



HAZELCAST

Fabriquer une application scalable avec HAZELCAST & Docker

PLAN

Ce que nous allons voir...



- Histoire & Présentation
- Cache distribué & stratégies
- Structures de données
- Fonctions sympas
- Une approche avec Docker
- Une démonstration

Pourquoi Hazelcast ?

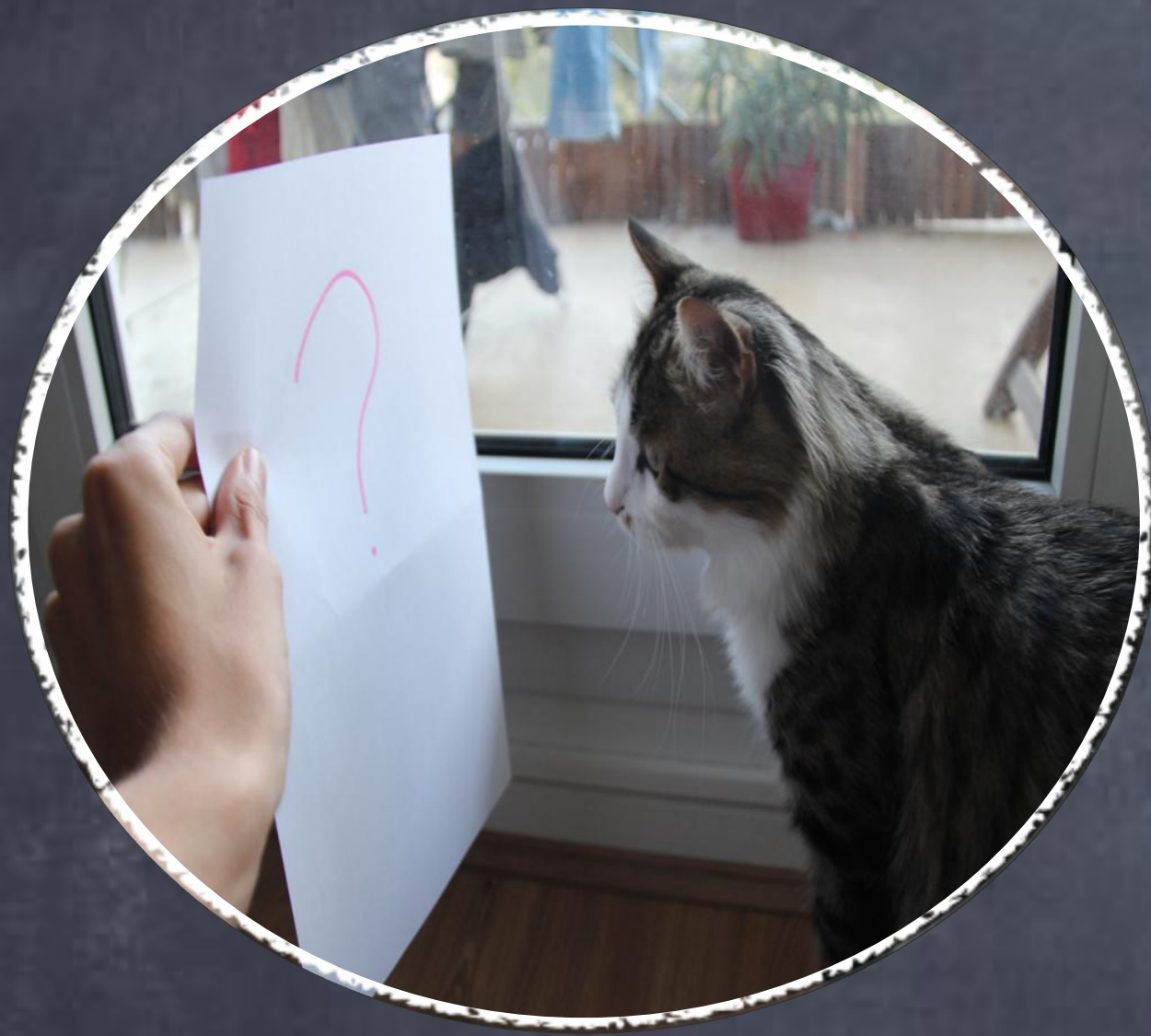
Pourquoi pas...



