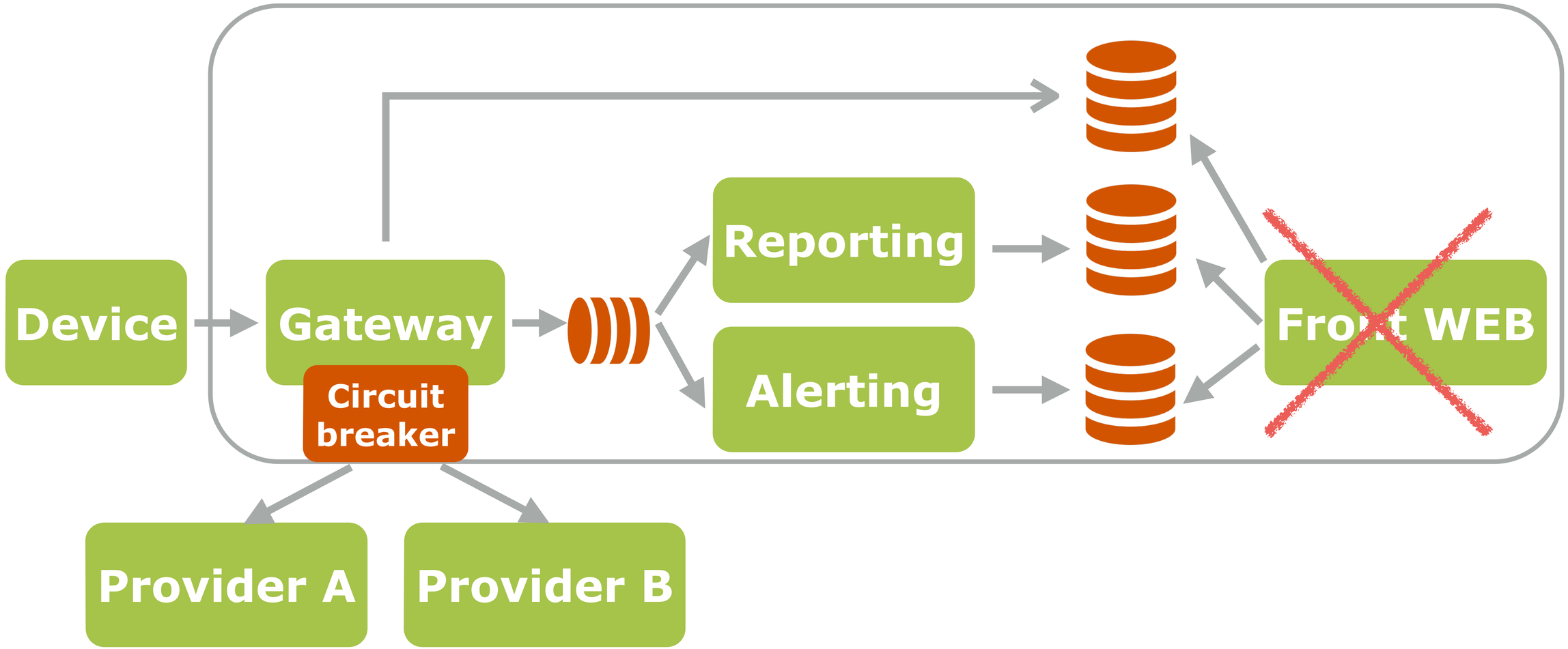
Architecture



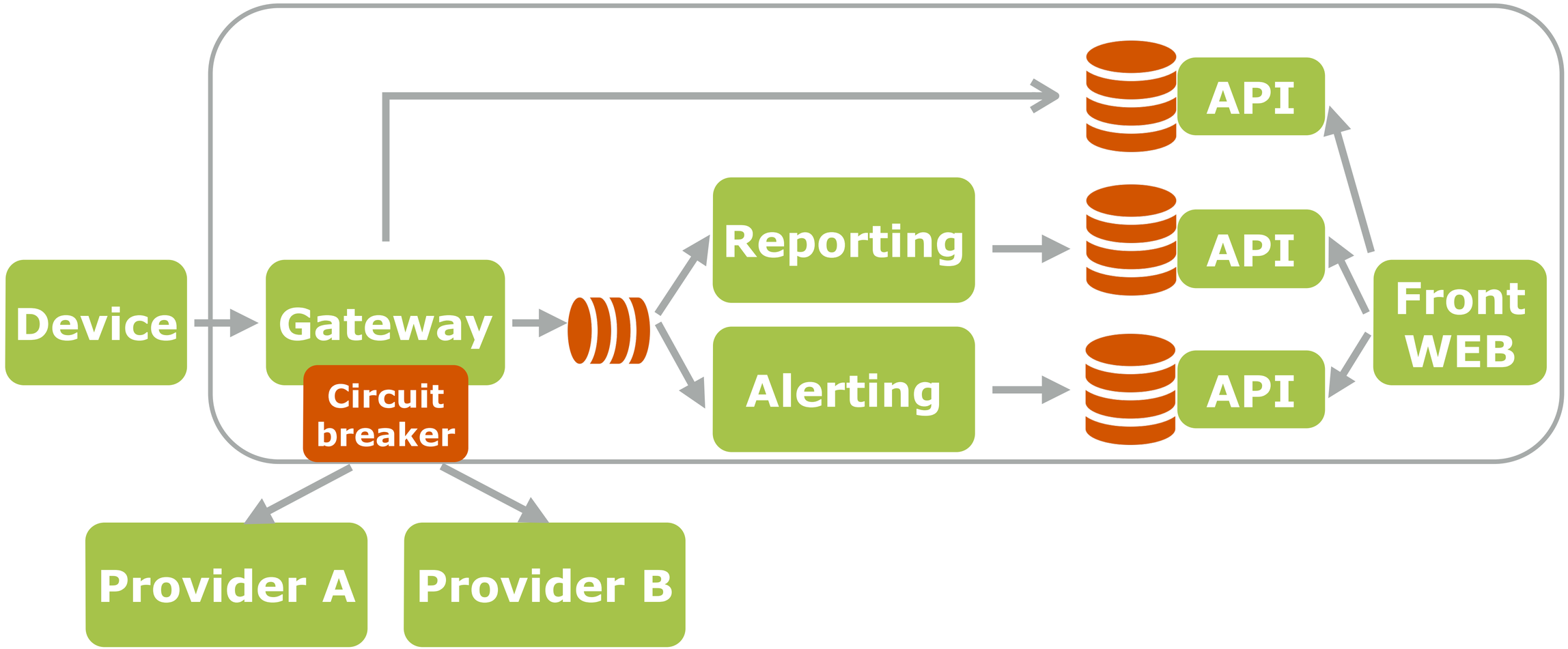
Architecture



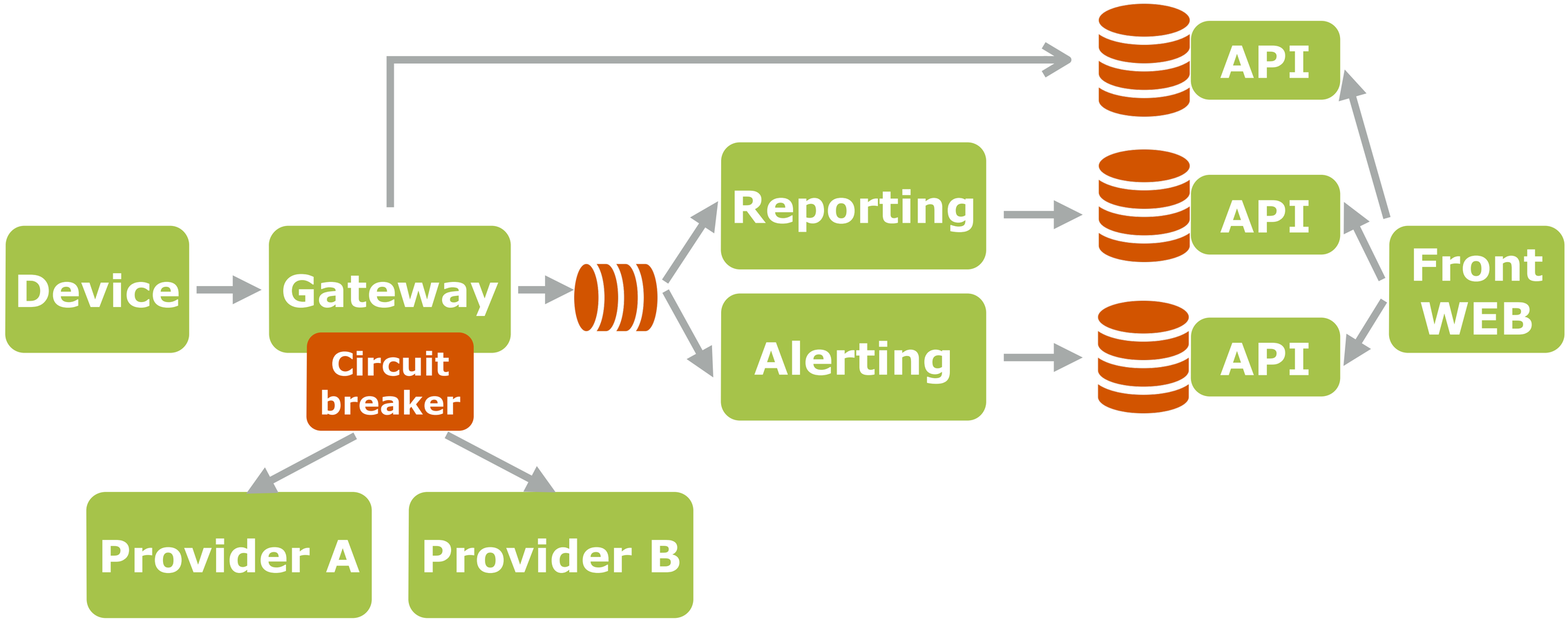
Architecture



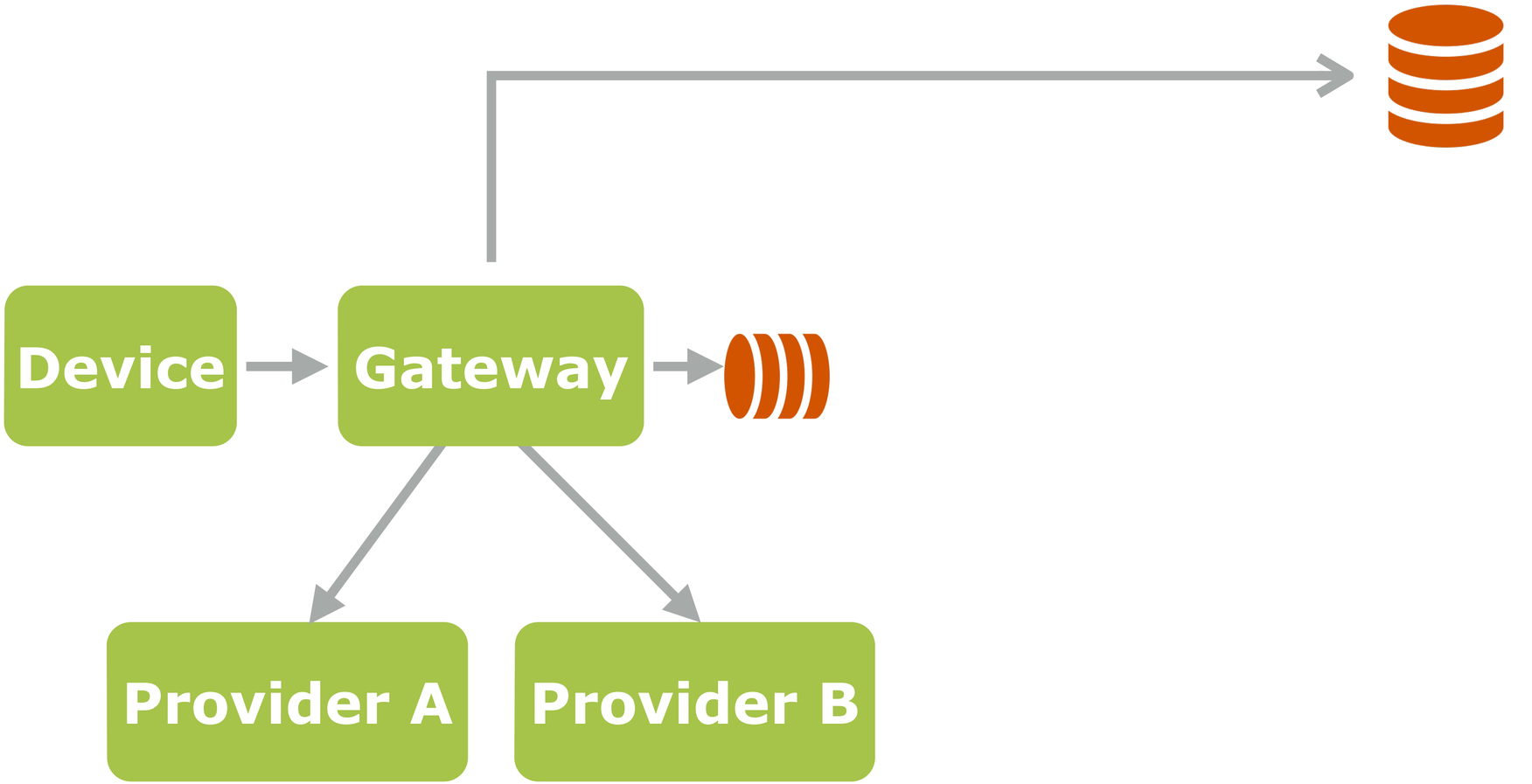
Architecture



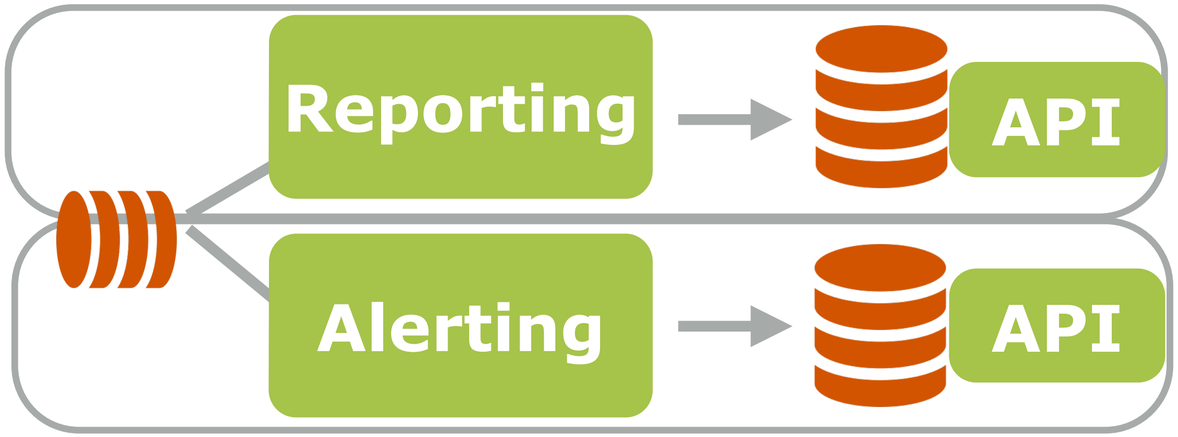
Architecture



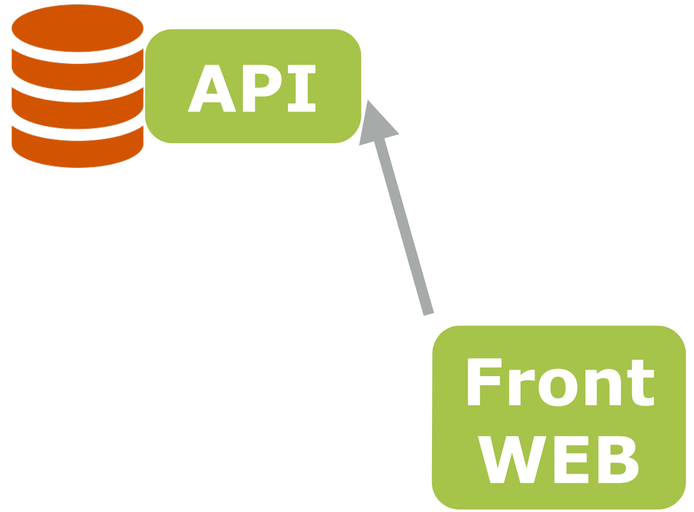
Contexte transactionnel



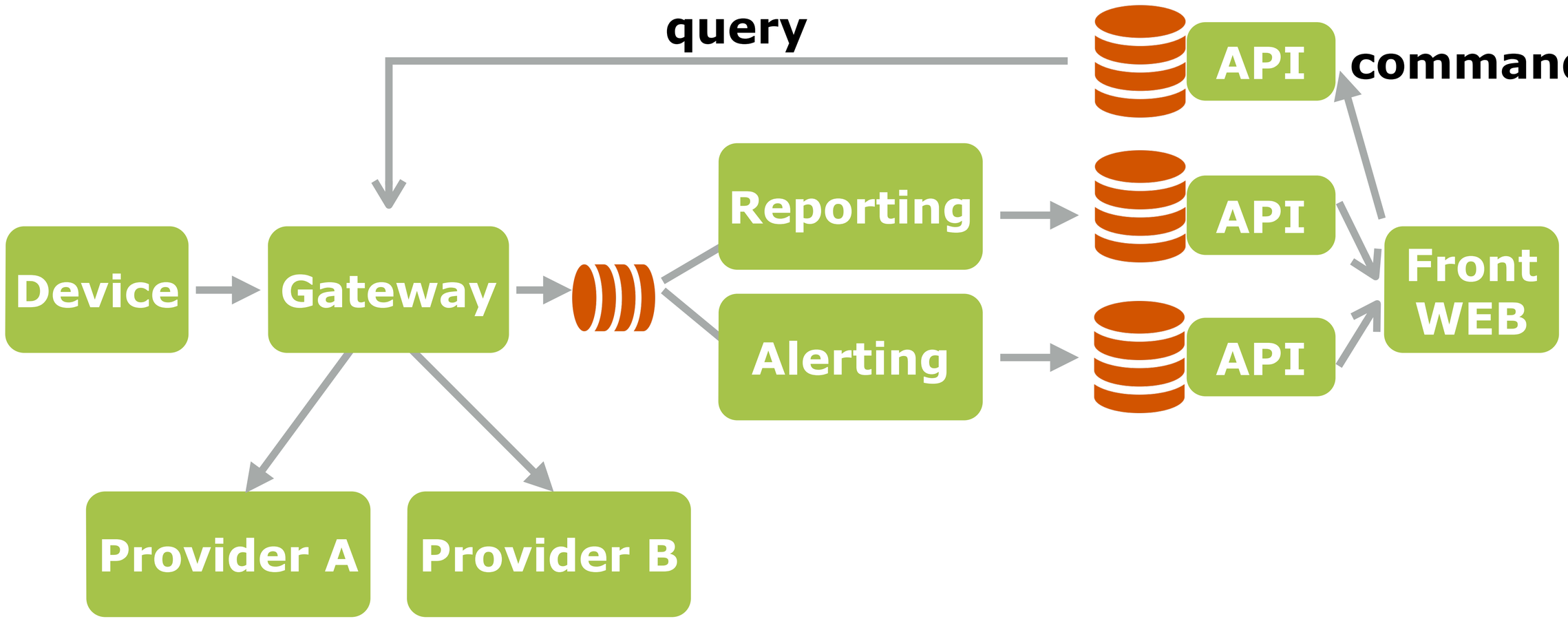
Contexte transactionnel



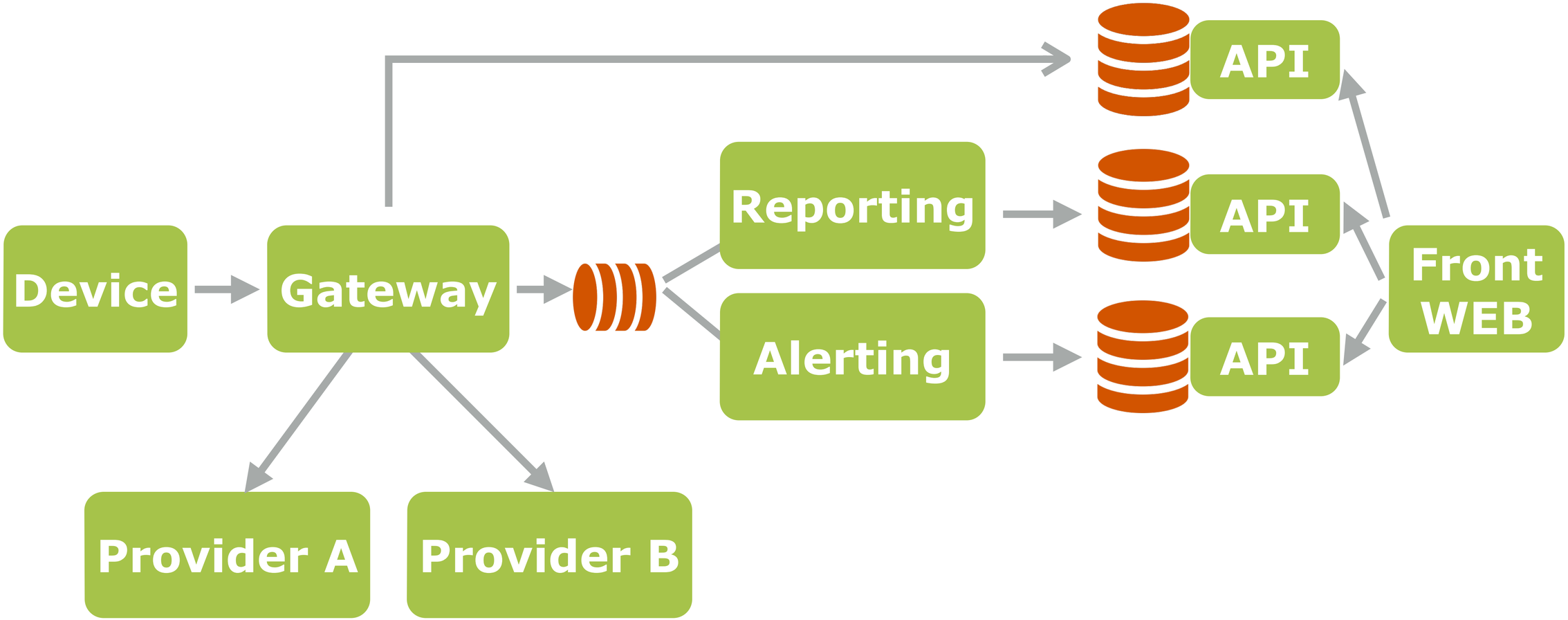
Contexte transactionnel



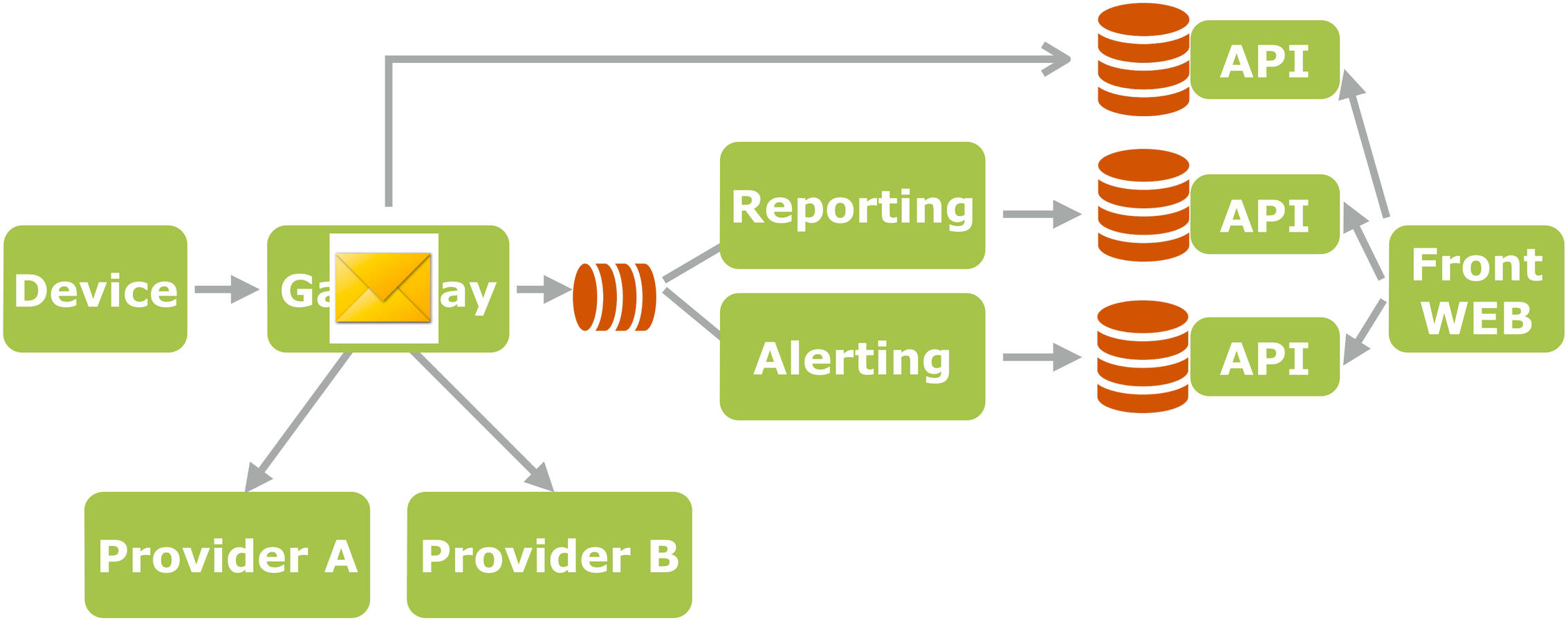
CQRS


query
command
Device
Gateway
Reporting
Alerting
API
API
API
Front
WEB
Provider A
Provider B

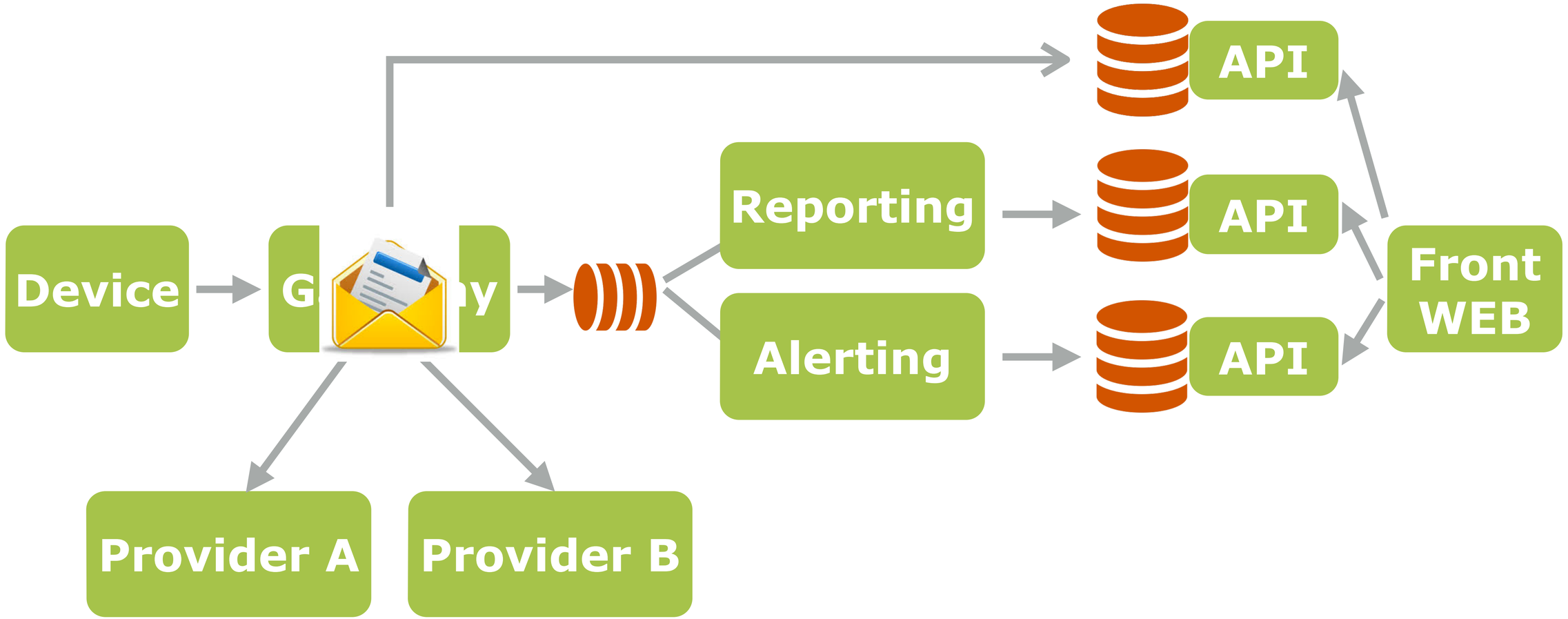
Dynamique du système



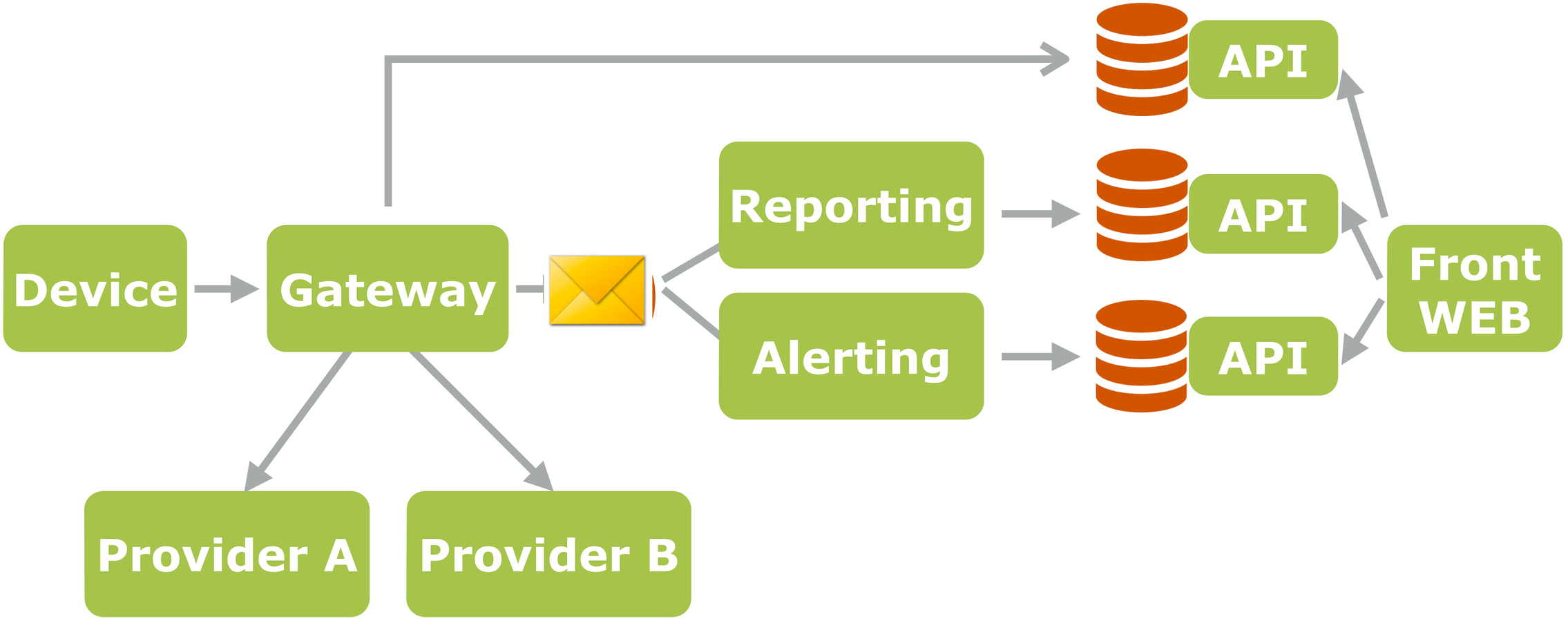
Dynamique du système



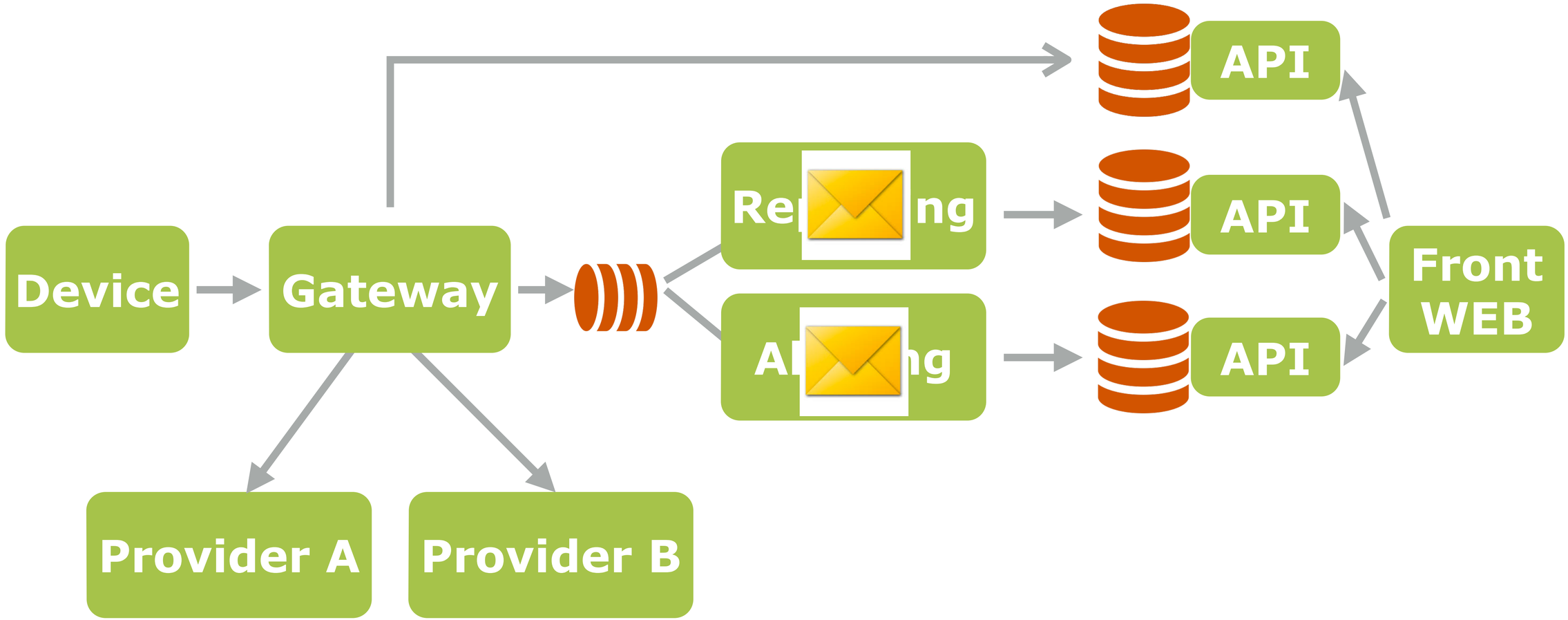
Dynamique du système



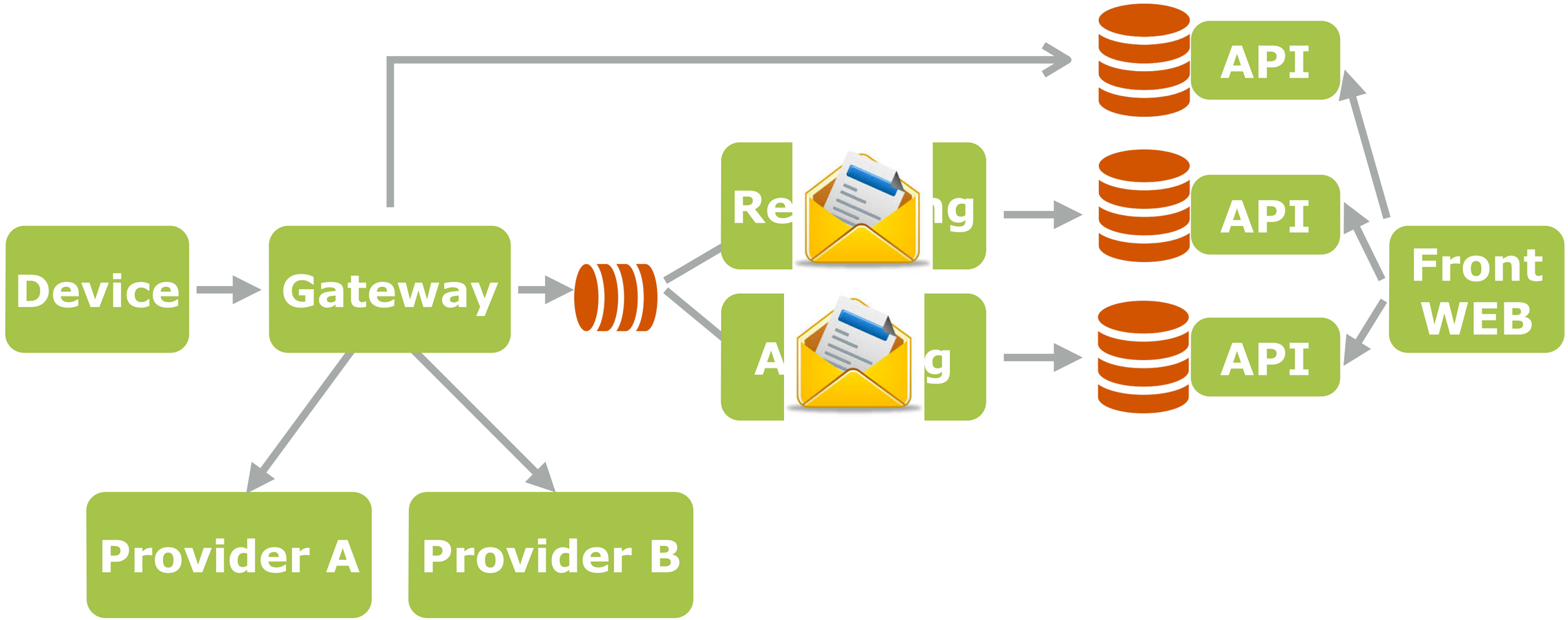
Dynamique du système



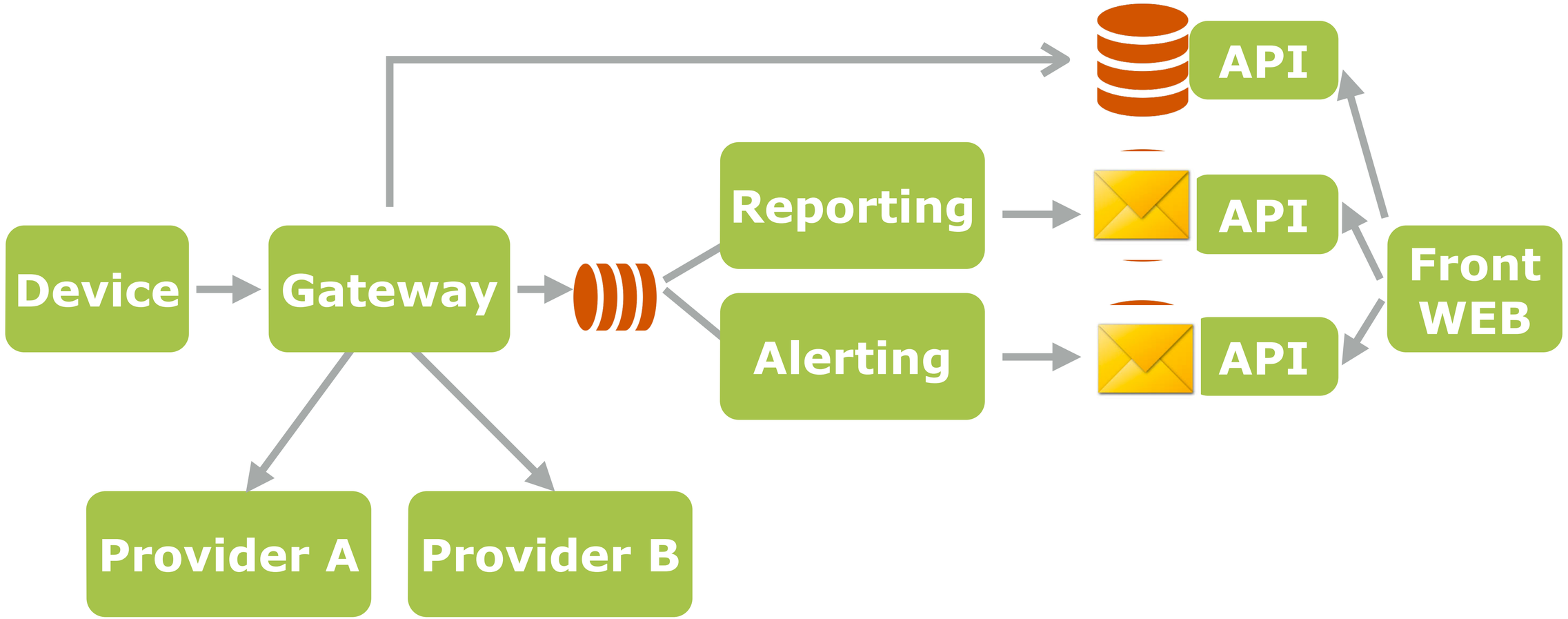
Dynamique du système



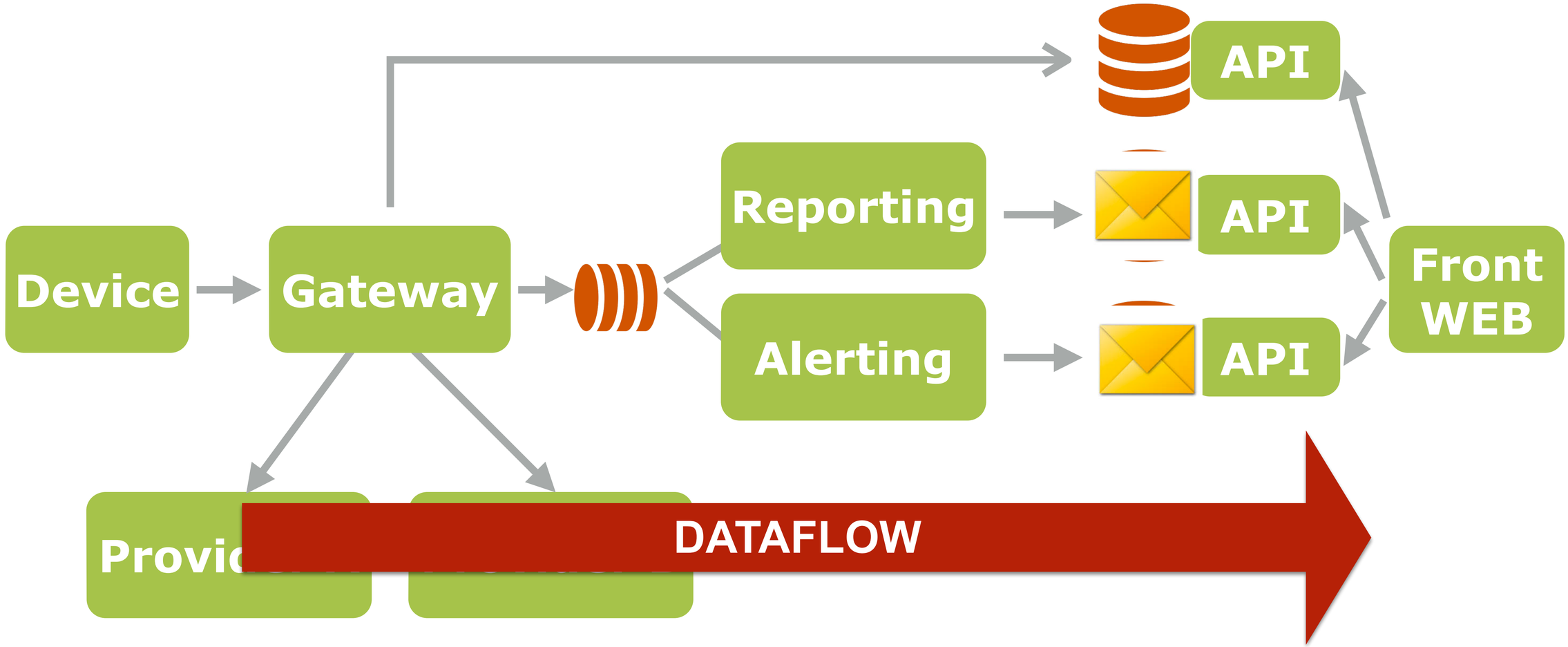
Dynamique du système



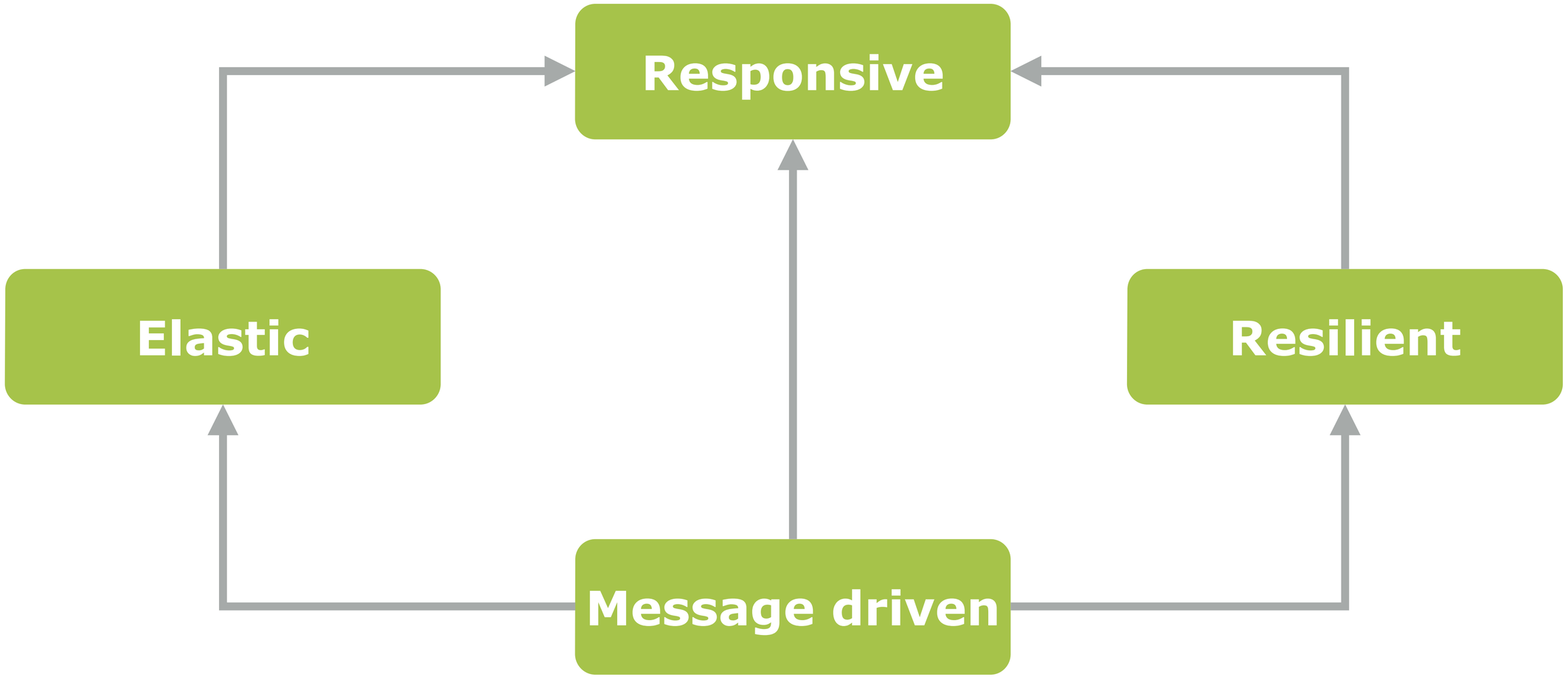
Dynamique du système



Dynamique du système



Reactive Manifesto

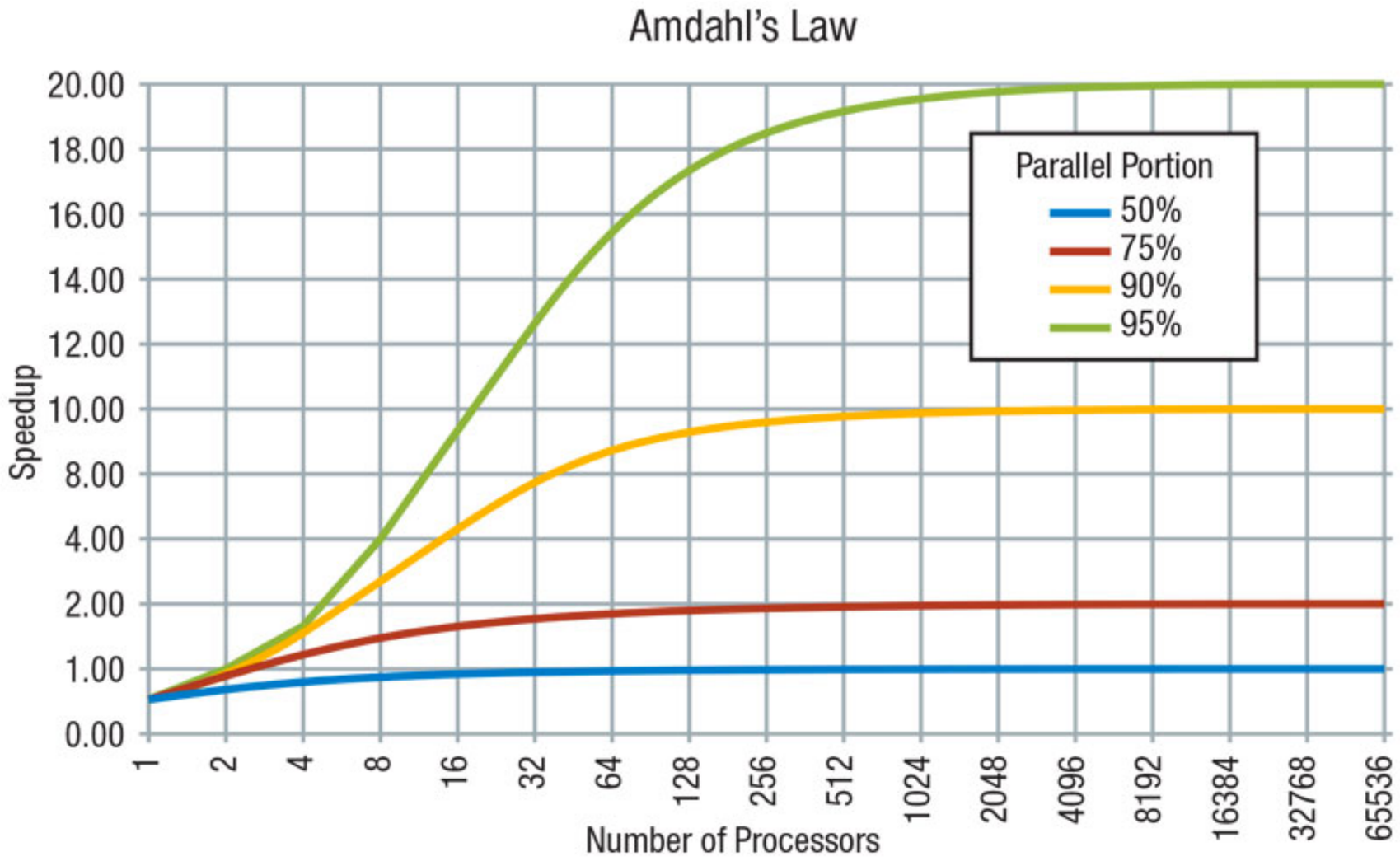


www.reactivemanifesto.org

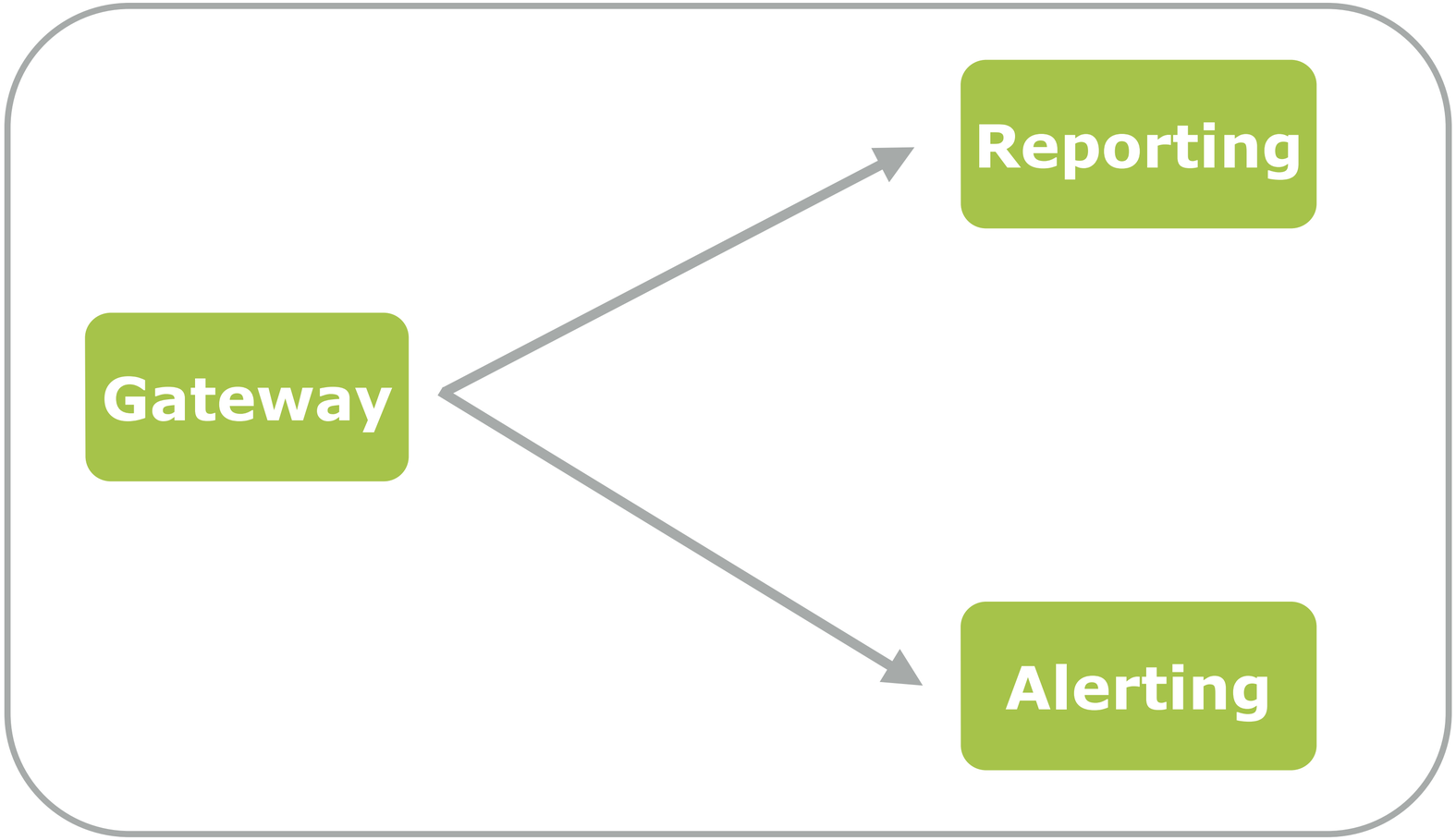
Asynchronisme

```
//asynchrone
def handle(event): Result = {
  val f1 = Future(report(event))
  val f2 = Future(alert(event)))
  Await.result(sequence(f1,f2))
  Success
}

// asynchrone et non bloquant :)
def handle(event): Future[Result] = {
  val f1 = Future(report(event))
  val f2 = Future(alert(event)))
  sequence(f1, f2).map(_ => Success)
}
```

