Open APIs for Open Minds

FIWARE Data Management in High Availability

Federico M. Facca (Martel Innovate)
Head of Martel Lab, FIWARE TSC Member
federico.facca@martel-innovate.com
@chicco785

FIWARE

Outline

- Basic High Availability Principles
- How to apply HA principles to FIWARE?
- On going and future activities in FIWARE

FIWARE

Basic High Availability Principles

FIWARE

What is High Availability?

High availability *is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period*

[https://en.wikipedia.org/wiki/High_availability]

FIWARE

Why do I need high availability?

Keep your customers happy

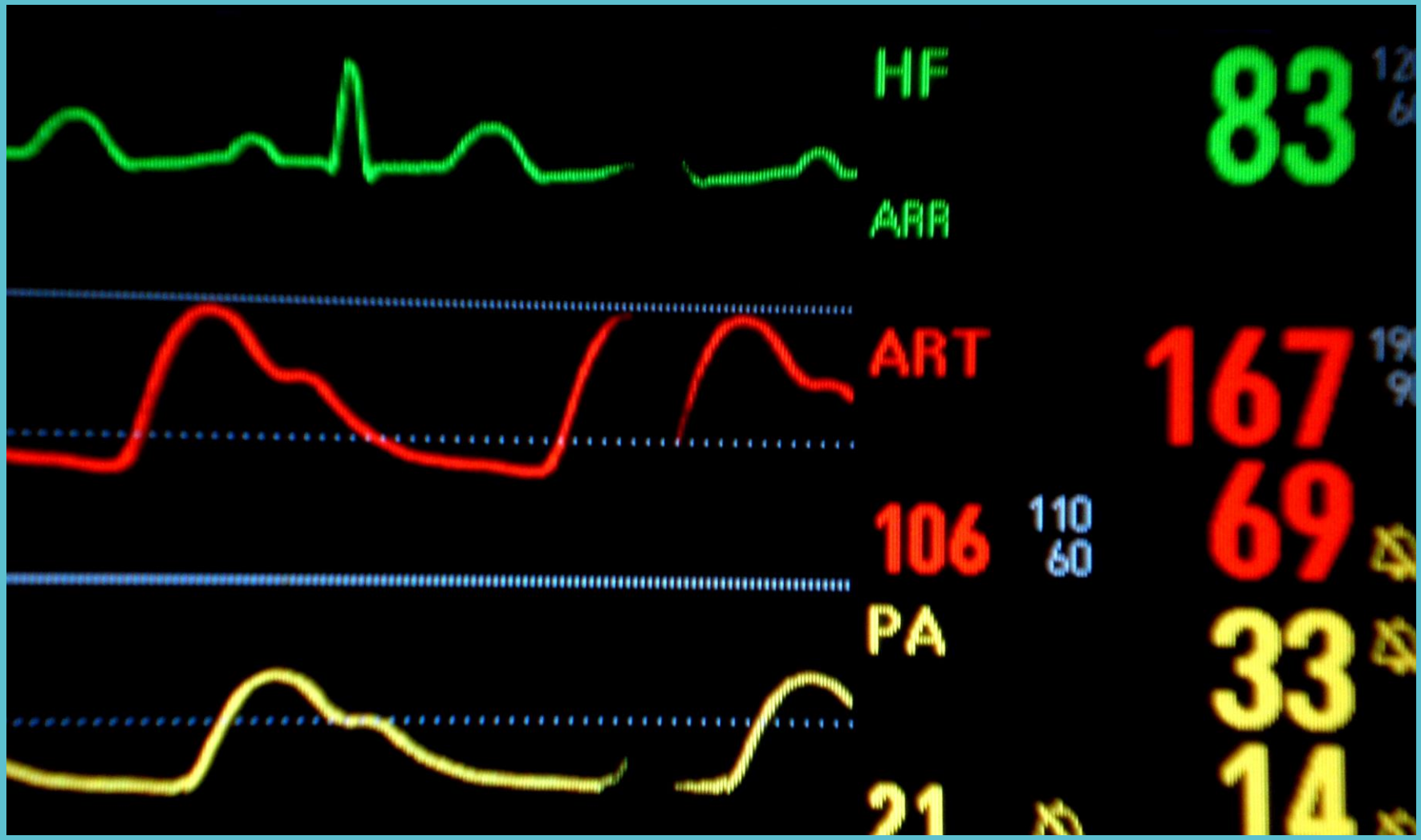
FIWARE

Eliminate single points of failure

Ensure reliable crossover

FIWARE

Detect failures as they occur



FIWARE

Cool I want to have my service in High Availability!