- Parce-que Hazelcast ça ressemble un peu à Azelart
- Une documentation plutôt clair
- Besoin dans le cadre d'un projet perso
- Démarrage rapide
- Open Source - Apache 2 Licence
- Quelques fonctions magiques

Hazelcast Kesako ?

Une data grid ou un cache



- Cache de données mémoire (en amont d'une BDD)
- Stocker des données comme une session WEB
- Stockage partagé
- Une alternative à memcached
- A suivre...

Qui ?

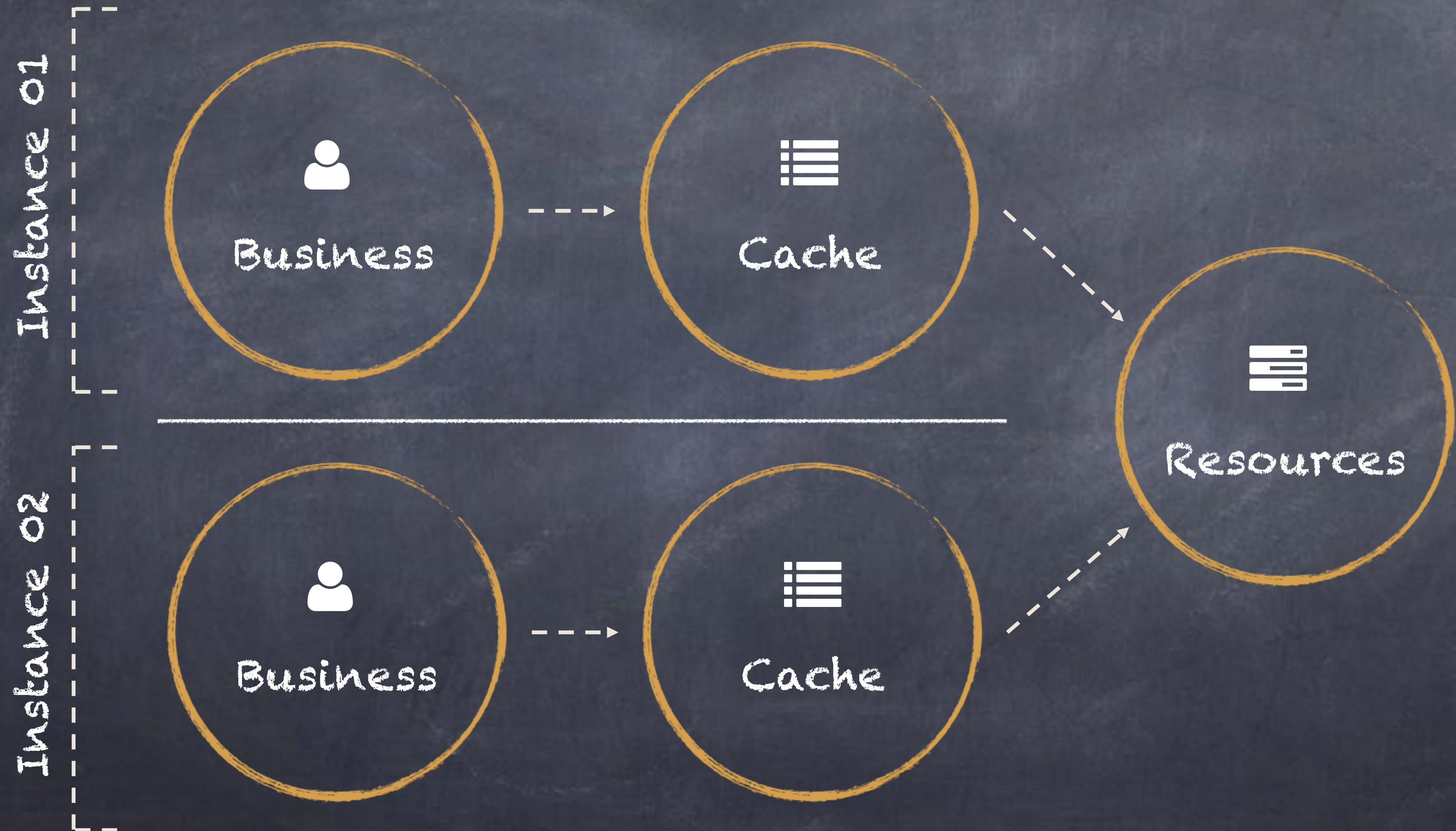
Une data grid ou un cache



- Palo Alto, Californie
- 2013 : réception de fond de Bain Capital
- Janvier 2014 : Greg Luck à rejoint Hazelcast (CTO)