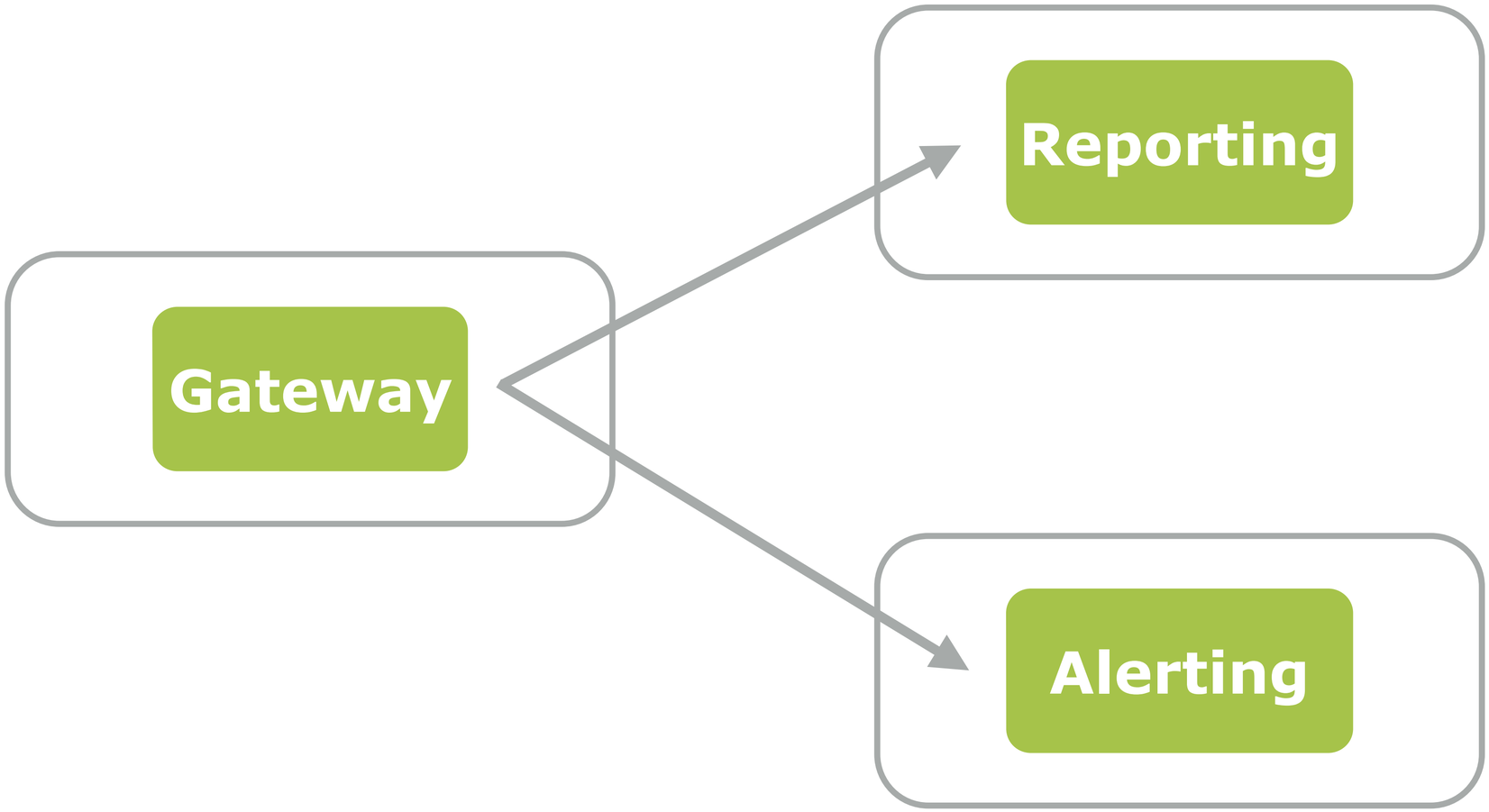
Loi d'Amdahl


Amdahl's Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
4.00
2.00
1.00
0.00
Parallel Portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

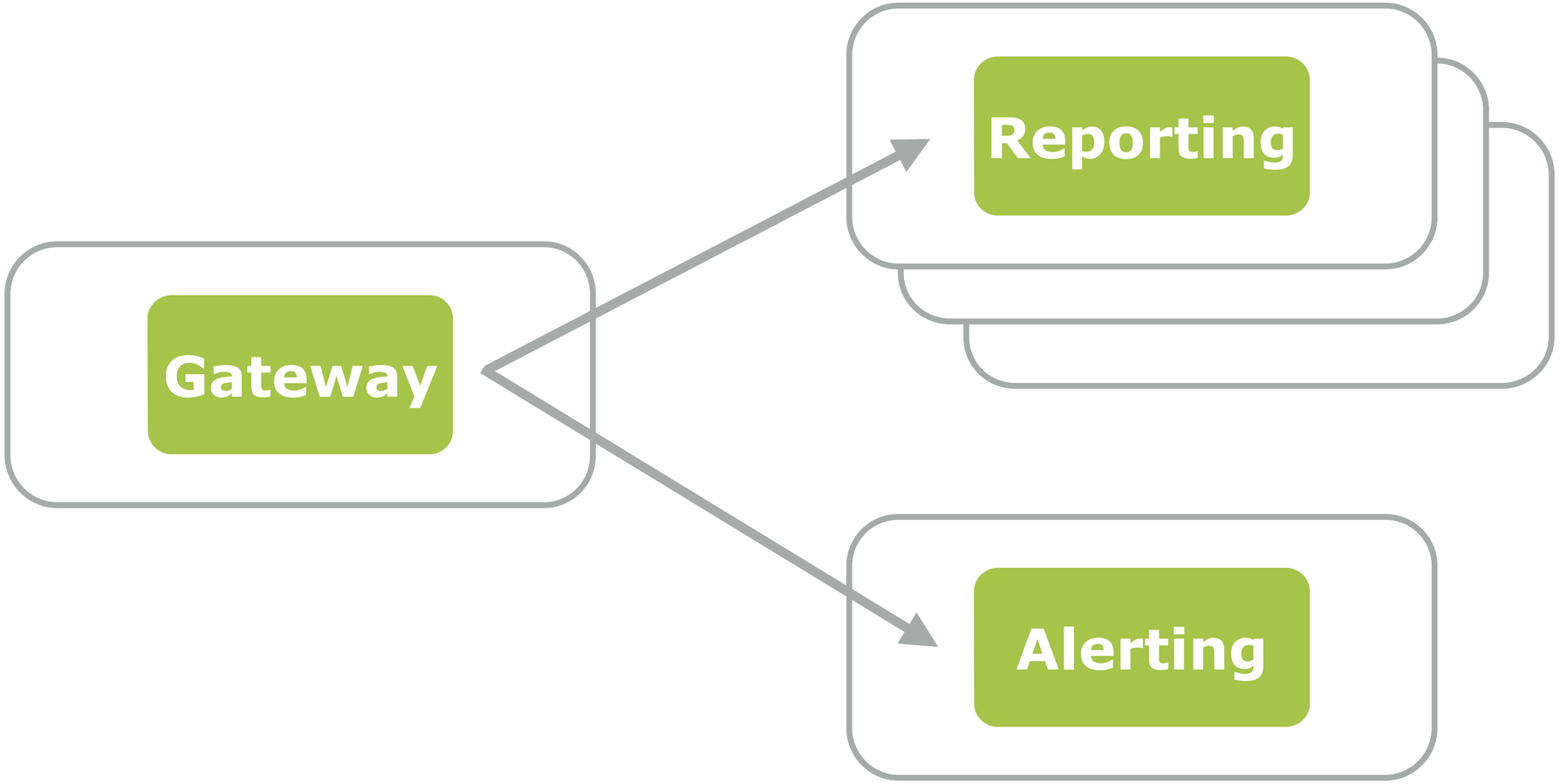
De Parallèle vers ...



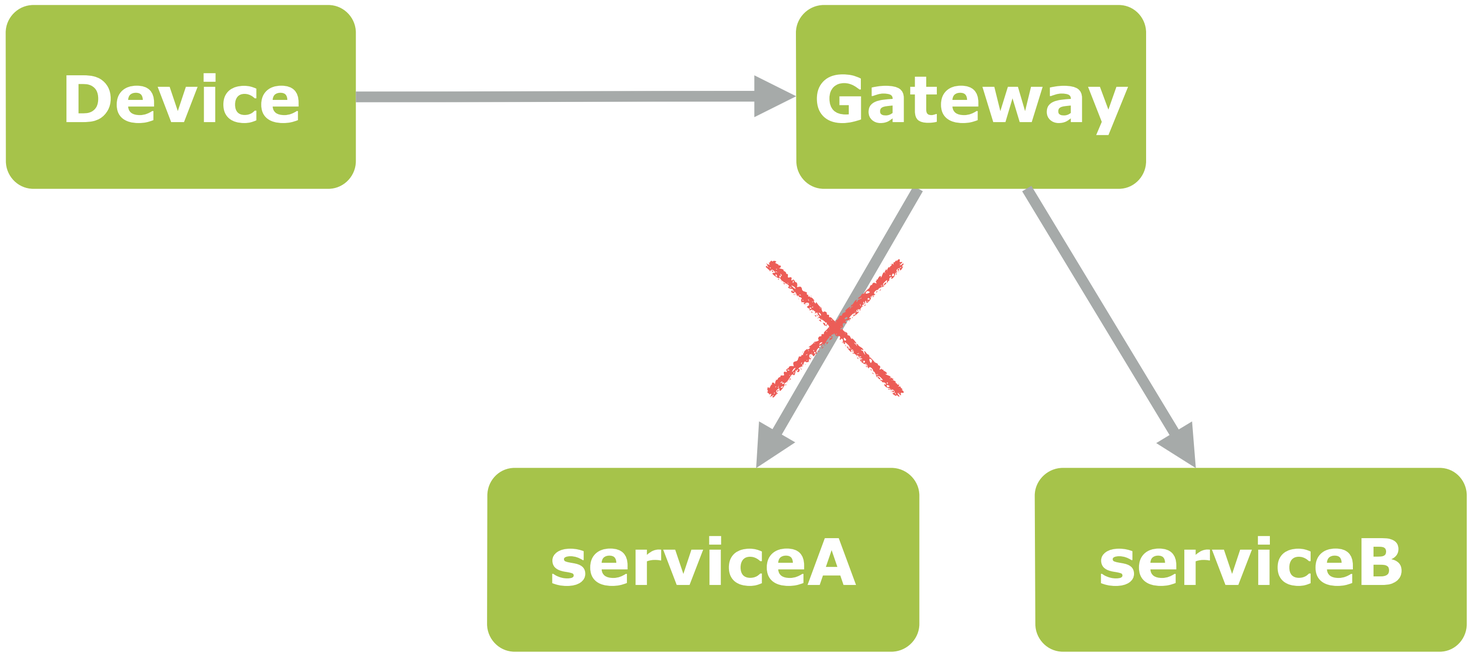
Distribué



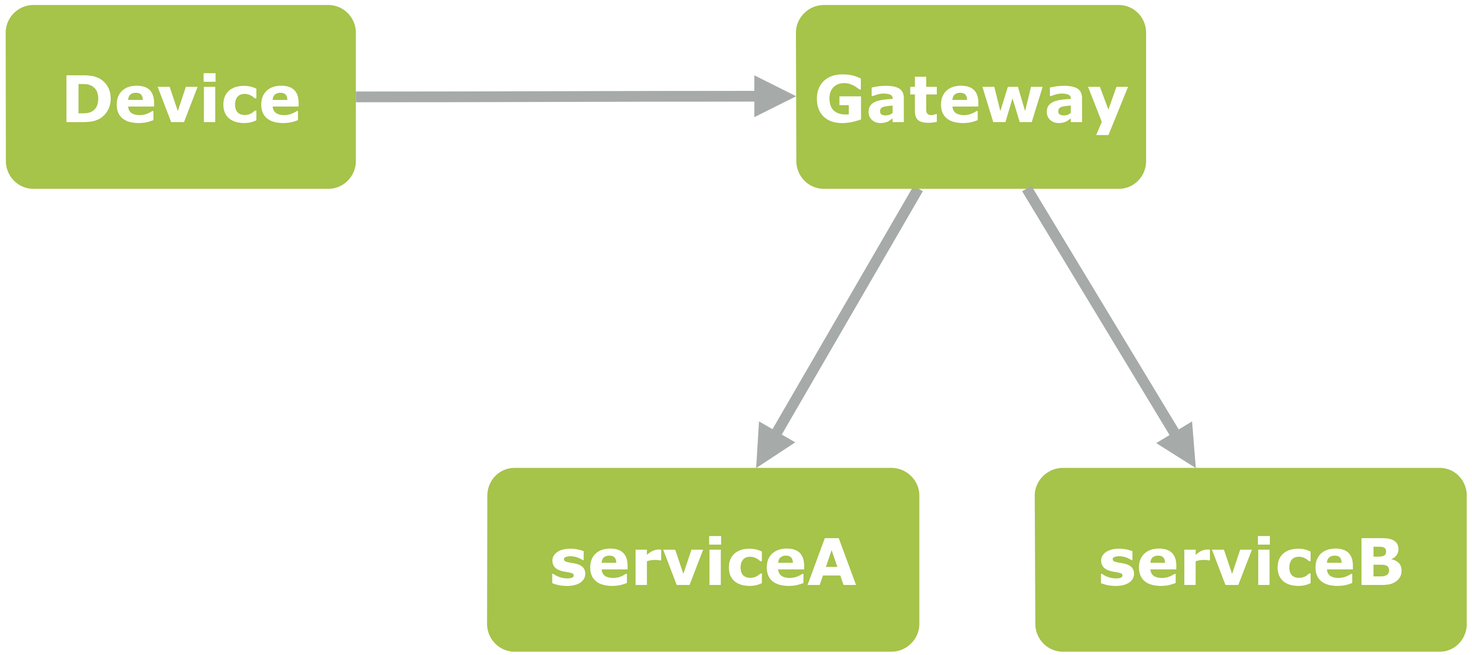
Elasticité



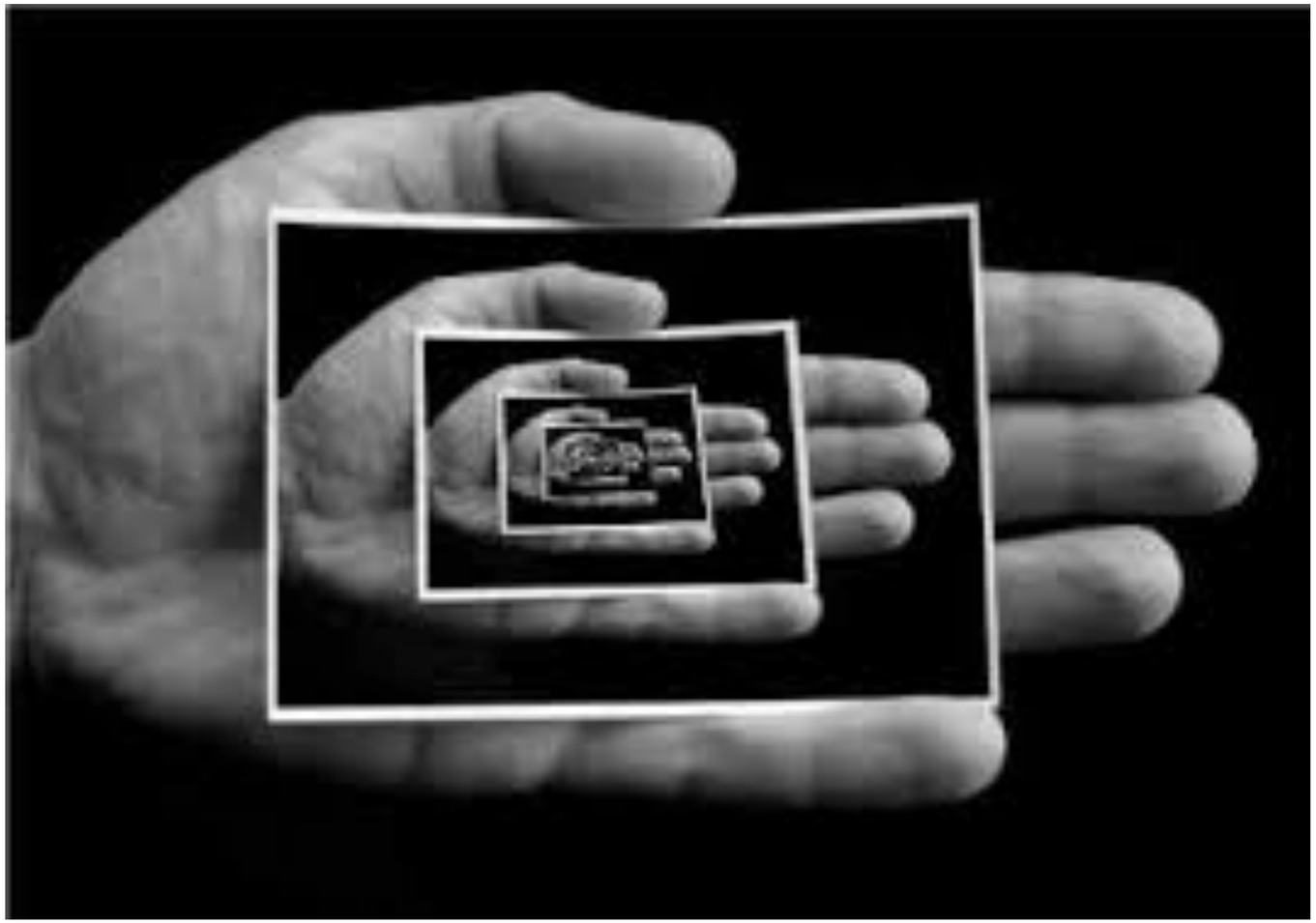
Fonctionnement partiel



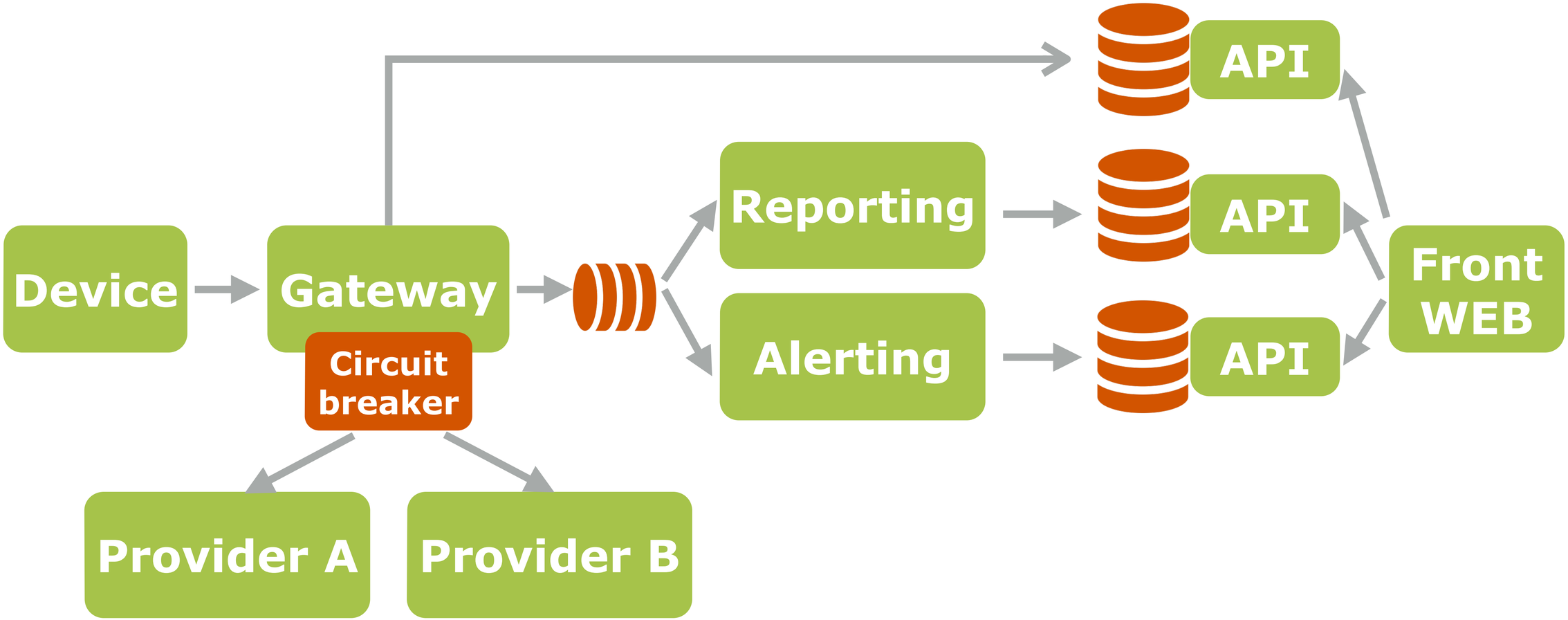
Bounded Latency



Système de systèmes…



Architecture



DEVOXX™ France

Agenda

- Contexte
- Comment concevoir ?
- **Comment développer ?**
- Comment déployer ?
- Comment monitorer ?

DEVOXX™ France

Framework asynchrone JVM

La programmation concurrente est difficile.

Framework asynchrone JVM

La programmation concurrente est difficile.

Callbacks?

Framework asynchrone JVM

La programmation concurrente est difficile.

ExecutorService?

Framework asynchrone JVM

La programmation concurrente est difficile.