To define an HA architecture for your service, you need to understand how you service works

FIWARE

Stateless vs Stateful services

- Stateless services
  - The output of the service depends only on the input
  - Easy to scale and distribute
- Stateful
  - The output of the service depends on the input and on a set of information stored by the service itself
  - Not so easy to scale and distribute (maintaining a consistent state)

If your services is stateless, things are very easy

FIWARE

CAP Theorem

- The **CAP theorem** states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
  - Consistency: Every read receives the most recent write or an error
  - Availability: Every request receives a response, without guarantee that it contains the most recent version of the information
  - Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped by the network between nodes
- i.e. when you implement HA in a stateful service, you can [illegible] being [illegible] *[illegible] AP and eventually consistent*

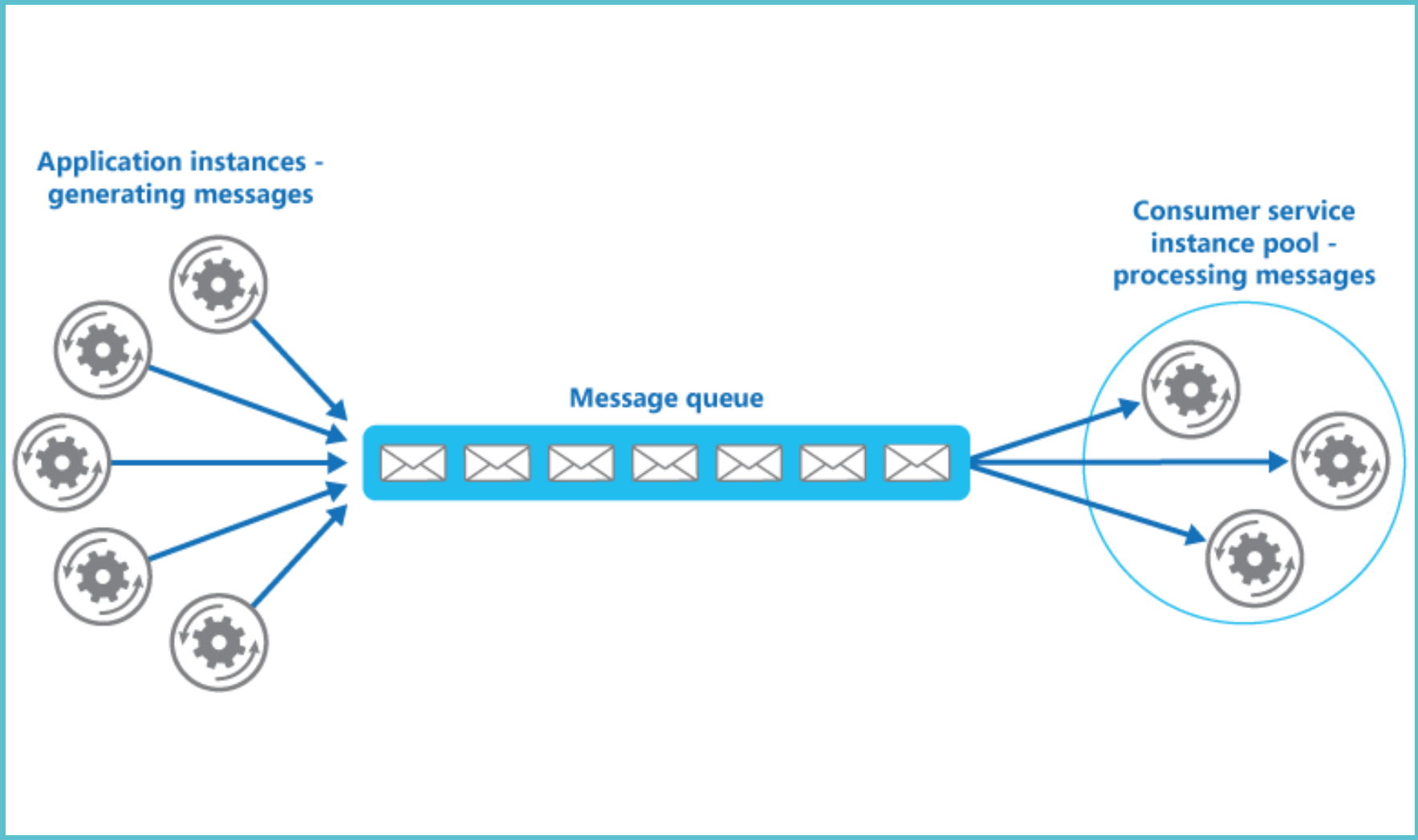
When your services is stateful, you need to decide what you are ready to give up (or eventually the specific database you use is already deciding for you)