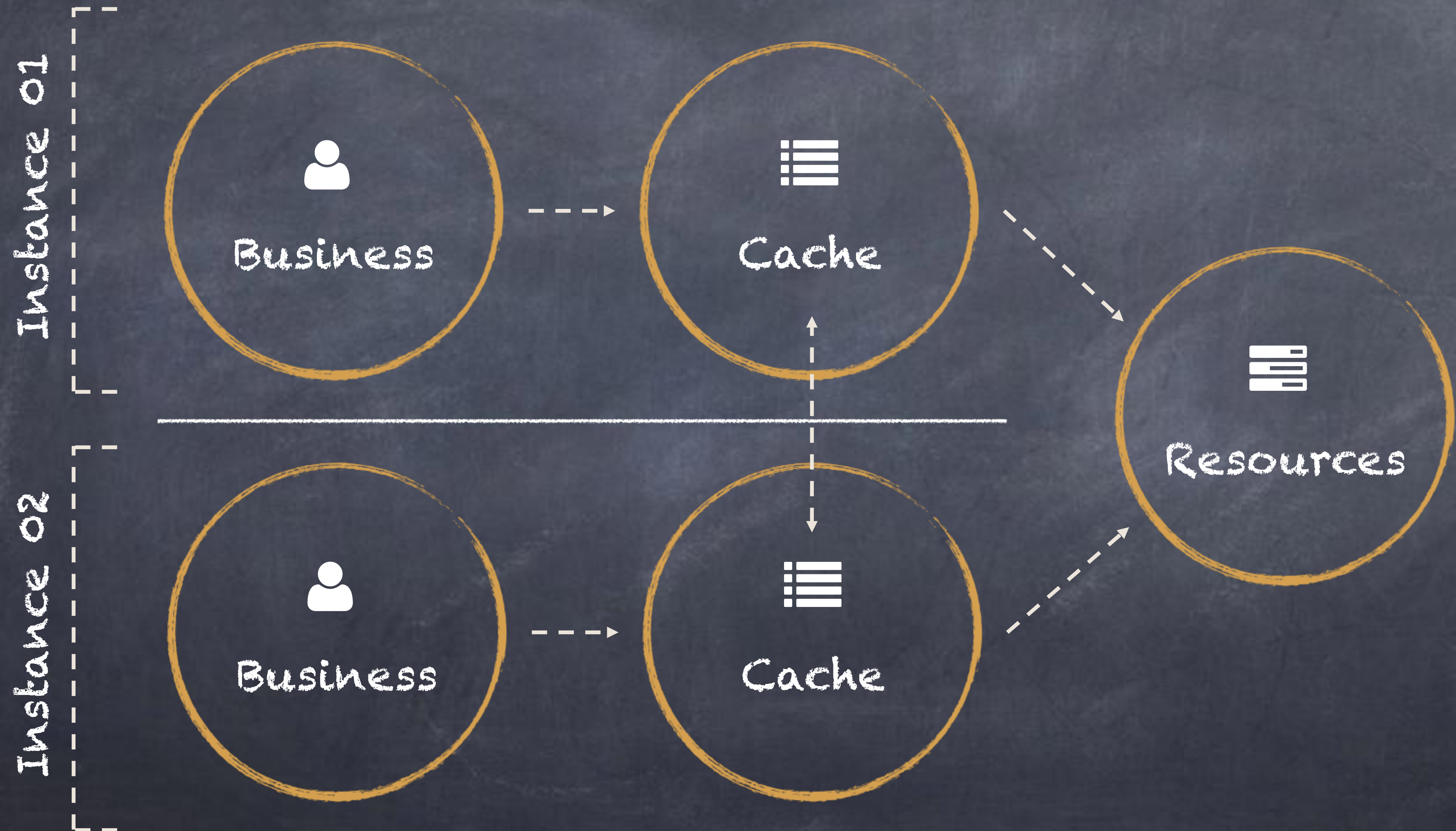
CACHES NON DISTRIBUES

Les caches des différentes instances ne communiquent pas



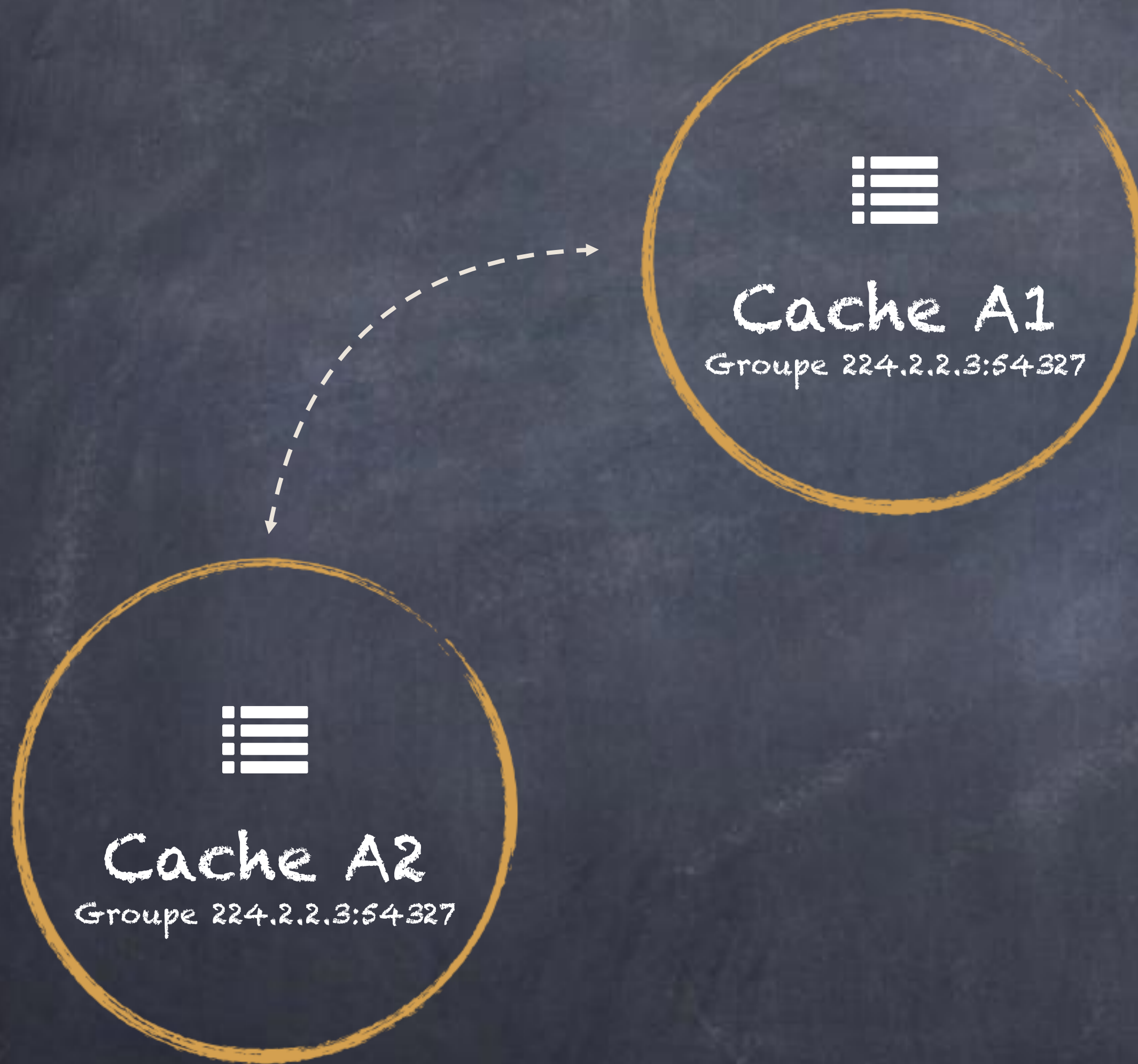
CACHES DISTRIBUTES

Les caches des différentes instances communiquent



HAZELCAST EST UN CACHE DISTRIBUE

Groupe de diffusion : Multicast



💡 Configuration

```
<group>
  <name>dev</name>
  <password>dev-pass</password>
</group>
<network>
  <join>
    <multicast enabled="true">
      <multicast-group>224.2.2.3</multicast-group>
      <multicast-port>54327</multicast-port>
    </multicast>
  </join>
</network>
```