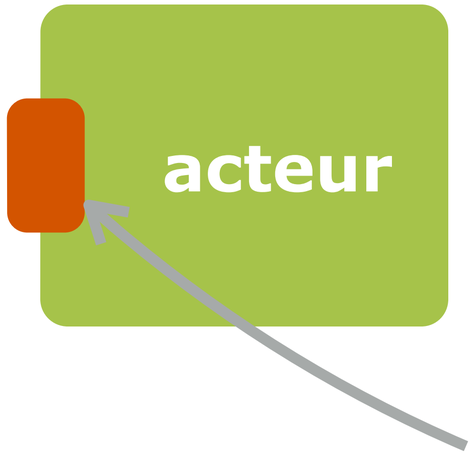
Modèle acteur

Framework asynchrone JVM

Modèle acteur

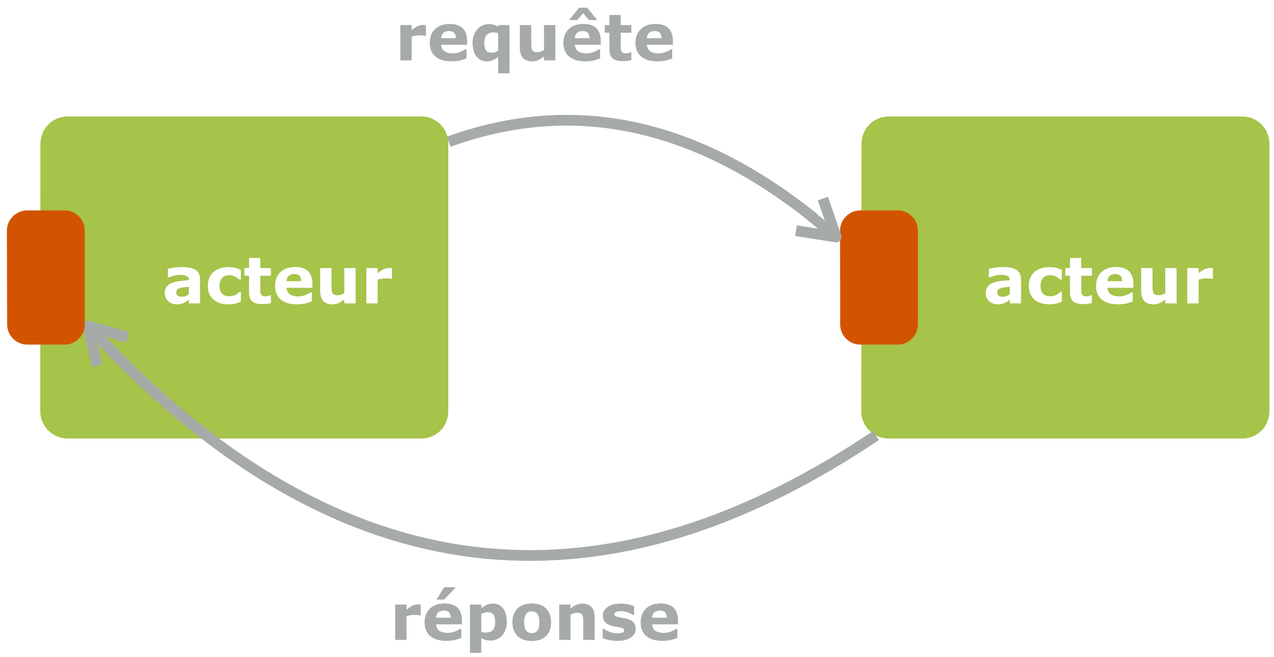
Modèle acteur

Modèle acteur

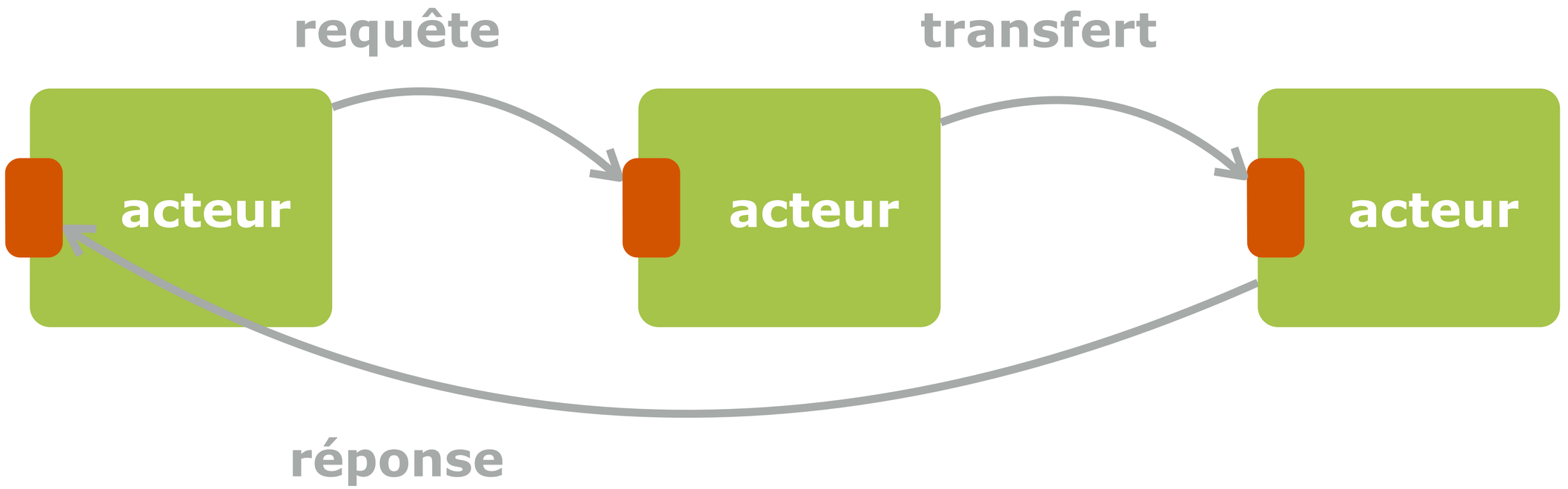


la boîte aux lettres
seul élément public d'un acteur

Modèle acteur

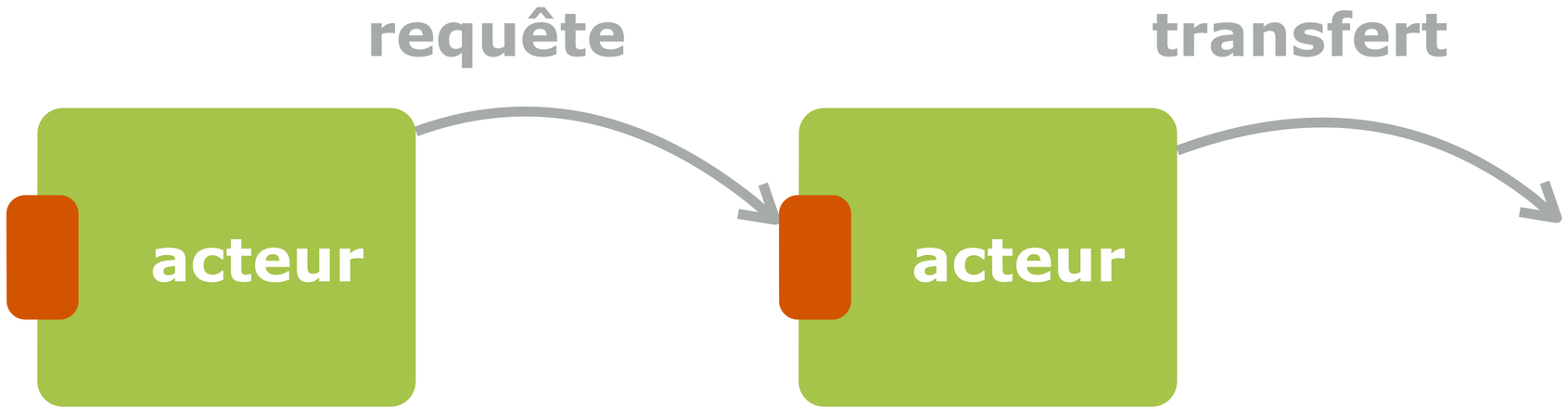


Modèle acteur

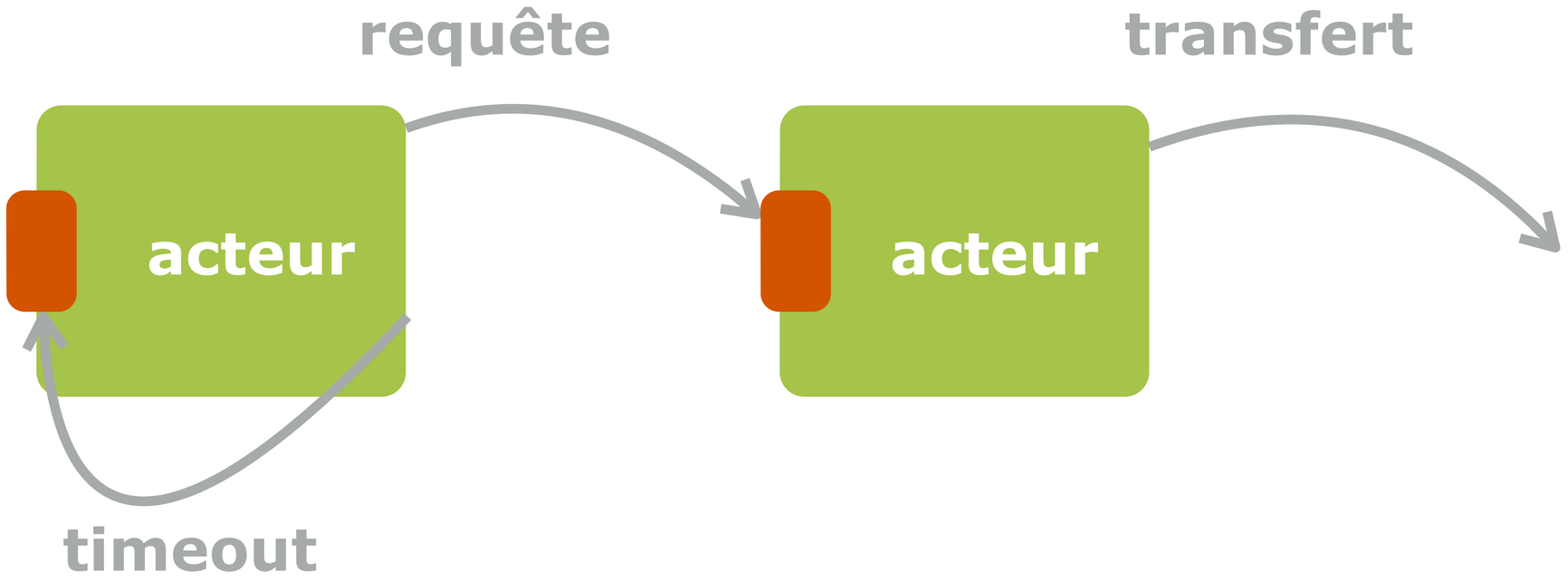


Modèle acteur

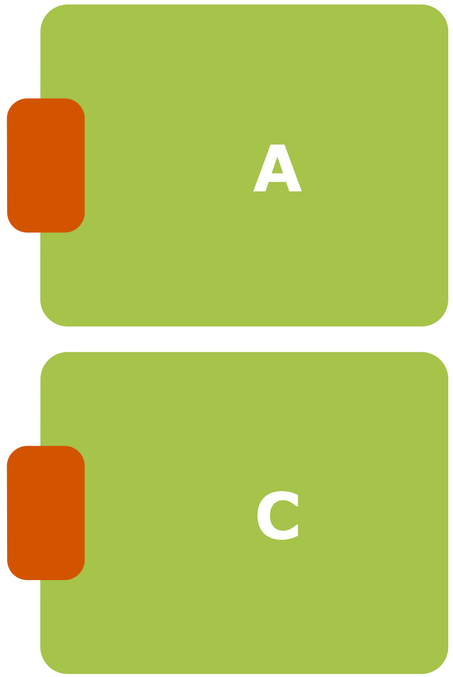
Modèle acteur

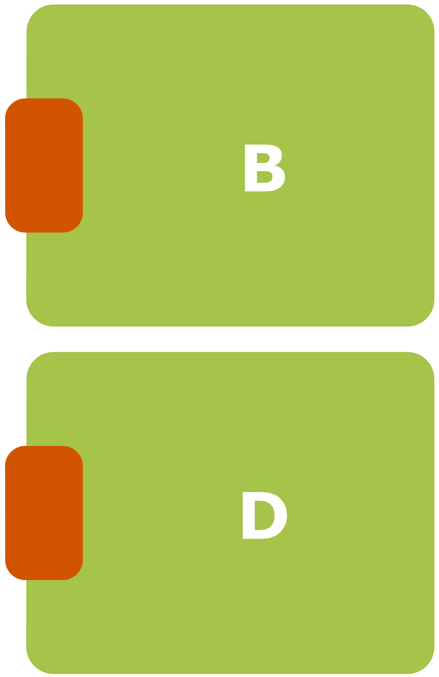


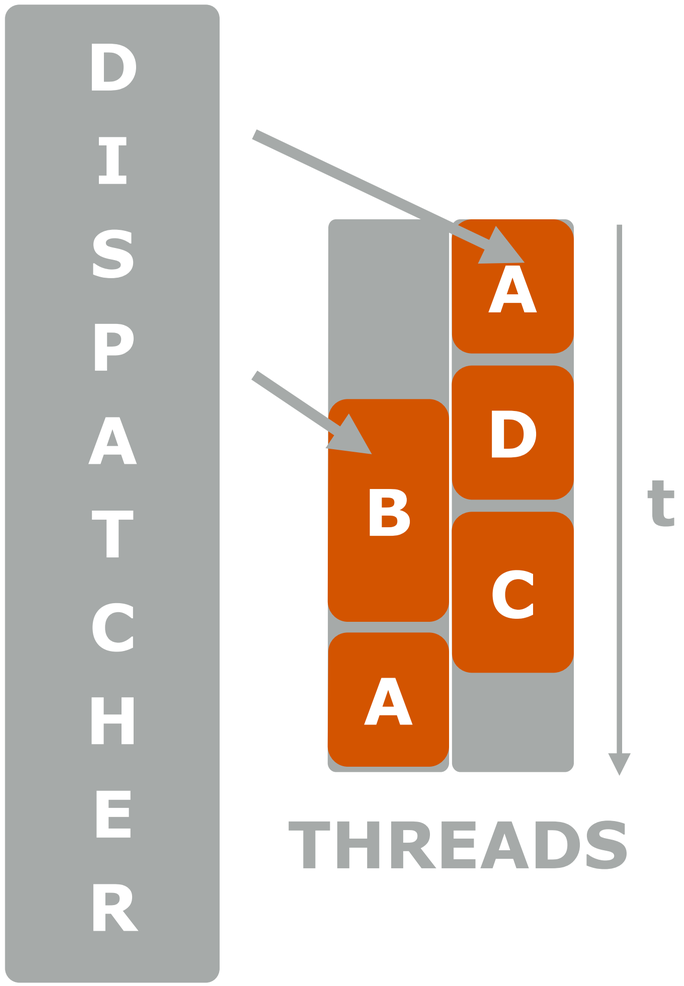
Modèle acteur



Modèle acteur





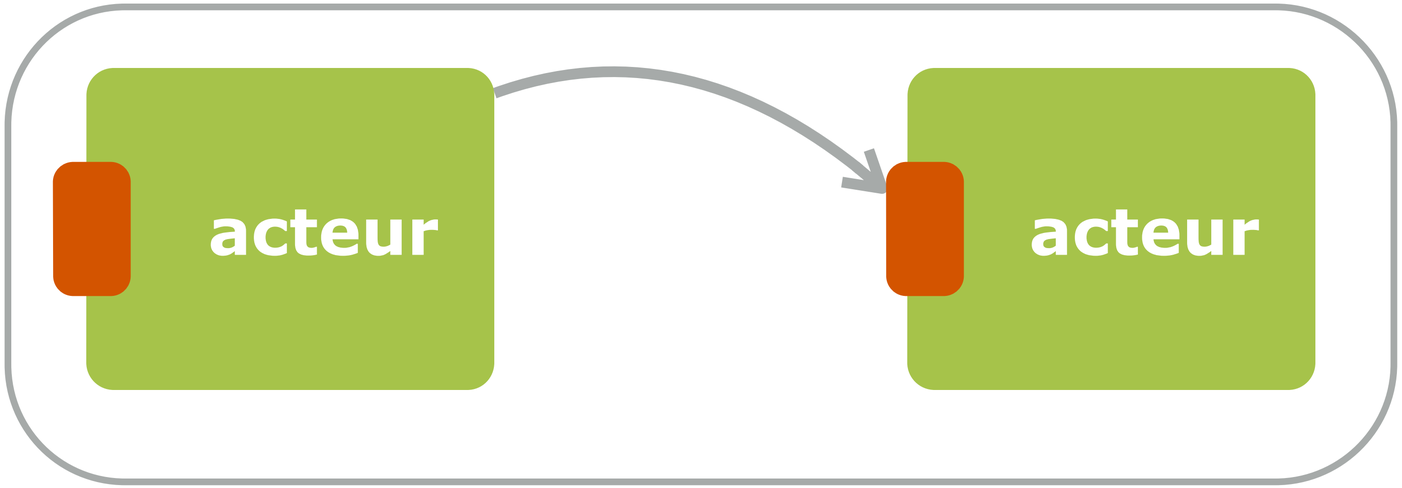


DEVOXX™ France

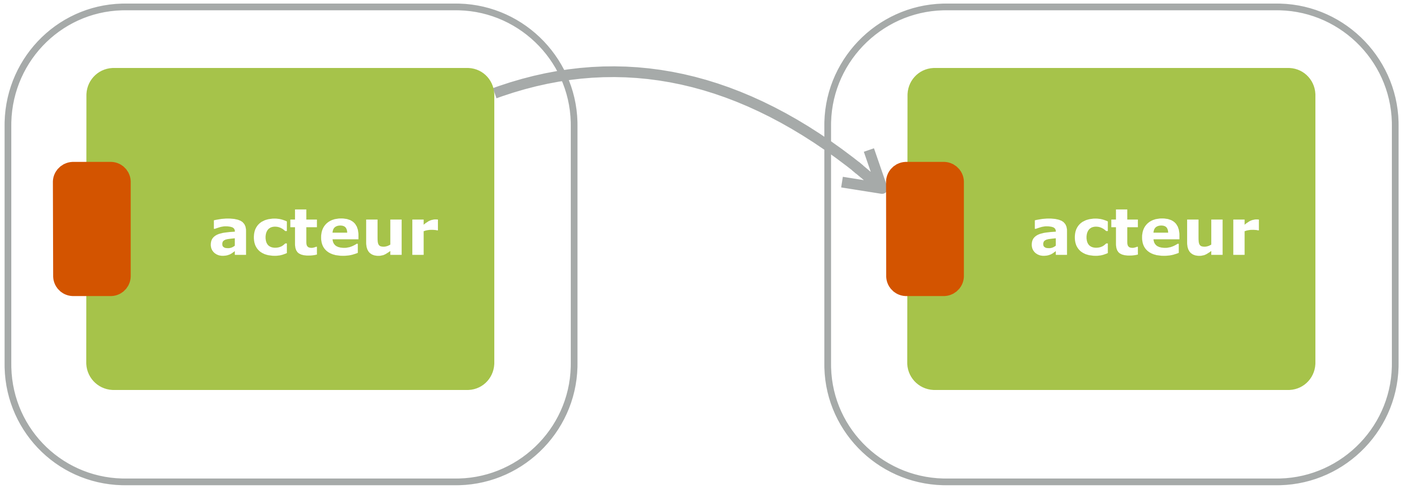
Framework asynchrone JVM

Développer en local avec les contraintes du distribué

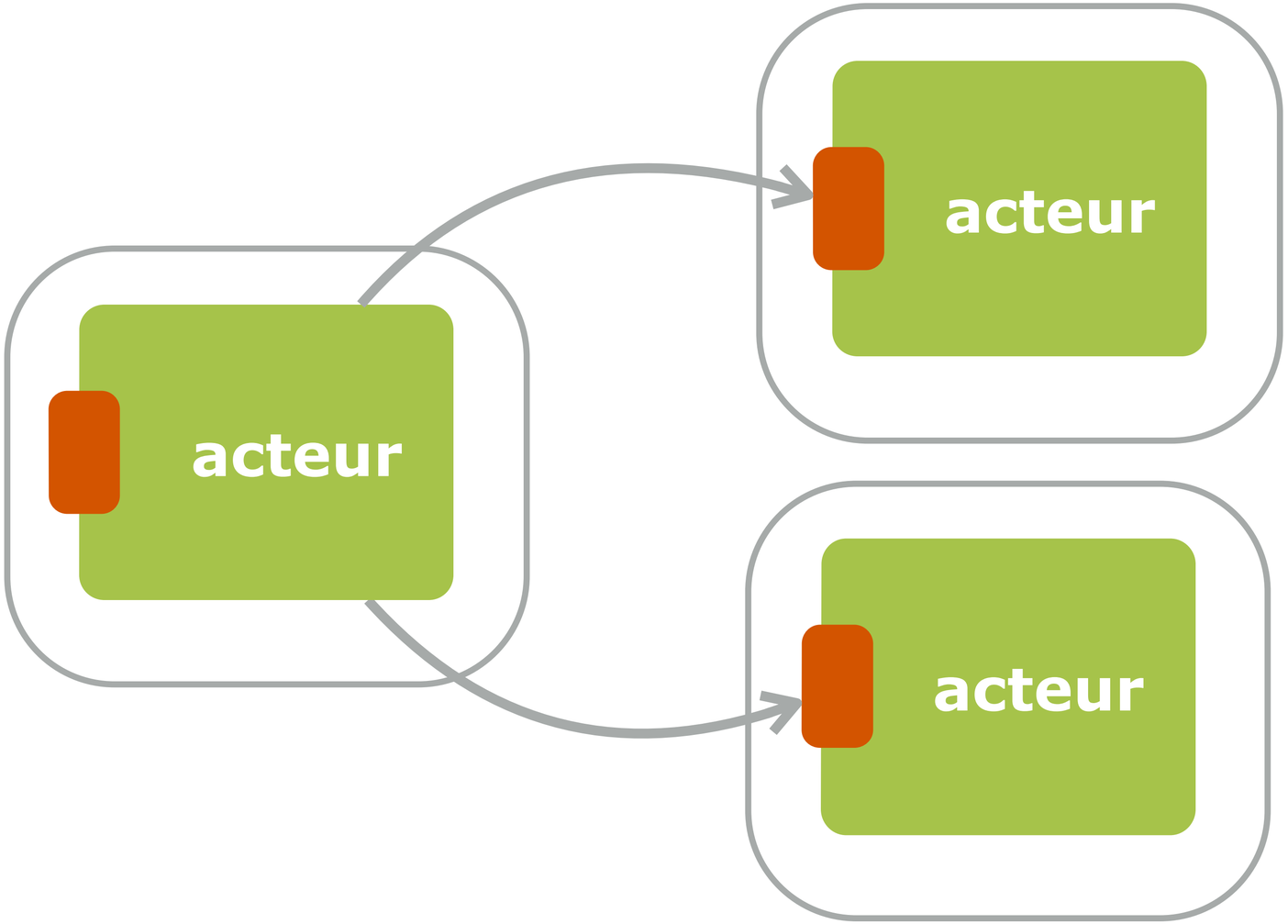
Modèle acteur



Modèle acteur

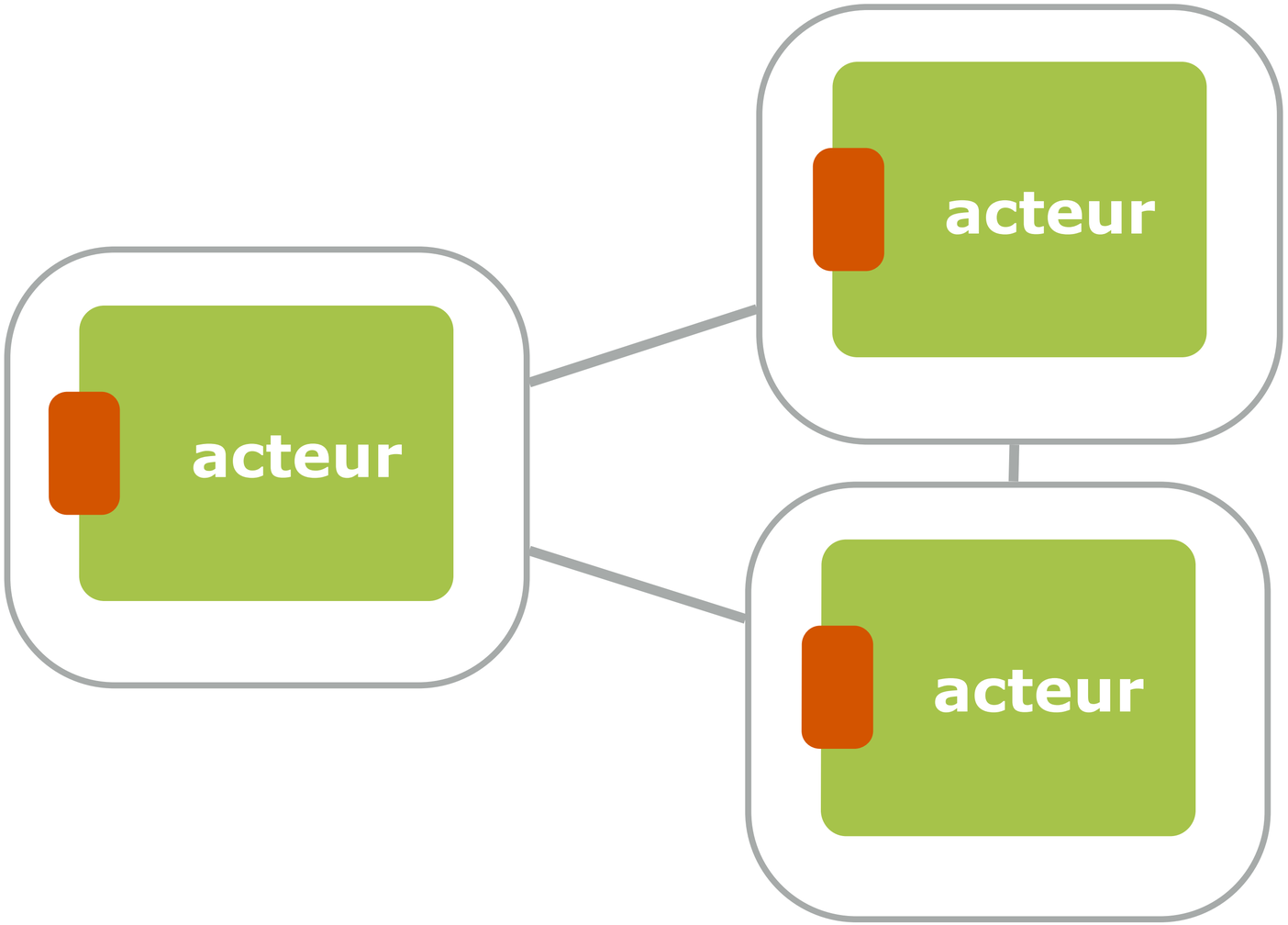


akka-cluster



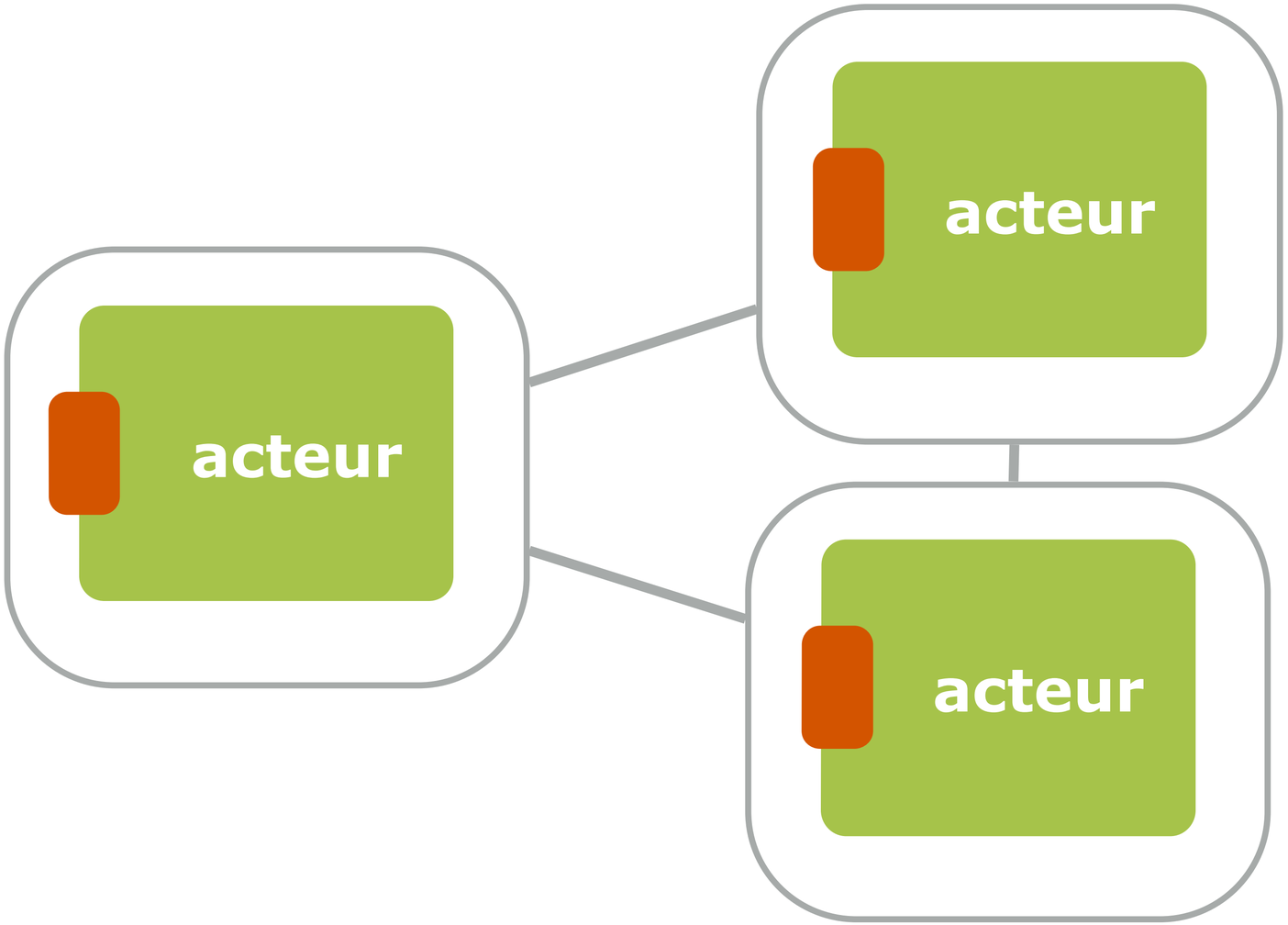
akka-cluster

akka-cluster



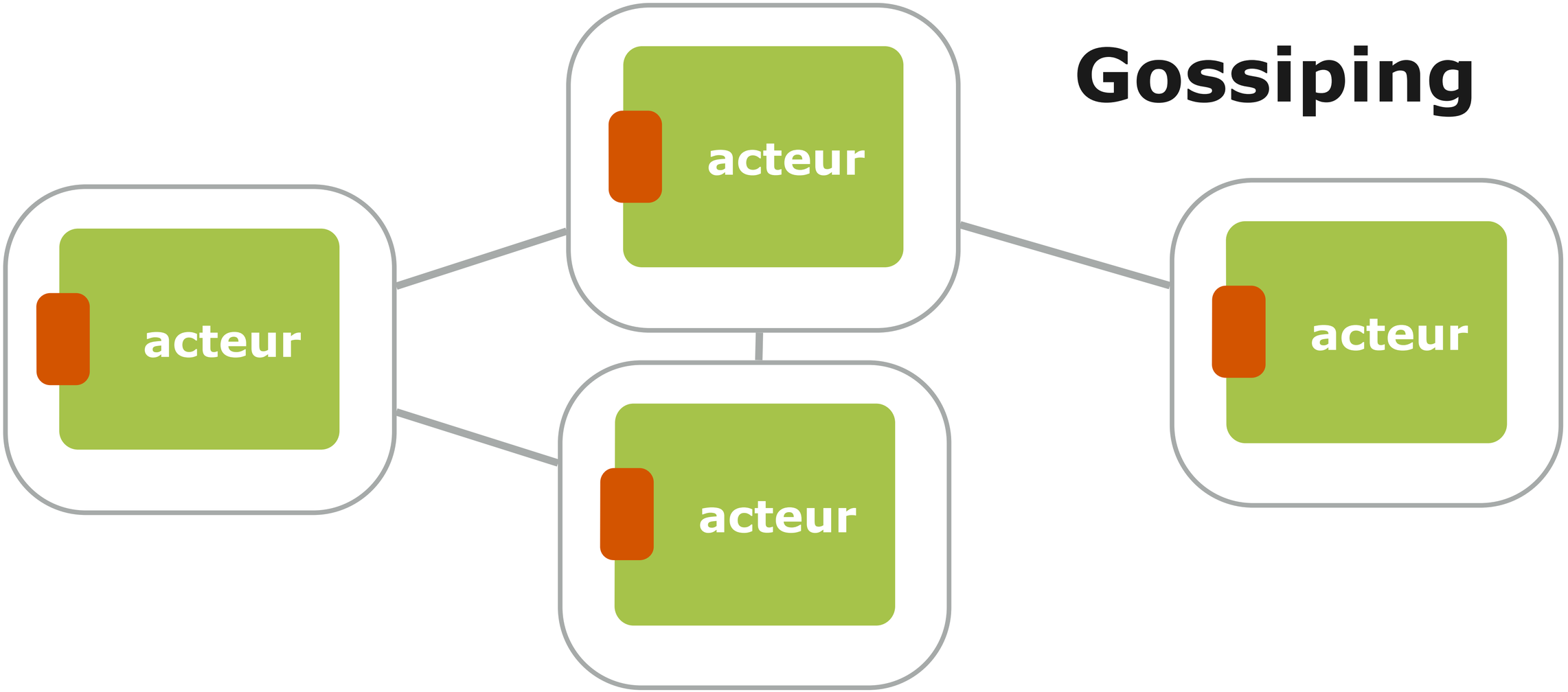
réseau point-à-point

akka-cluster

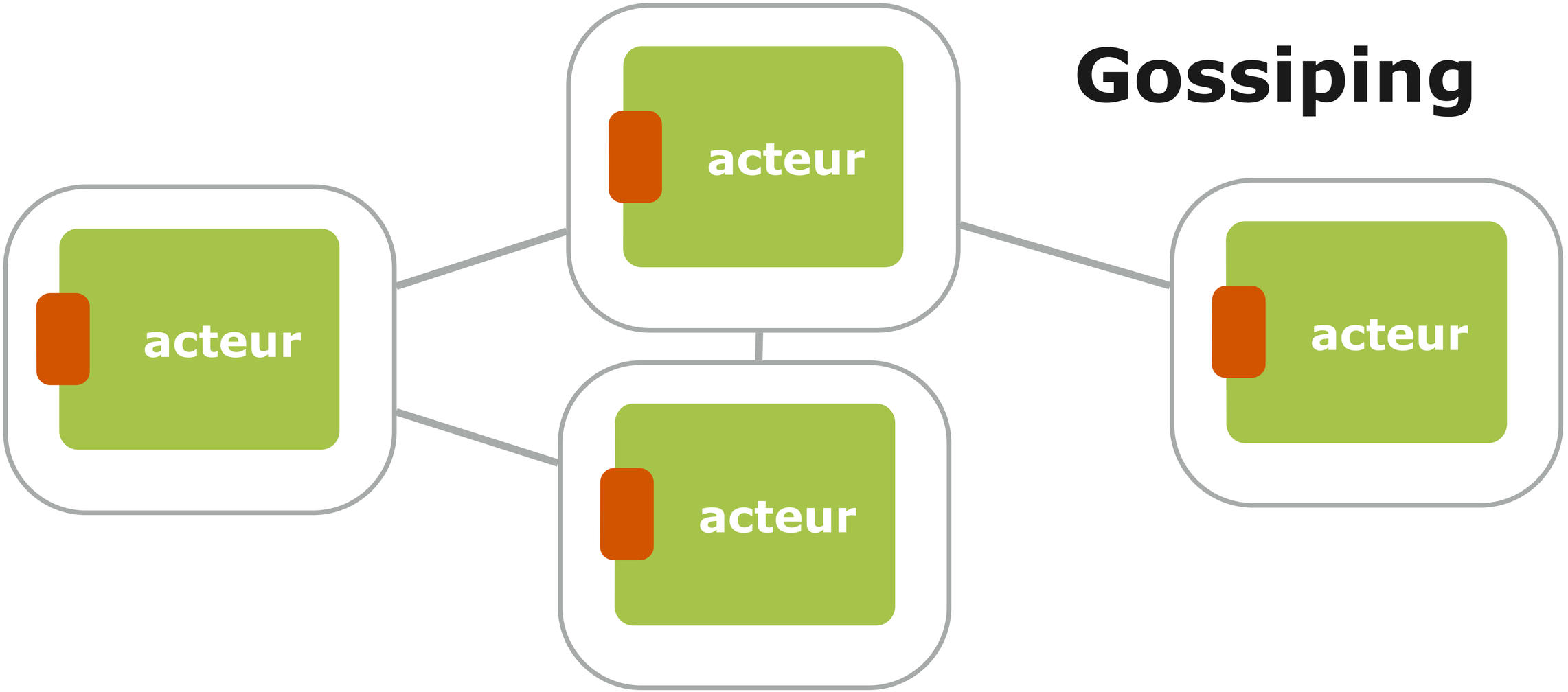


Gossiping

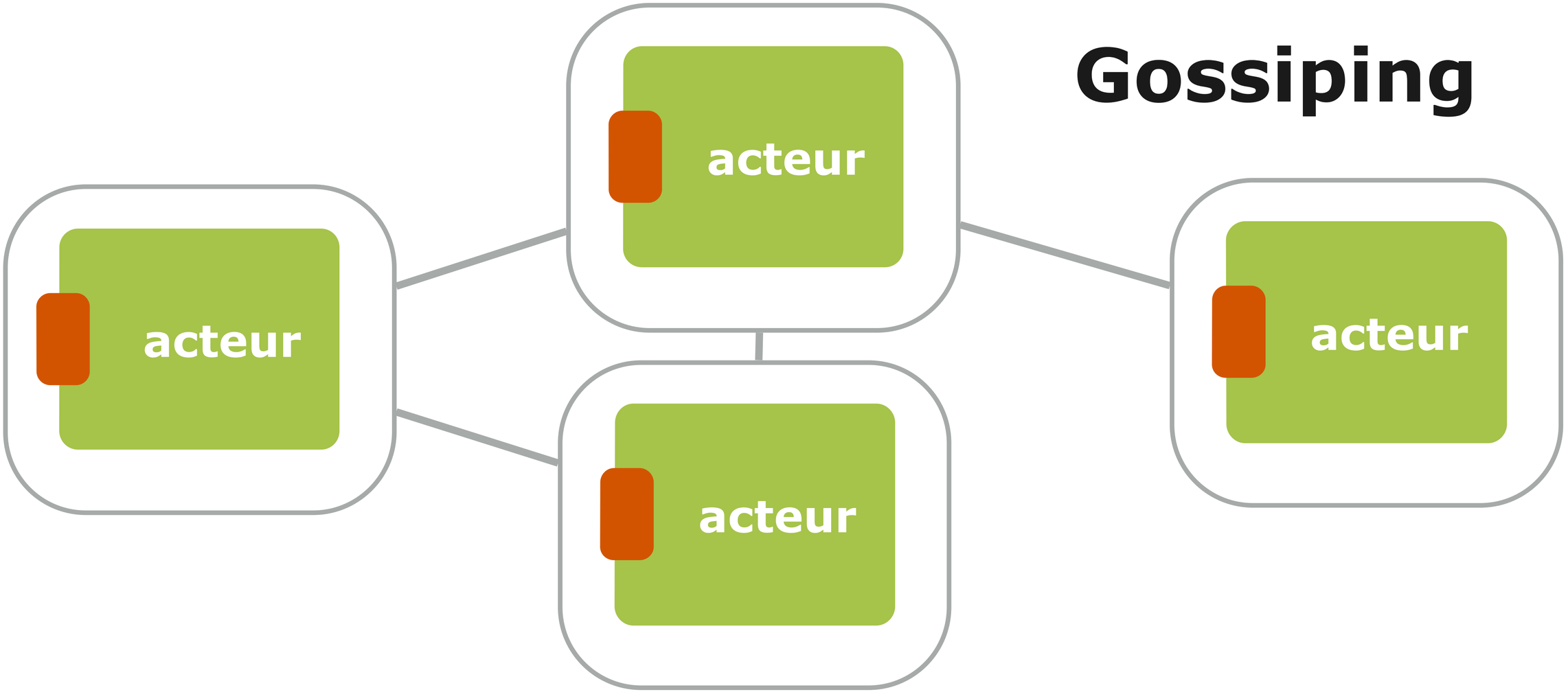
akka-cluster



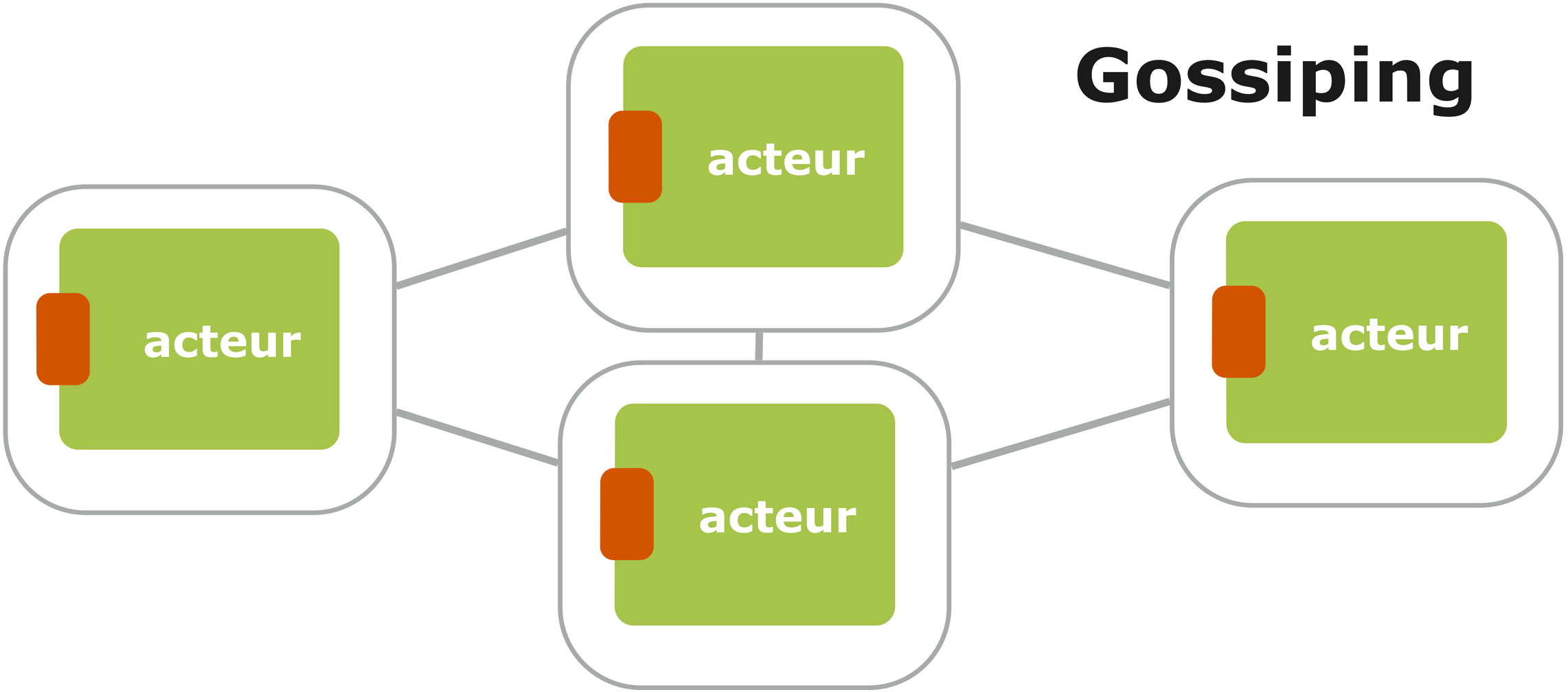
akka-cluster



akka-cluster

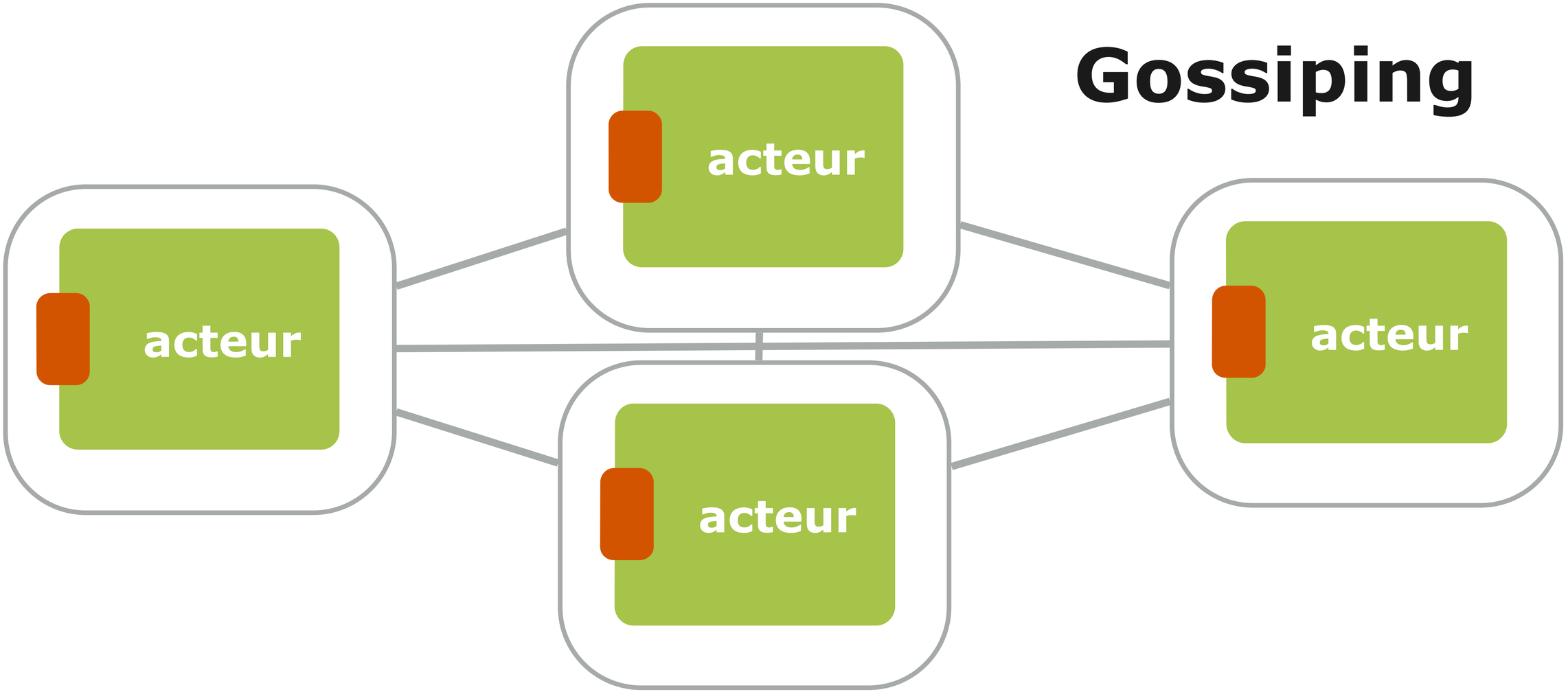


akka-cluster



DEVOXX France

akka-cluster



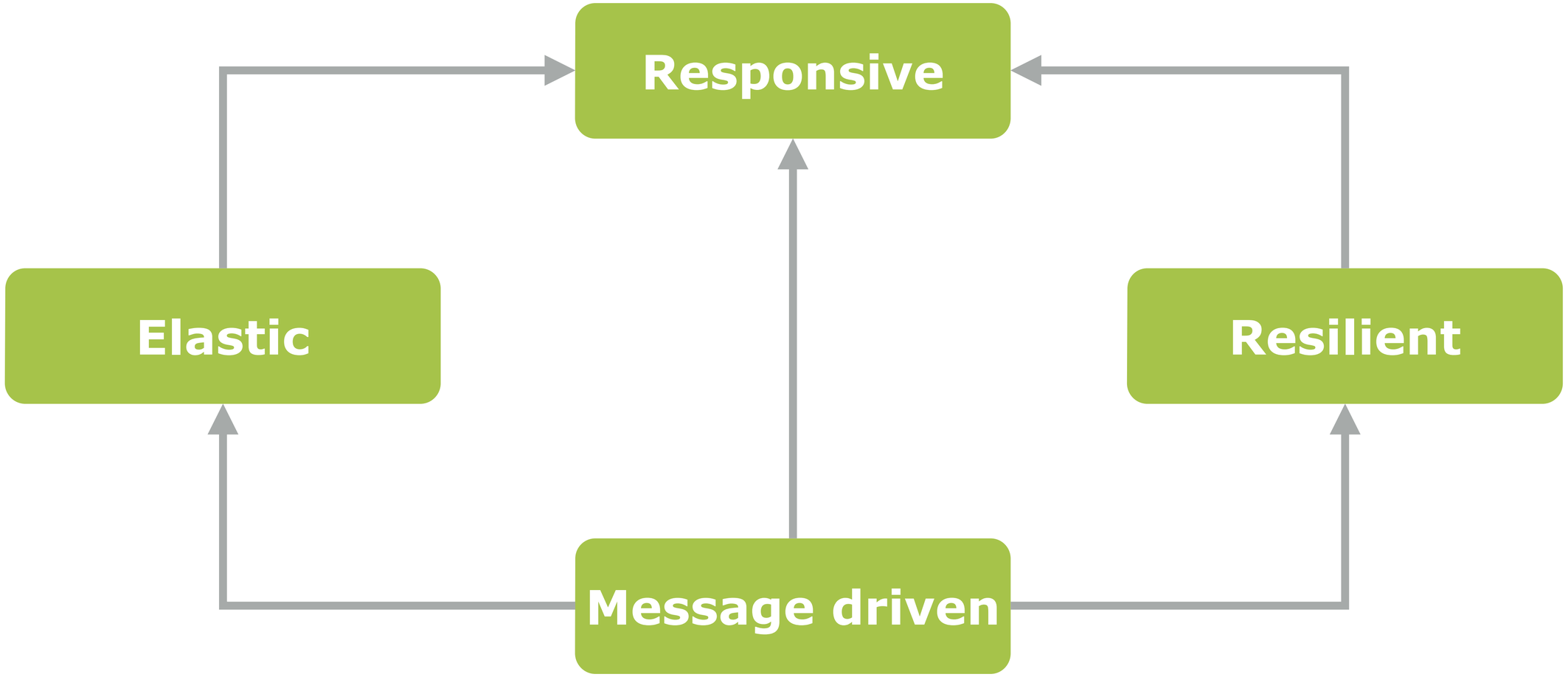
akka-cluster

```
akka.actor.deployment {
  /orchestration {
    router = round-robin-group
    nr-of-instances = 100
    routees.paths = ["/user/orchestration"]
    cluster {
      enabled = on
      allow-local-routees = off
      use-role = orchestration
    }
  }
}
```

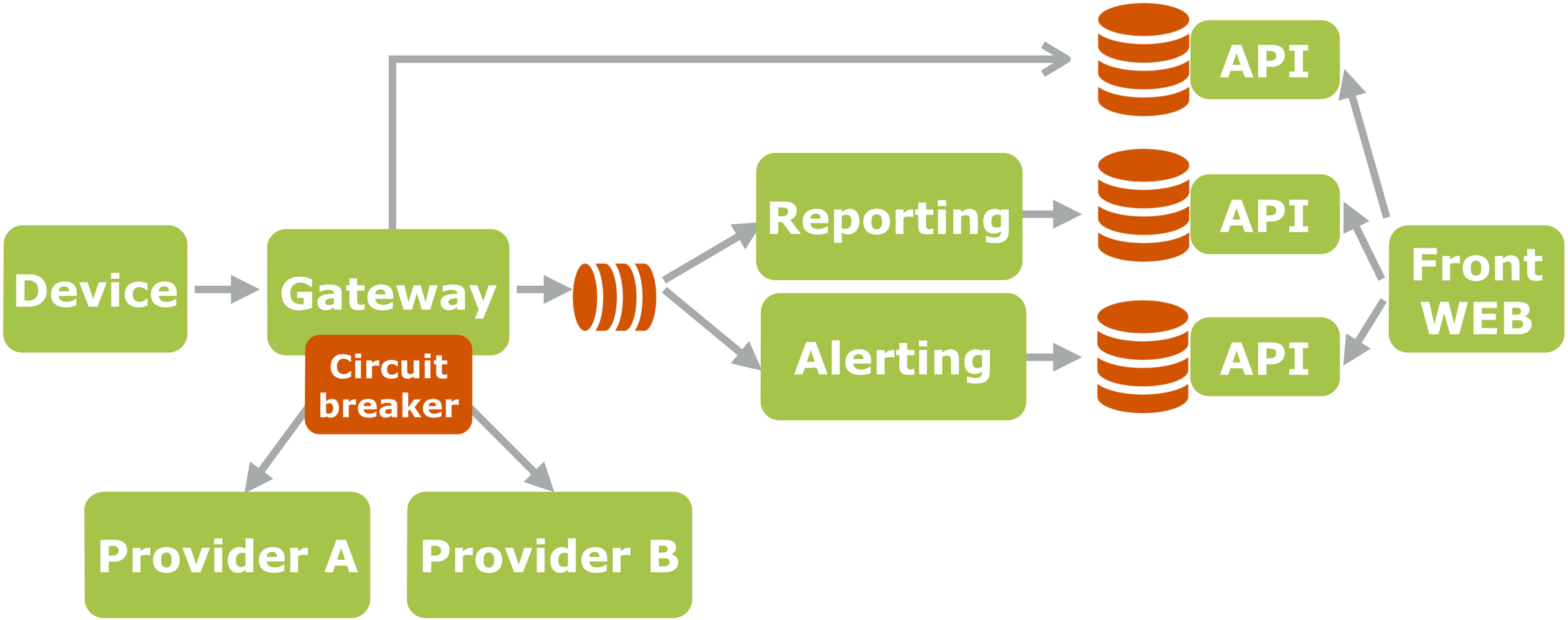
akka-cluster

akka-cluster est la clé de l'élasticité et de la résilience du système.

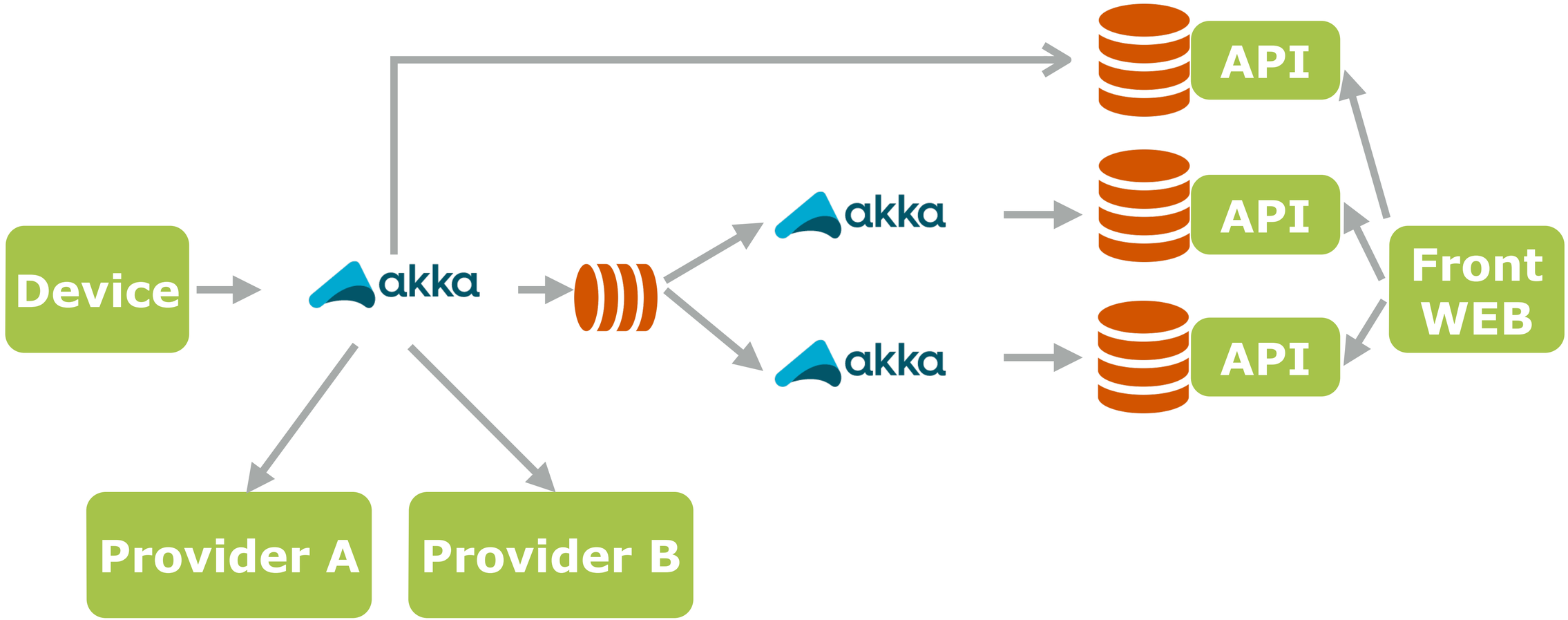
Reactive Manifesto et Akka



Technos


Device
Gateway
Circuit breaker
Provider A
Provider B
Reporting
Alerting
API
API
API
Front WEB

Technos

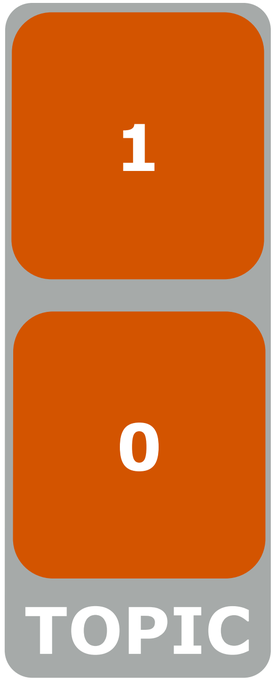

Device
akka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Journal distribué

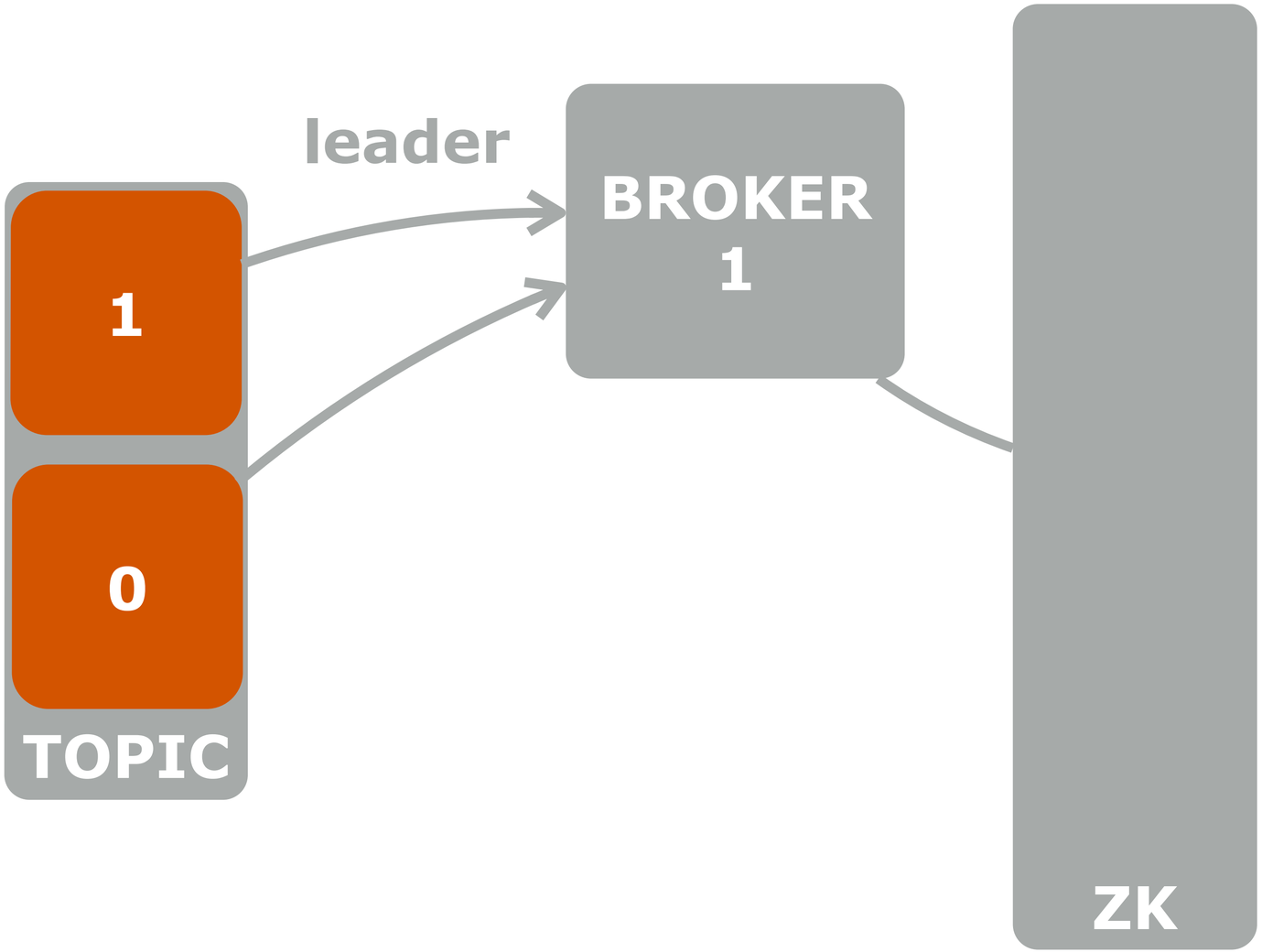
Journal distribué

- **Développé à l'origine chez LinkedIn**
- **Rapide, robuste, scalable**
- **File d'attentes, agrégation logs, streaming**