FIWARE

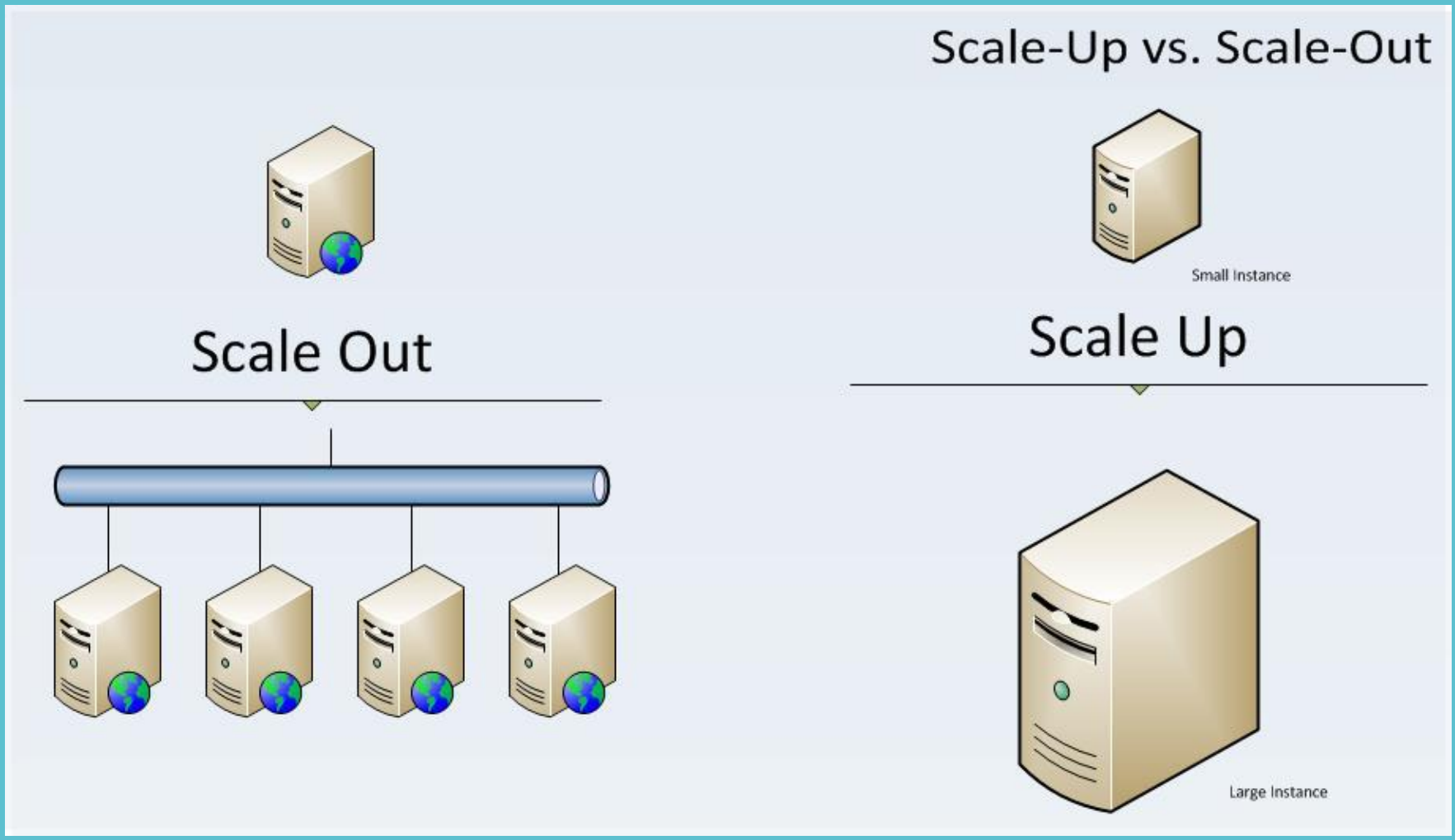
How HA relates to Cloud architectures?