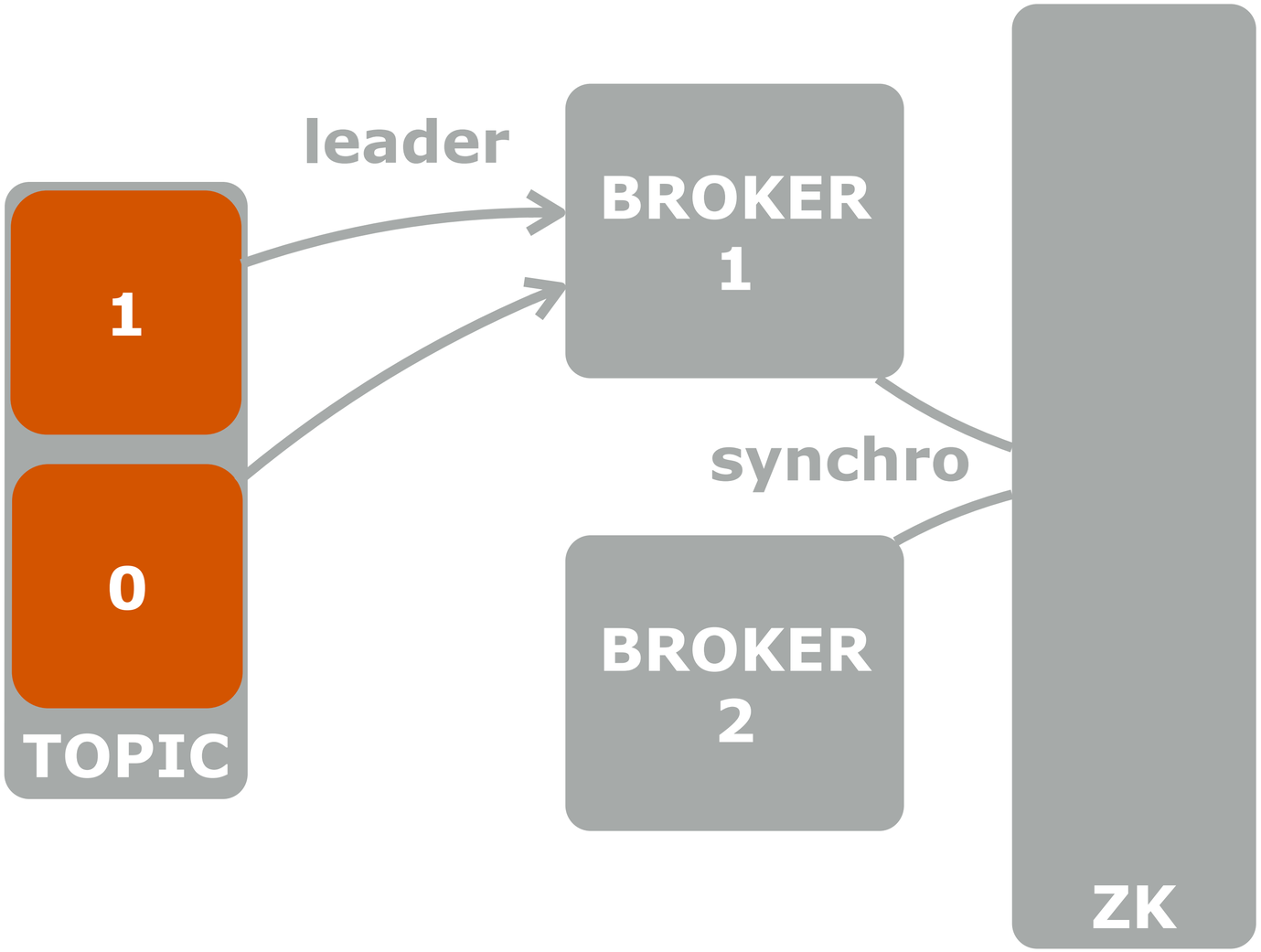
Journal distribué



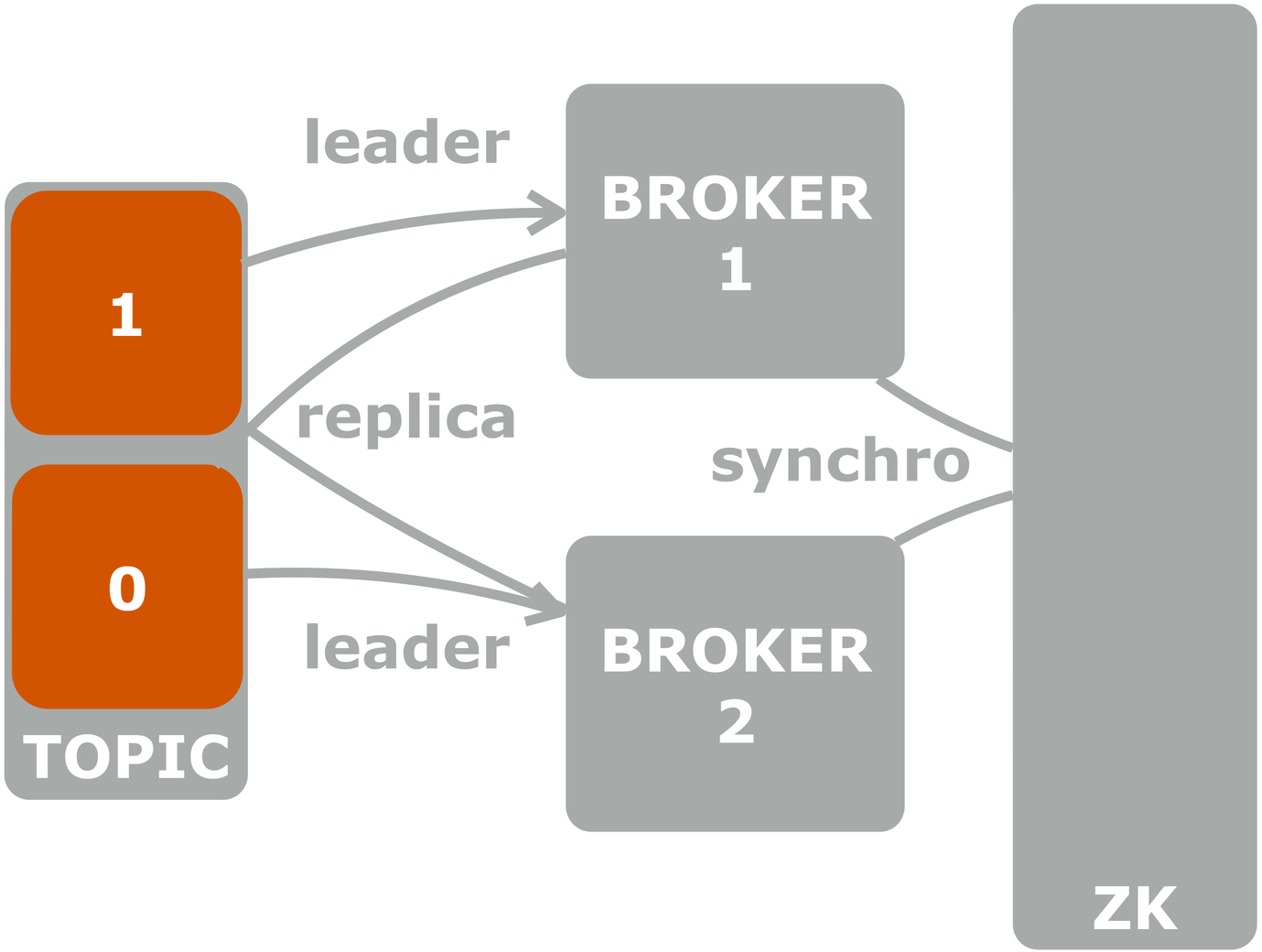
Journal distribué



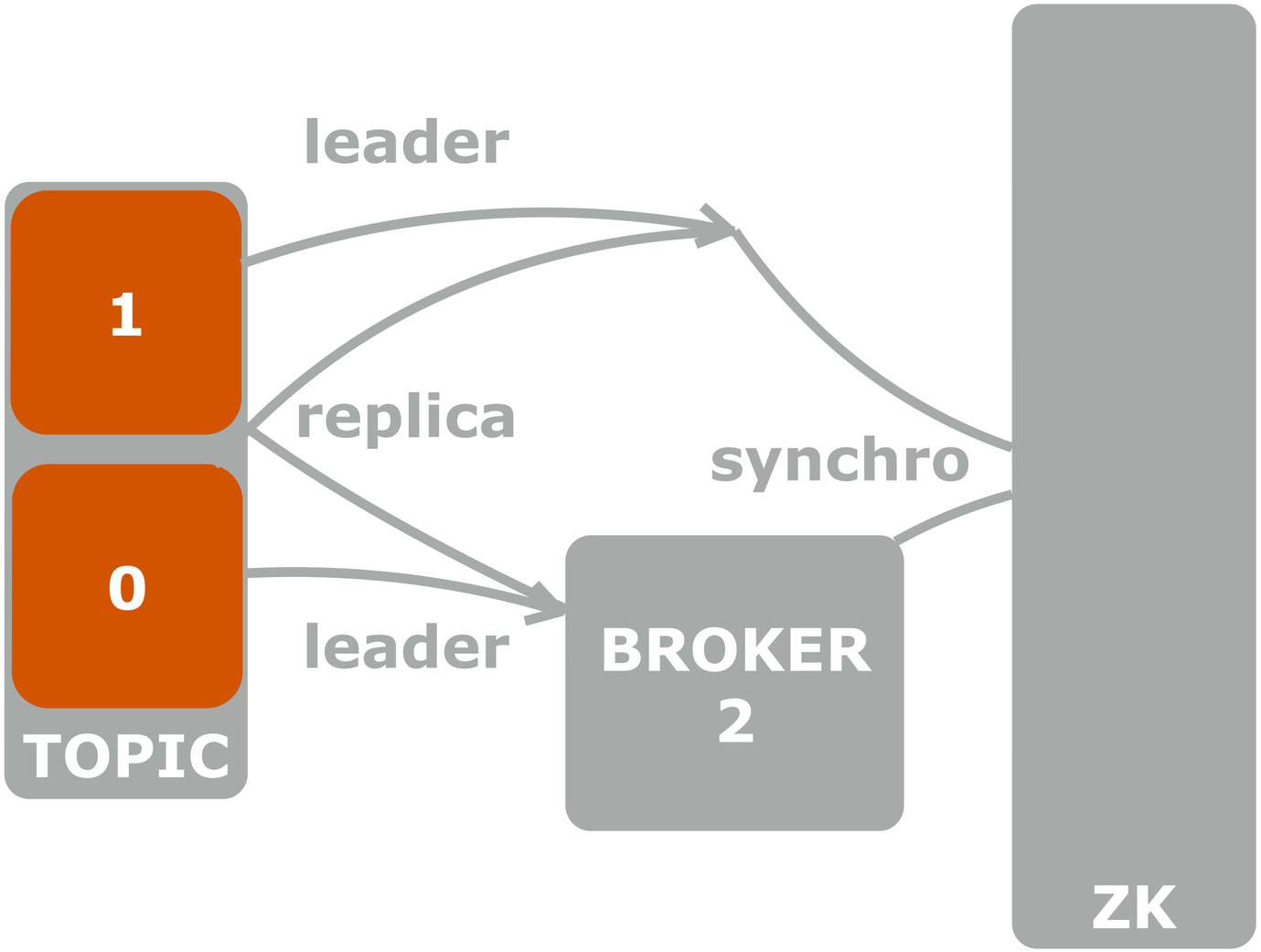
Journal distribué



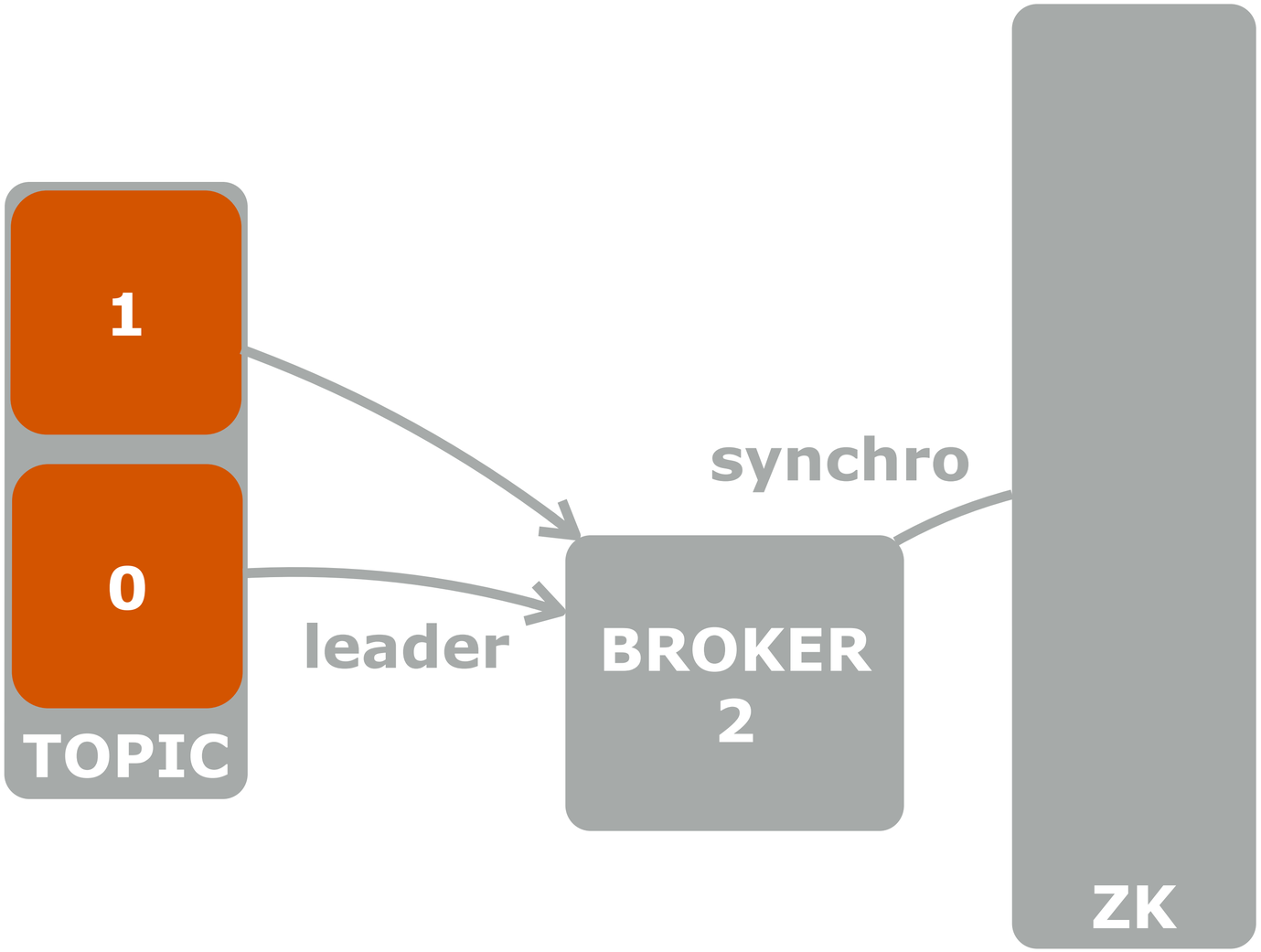
Journal distribué



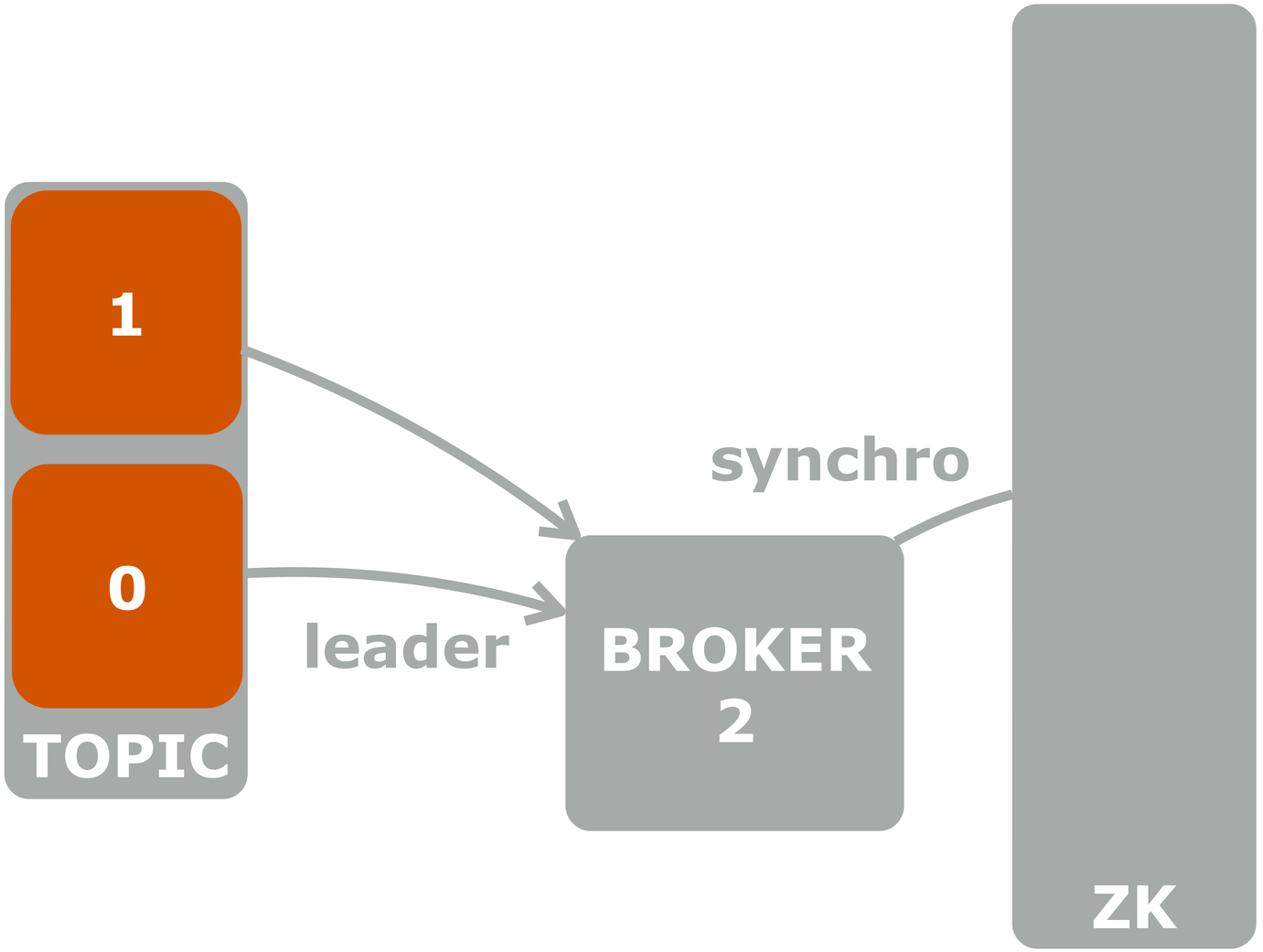
Journal distribué



Journal distribué

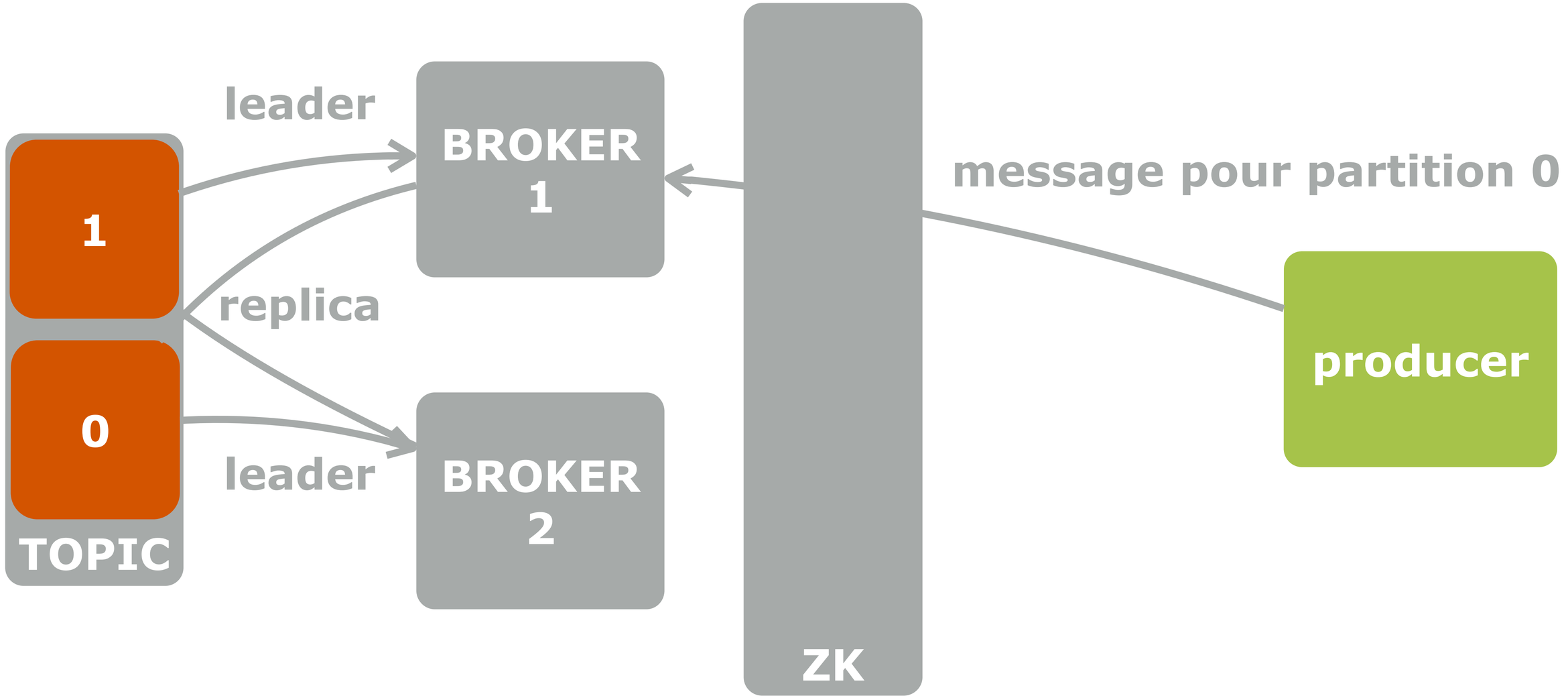


Journal distribué

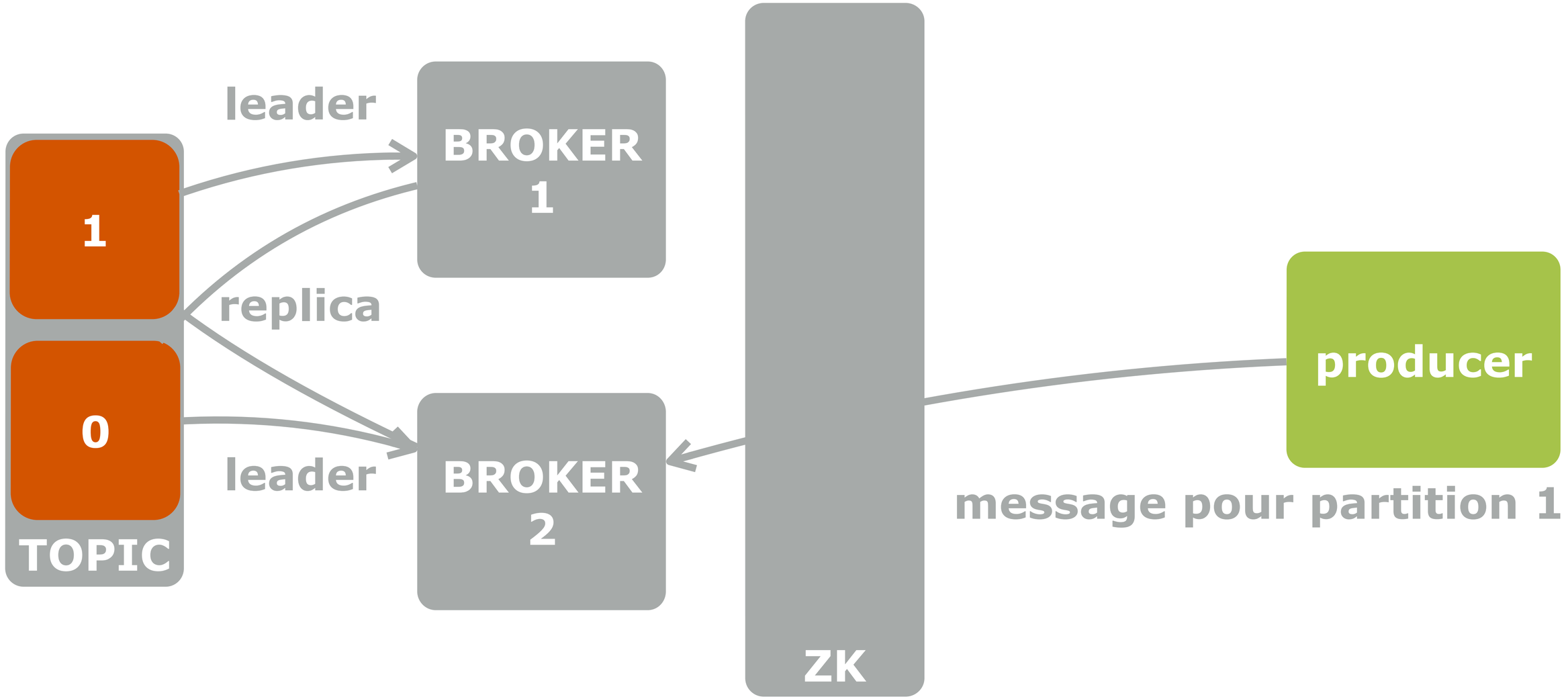


- Élasticité
- Résilience

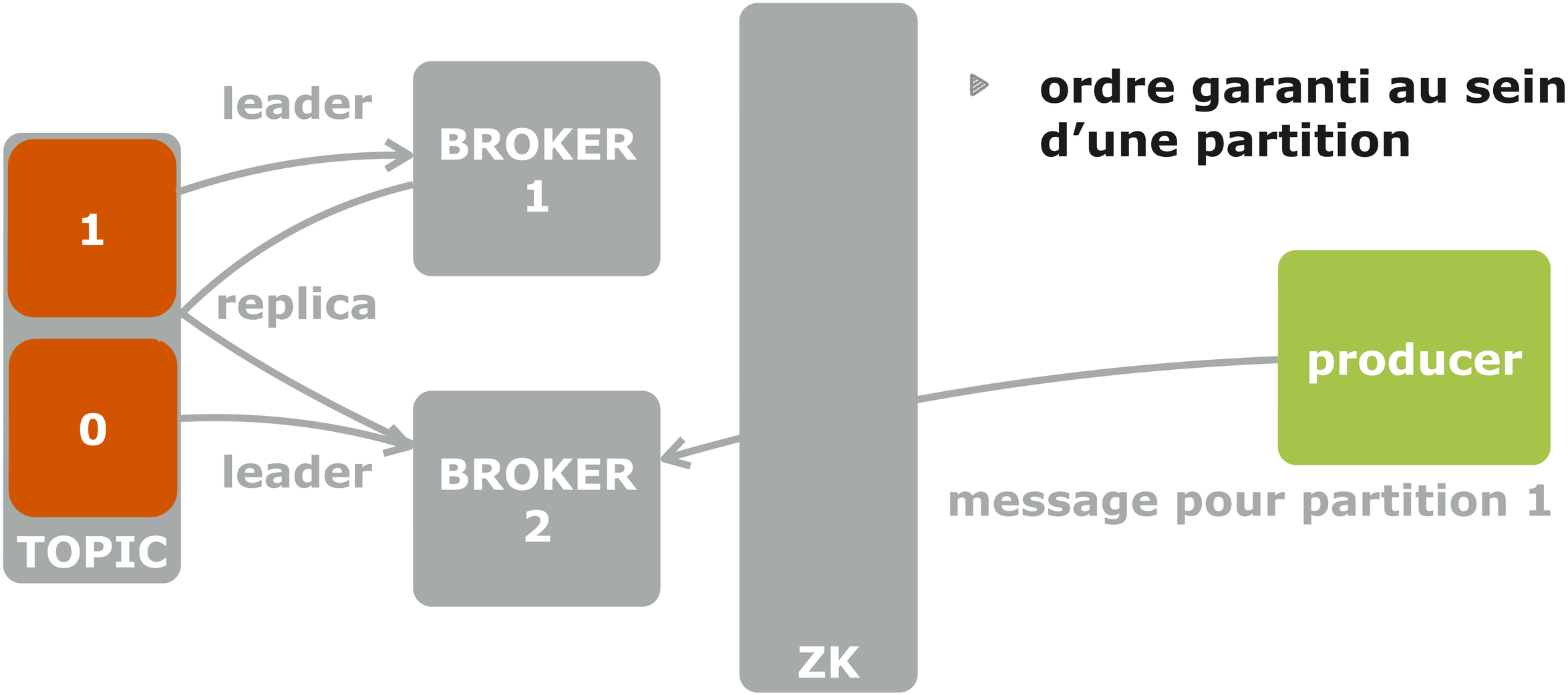
Journal distribué



Journal distribué

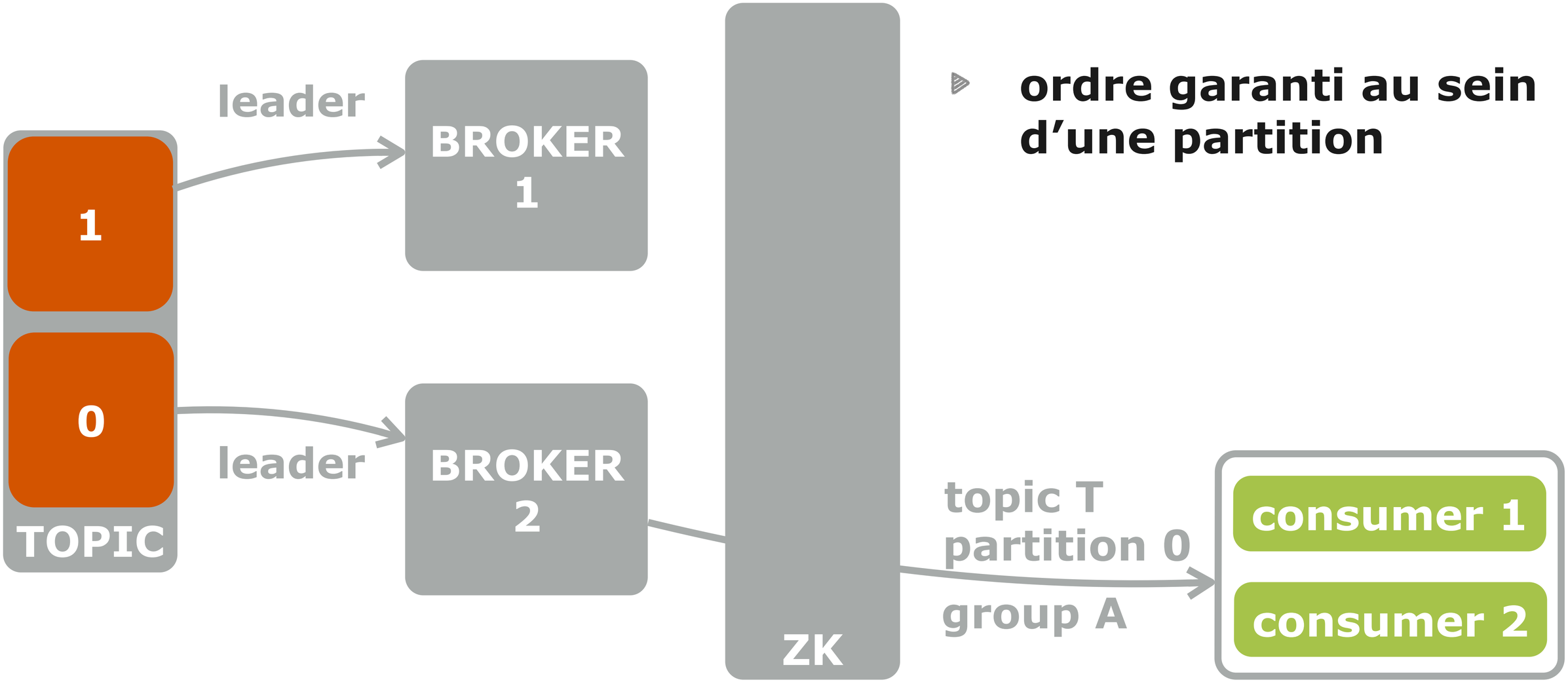


Journal distribué



- **ordre garanti au sein d'une partition**

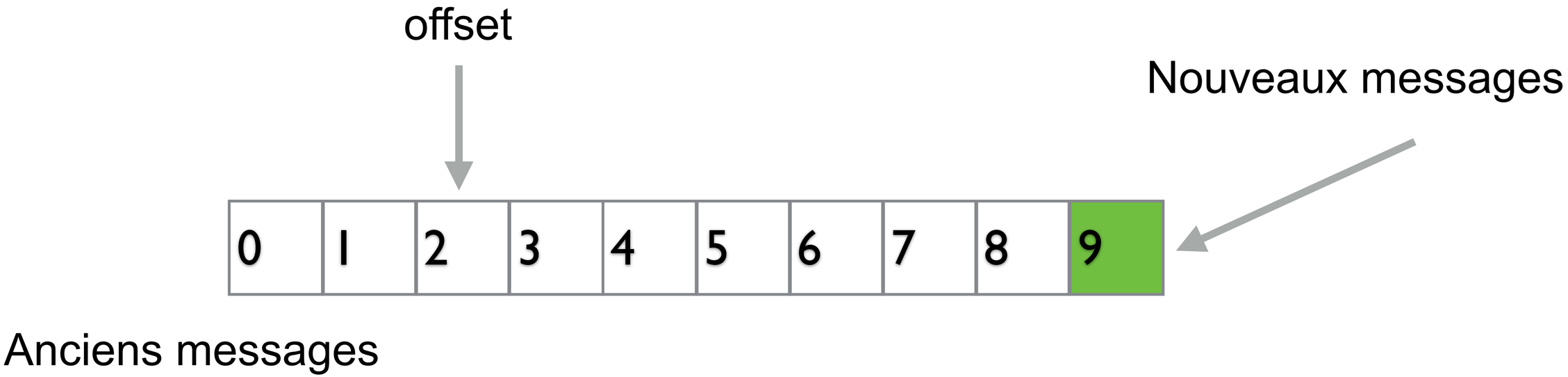
Journal distribué



- **ordre garanti au sein d'une partition**

Kafka topic

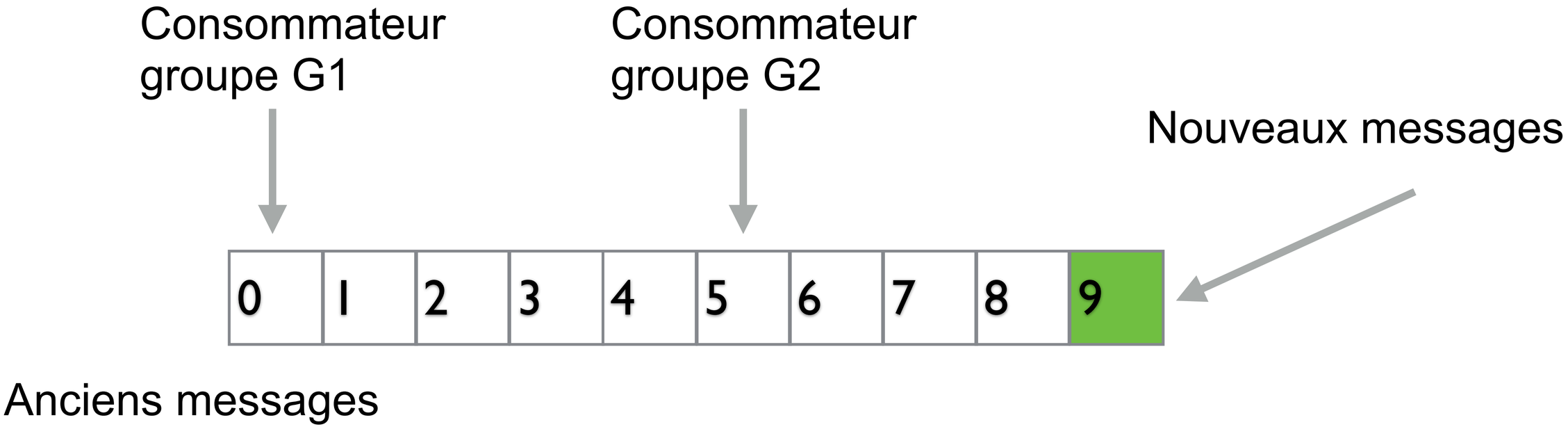
Apache Kafka
A high-throughput distributed messaging system.



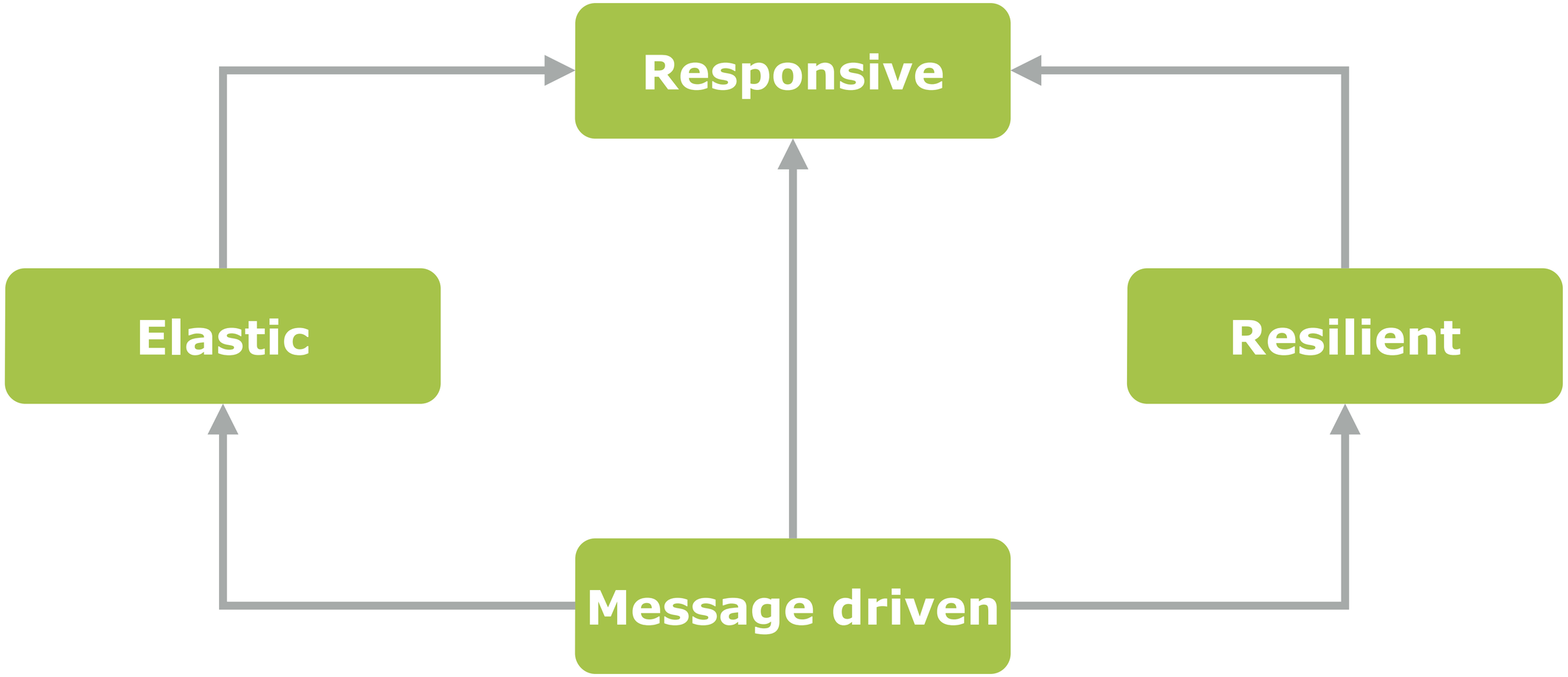
DEVOXX France

Kafka topic

Apache Kafka
A high-throughput distributed messaging system.

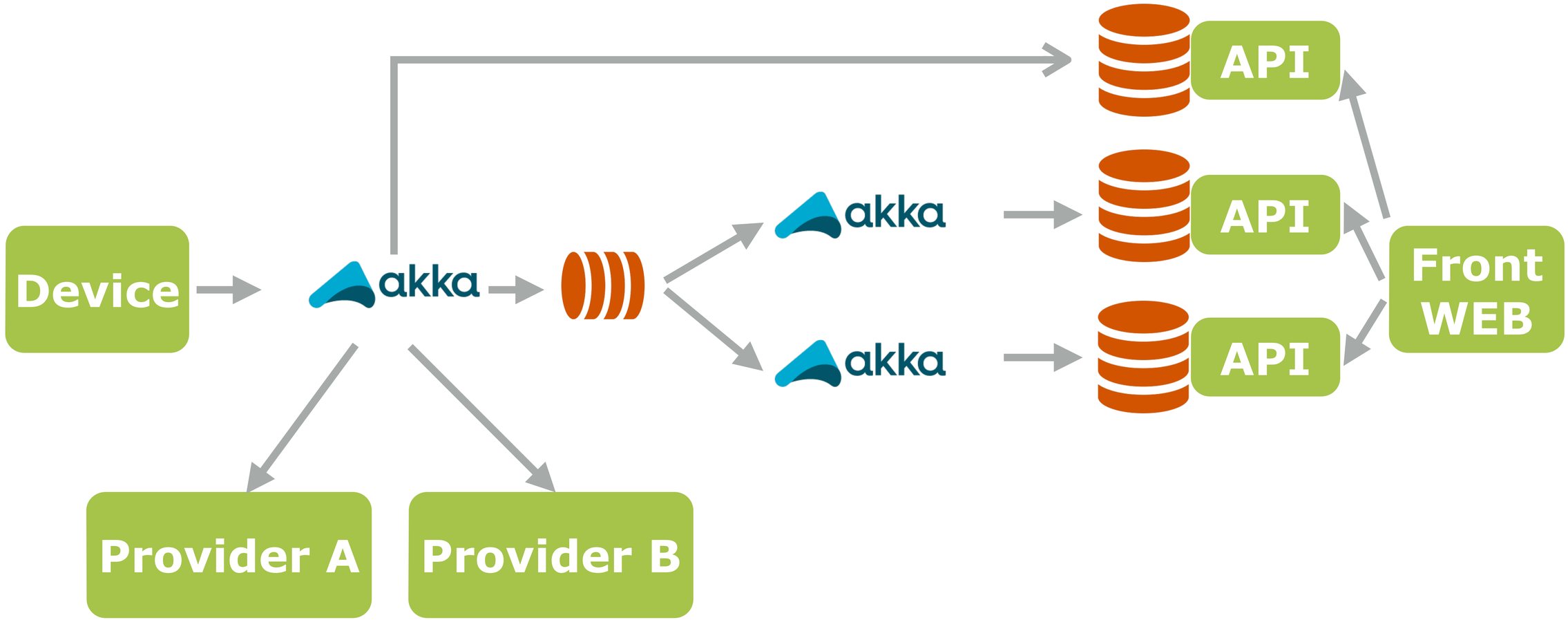


Reactive Manifesto et Kafka

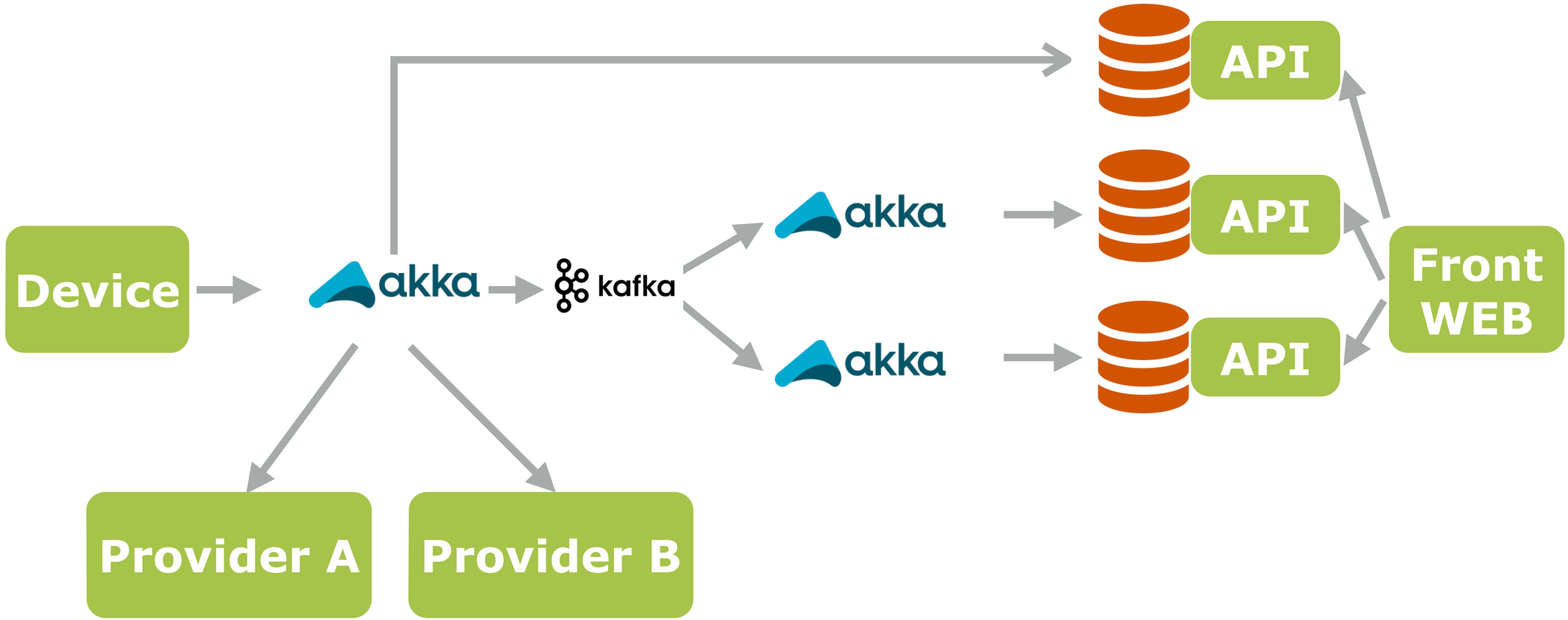


DEVOXX France

Technos


Device
akka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Technos


Device
akka
kafka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Elasticsearch

elasticsearch

DEVOXX™ France

Reporting

- Recherche full-text
- Aggregations

Alerting

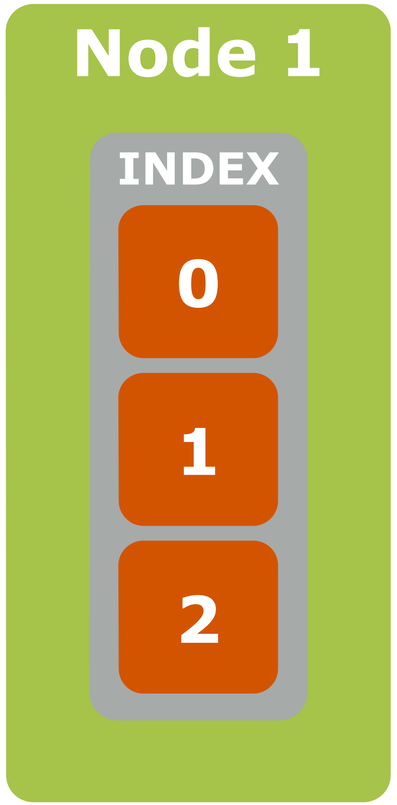
- quantité
- seuil
- période de temps
- filtres

Document

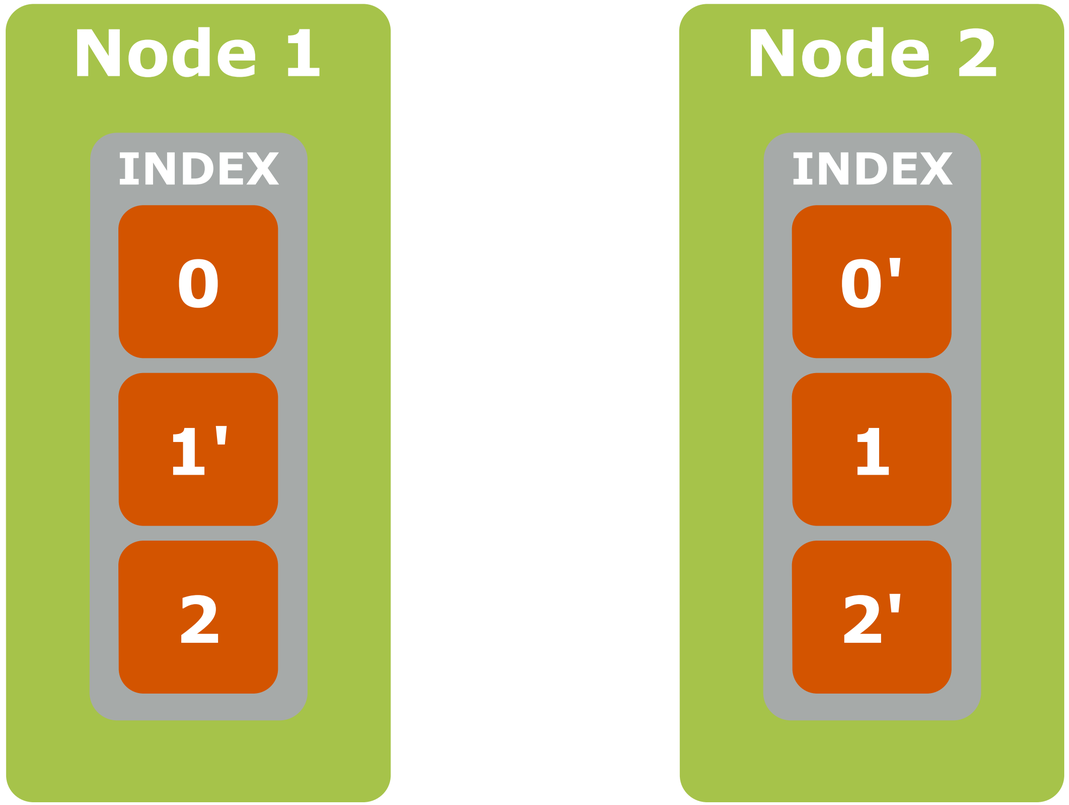
Event

```
{
    "device"  : "device1234"
    "msg" : "some message"
    "timestamp" : "2015-04-08T13:30:00.000Z"
}
```

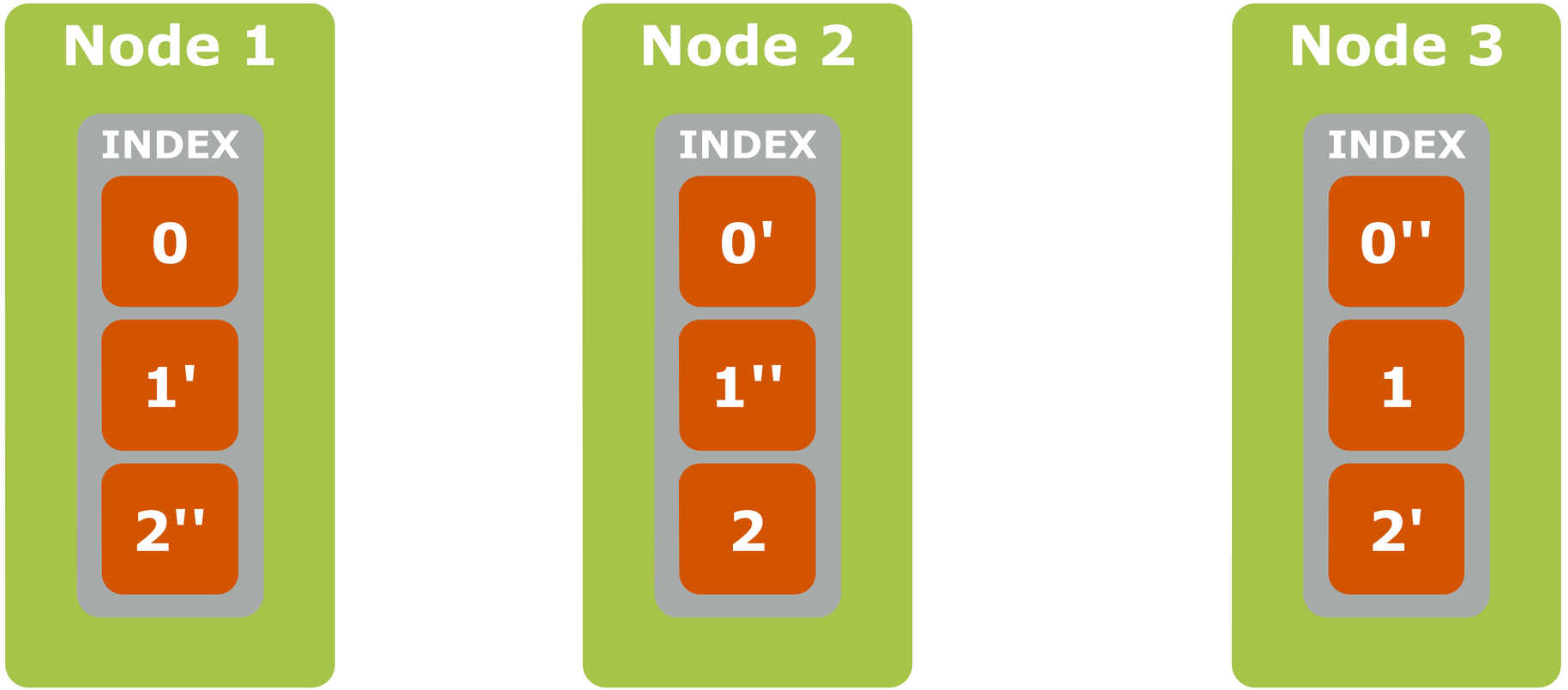
Elastique - Shards



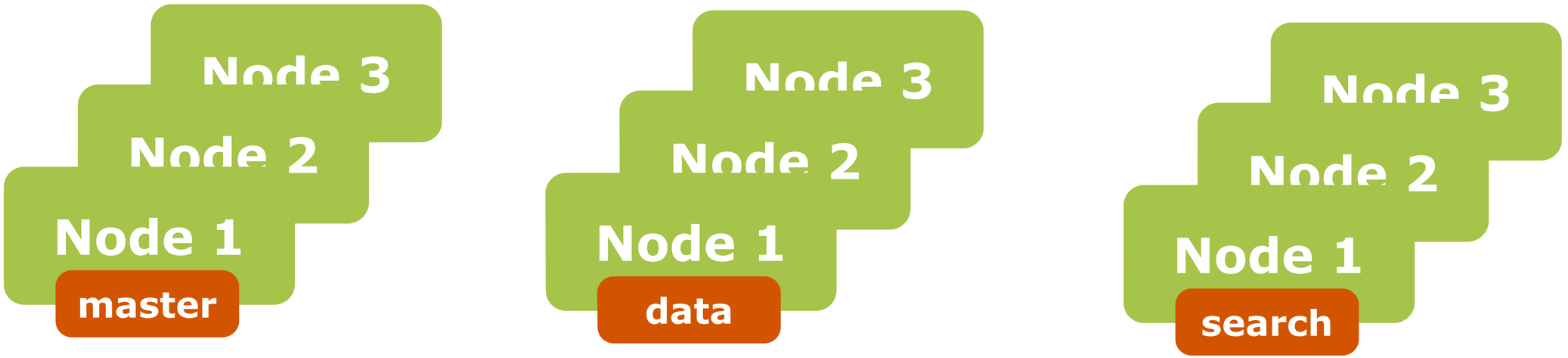
Elastique - Shards



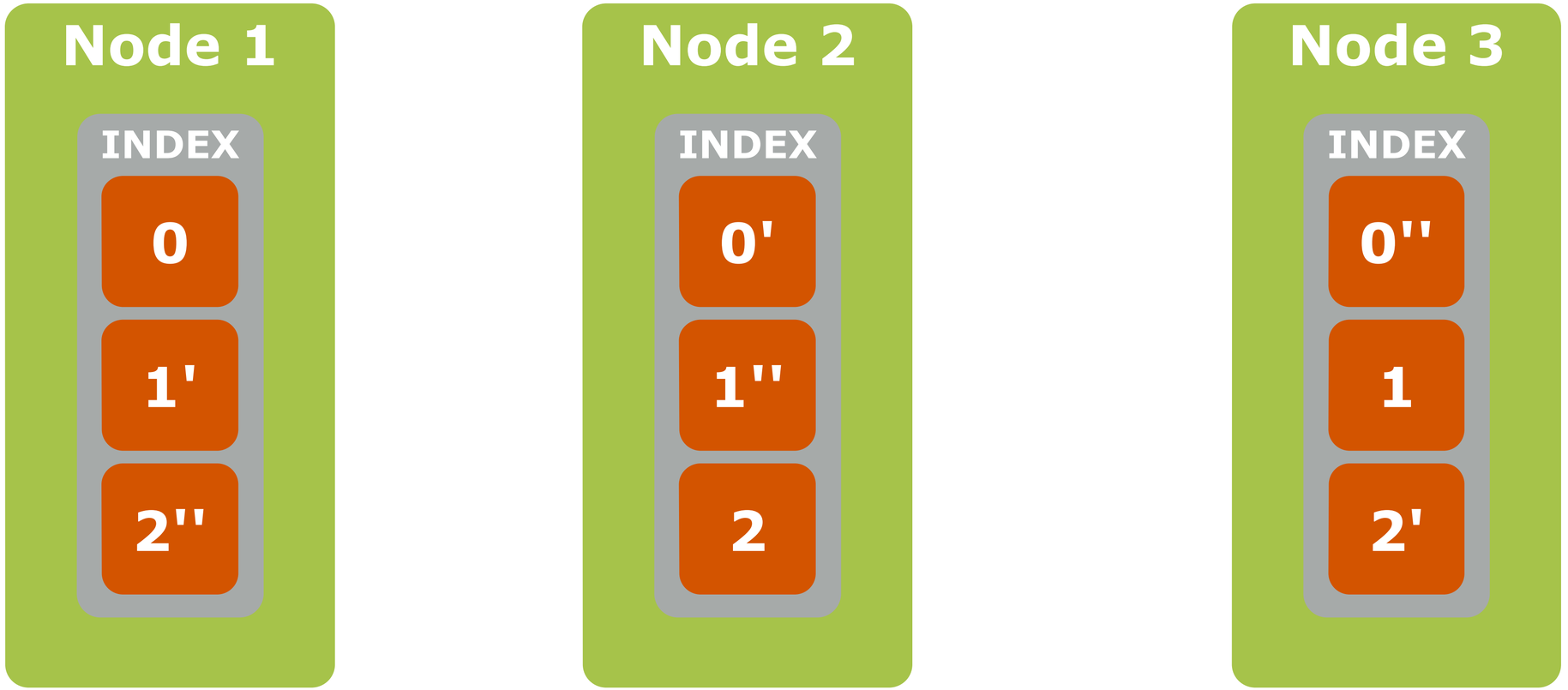
Elastique - Shards



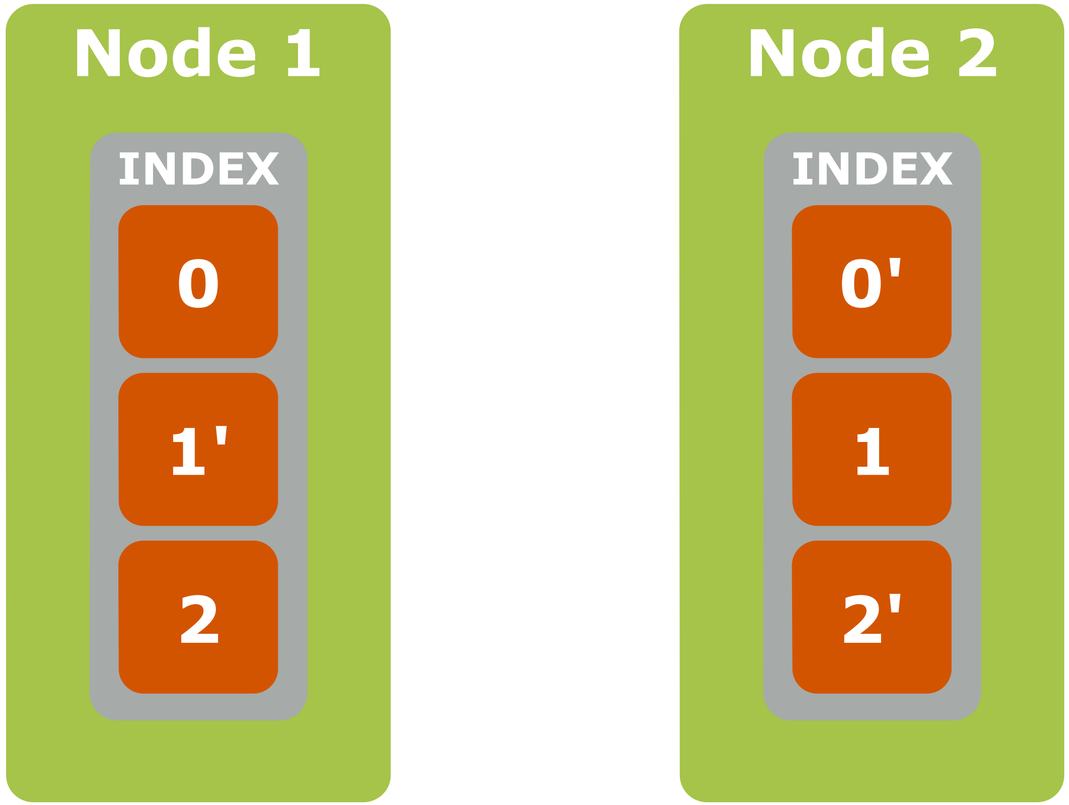
Elastique - Noeuds



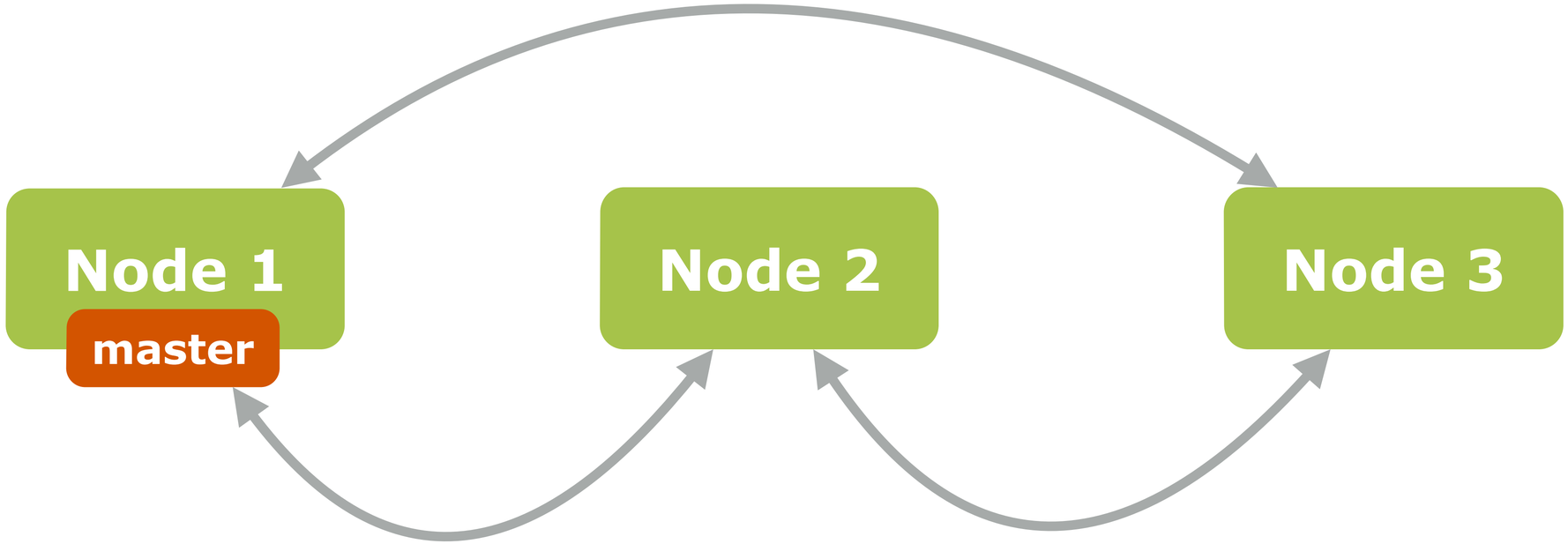
Résilient - Shards



Résilient - Shards

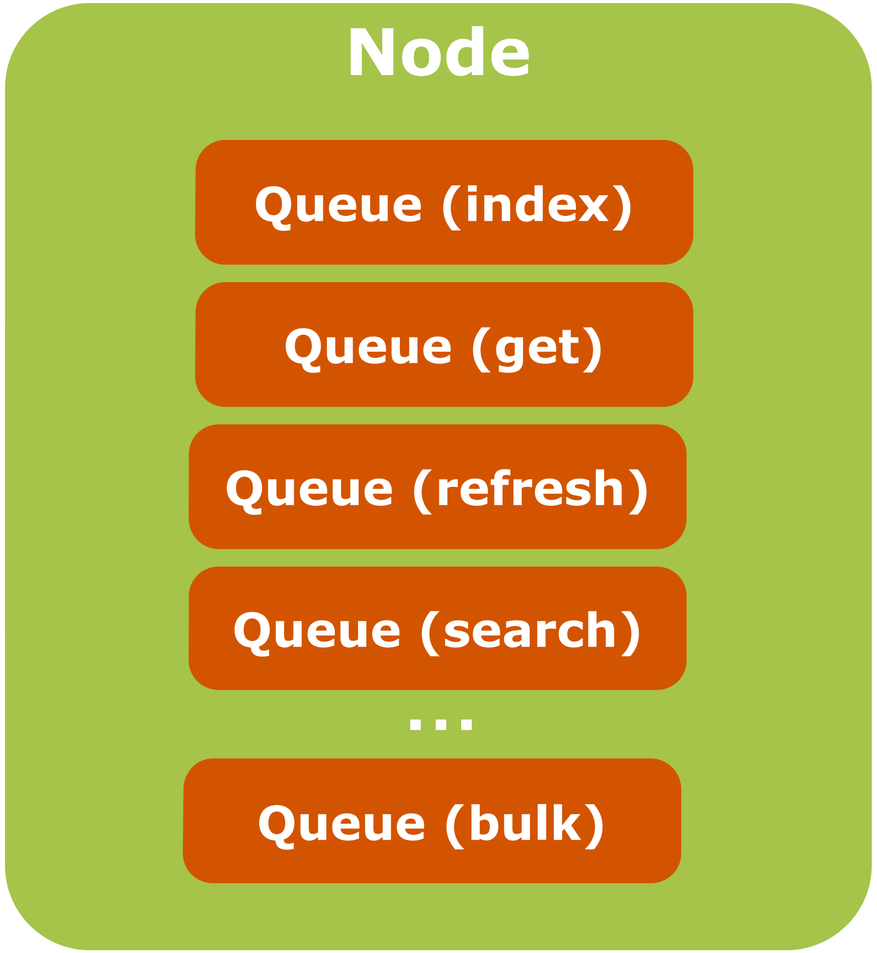


Résilient - Noeuds

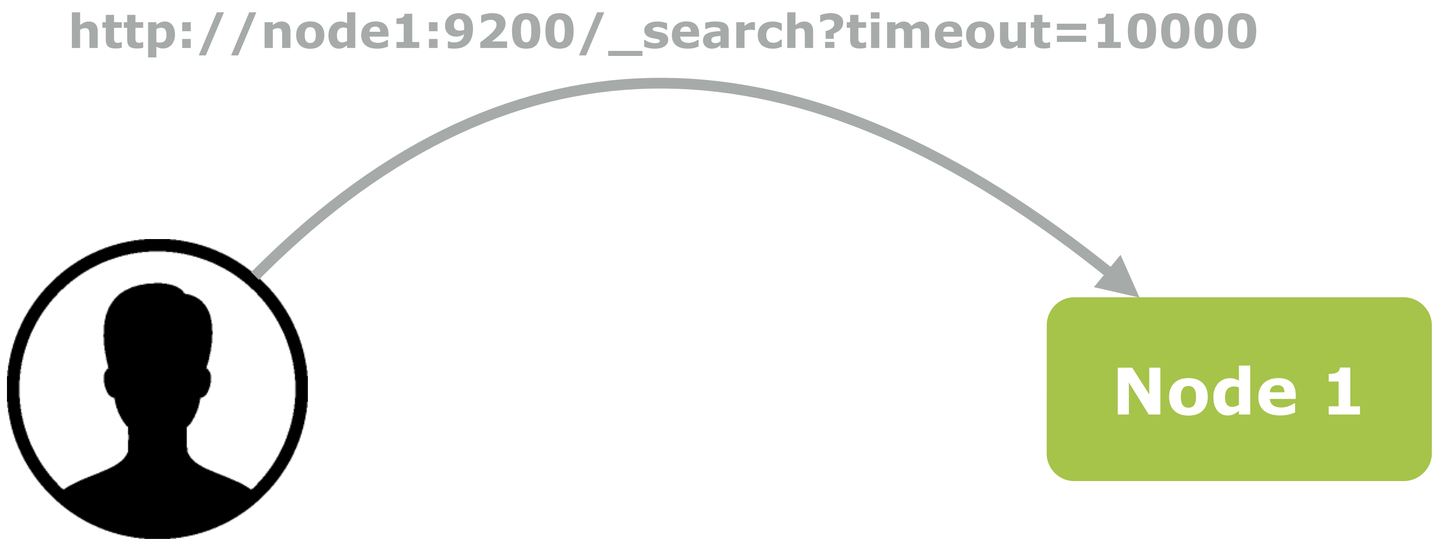


Résilient - Noeuds

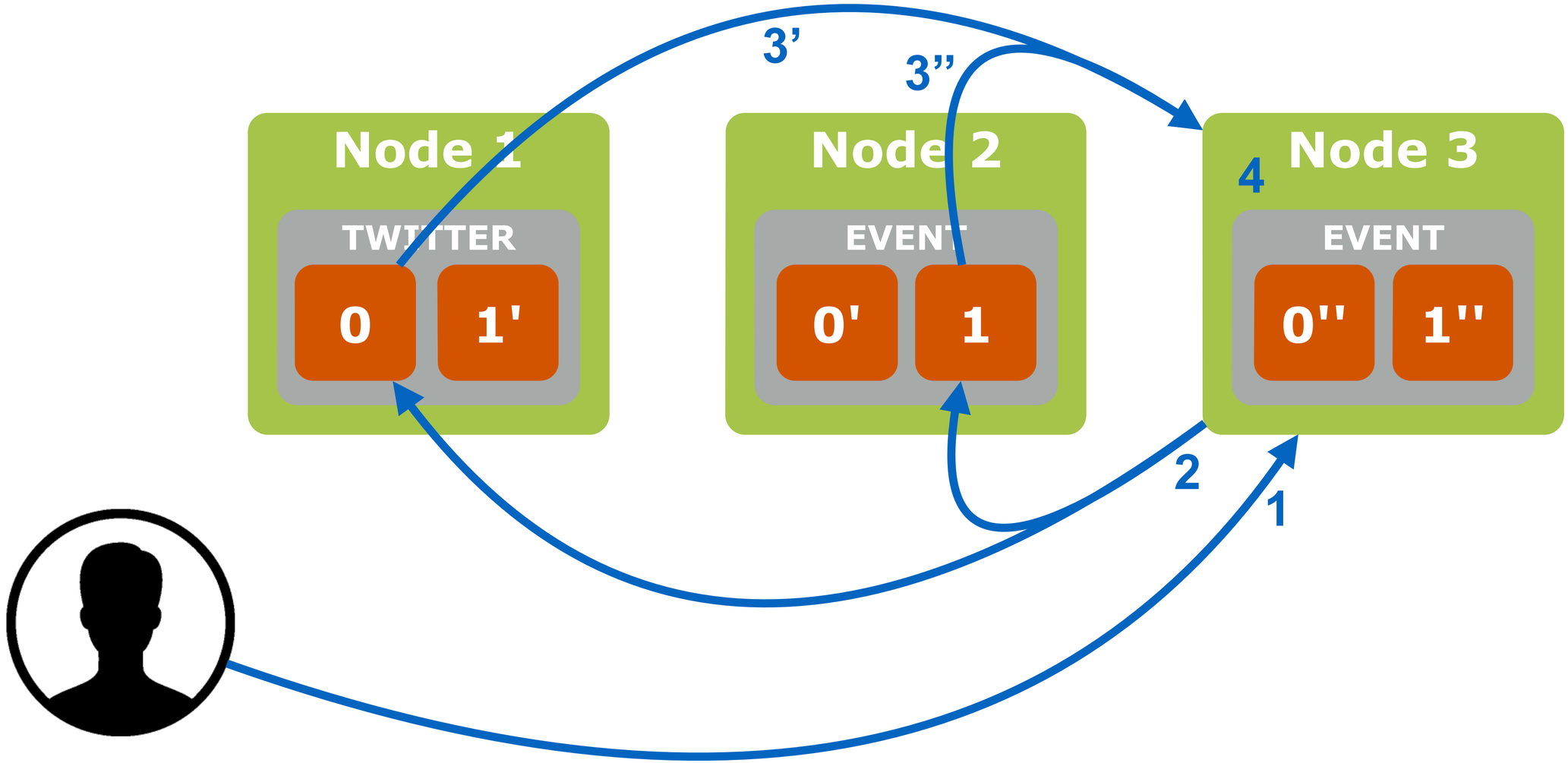
Bounded Queues



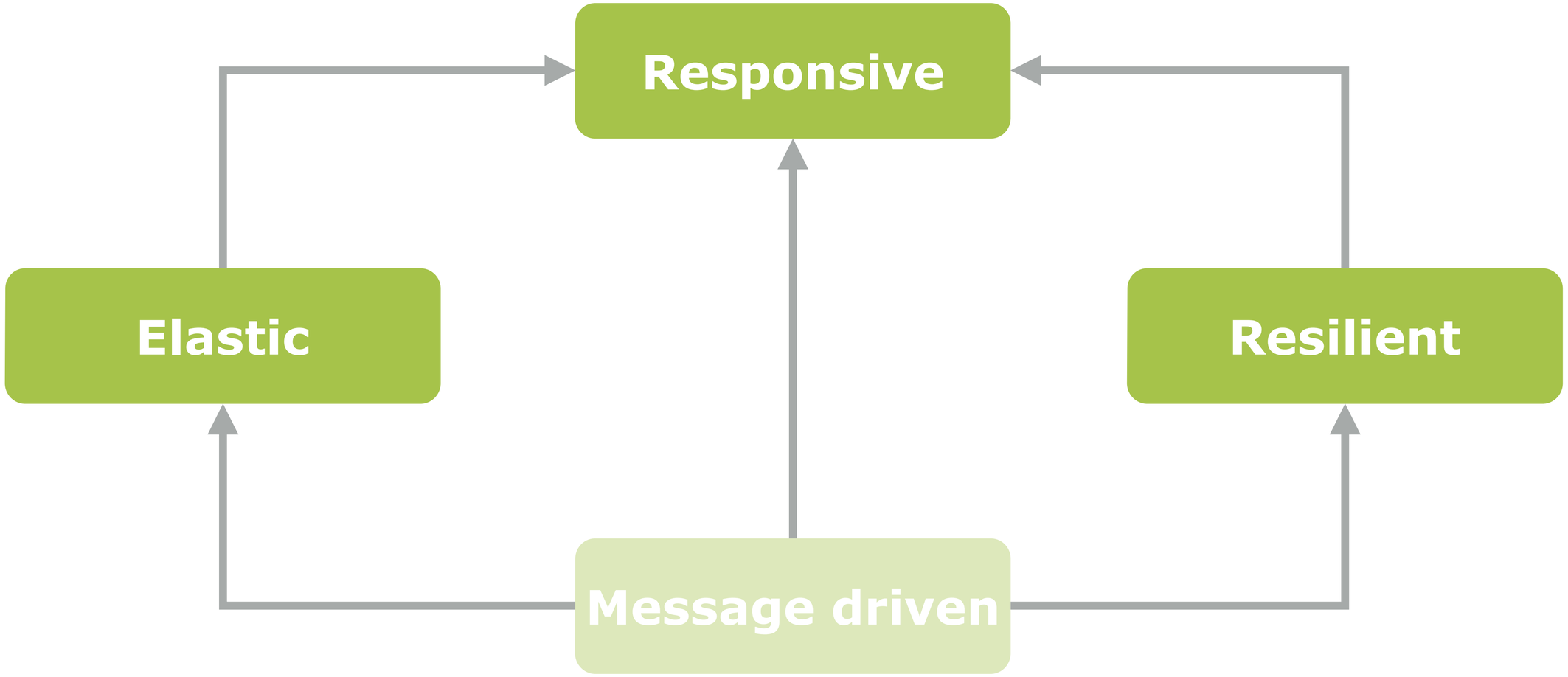
Responsive - Timeout



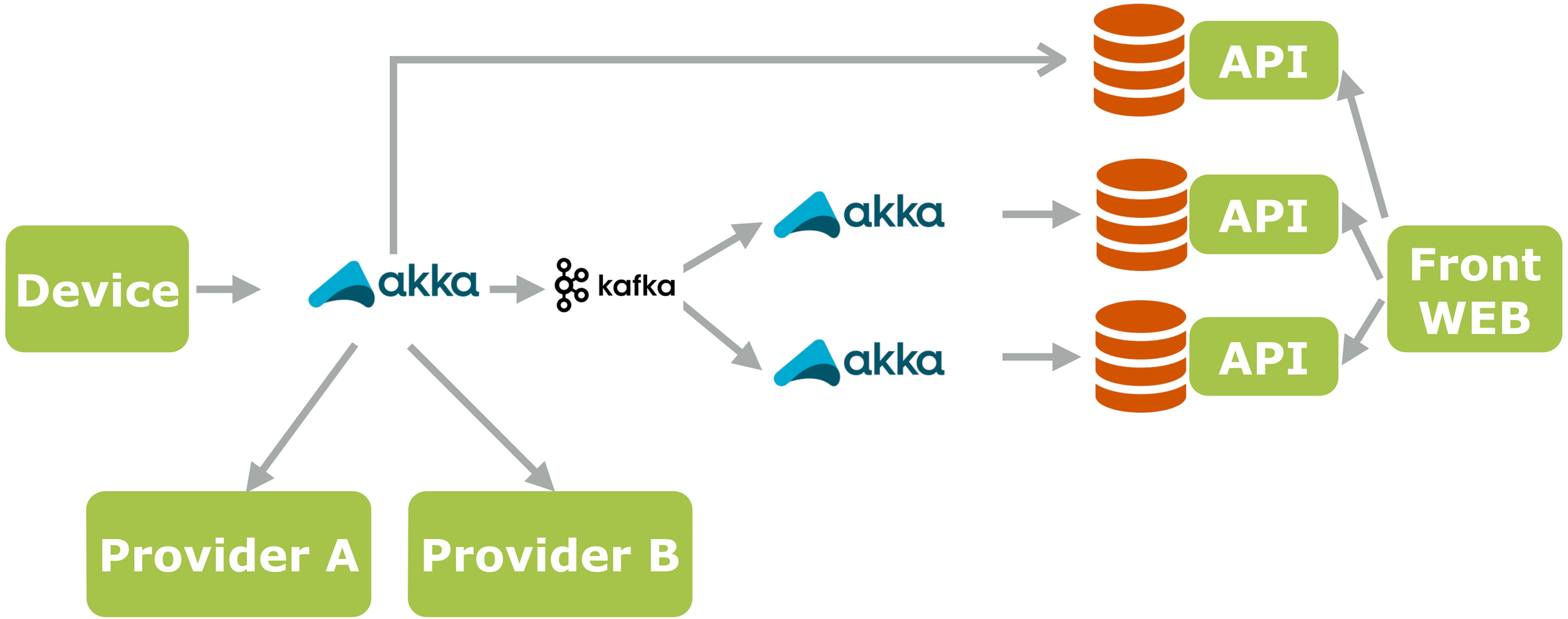
Recherche



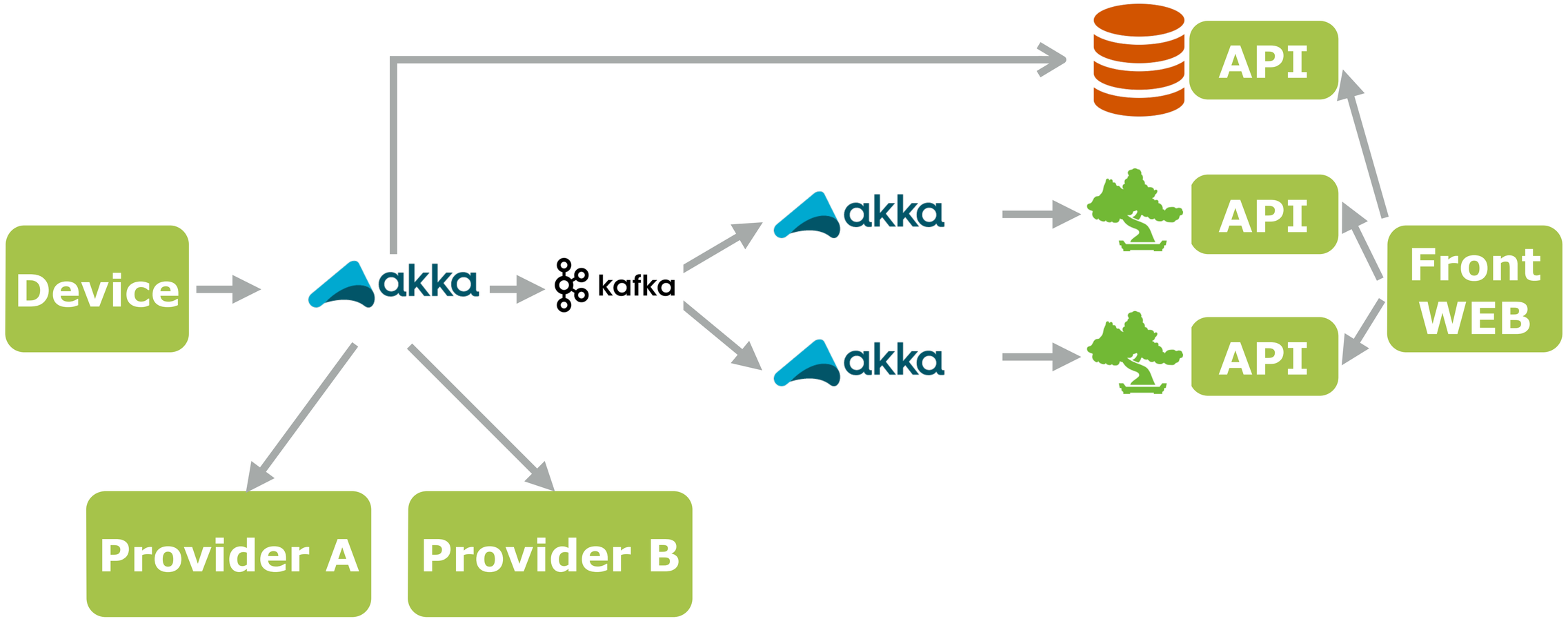
Reactive Manifesto et ES



Technos


Device
akka
kafka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Technos


Device
akka
kafka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Couchbase

Couchbase

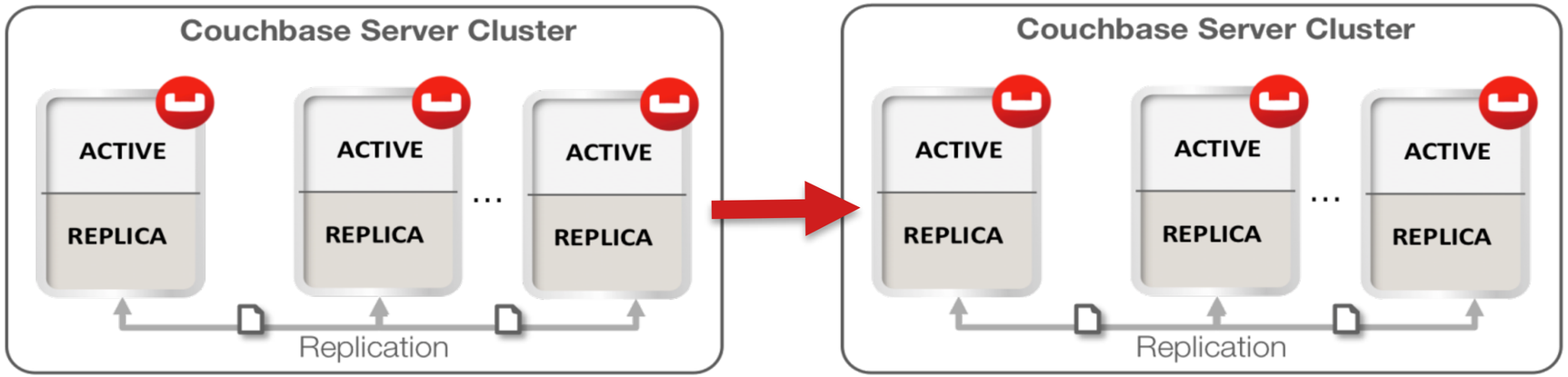
DEVOXX™ France

Couchbase

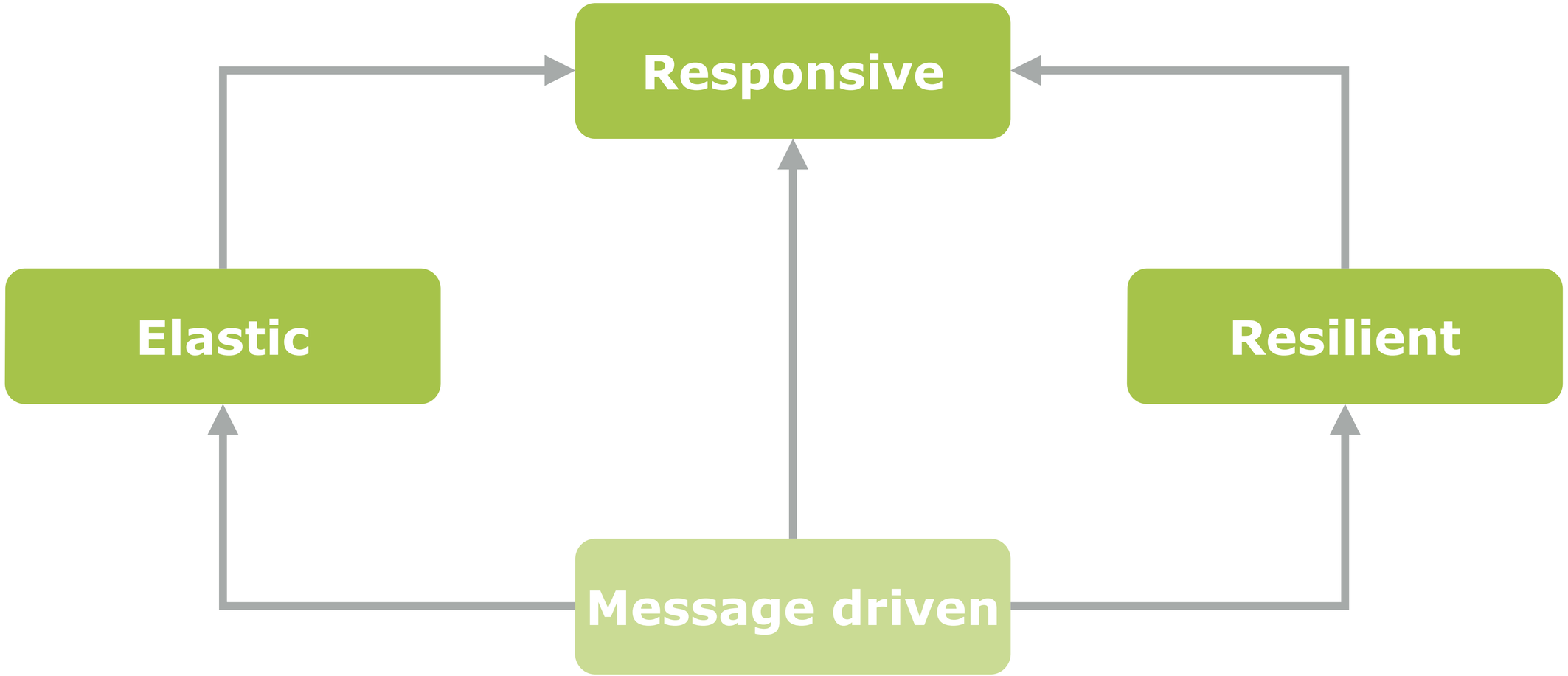
- Gestion documents
- Schemaless
- Clé / valeur
- Cache distribué

Couchbase

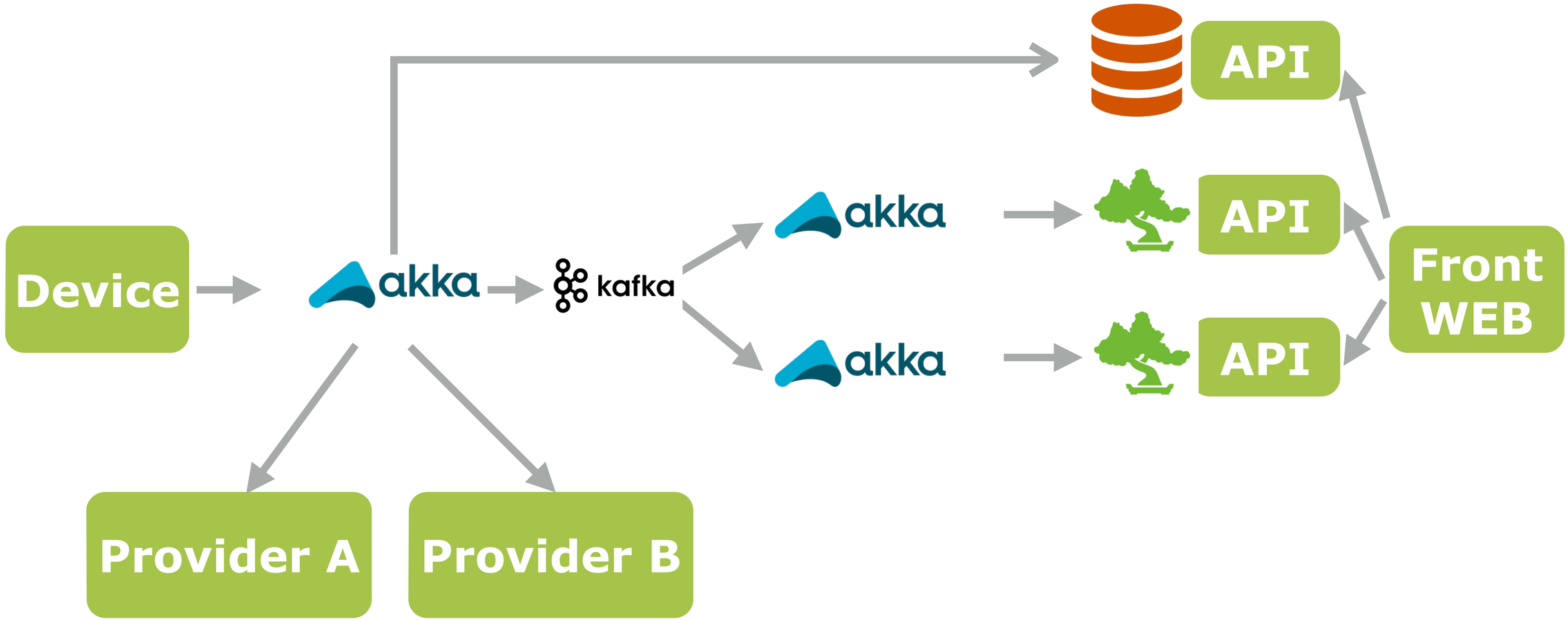
- Fail over node
- Cross data center replication



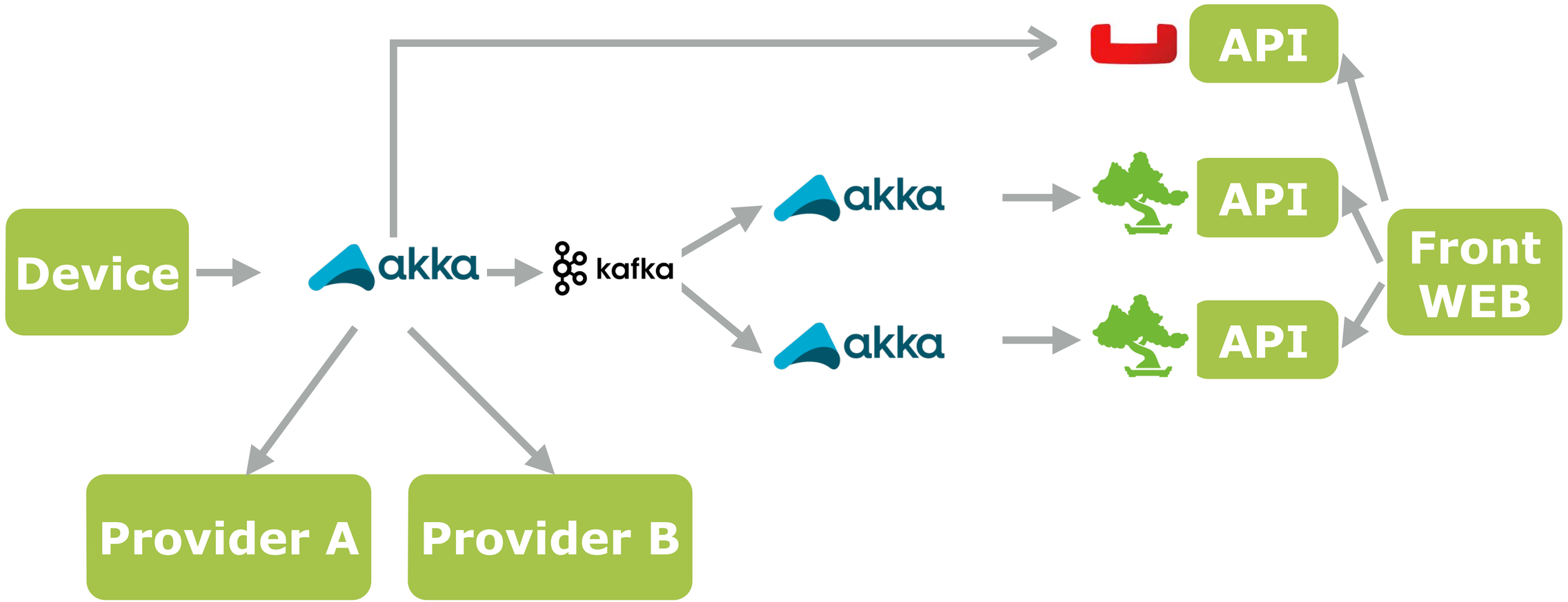
Reactive Manifesto et CB



Technos


Device
akka
kafka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Technos



PLAY / REACT JS

DEVOXX™ France

DEVOXX™ France

PLAY

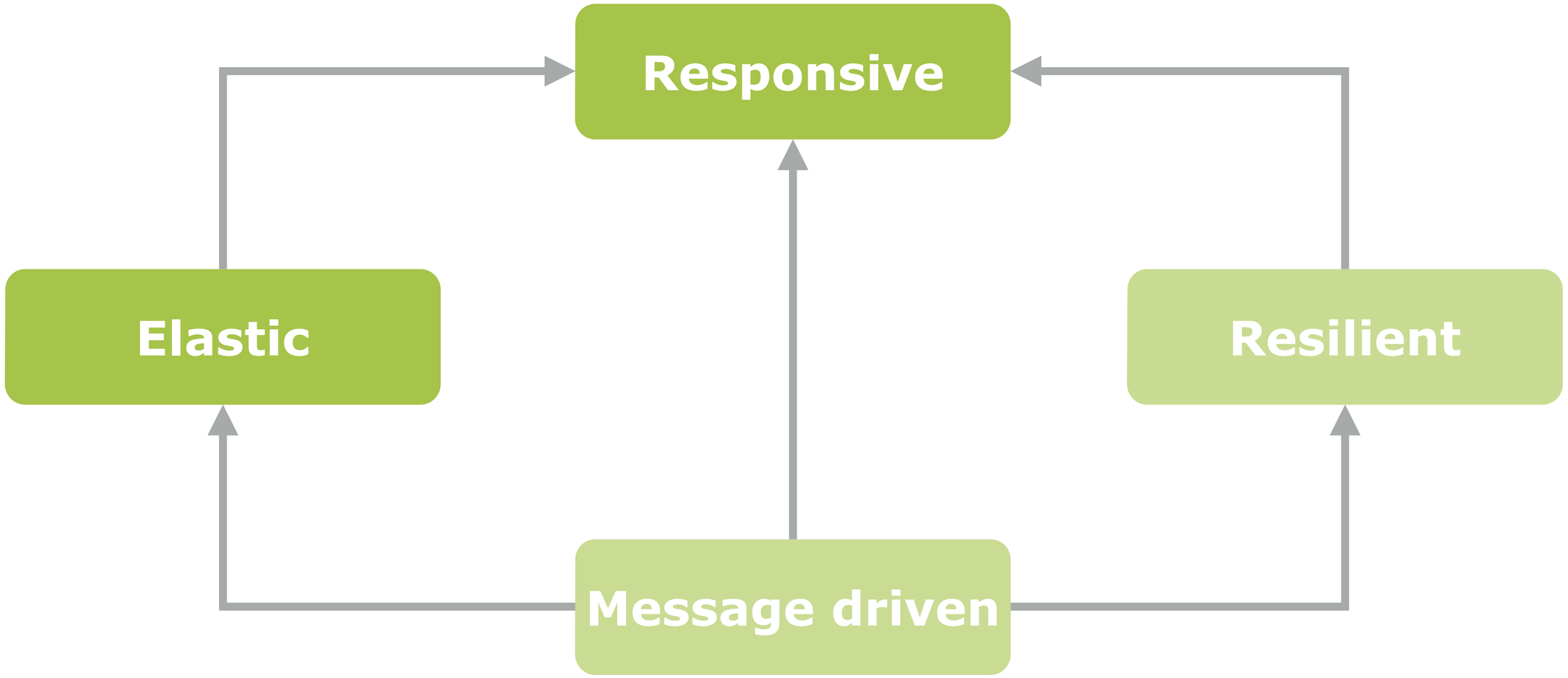
- Formulaires, Json, validation
- Intégration Scala et Akka
- IO non bloquant
- Sans état