- You do not need Cloud solutions to implement high availability but…
- Cloud solutions simplifies the implementation of High Available architectures
- High Available architectures are a prerequisite to implement many scalable services

Queue centric workflow patterns



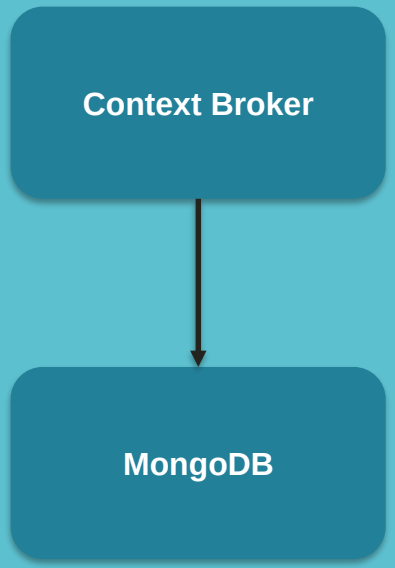
Scalability patterns



How to apply HA principles to FIWARE?

Context Broker

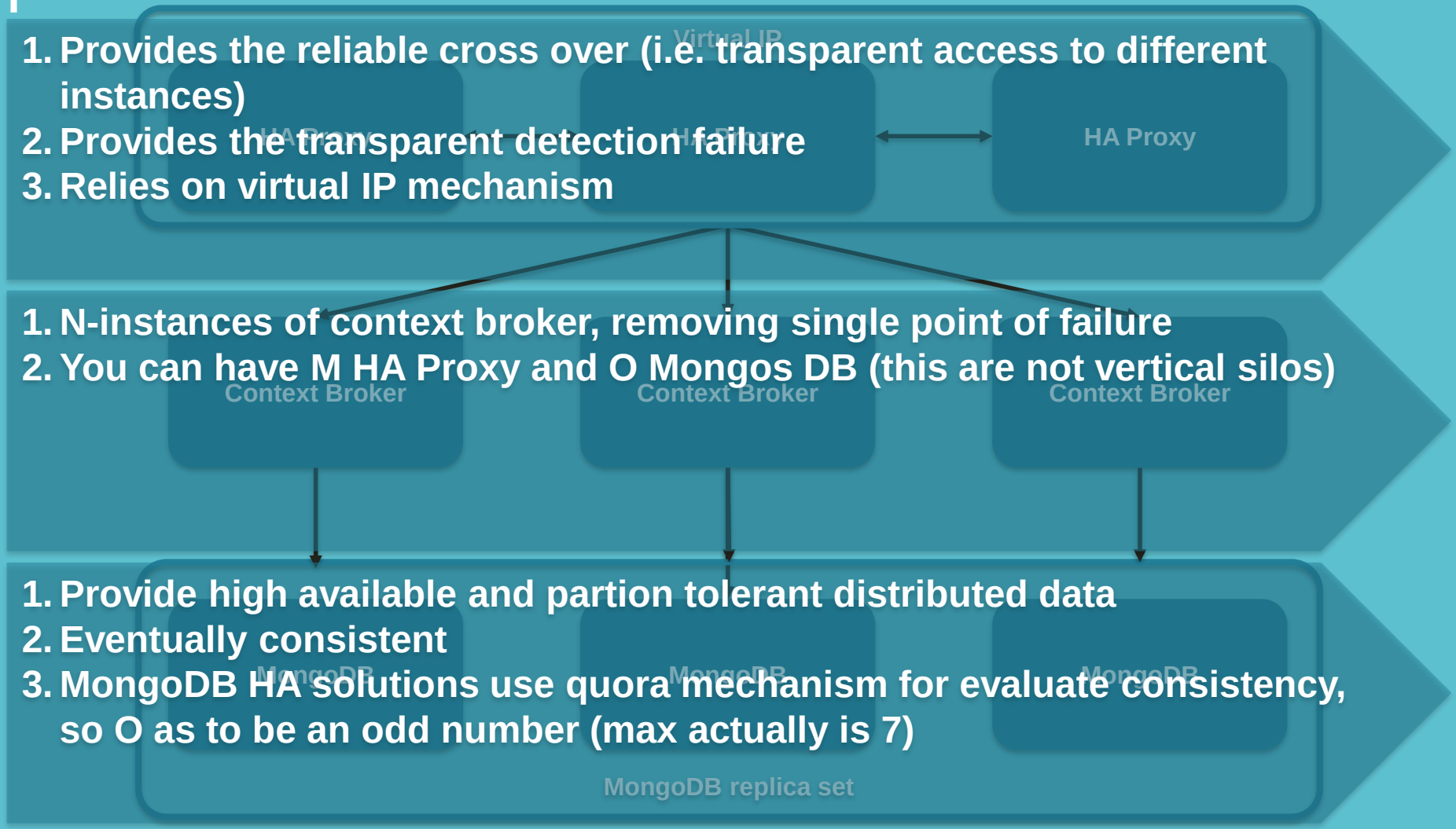
- Context Broker is perhaps the most used GE ☺
- It includes to components:
  - The API
  - The Backend
- The API is HTTP based
- The Backend in based on MongoDB