DEVOXX France

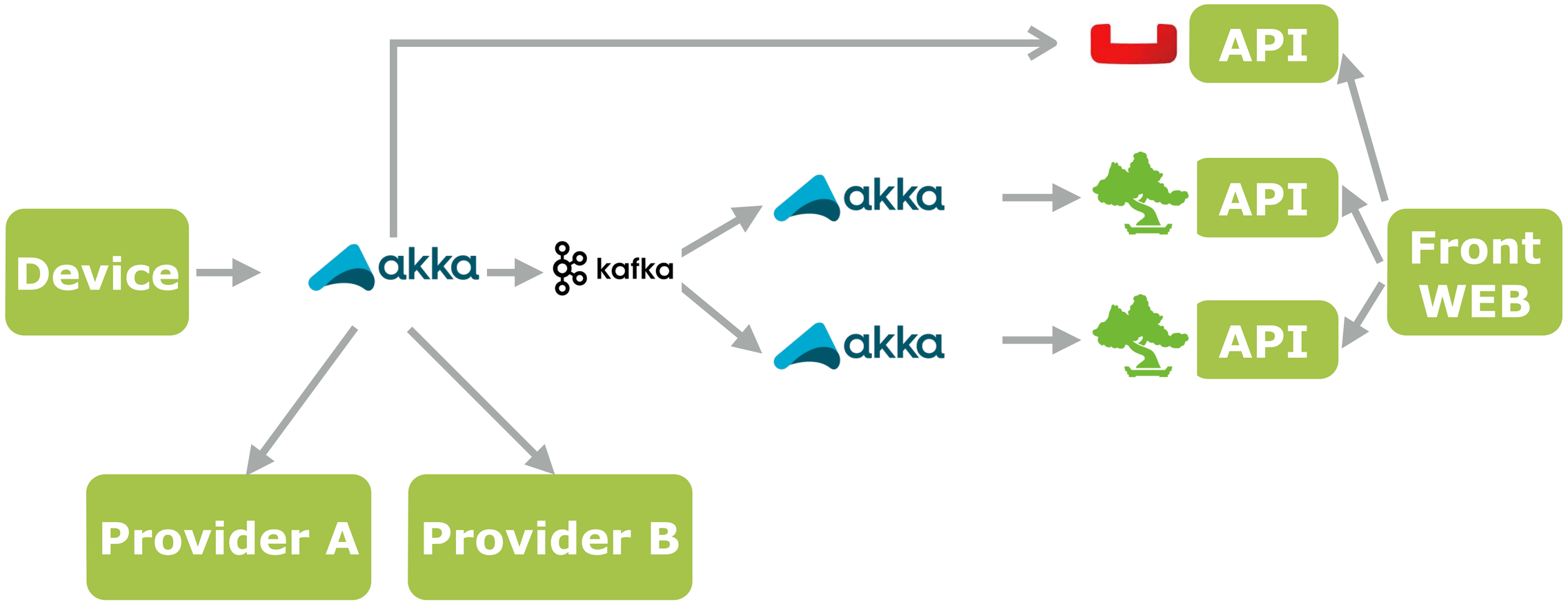
REACT JS

- Créé par Facebook et Instagram
- Composants
- Data flow unique
- Performant

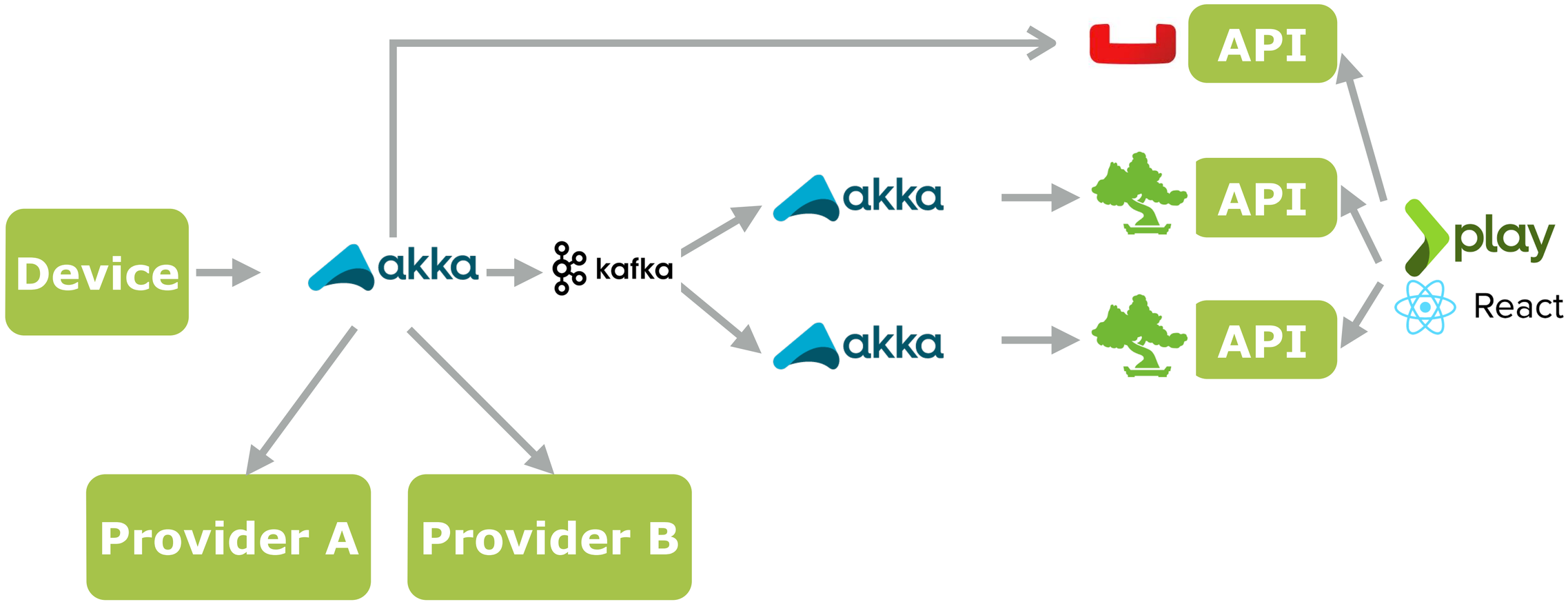
Reactive Manifesto et Play



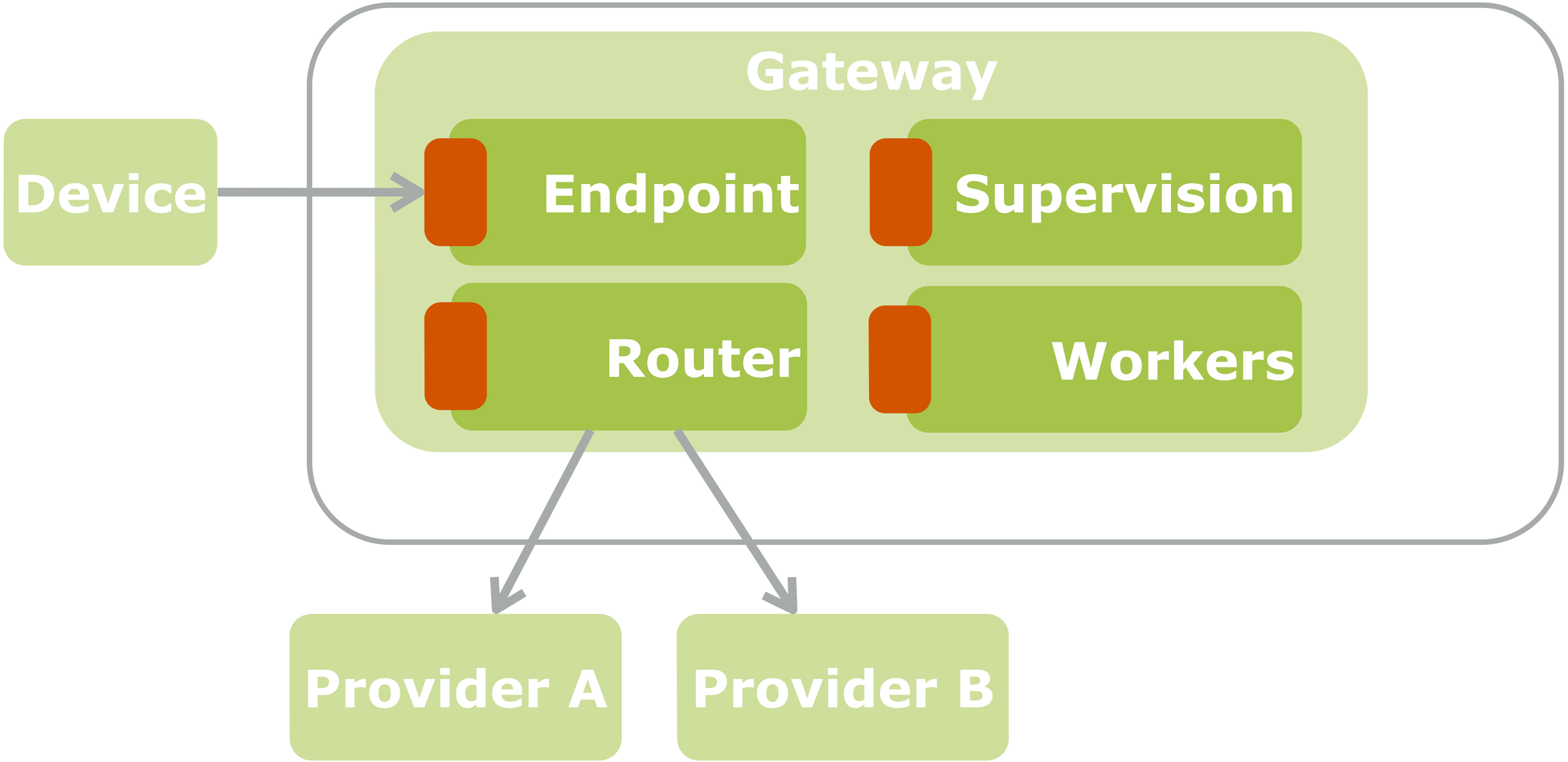
Technos



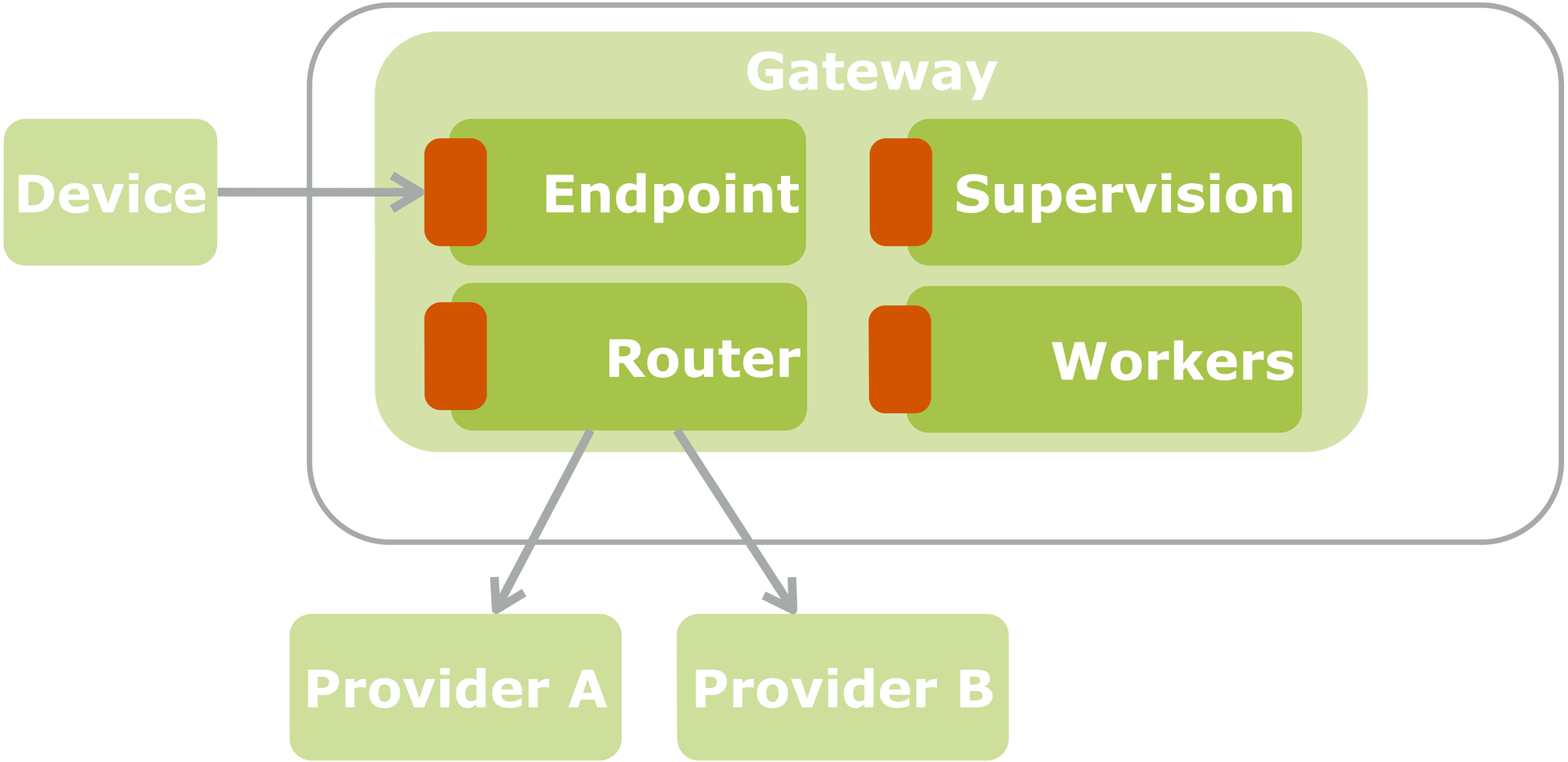
Technos


Device
akka
kafka
akka
akka
API
API
API
play
React
Provider A
Provider B

Testing


Gateway
Device
Endpoint
Supervision
Router
Workers
Provider A
Provider B

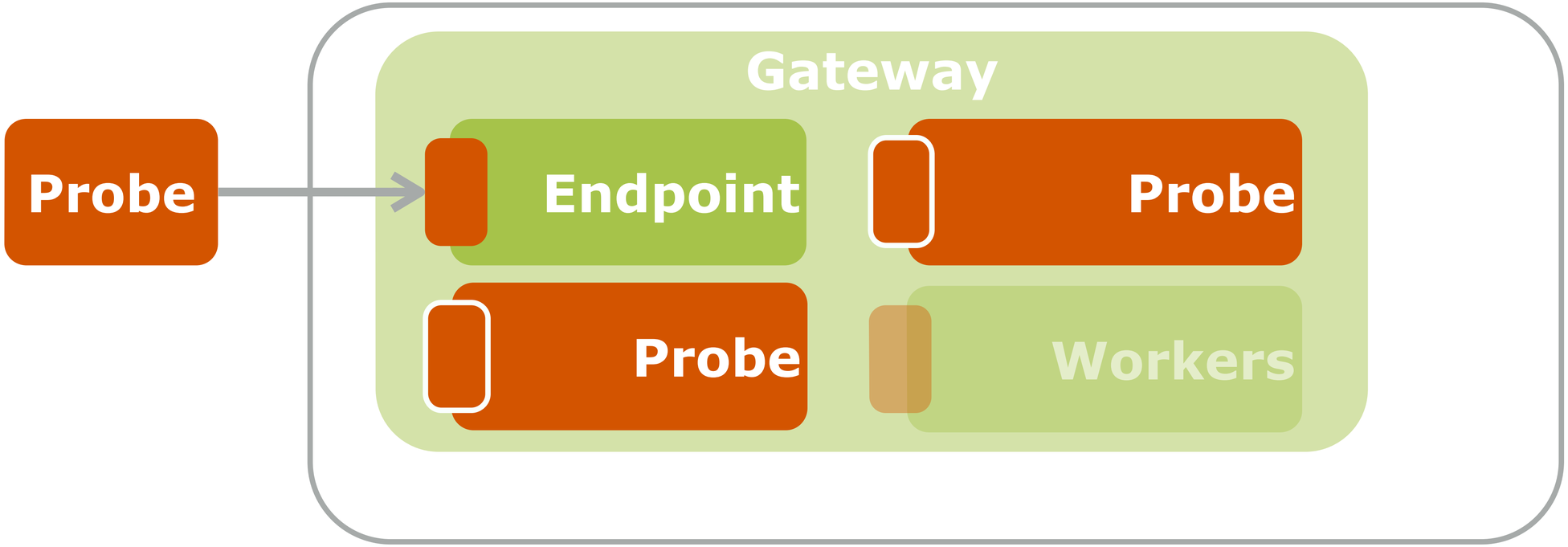
Test unitaire



akka testkit

DEVOXX France

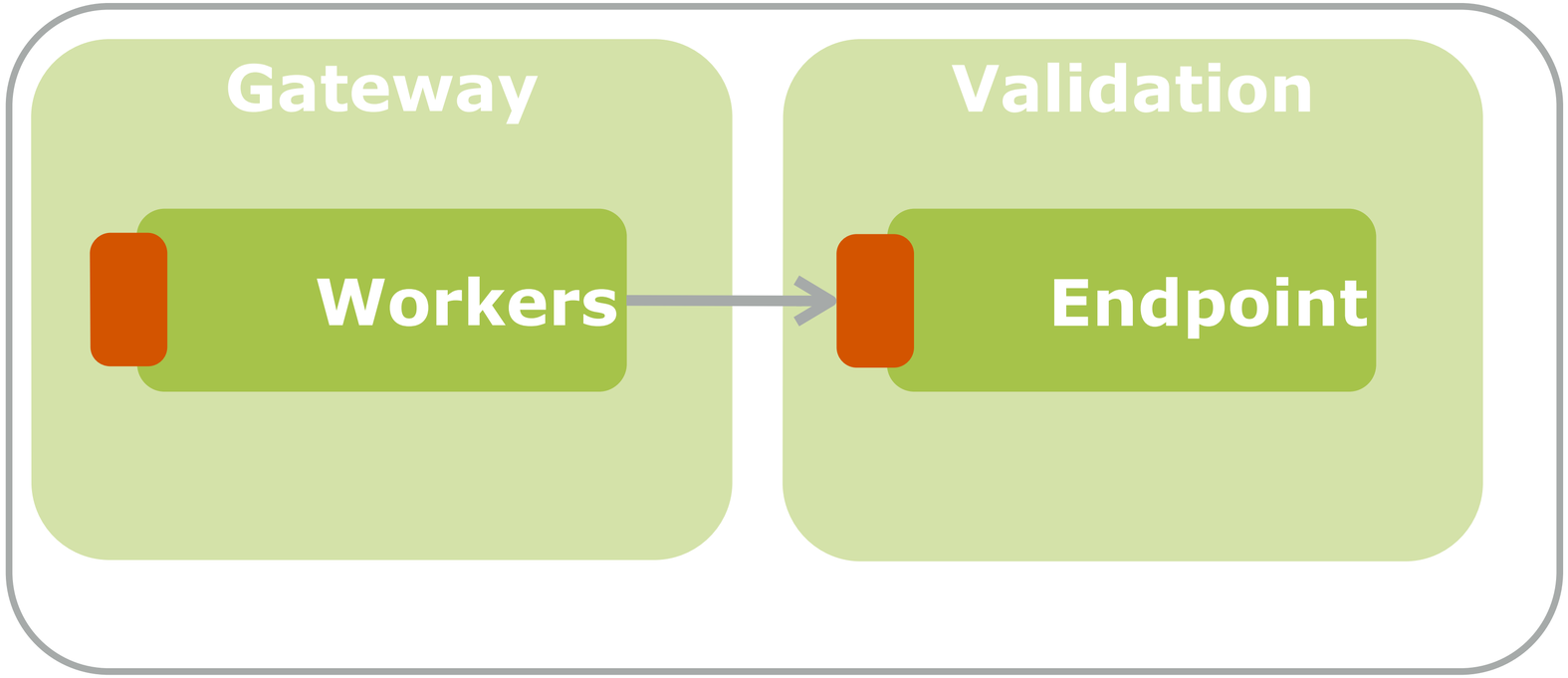
Test unitaire



akka testkit

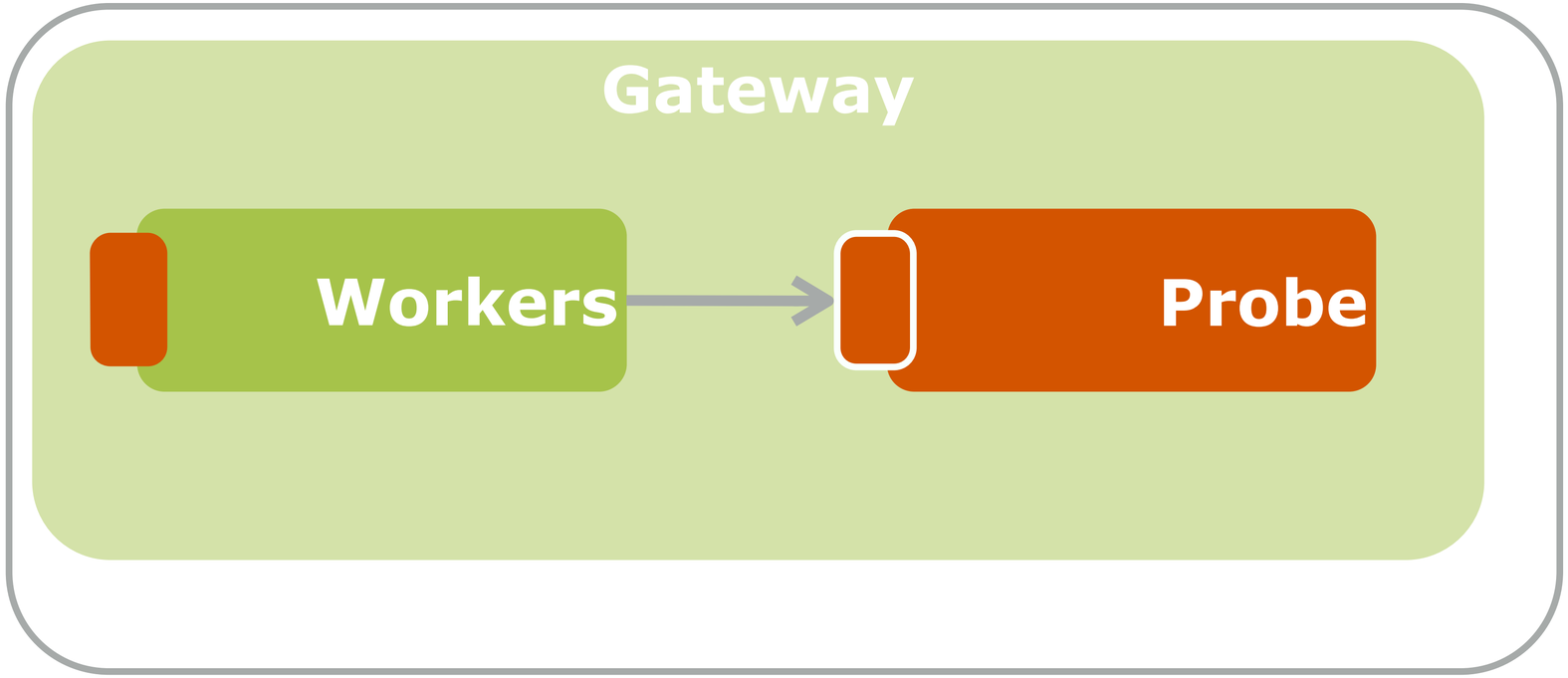
DEVOXX France

Test unitaire



akka testkit

Test unitaire



akka testkit

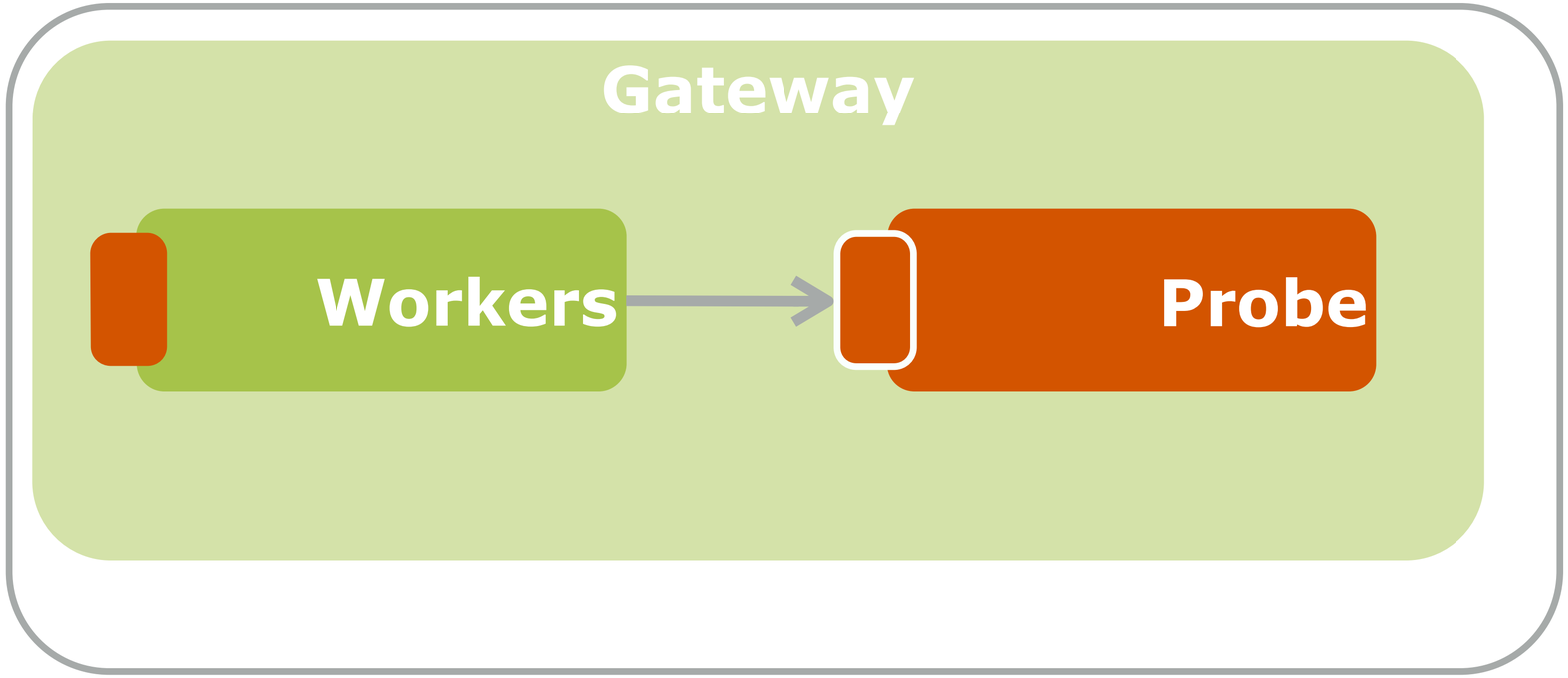
Test unitaire

```
describe("A kafka source actor") {

  it("should be able to pump a given number of message from kafka") {
    val probe = TestProbe()

    val reader = system.actorOf(Props(classOf[KafkaSource], probe))

    //send 11 messages on a topic

    probe.send(reader, 10)
    probe.receiveN(10, max = 3.seconds) shouldBe messages.take(10)

    probe.expectNoMsg(1.second)
  }
}
```

Test unitaire

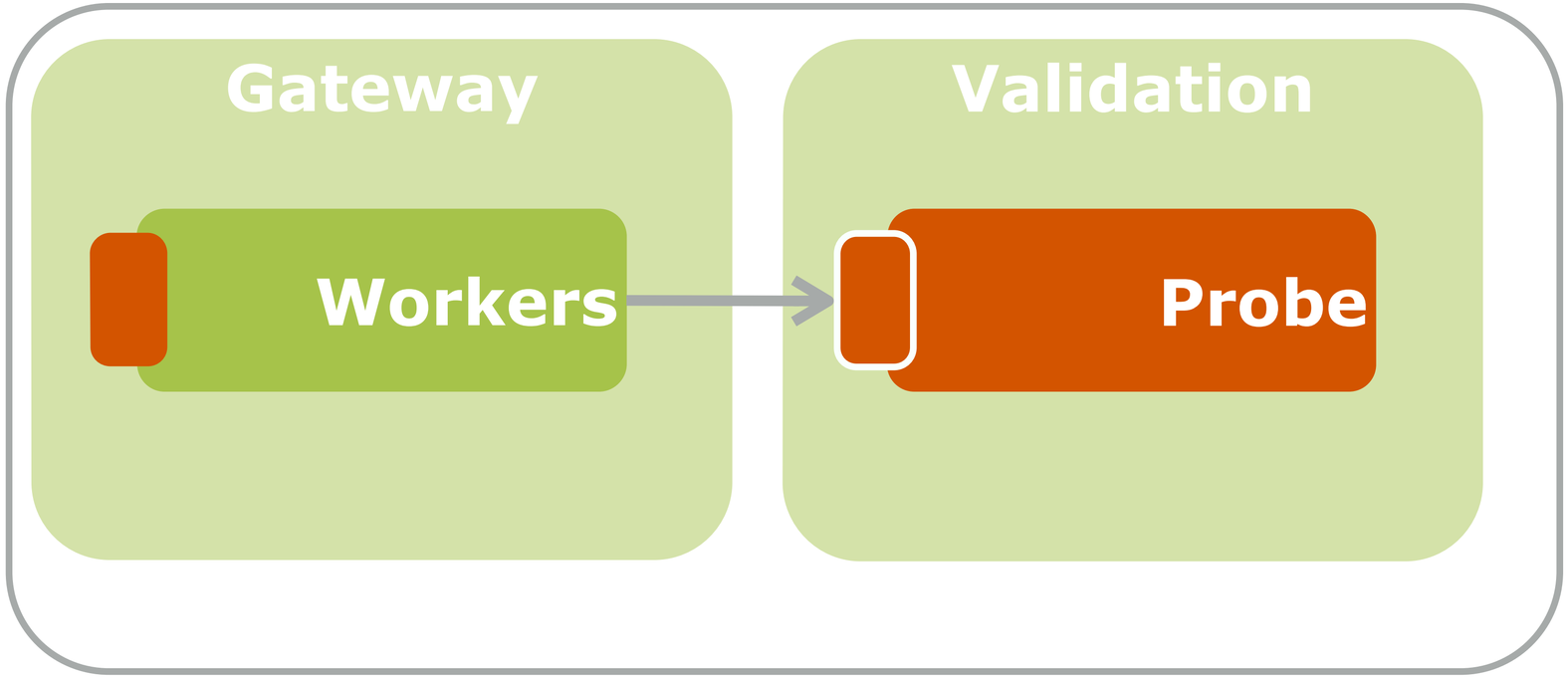
```
describe("A terminal session") {
  it("should have an idle timeout") {

    implicit val system = ActorSystem("test",
   ConfigFactory.parseString("session.timeout=\"500 milliseconds\""))

    val probe = TestProbe()

    val session = system.actorOf(DeviceSession.props)
    probe.watch(session)

    probe.expectTerminated(session, 1.second)
  }
}
```

Test multi-jvm



sbt
multi-jvm

Test multi-jvm

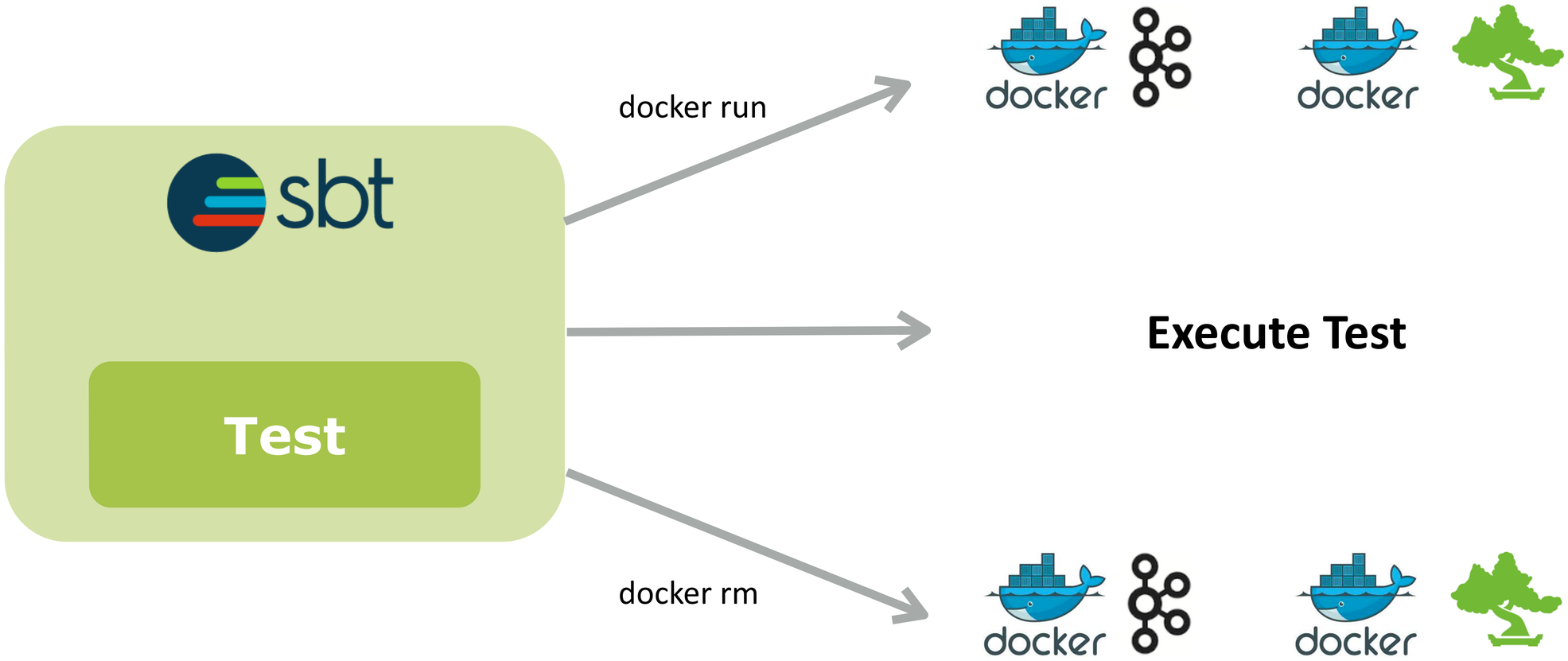


sbt
multi-jvm

Test multi-jvm

```
"form a cluster" in {
    runOn(clusterNodes: _*) {
     SeedDiscovery.joinCluster(system)
      enterBarrier("deployed")
}

"support network partition" in {
  runOn(node1) {
    for {
      from <- group_1
      to <- group_2
    } {
      testConductor.blackhole(from, to, Direction.Both).await
    }
  }
```

Test intégration



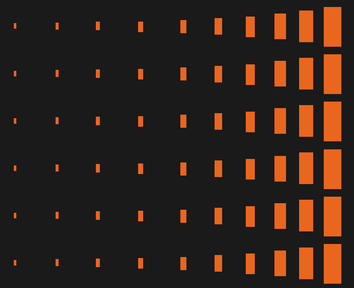
Versioning

NOT SUPPORTED YET

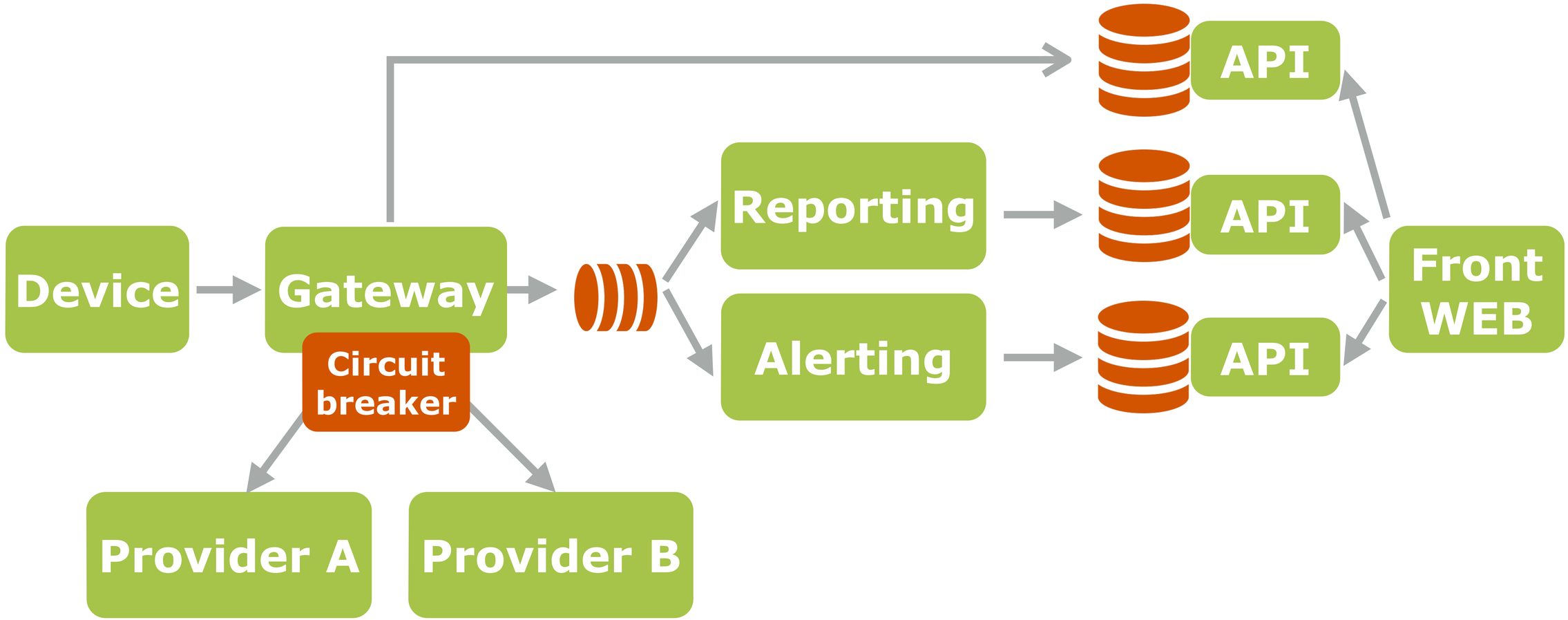
Versioning

- **JSON**
- **Scala**
- **Apache Avro**

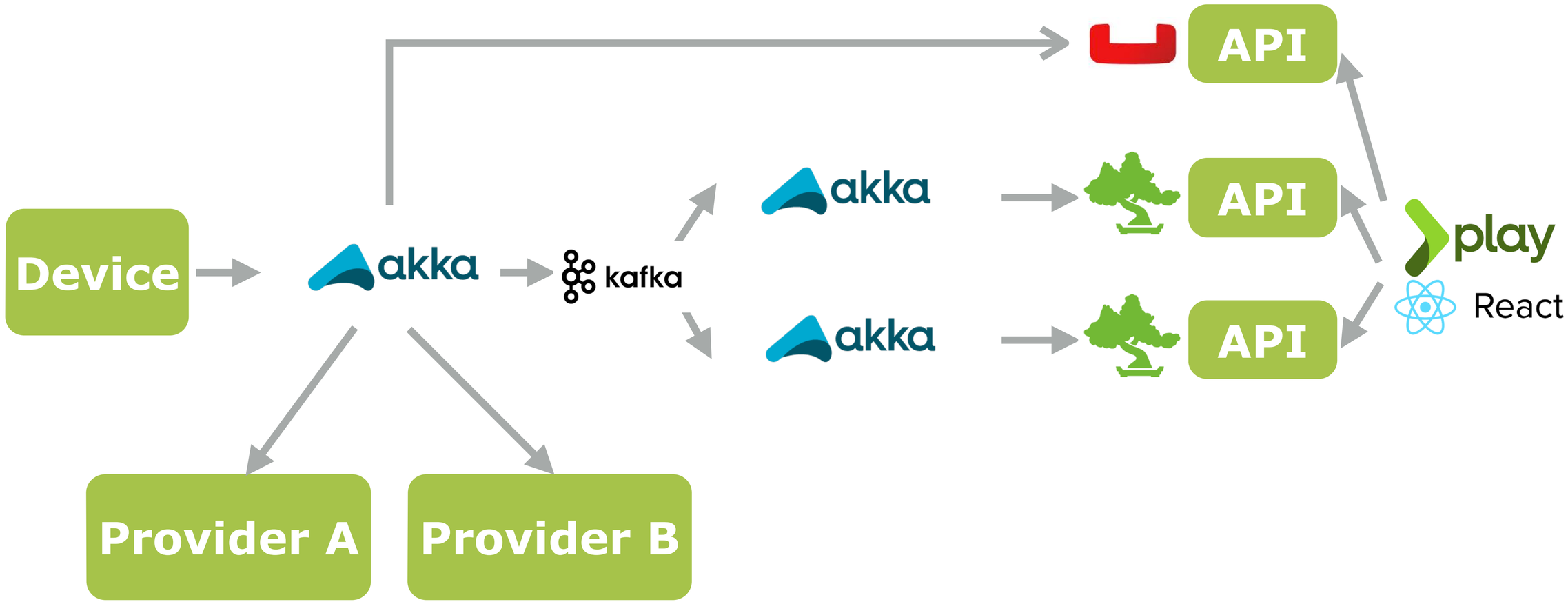
QUESTIONS ?



Résumé


Device
Gateway
Circuit breaker
Provider A
Provider B
Reporting
Alerting
API
API
API
Front WEB

Résumé


Device
akka
kafka
akka
akka
API
API
API
play
React
Provider A
Provider B

DEVOXX™ France

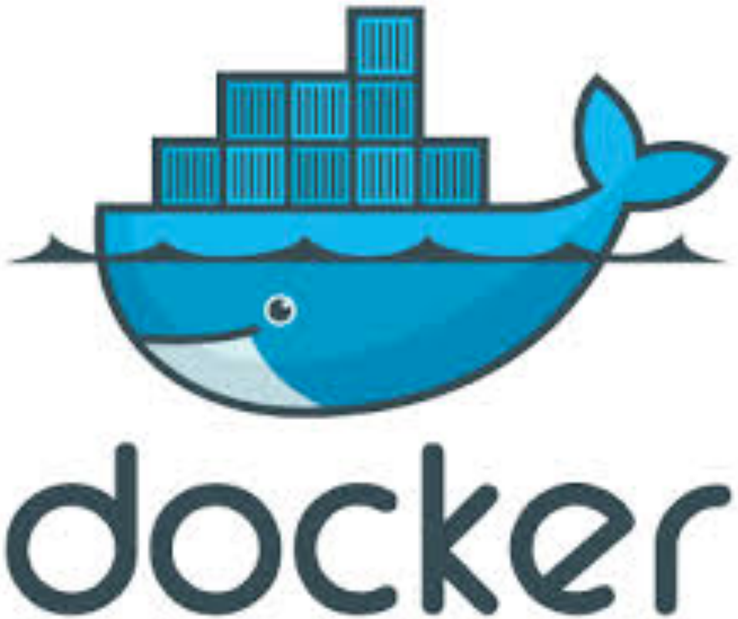
Agenda

- Contexte
- Comment concevoir ?
- Comment développer ?
- **Comment déployer ?**
- Comment monitorer ?

DEVOXX™ France

Thanks

Docker

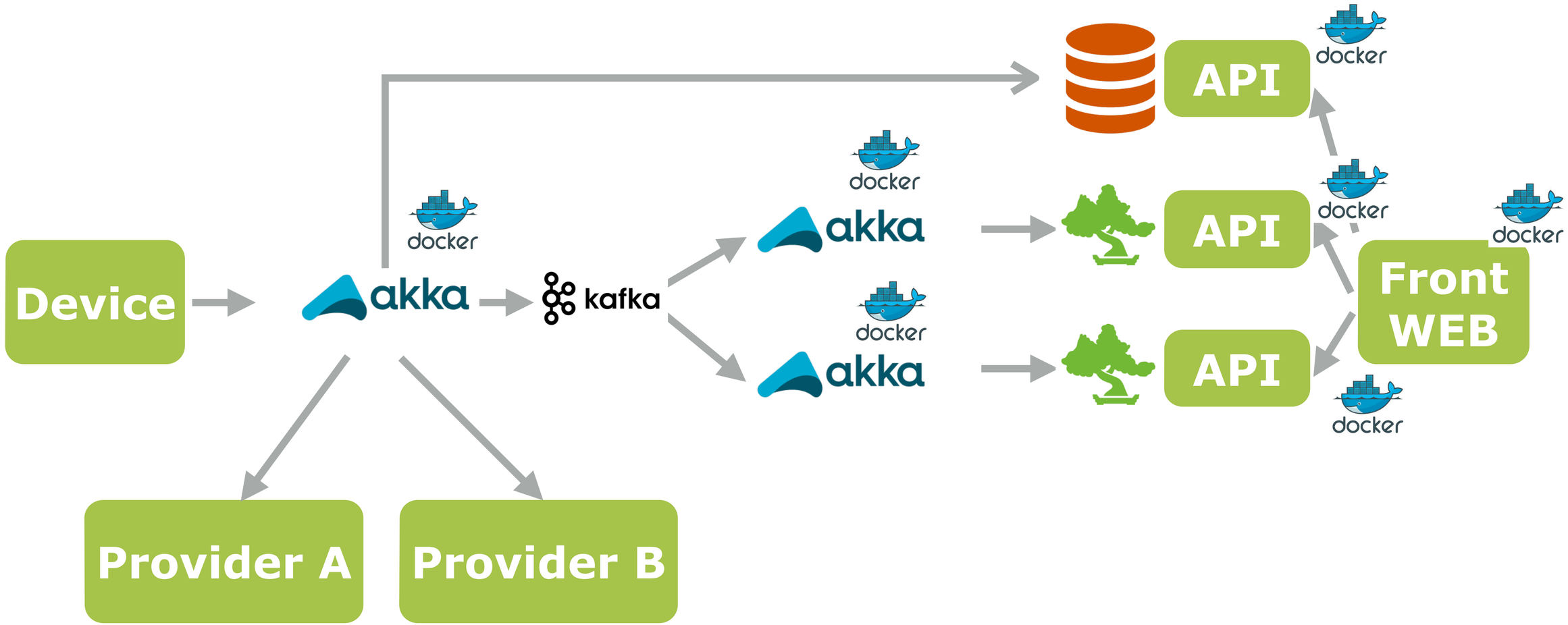


DEVOXX France

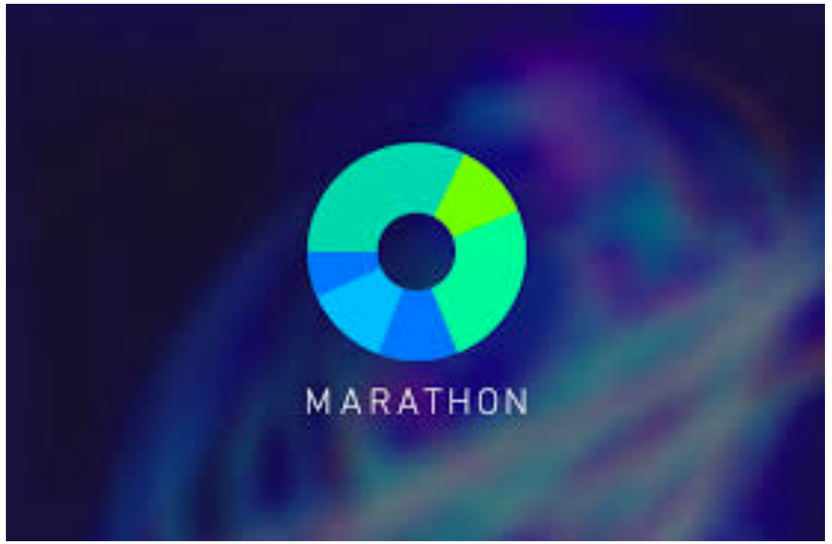
Docker

```
FROM ubuntu

RUN apt-get install -y wget software-properties-common
RUN apt-add-repository ppa:ansible/ansible
RUN apt-get update && apt-get install -y ansible

RUN wget https://bootstrap.pypa.io/ez_setup.py -O - | python
RUN easy_install pip
RUN pip install boto

ENV ANSIBLE_HOST_KEY_CHECKING=False
```

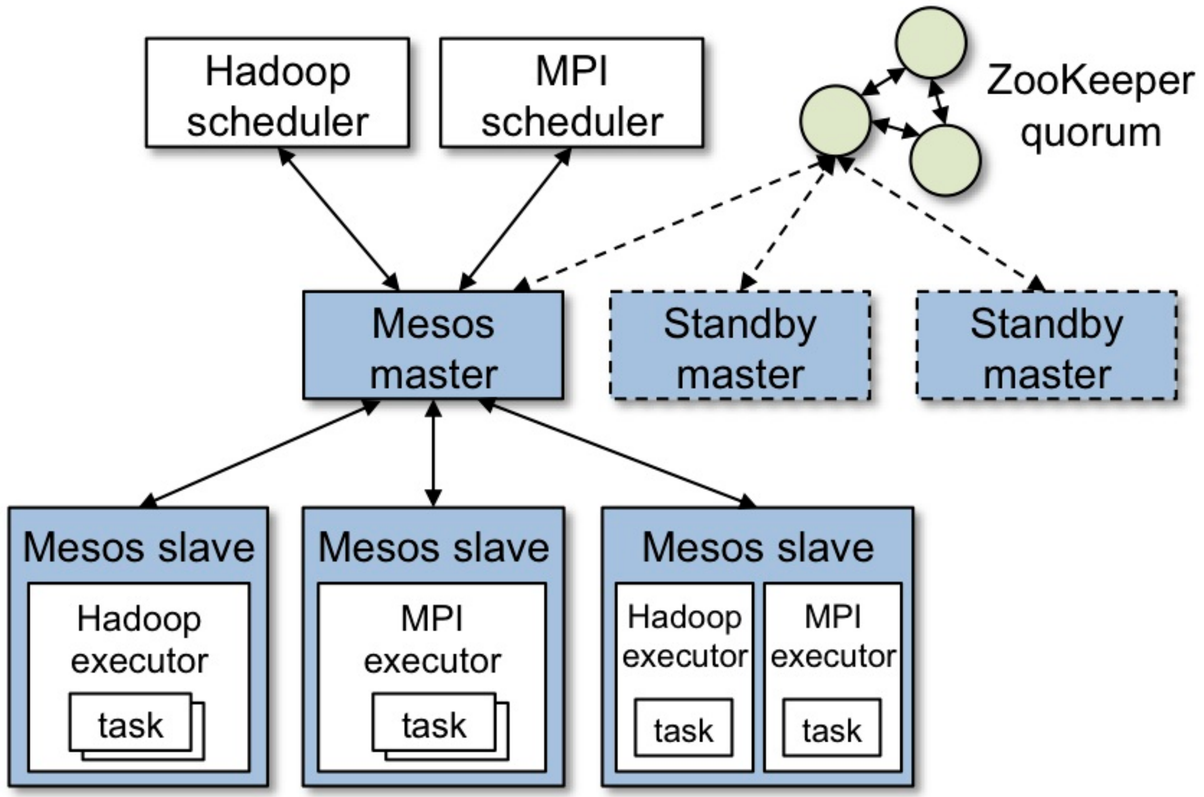
Technos


Device
akka
kafka
docker
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

Mesos / Marathon

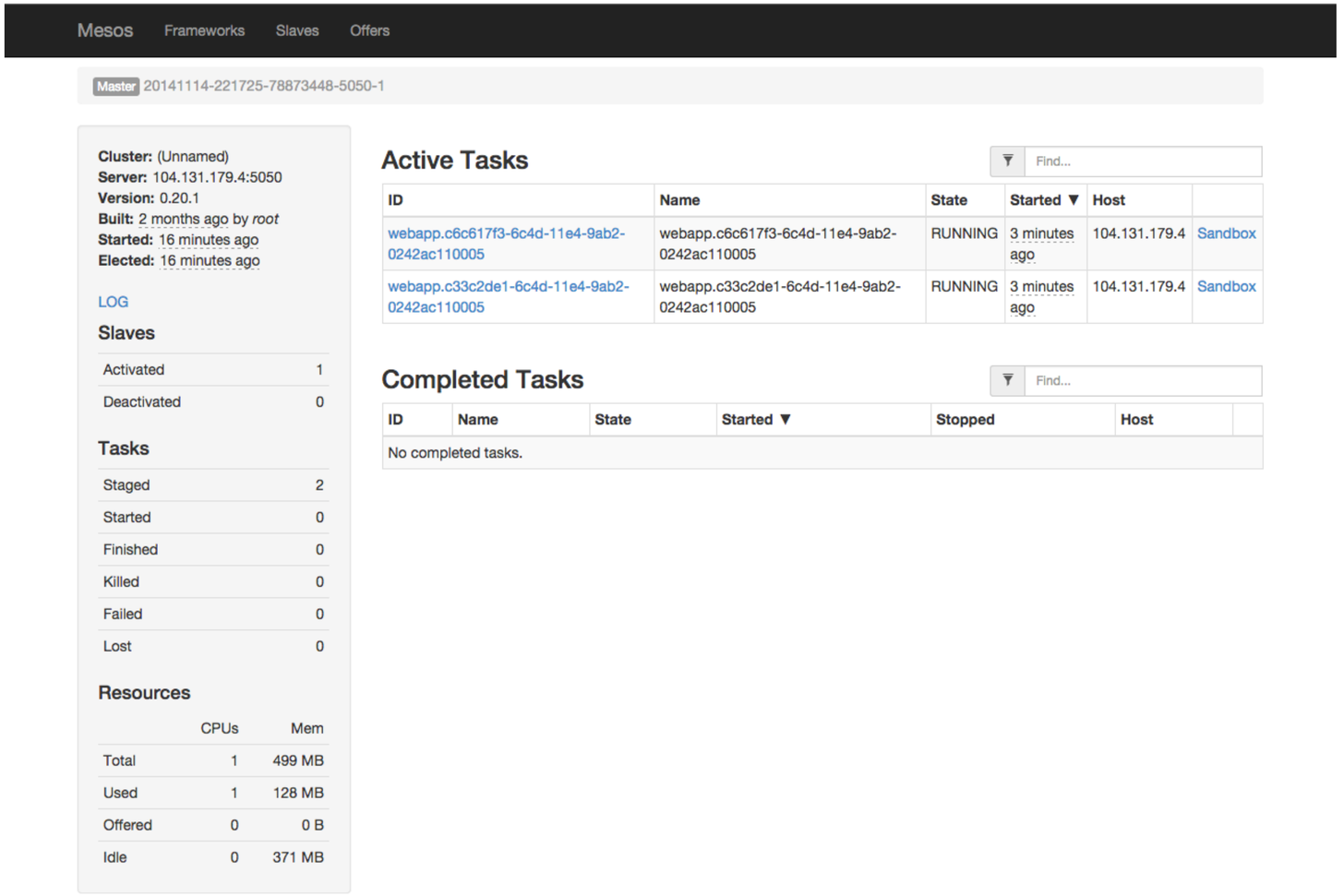


Mesos


Hadoop scheduler
MPI scheduler
ZooKeeper quorum
Mesos master
Standby master
Standby master
Mesos slave
Hadoop executor
task
Mesos slave
MPI executor
task
Mesos slave
Hadoop executor
task
MPI executor
task

Mesos

- Distribution des ressources
- Détection des erreurs



Marathon

- **Execution d’applications**
- **API Rest**
- **Beaucoup d’options**

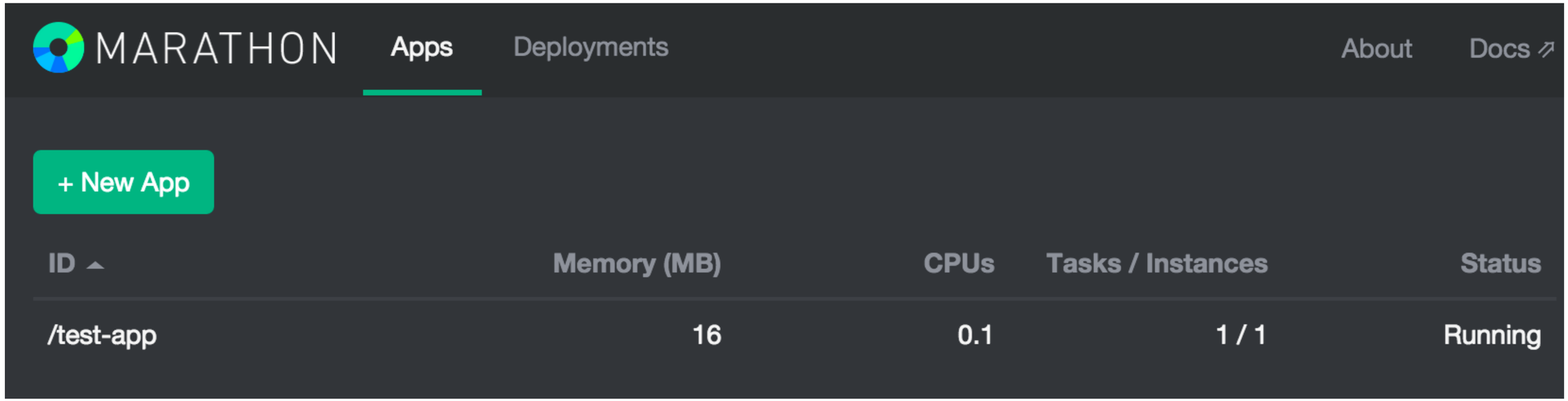
Marathon Config

```
{
 "id": "provisioning",
 "instances": 1,
 "cpus": 2.0,
 "mem": 512,
 "ports":[9000],
 "container": {
   "type": "DOCKER",
   "docker": {
     "image": "project/web:0crec10a90724f791caaf95cbc62ea61abbd6376",
     "network": "BRIDGE",
     "portMappings": [
        { "containerPort": 9000, "servicePort": 9000,
          "hostPort": 0, "protocol": "tcp" }
     ]
    }
  }
}
```

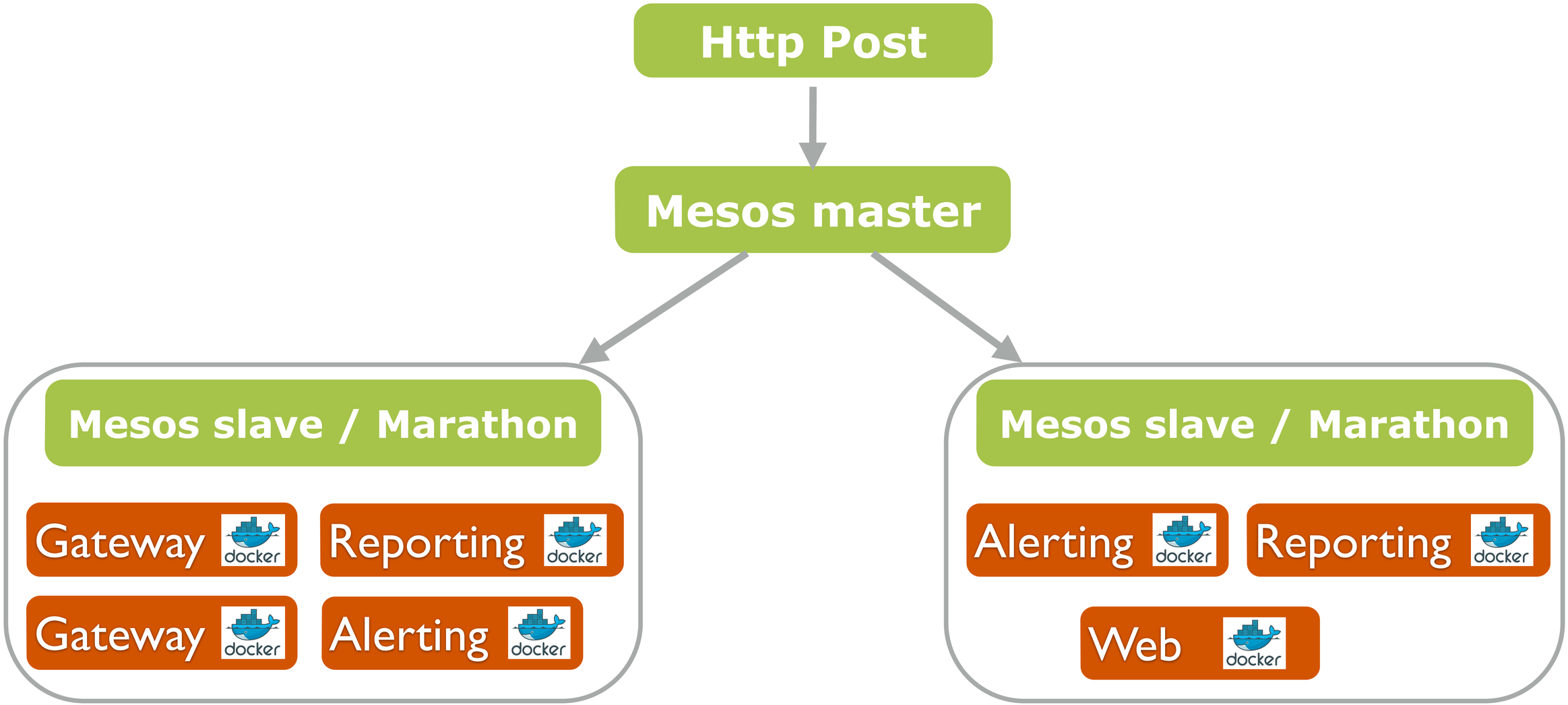
Marathon Config

```
…
"healthChecks": [
    {
      "path":"/health",
      "portIndex":0,
      "protocol":"HTTP",
      "gracePeriodSeconds":3,
      "intervalSeconds":10,
      "timeoutSeconds":10,
      "maxConsecutiveFailures":3
    }
],
…
"upgradeStrategy": {
    "minimumHealthCapacity": 0
},
…
```

Marathon



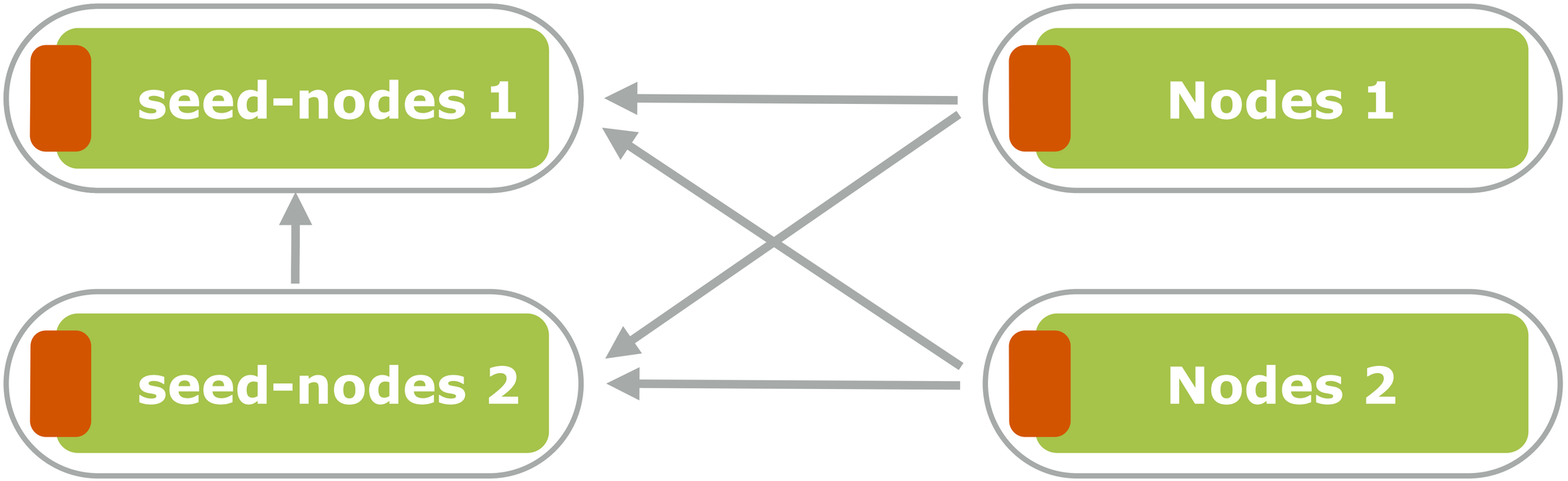
Architecture



DEVOXX France

Découverte de services

Découverte de services



Découverte de services

- Akka 2.4 snapshot
- Intégration avec Docker


```
remote {

  enabled-transports = ["akka.remote.netty.tcp"]

  netty.tcp {

    hostname = "127.0.0.1"
    port = 2551
  }
}
```

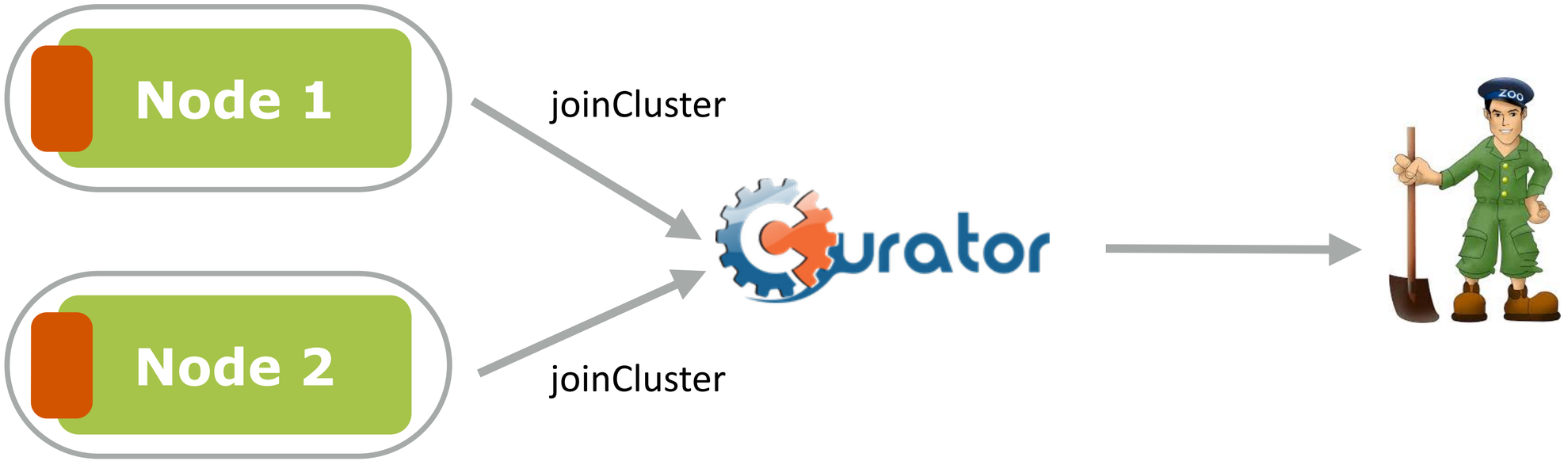
Découverte de services

- Akka 2.4 snapshot
- Intégration avec Docker


```
remote {

  enabled-transports = ["akka.remote.netty.tcp"]

  netty.tcp {

    hostname = "127.0.0.1"
    port = 2551

    bind-hostname = "192.132.122.12"
    bind-port = 32001
  }
}
```

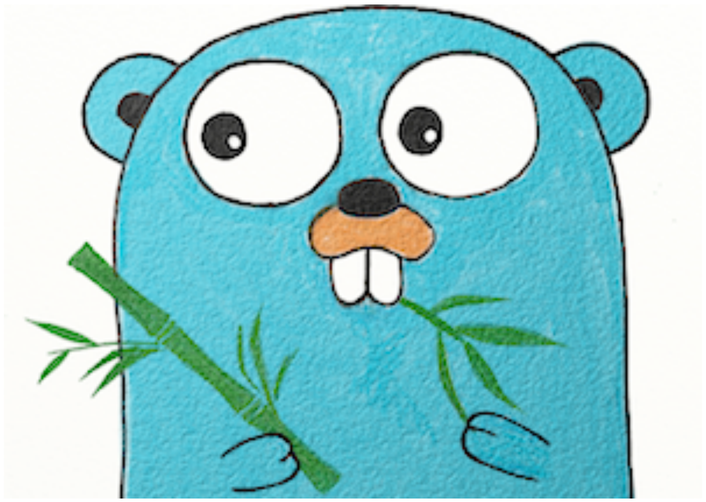
Découverte de services

Découverte de services



DEVOXX™ France

Bamboo

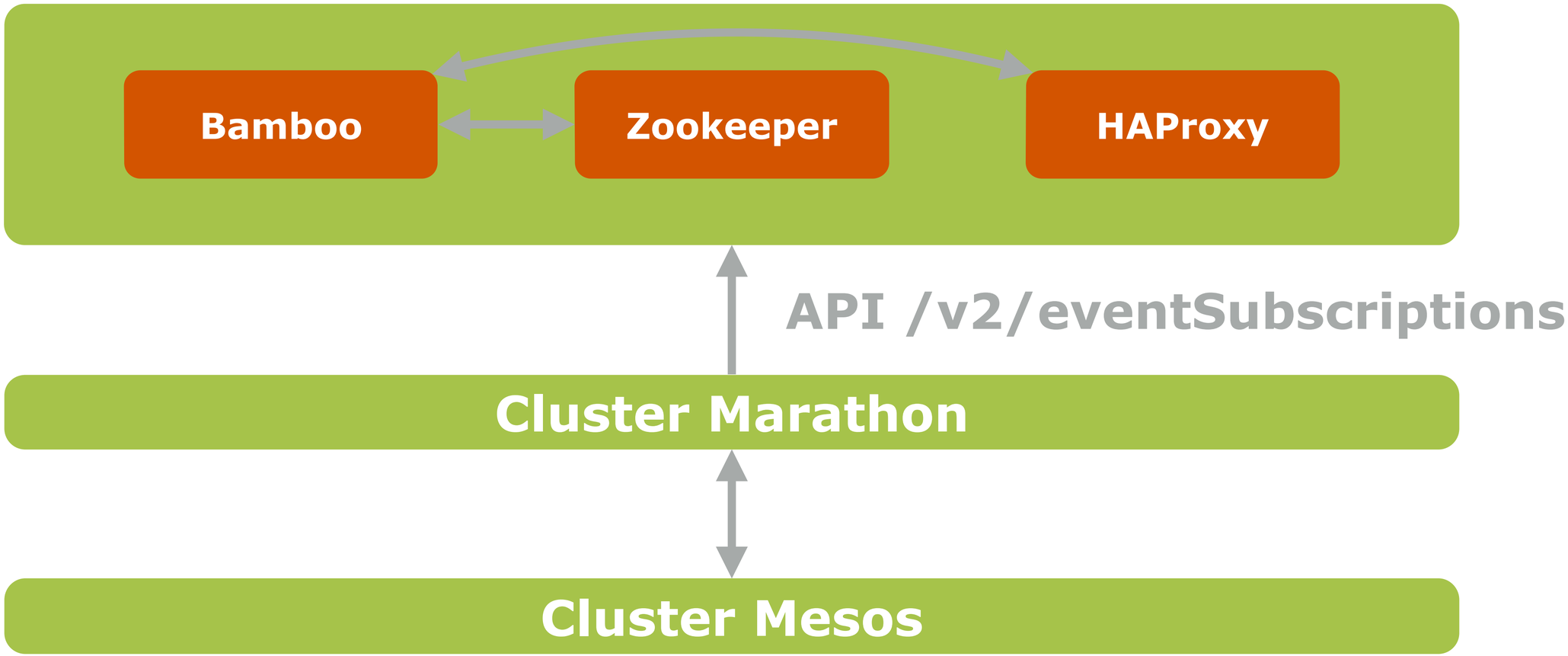


DEVOXX™ France

Problème

- IP dynamique
- Port dynamique
- Nombre d'instances
- Sous domaines (web, monitoring, …)

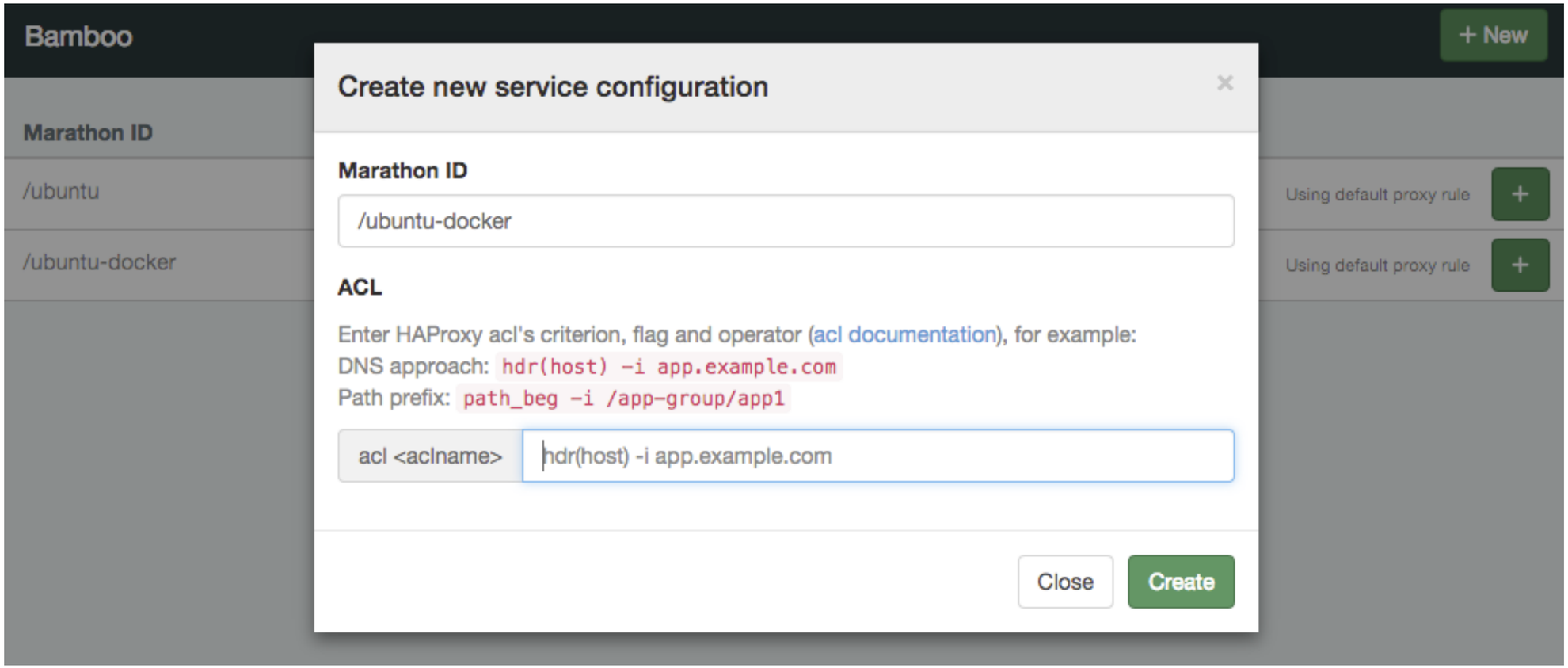
Bamboo



Bamboo: UI

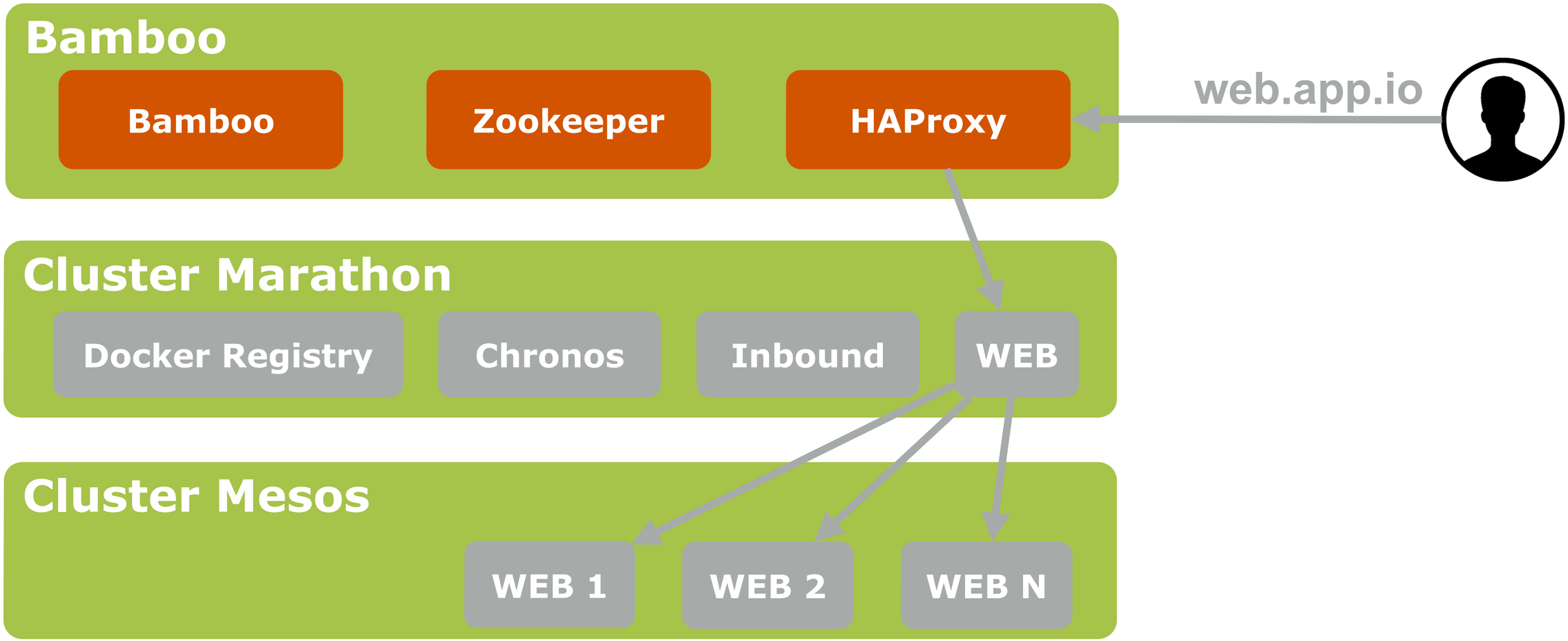

Bamboo
+ New
Marathon ID
ACL
Instances
/ubuntu
1
Using default proxy rule
+
/ubuntu-docker
hdr(host) -i ubuntu-docker.example.com
2

Bamboo: UI


Bamboo
+ New
Marathon ID
/ubuntu
/ubuntu-docker
Using default proxy rule
Using default proxy rule
Create new service configuration
Marathon ID
/ubuntu-docker
ACL
Enter HAProxy acl's criterion, flag and operator (acl documentation), for example:
DNS approach: hdr(host) -i app.example.com
Path prefix: path_beg -i /app-group/app1
acl <aclname>
hdr(host) -i app.example.com
Close
Create

DEVOXX France

Bamboo



Ansible

ANSIBLE

DEVOXX™ France

Problème

- Kafka
- Couchbase
- Marathon
- Mesos master / slave
- Elasticsearch master / search / data
- GoCD server / agent
- …

DEVOXX™ France

Introduction

- **Infrastructure as code**
- **Agent-less**
- **Python / Boto**

Inventaire

```
$ cat inventory

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
```

Modules

- Commands (command, shell, script)
- Files (file, copy, unarchive, ...)
- System (service, hostname, cron, ...)
- Packaging (apt, yum, ...)
- Source Control (git, svn, ...)
- Notifications (mail, irc, slack, ...)

Tasks

```
$ cat tasks/main.yml

- name: Create /mnt/data directory
  file:
    path=/mnt/data/elasticsearch
    state=directory

- name: Install Elasticsearch
  apt:
    name=elasticsearch
    state=latest
    force=yes
  notify: Restart Elasticsearch
```

Rôles

```
$ tree -a .

├── files
│   └── template.json
├── handlers
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
│   └── elasticsearch.conf.j2
└── vars
    └── .gitkeep
```

Variables

```
$ cat group_vars/elasticsearch

elasticsearch:
  version: 1.4
  clustername: elasticsearch-cluster
  nodename: {{ hostvars[inventory_hostname]['ec2_id'] }}
  quorom: 1
  replicas: 2
  plugins:
    - head
    - paramedic
```

Templates

```
discovery.zen.ping.multicast.enabled: false

discovery.zen.ping.unicast.hosts:
[
   {% for host in groups[group_name] %}

      "{{ hostvars[host]['ansible_eth0'].ipv4.address }}"

   {% if not loop.last %},{% endif %}

   {% endfor %}
]
```

Playbooks

```
$ cat elasticsearch.yml

- hosts: es_nodes
  user: ubuntu
  sudo: yes

  roles:
    - all
    - java
    - elasticsearch
```

DEVOXX France

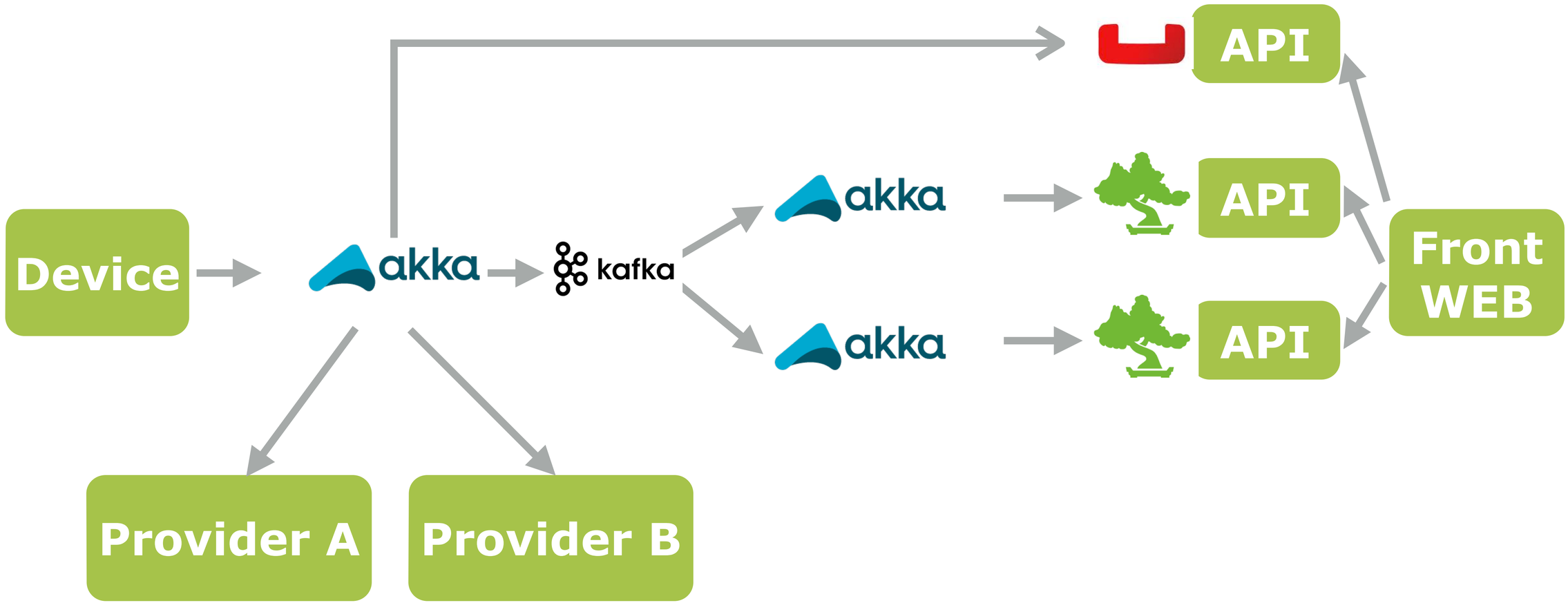
Résultats

```
TASK: [java | Install Oracle Java] ********************************************
ok: [52.17.94.140] => (item=oracle-java8-installer)
ok: [52.16.220.194] => (item=oracle-java8-installer)
ok: [52.17.93.82] => (item=oracle-java8-installer)

TASK: [java | Install OpenJDK] ************************************************
skipping: [52.16.220.194]
skipping: [52.17.94.140]
skipping: [52.17.93.82]

TASK: [elasticsearch | Check mounted volumes] *********************************
changed: [52.16.220.194]
changed: [52.17.94.140]
changed: [52.17.93.82]
```

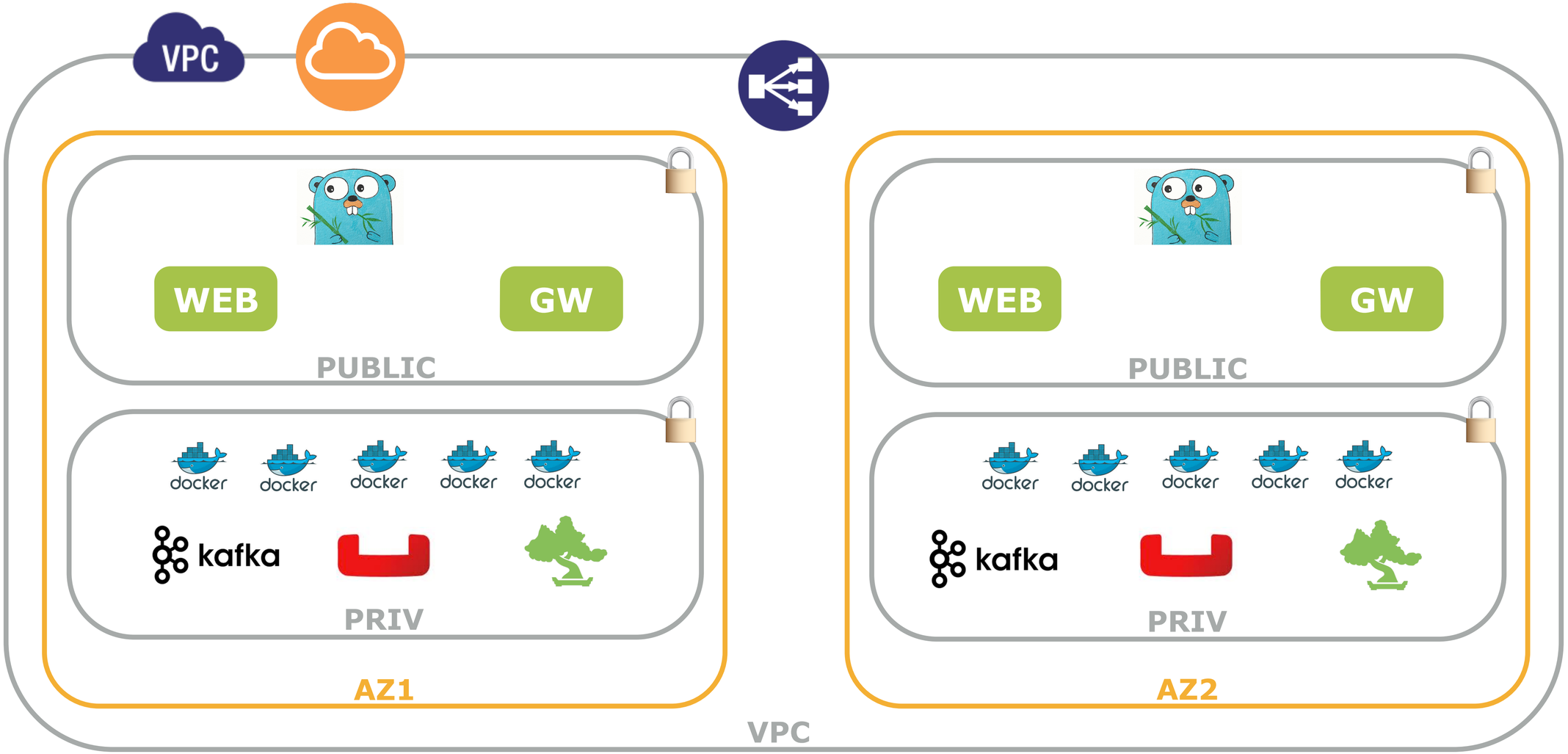
Résultats

- Création instances EC2
- Configuration DNS (Route 53)
- Configuration instances
- Ré-utilisation des rôles (all, java, zk, ...)

Technos

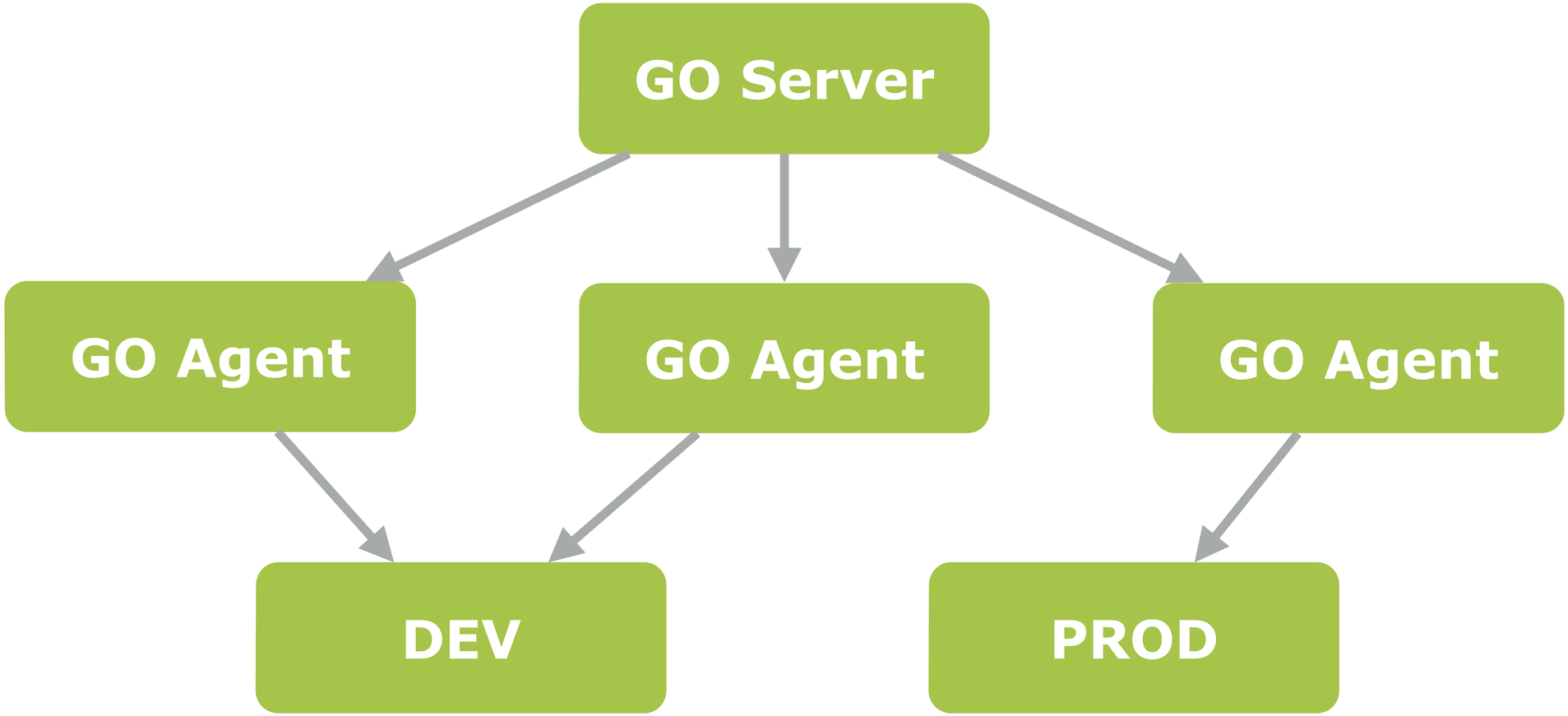

Device
akka
kafka
akka
akka
API
API
API
Front
WEB
Provider A
Provider B

VPC + CROSS AZ

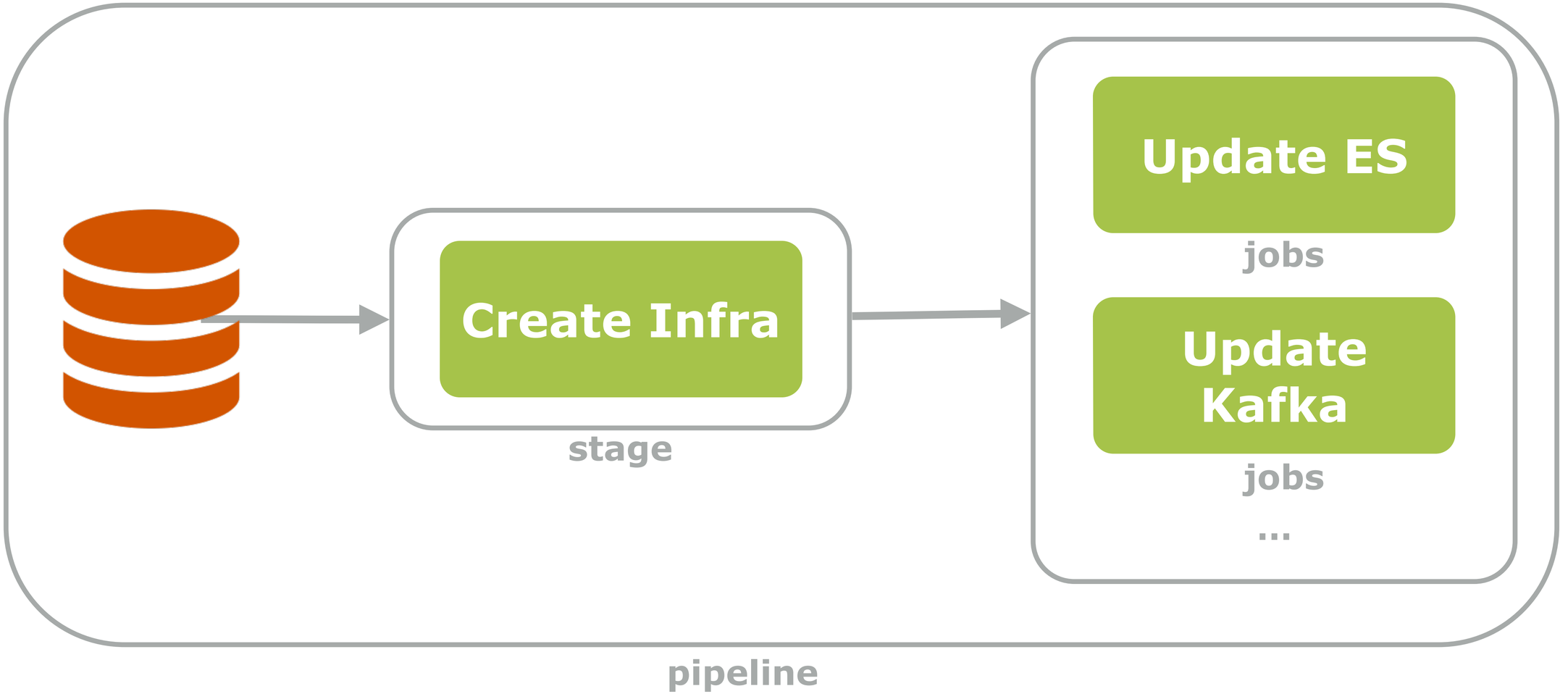

VPC
WEB
GW
PUBLIC
docker
docker
docker
docker
docker
kafka
PRIV
AZ1
WEB
GW
PUBLIC
docker
docker
docker
docker
docker
kafka
PRIV
AZ2
VPC