- How to make it high available?
  - An easy crossover mechanism for HTTP APIs are Load Balancers
  - MongoDB has is proprietary HA mechanism (replica set)

FIWARE

Context Broker



Example configuration

- Load Balancer 1: *lb1.example.com*, IP address: *192.168.0.100*
- Load Balancer 2: *lb2.example.com*, IP address: *192.168.0.101*
- Context Broker 1: *ctx1.example.com*, IP address: *192.168.0.102*
- Context Broker 2: *ctx2.example.com*, IP address: *192.168.0.103*
- Mongo DB 1: *mdb1.example.com*, IP address: *192.168.0.104*
- Mongo DB 2: *mdb2.example.com*, IP address: *192.168.0.105*
- Mongo DB 3: *mdb3.example.com*, IP address: *192.168.0.106*

- *Shared IP=192.168.0.99*

HA Proxy Installation (LB1/LB2)

- Install HA Proxy (ubuntu)
  - sudo apt-get install haproxy
- Configure HA Proxy to start at boot time
  - sudo nano /etc/default/haproxy
  - change the value of **ENABLED** to "**1**"
- Configure HA Proxy
  - sudo nano /etc/haproxy/haproxy.cfg

```
...

defaults
    log     global
    mode    tcp
    option  tcplog

...

frontend www
    bind
load_balancer_anchor_IP:1026
    default_backend ctx_pool


backend ctx_pool
    balance roundrobin
    mode tcp
    server ctx1 ctx1_private_IP:1026
check
    server ctx2 ctx2_private_IP:1026
check
```

Install keepalived active/passive (LB1/LB2)

- Install keepalived (ubuntu)
  - sudo apt-get install keepalived
- Ensure HA Proxy will be able to bind to non local addresses
  - sudo nano /etc/sysctl.conf
  - change the value of **net.ipv4.ip_nonlocal_bind** to "**1**"
  - sudo sysctl -p
- Configure keepalived
  - sudo nano /etc/keepalived/keepalived.conf
  - For LB2, change **state** to **BACKUP** and **priority** to **100**

```
global_defs {
   …
}

# Check if haproxy is still working

vrrp_script chk_haproxy {
    script "killall -0 haproxy"
    interval 2
    weight 2
}

# Configuation for the virtual
Interface

vrrp_instance VI_1 {
    interface eth0
    state MASTER
    priority 101
    virtual_router_id 51
    virtual_ipaddress {
        192.168.0.99
    }
    track_script {
      chk_haproxy
    }
}
```

Active/Passive is not the best solution… you can also run an Active/Active load balancer. That is a bit more complex though.
Keepalived can be replaced with more complex monitor and management solutions
like corosynch/pacemaker combination

FIWARE

Install MongoDB Replica Set (MDB1, MDB2, MDB3)

- Install MongoDB (ubuntu xenial)
  - sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
  - echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
  - sudo apt-get update
  - sudo apt-get install -y mongodb-org
- Start mongodb on each node
  - mongod --replSet ”orion_rs"
- Access a mongodb instance and configure the replicaset
  - mongo
  - In mongo console type:
    - rs.initiate()
    - rs.add(" *mdb2.example.com* ")
    - rs.add(" *mdb3.example.com* ")
    - rs.conf()
    - rs.status()

Install ContextBroker (CTX1, CTX2)

- Install Context Broker (centos/rh)
  - Create a file fiware.repo in /etc/yum.repos.d/ with the following lines:
    - [fiware]
    - name=Fiware Repository
    - baseurl=http://repositories.lab.fiware.org/repo/rpm/$releasever
    - gpgcheck=0
    - enabled=1
  - yum install contextBroker
- Configure the Context Broker
  - sudo nano /etc/sysconfig/contextBroker
- Start the Context Broker
  - /etc/init.d/contextBroker start

```
…
BROKER_DATABASE_HOST=mdb1.example.com,mdb2.example.com,mdb3.example.com
BROKER_DATABASE_NAME=orion

# Replica set configuration. Note that if you set this parameter, the BROKER_DATBASE_HOST is interpreted as the list of host (or host:port) separated by commas to use as## replica set seed list (single element lists are also allowed). If BROKER_DATABASE_RPL_SET parameter is unset, Orion CB assumes that the BROKER_DATABASE_HOST is an stand-alone mongod instance

BROKER_DATABASE_RPLSET=orion_rs
```


FIWARE

You made it!

Additional considerations

- Scale up
  - Scaling up context broker processing capacity requires only to add a new entry in the HA Proxy and deploy a new instance of context broker
- Hardware failures
  - If all your service instances (context broker, ha proxy, mongodb) run on the same physical server you achieved only HA within respect software failures
- Shared configuration
  - For many of the services large part of the configuration is shared, plans for easy ways to keep it in synch (e.g. NFS, github)

What about other GEs?

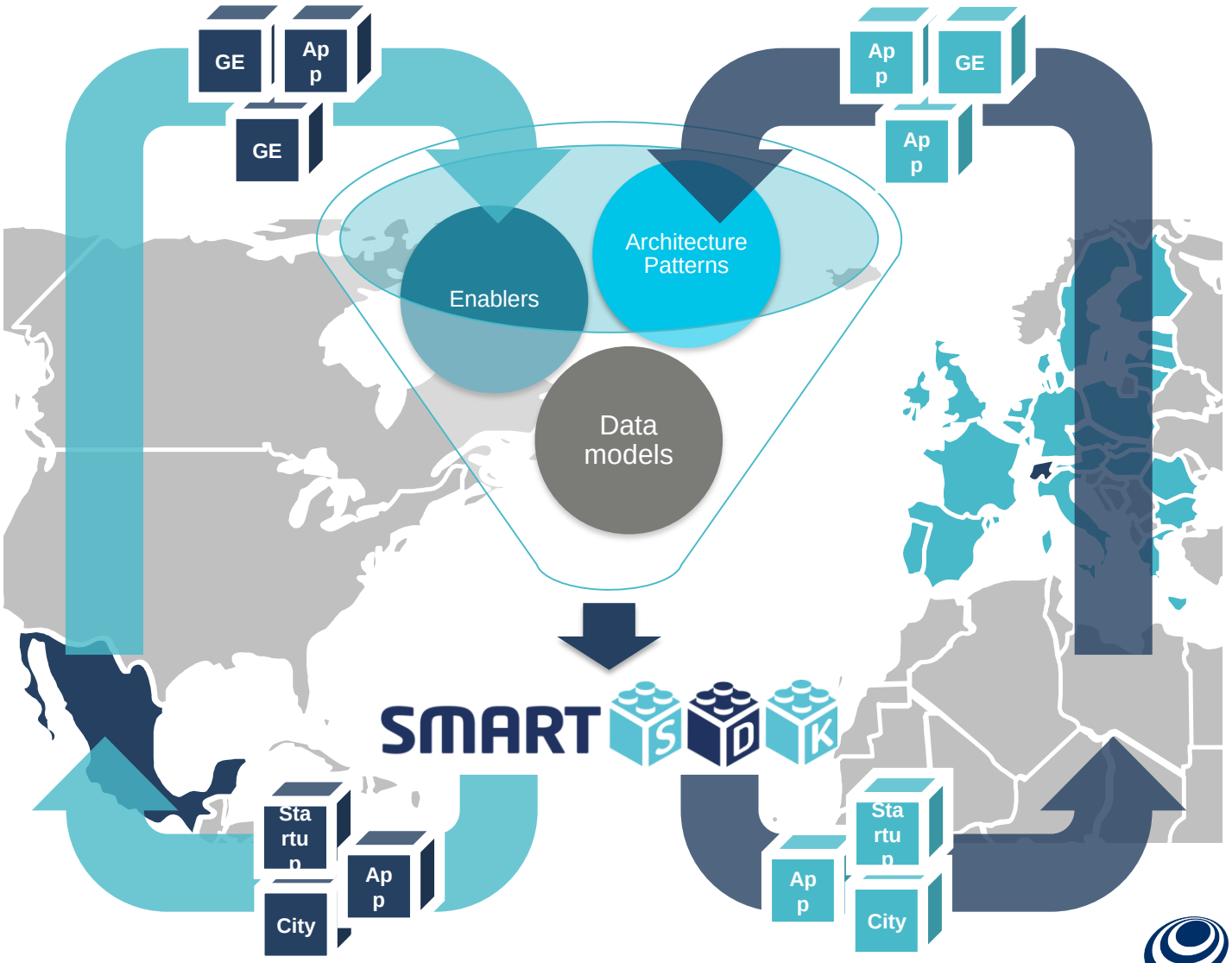
- STH and IoT Agent have similar architecture to the Context Broker
  - You can adopt a similar strategy
- Cygnus is based on Apache FLUME
  - It can be configured HA with Active/Passive modality using a load balancer and at least 2 Cygnus agents
- Some are HA by design
  - COSMOS is based on Hadoop and Hadoop is basically an HA cluster