GO Continuous Delivery

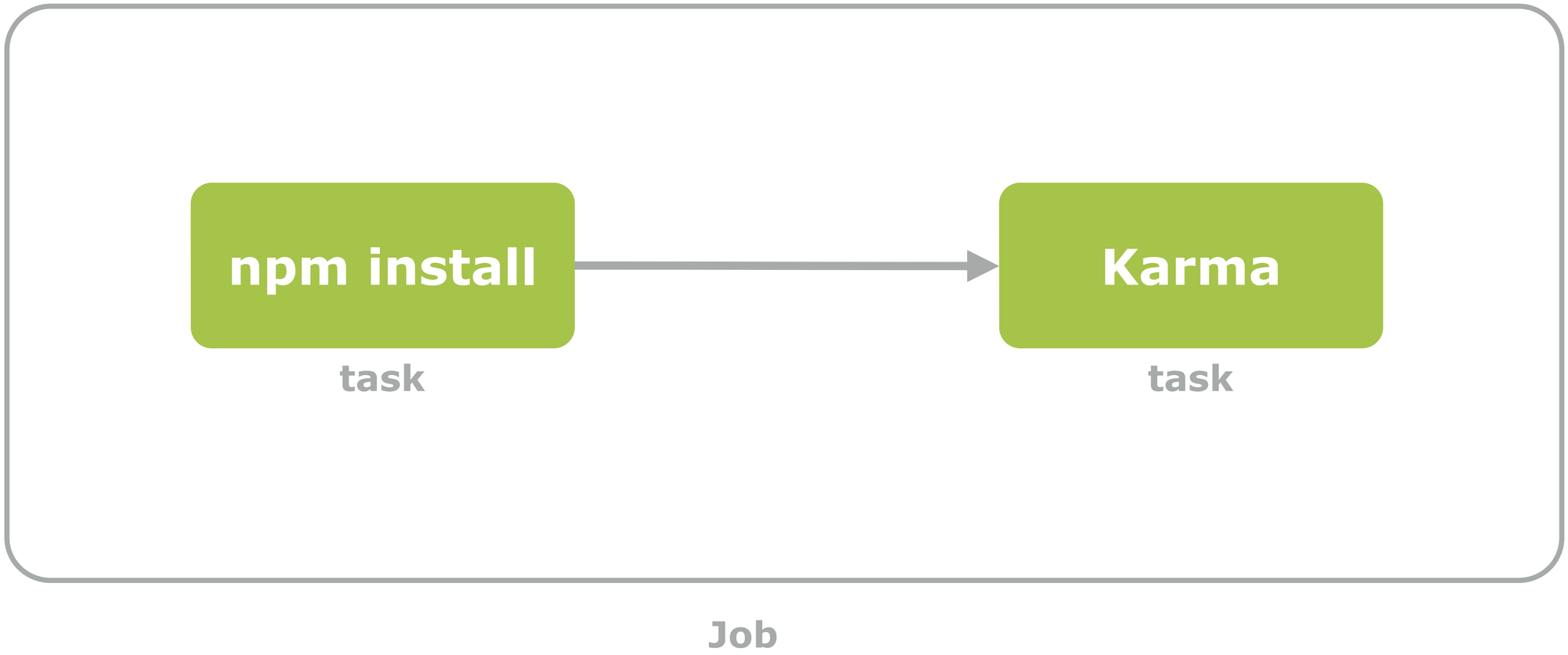
GO CD: Principes



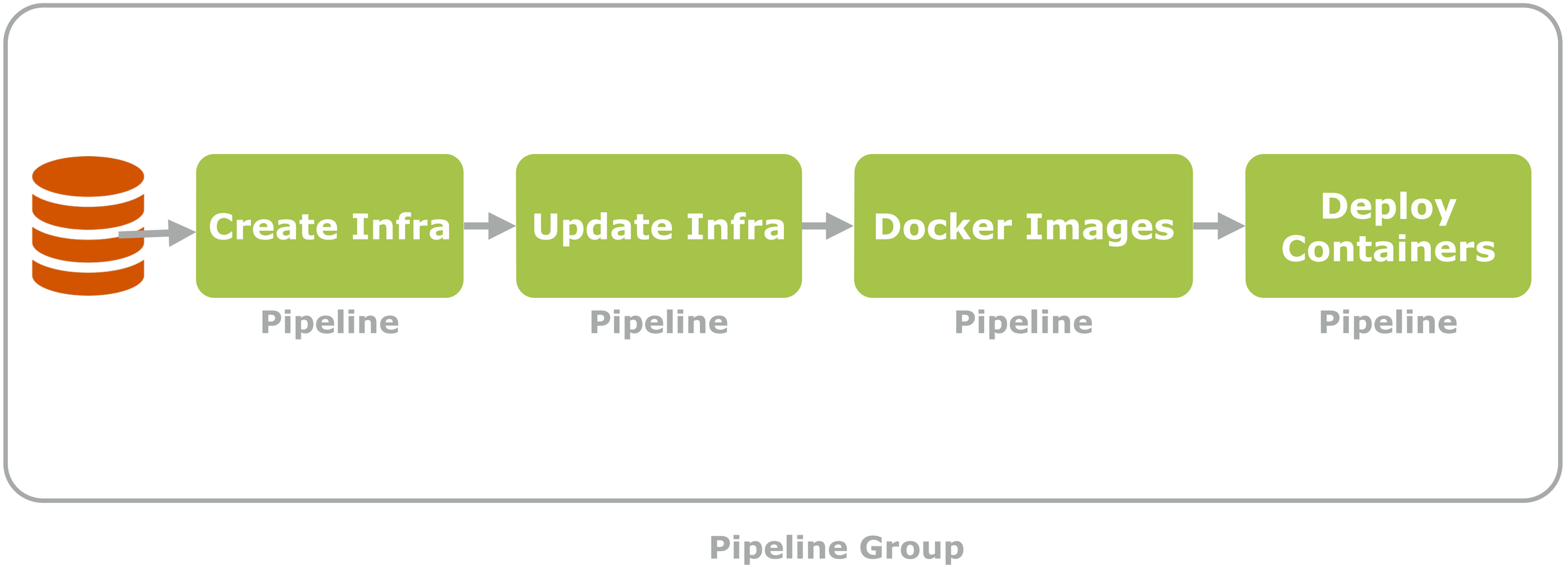
GO CD: Principes



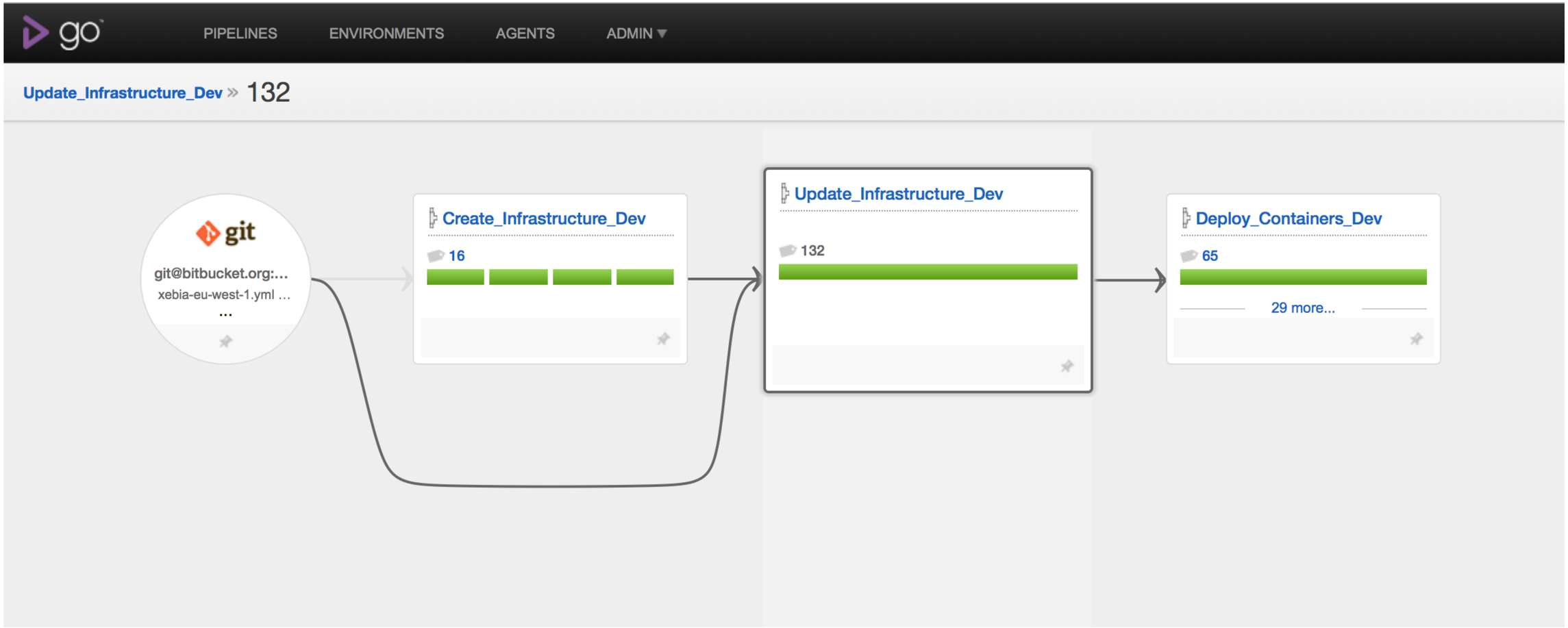
GO CD: Principes



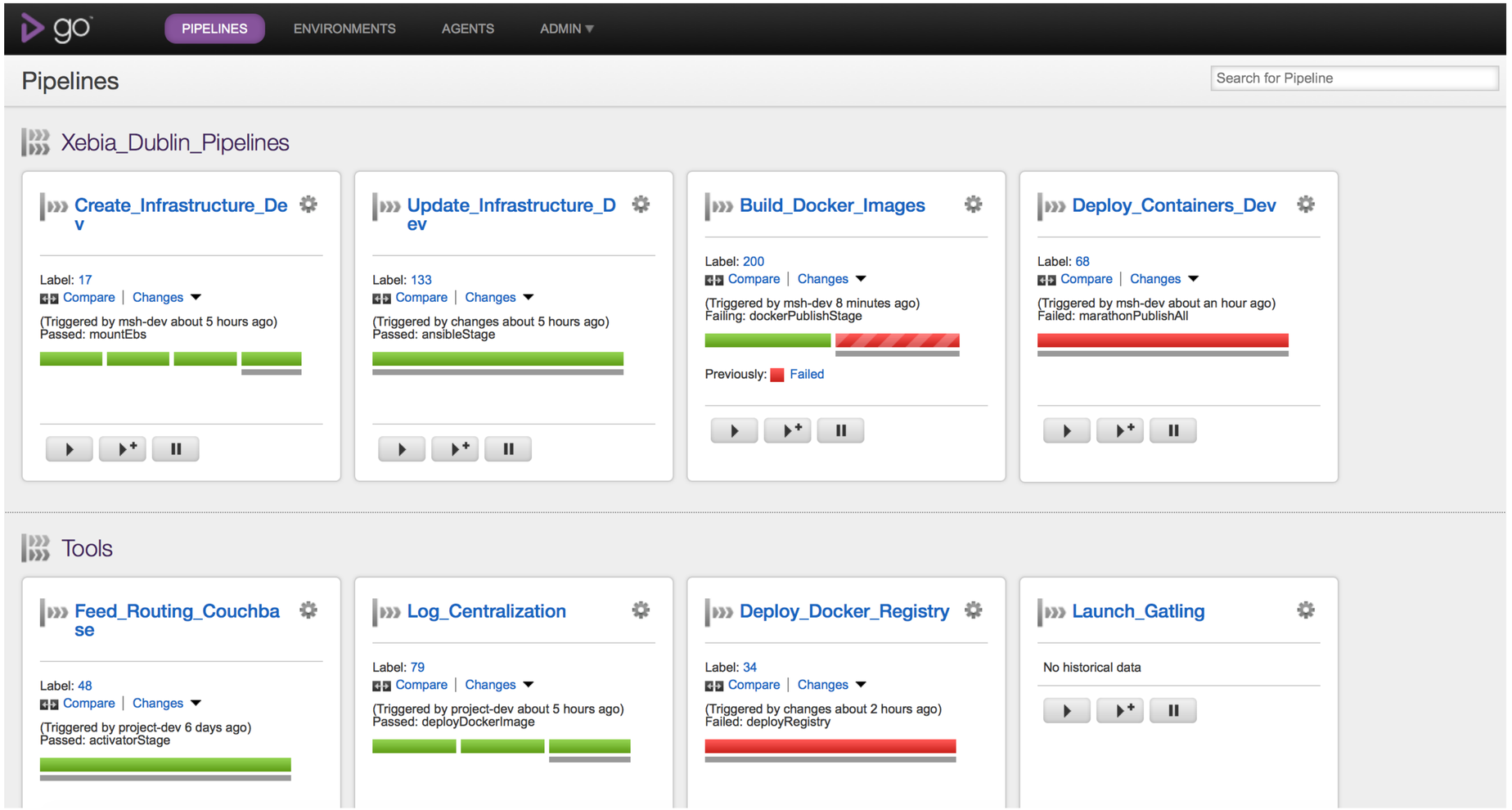
GO CD: Principes



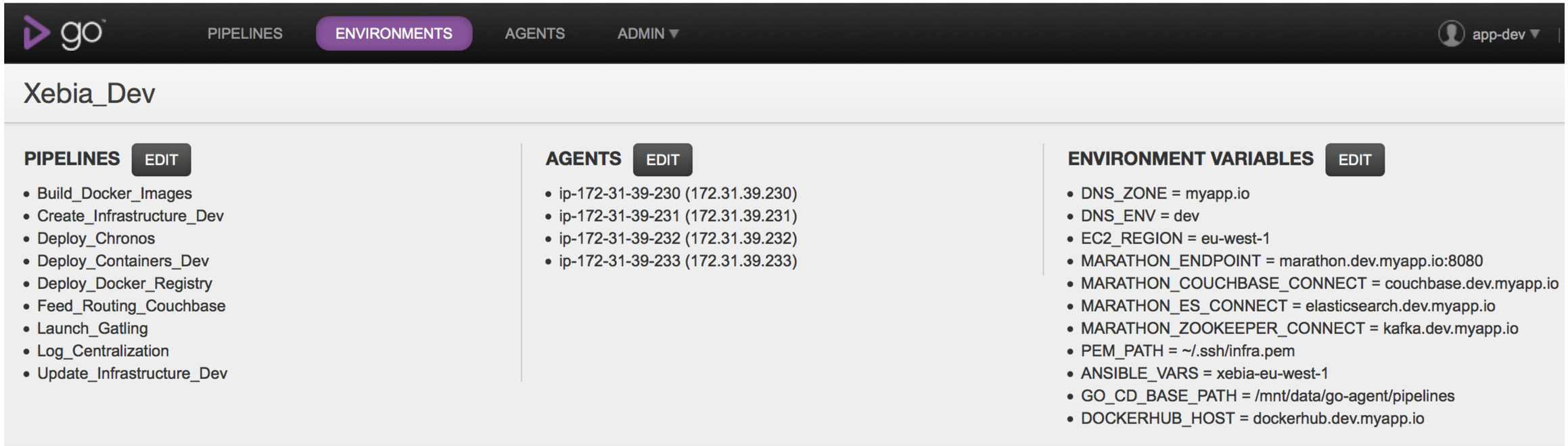
GO CD: UI



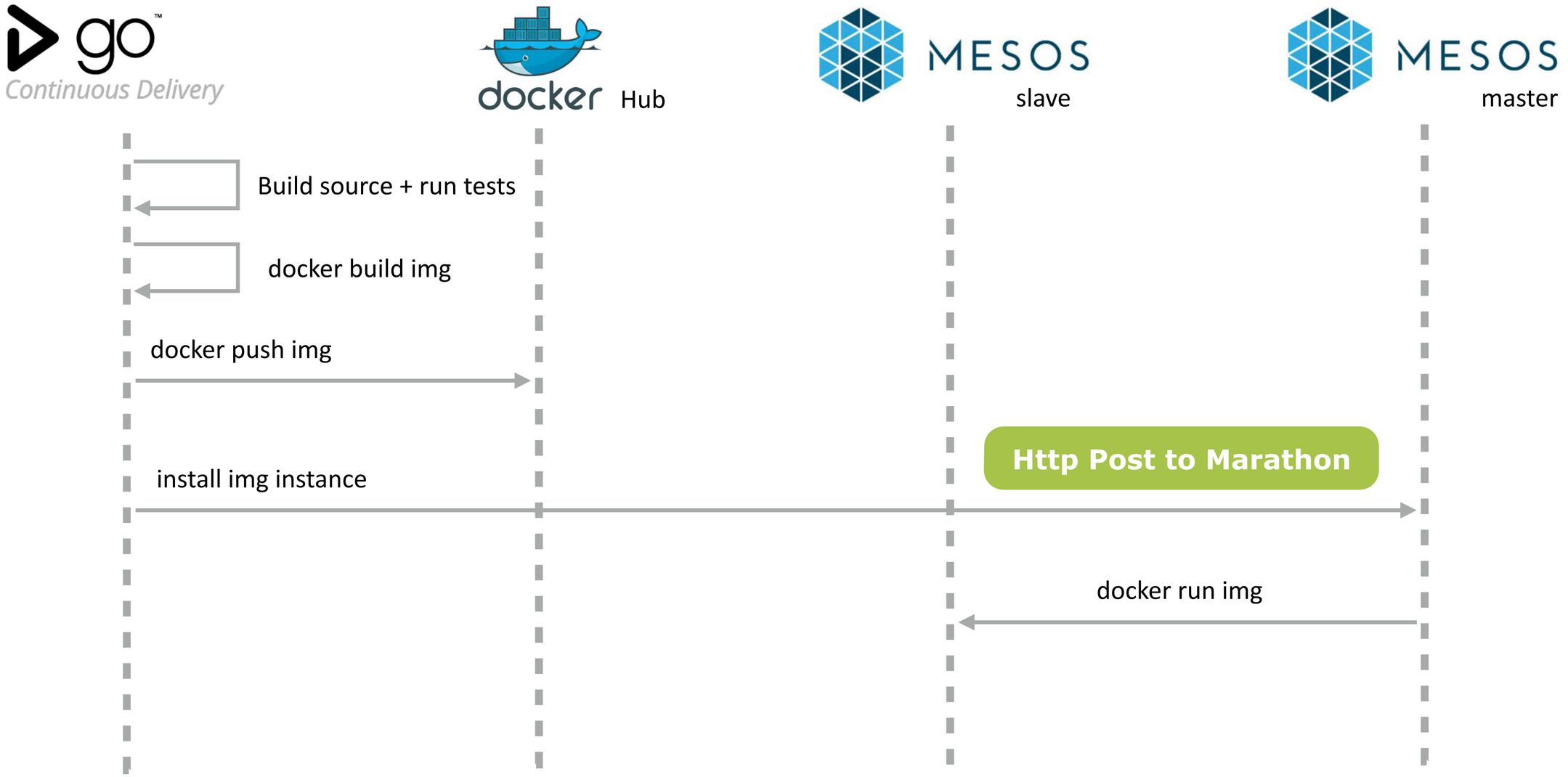
GO CD: UI


go
PIPELINES
ENVIRONMENTS
AGENTS
ADMIN
Pipelines
Search for Pipeline
Xebia_Dublin_Pipelines
Create_Infrastructure_Dev
Label: 17
Compare | Changes
(Triggered by msh-dev about 5 hours ago)
Passed: mountEbs
Update_Infrastructure_Dev
Label: 133
Compare | Changes
(Triggered by changes about 5 hours ago)
Passed: ansibleStage
Build_Docker_Images
Label: 200
Compare | Changes
(Triggered by msh-dev 8 minutes ago)
Failing: dockerPublishStage
Previously: Failed
Deploy_Containers_Dev
Label: 68
Compare | Changes
(Triggered by msh-dev about an hour ago)
Failed: marathonPublishAll
Tools
Feed_Routing_Couchbase
Label: 48
Compare | Changes
(Triggered by project-dev 6 days ago)
Passed: activatorStage
Log_Centralization
Label: 79
Compare | Changes
(Triggered by project-dev about 5 hours ago)
Passed: deployDockerImage
Deploy_Docker_Registry
Label: 34
Compare | Changes
(Triggered by changes about 2 hours ago)
Failed: deployRegistry
Launch_Gatling
No historical data

TEMPLATES

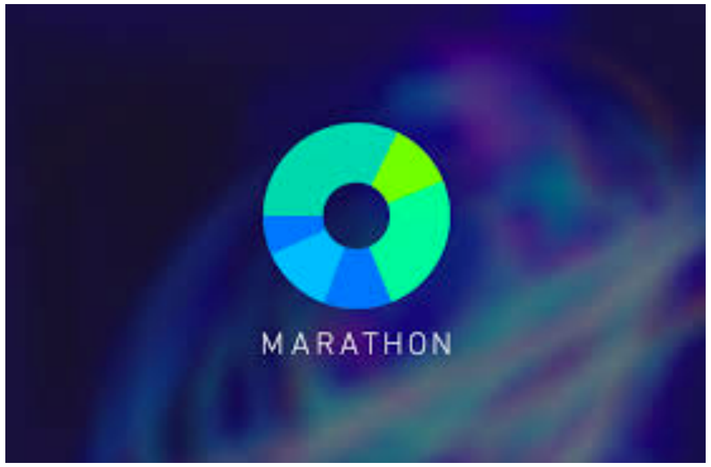

go
PIPELINES
ENVIRONMENTS
AGENTS
ADMIN
app-dev
Xebia_Dev
PIPELINES
EDIT
Build_Docker_Images
Create_Infrastructure_Dev
Deploy_Chronos
Deploy_Containers_Dev
Deploy_Docker_Registry
Feed_Routing_Couchbase
Launch_Gatling
Log_Centralization
Update_Infrastructure_Dev
AGENTS
EDIT
ip-172-31-39-230 (172.31.39.230)
ip-172-31-39-231 (172.31.39.231)
ip-172-31-39-232 (172.31.39.232)
ip-172-31-39-233 (172.31.39.233)
ENVIRONMENT VARIABLES
EDIT
DNS_ZONE = myapp.io
DNS_ENV = dev
EC2_REGION = eu-west-1
MARATHON_ENDPOINT = marathon.dev.myapp.io:8080
MARATHON_COUCHBASE_CONNECT = couchbase.dev.myapp.io
MARATHON_ES_CONNECT = elasticsearch.dev.myapp.io
MARATHON_ZOOKEEPER_CONNECT = kafka.dev.myapp.io
PEM_PATH = ~/.ssh/infra.pem
ANSIBLE_VARS = xebia-eu-west-1
GO_CD_BASE_PATH = /mnt/data/go-agent/pipelines
DOCKERHUB_HOST = dockerhub.dev.myapp.io

Déploiement continu



Http post to Marathon

Build Marathon request with dynamic parameters



SBT

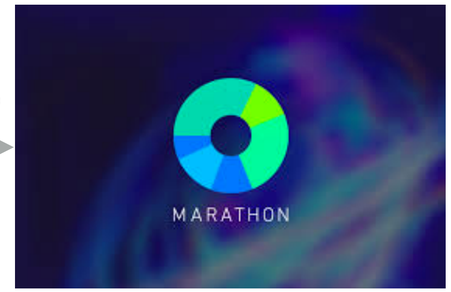
```
cpu in Marathon := 2

memory in Marathon := 512

instances in Marathon := 1

portsMapping in Marathon :=
  Seq(dynamicPort(9000, Protocol.TCP))

healthchecks in Marathon :=
  Seq(httpHealthCheck("/health", 3, 10, 10, 3))
```

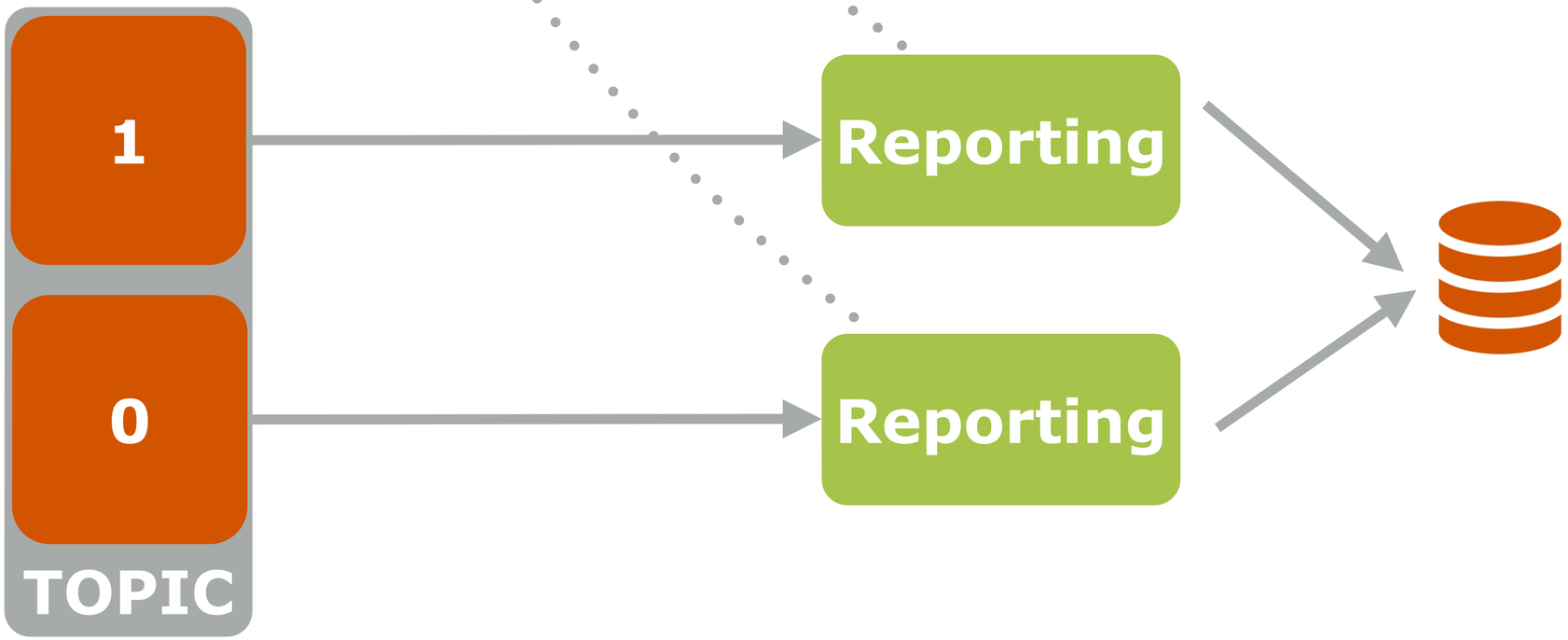
Kafka

2 partitions

sbt

deploy 2 instances





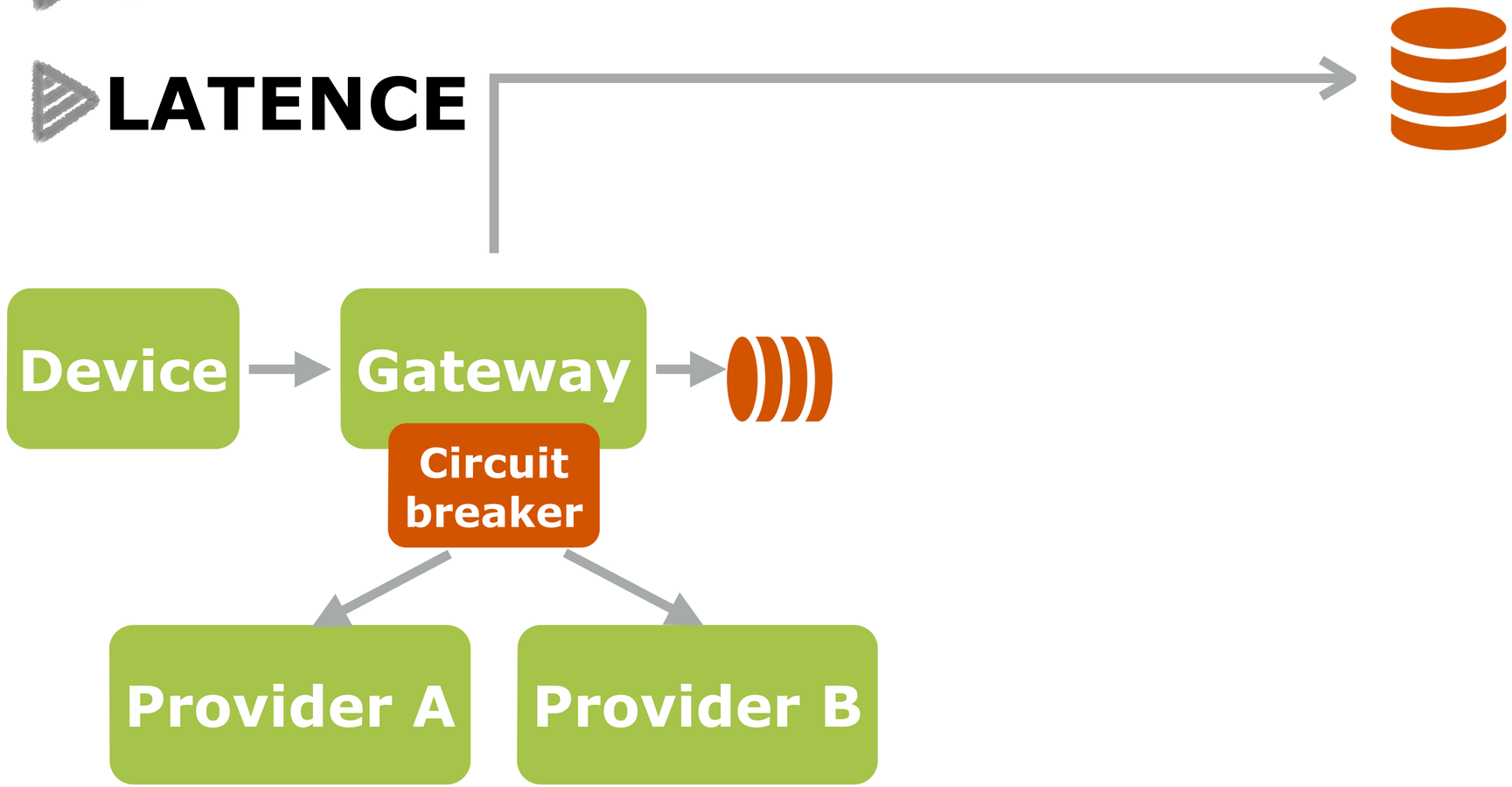
DEVOXX™ France

Agenda

- Contexte
- Comment concevoir ?
- Comment développer ?
- Comment déployer ?
- **Comment monitorer ?**

KPI

- UP TIME
- LATENCE



KPI

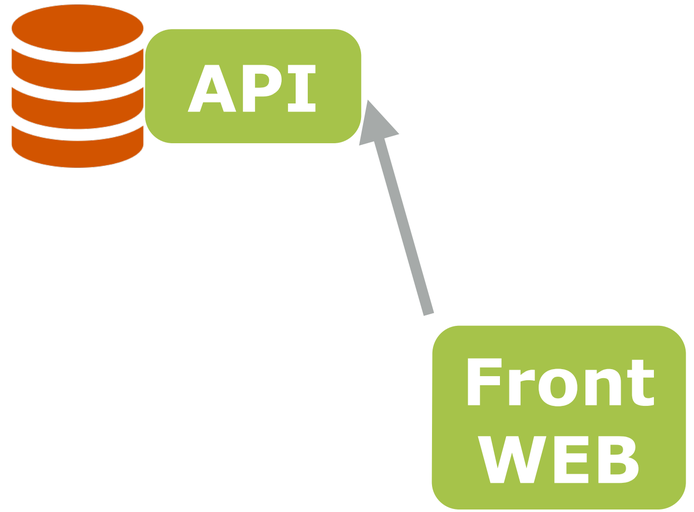
- DÉBIT

KPI

- LATENCE
- VOLUME

KPI

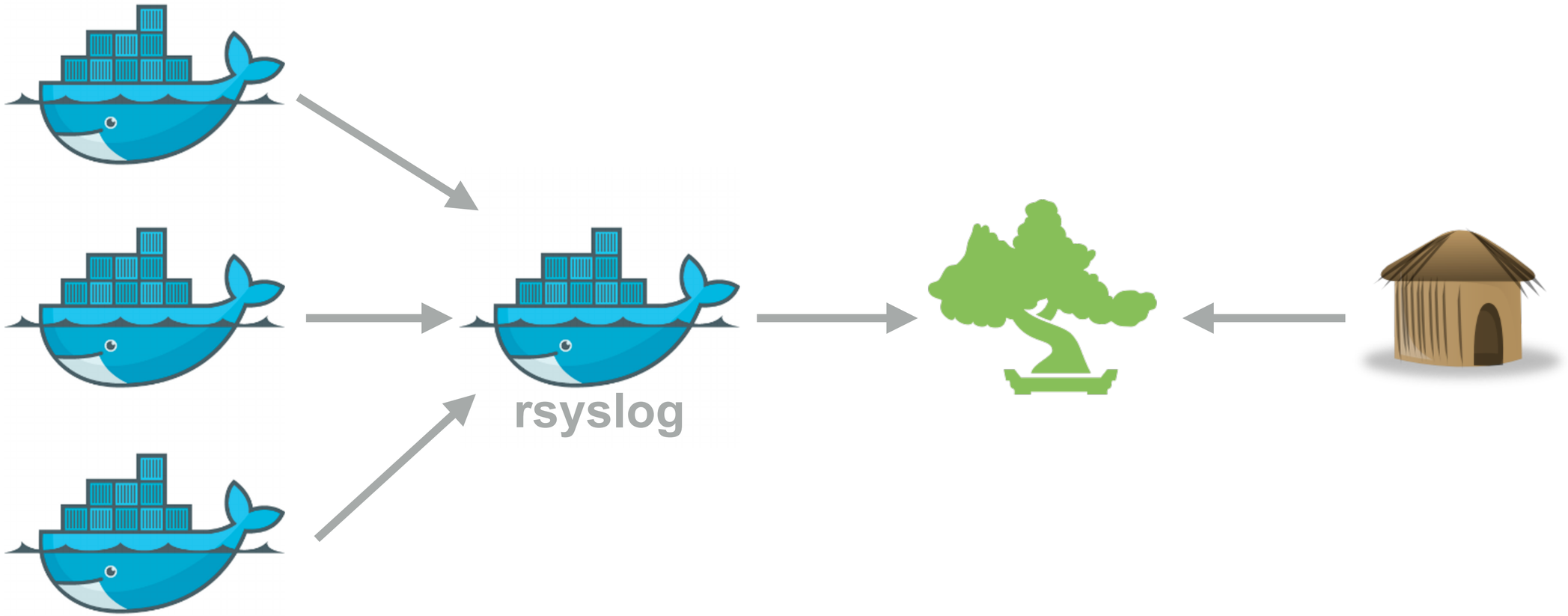
- COHERENCE



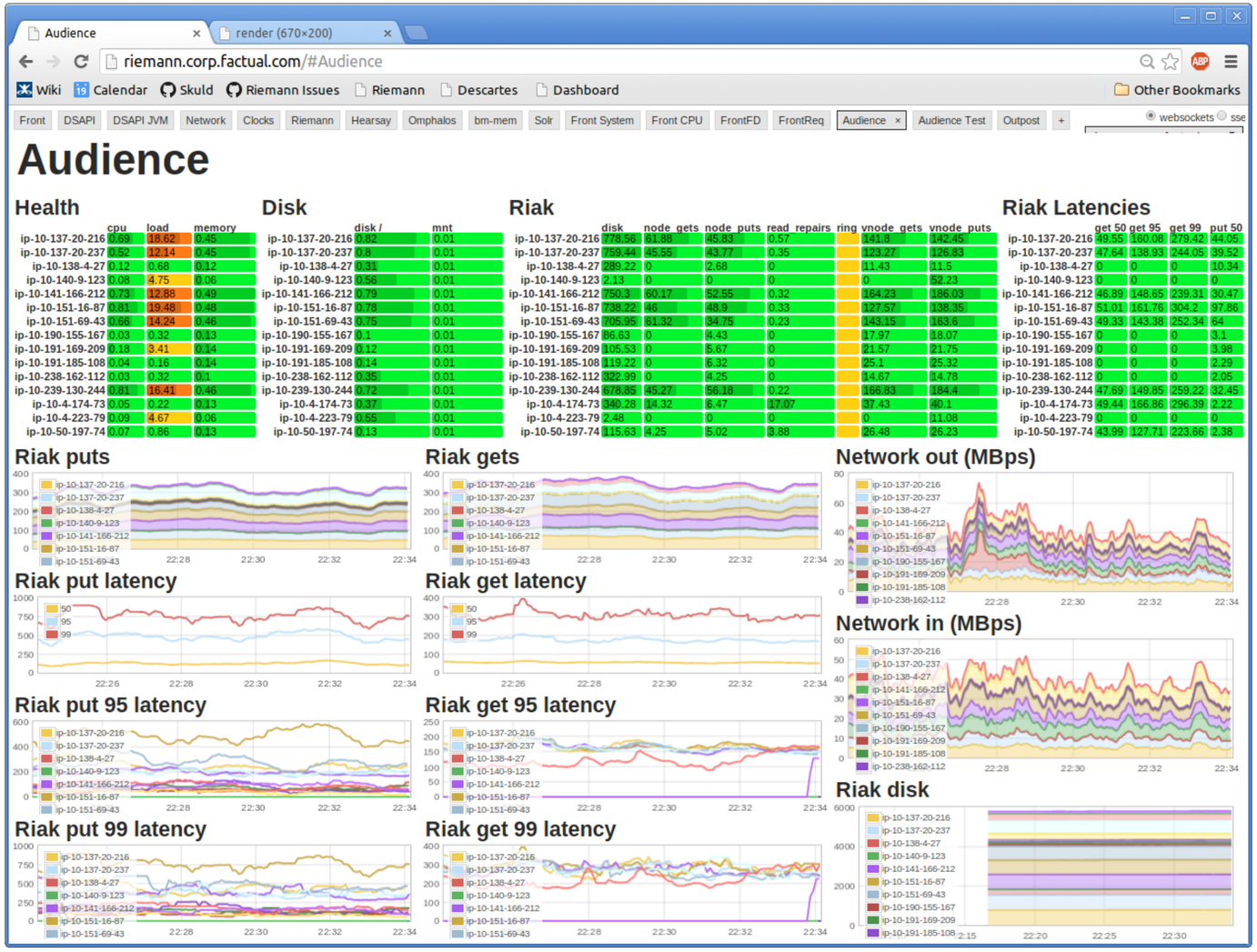
Cloud

PET vs CATTLE

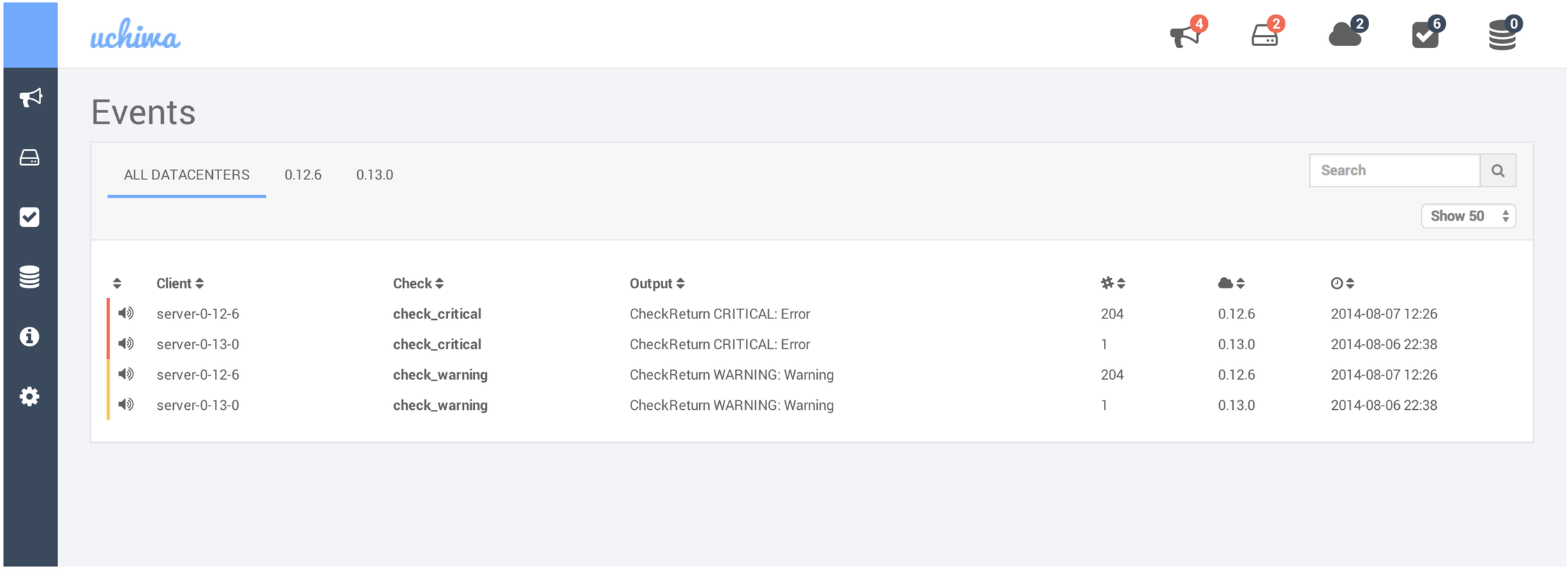
Rsyslog + ES + Kibana



Riemann ?


Audience
render (670×200)
riemann.corp.factual.com/#Audience
Wiki
Calendar
Skuld
Riemann Issues
Riemann
Descartes
Dashboard
Other Bookmarks
Front
DSAPI
DSAPI JVM
Network
Clocks
Riemann
Hearsay
Omphalos
bm-mem
Solr
Front System
Front CPU
FrontFD
FrontReq
Audience
Audience Test
Outpost
websockets
sse
Audience
Health
Disk
Riak
Riak Latencies
Riak puts
Riak gets
Network out (MBps)
Riak put latency
Riak get latency
Network in (MBps)
Riak put 95 latency
Riak get 95 latency
Riak disk
Riak put 99 latency
Riak get 99 latency

Sensu ?

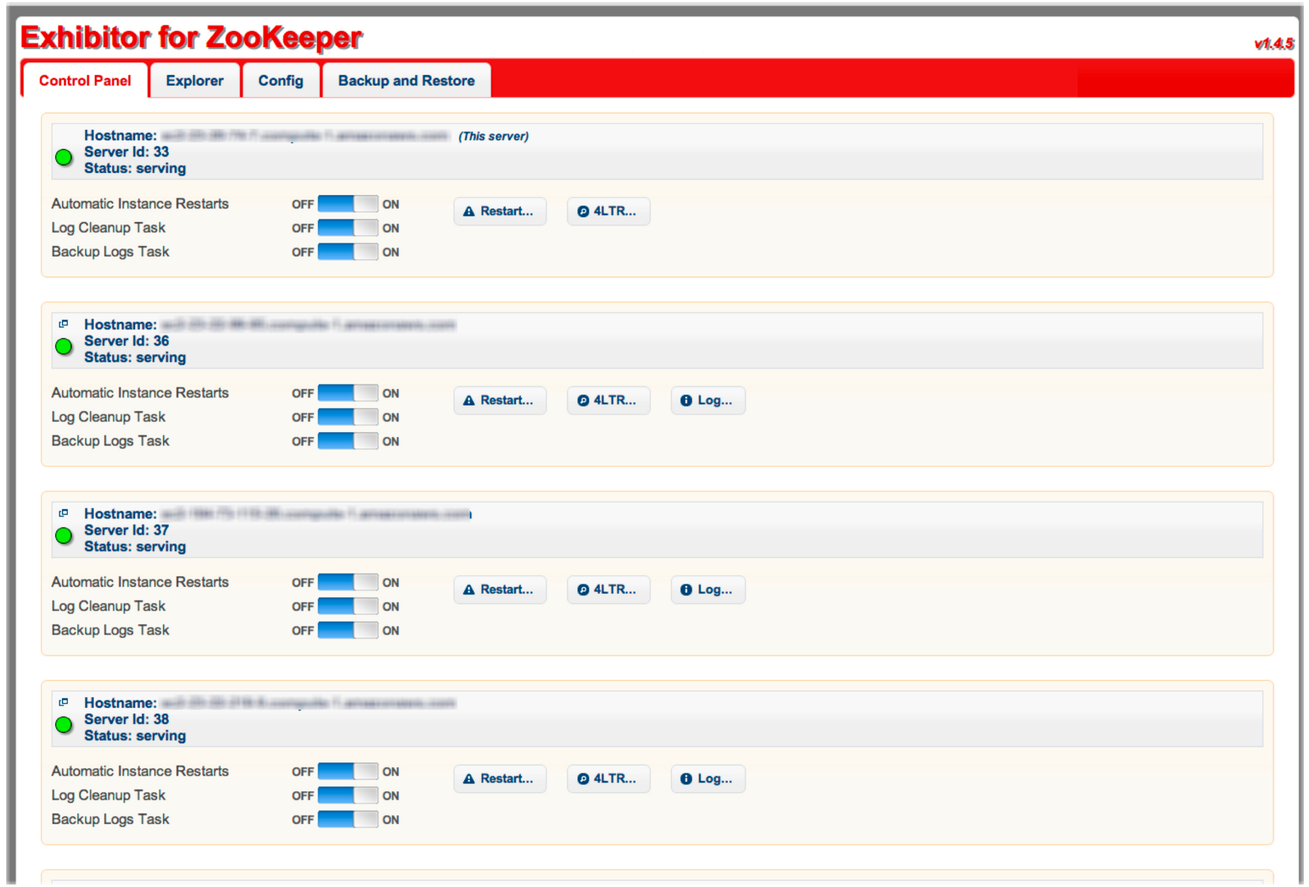

uchiwa
4
2
2
6
0
Events
ALL DATACENTERS
0.12.6
0.13.0
Search
Show 50
Client
Check
Output
server-0-12-6
check_critical
CheckReturn CRITICAL: Error
204
0.12.6
2014-08-07 12:26
server-0-13-0
check_critical
CheckReturn CRITICAL: Error
1
0.13.0
2014-08-06 22:38
server-0-12-6
check_warning
CheckReturn WARNING: Warning
204
0.12.6
2014-08-07 12:26
server-0-13-0
check_warning
CheckReturn WARNING: Warning
1
0.13.0
2014-08-06 22:38

DEVOXX France

Monitoring Zookeeper

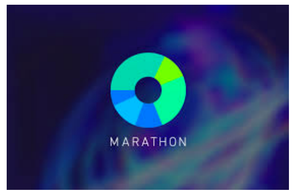
- Akka Cluster
- Kafka
- Mesos

NETFLIX

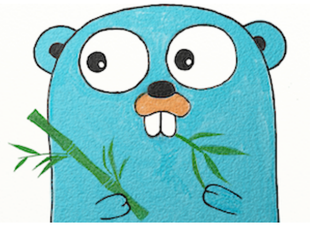


Bilan

Charge variable



Haute dispo

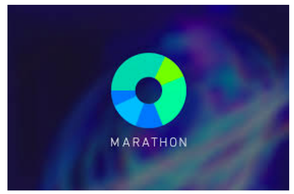


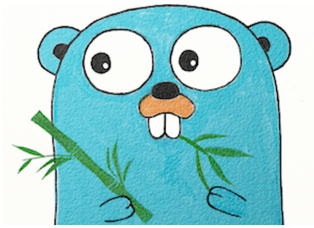
Bilan

Tolérance aux pannes

kafka

akka

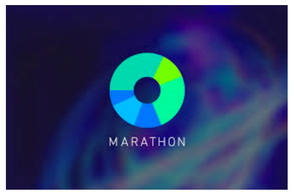




ANSIBLE

Déploiement service à chaud

akka



Bilan

A/B testing

En cours de réflexion, développement

Schemaless

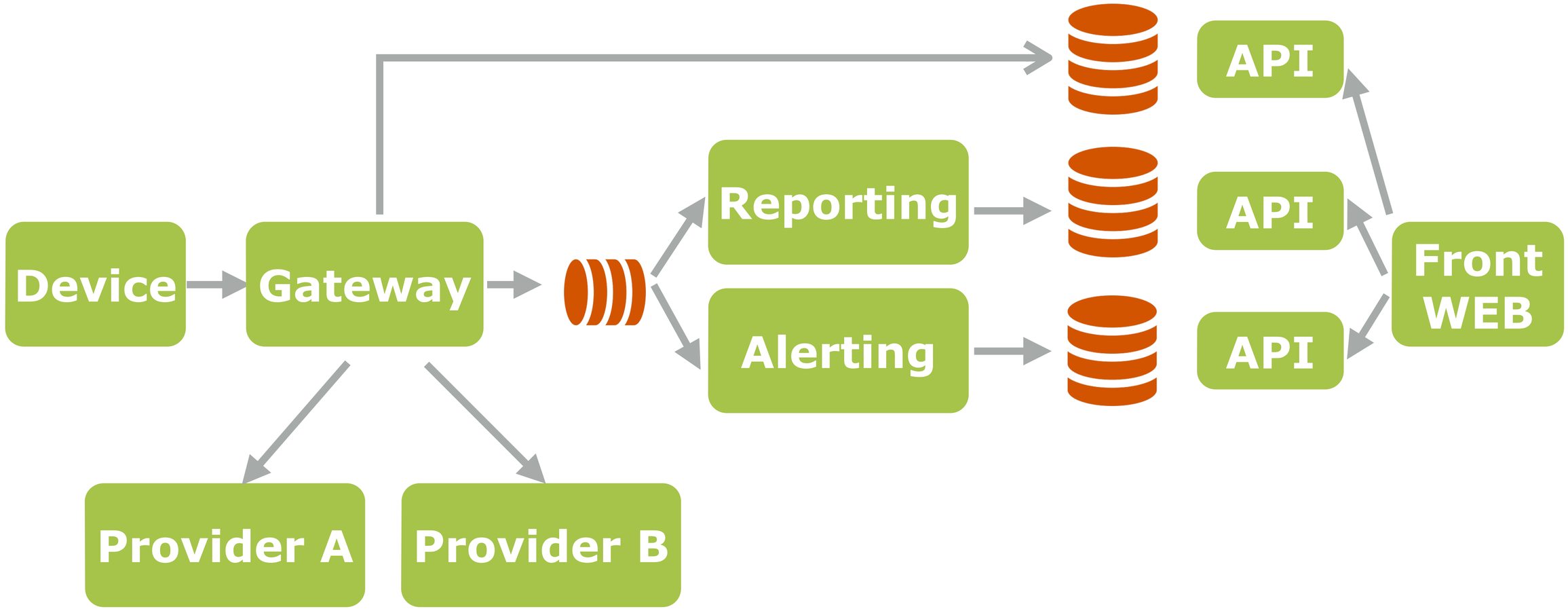
Facile à exploiter

DEVOXX™ France

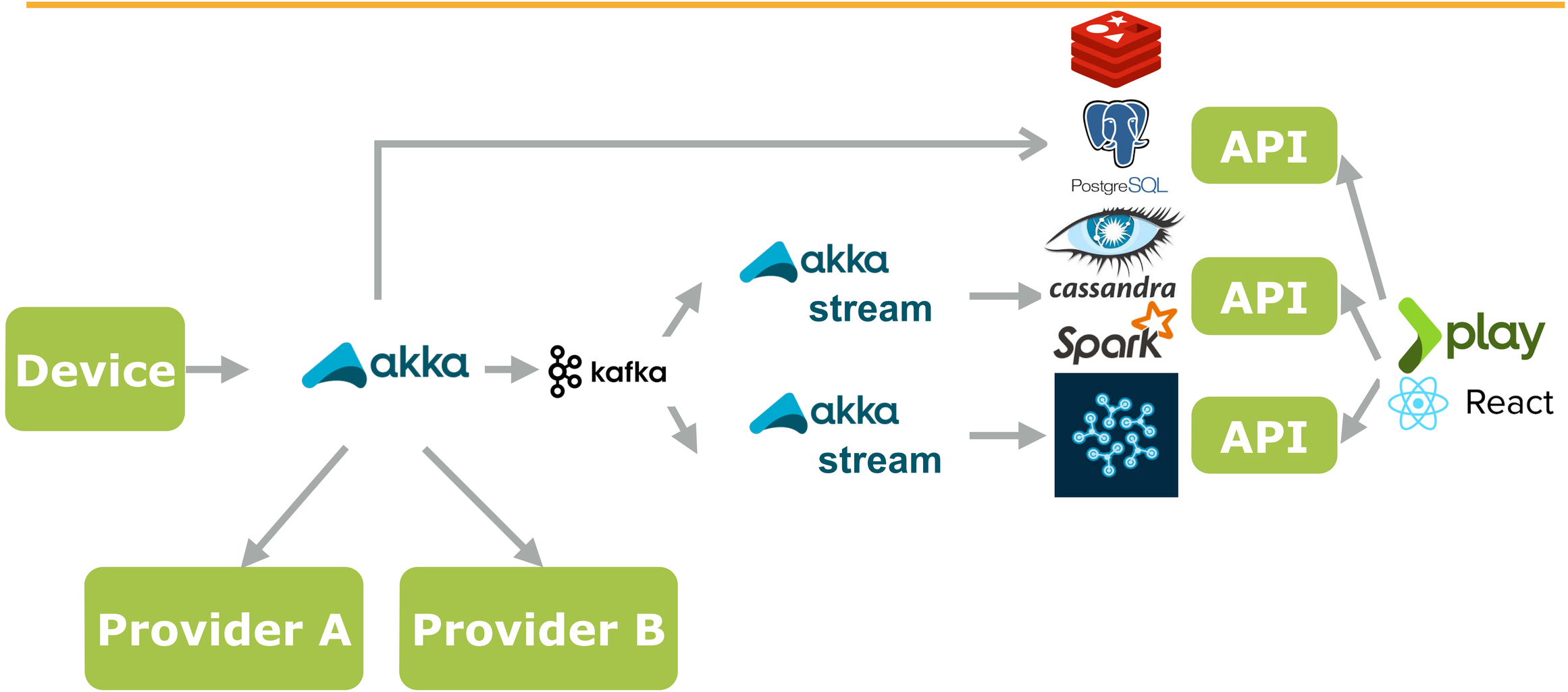
Améliorations

- **Minimum Vertical Slice**
- **Event Sourcing**

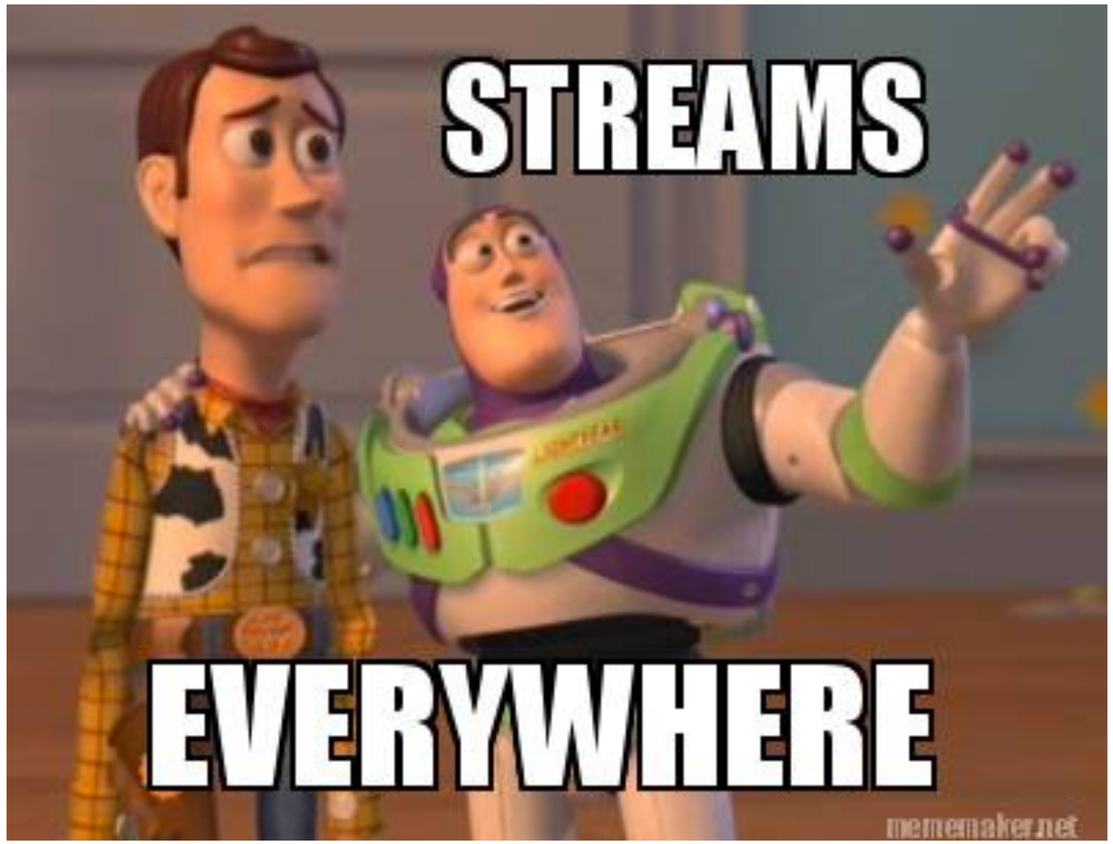
Améliorations



Améliorations



Améliorations

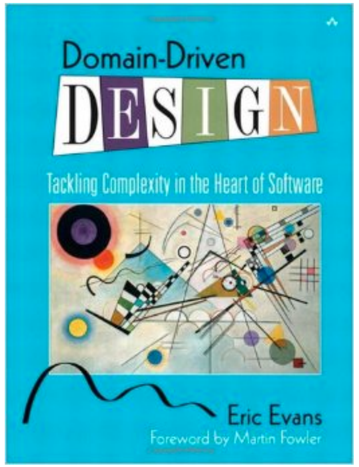


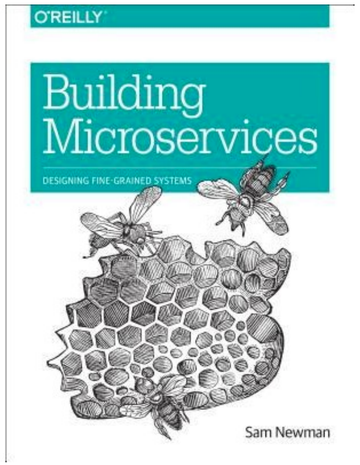
DEVOXX™ France

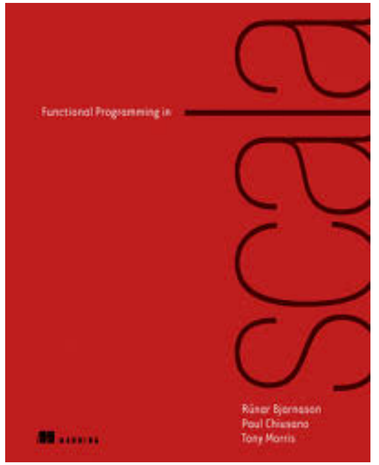
Prise de recul

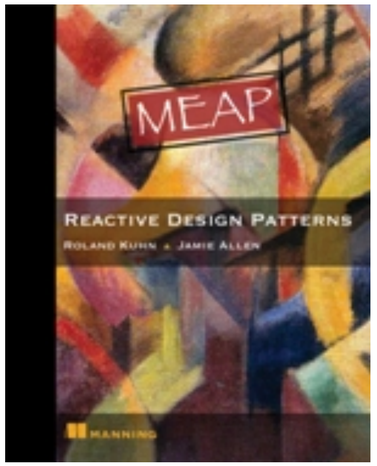
- **Déploiements**
- **Mise en place de tests**
- **Granularité des services**
- **Culture du client**

Bibliographie









Standing on the shoulders of giants

http://www.se-radio.net/2014/10/episode-213-james-lewis-on-microservices/

QUESTIONS ?