On going and future activities in FIWARE

Did it look complex?

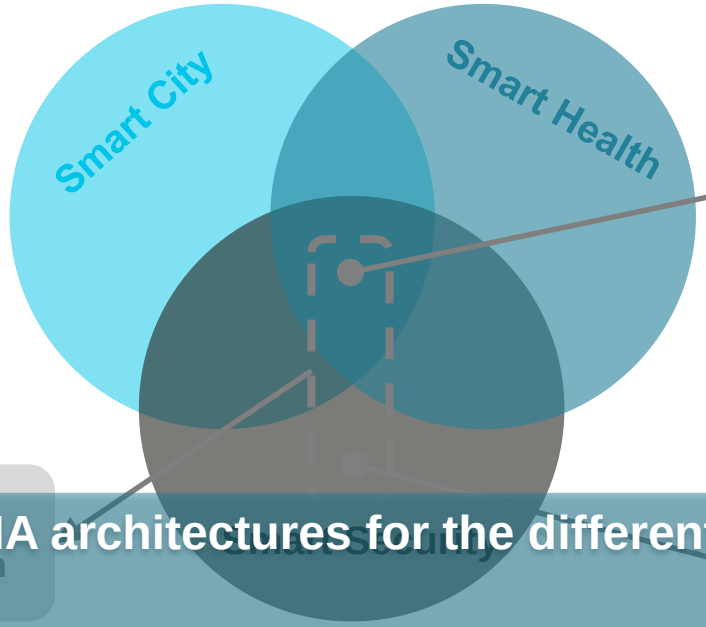
FIWARE

SMART SDK



FIWARE

SMART SDK



SMART SDK

- Common architecture patterns: e.g. scalability pattern
- Common generic enablers: e.g. orion context-broker
- Common data models: e.g. geo-location

- Specific architecture patterns: e.g. secured data access pattern
- Specific and customised generic enablers: e.g. security risk detection filters for kurento media server
- Specific data models: e.g. security's

1. Analyse HA architectures for the different Data and IoT Management enablers
2. Creating Docker compose recipes to allow easy deployment of HA enablers
3. Making them available in FIWARE Lab to experimenters

FIWARE

Do you have questions?
Do you want to contribute?

Contact Us

www.martel-innovate.com

Federico M. Facca
Head of Martel Lab
federico.facca@martel-innovate.com
Dorfstrasse 73 – 3073
Gümligen (Switzerland)
004178 807 58 38

FIWARE

Thank you!

http://fiware.org
Follow @FIWARE on Twitter