HAZELCAST EST UN CACHE DISTRIBUE


Classique : TCP-IP

💡 Configuration

```
<group>
  <name>dev</name>
  <password>dev-pass</password>
</group>
<network>
  <join>
    <tcp-ip enabled="true">
      <interface>192.168.0.100:5701</interface>
      <interface>192.168.0.101:5701</interface>
      <interface>192.168.0.102:5701</interface>
    </tcp-ip>
  </join>
</network>
```


Cache A1
IP 192.168.0.100


Cache A2
IP 192.168.0.101

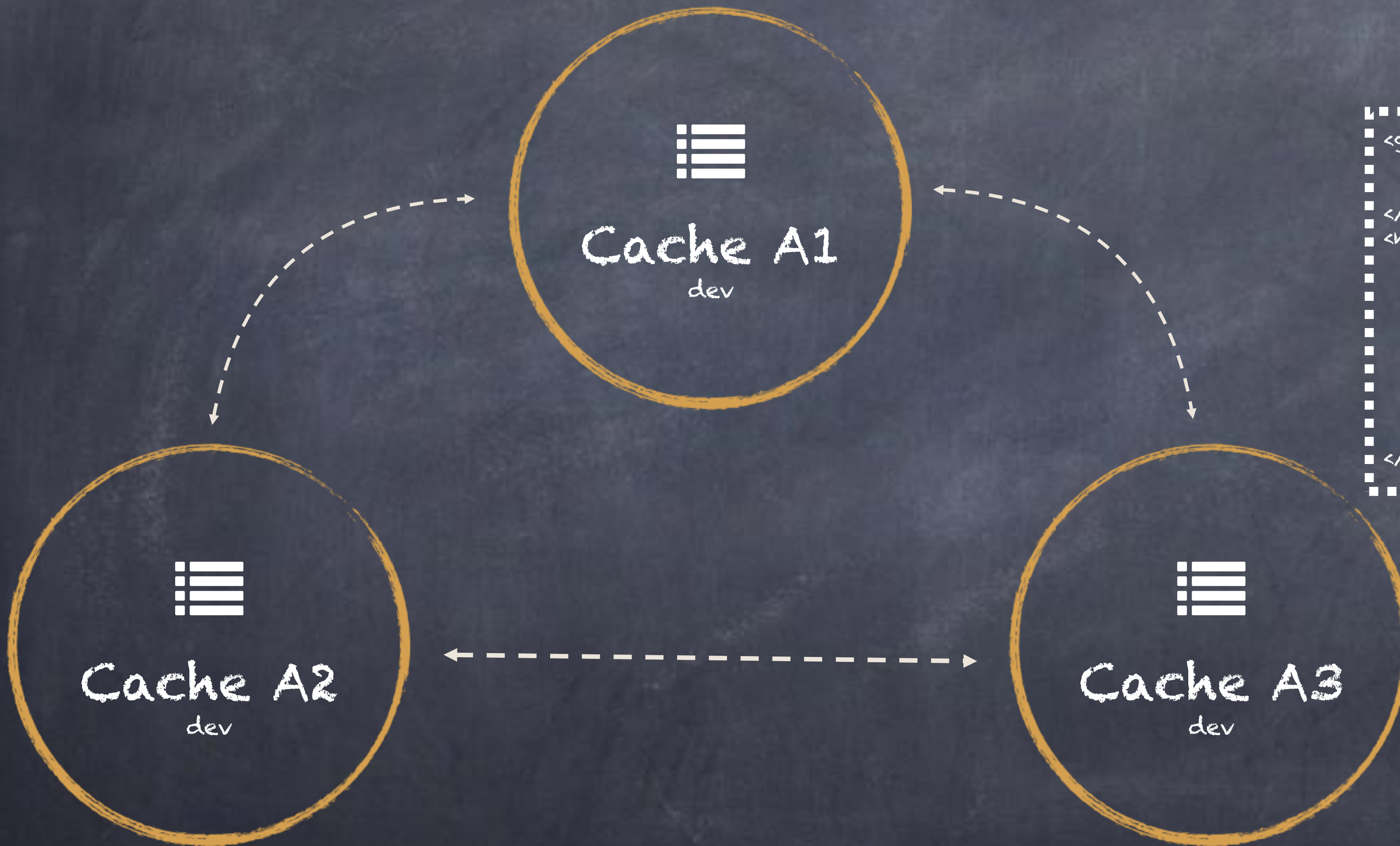

Cache A3
IP 192.168.0.102

HAZELCAST EST UN CACHE DISTRIBUE

Amazon EC2 Auto Discovery

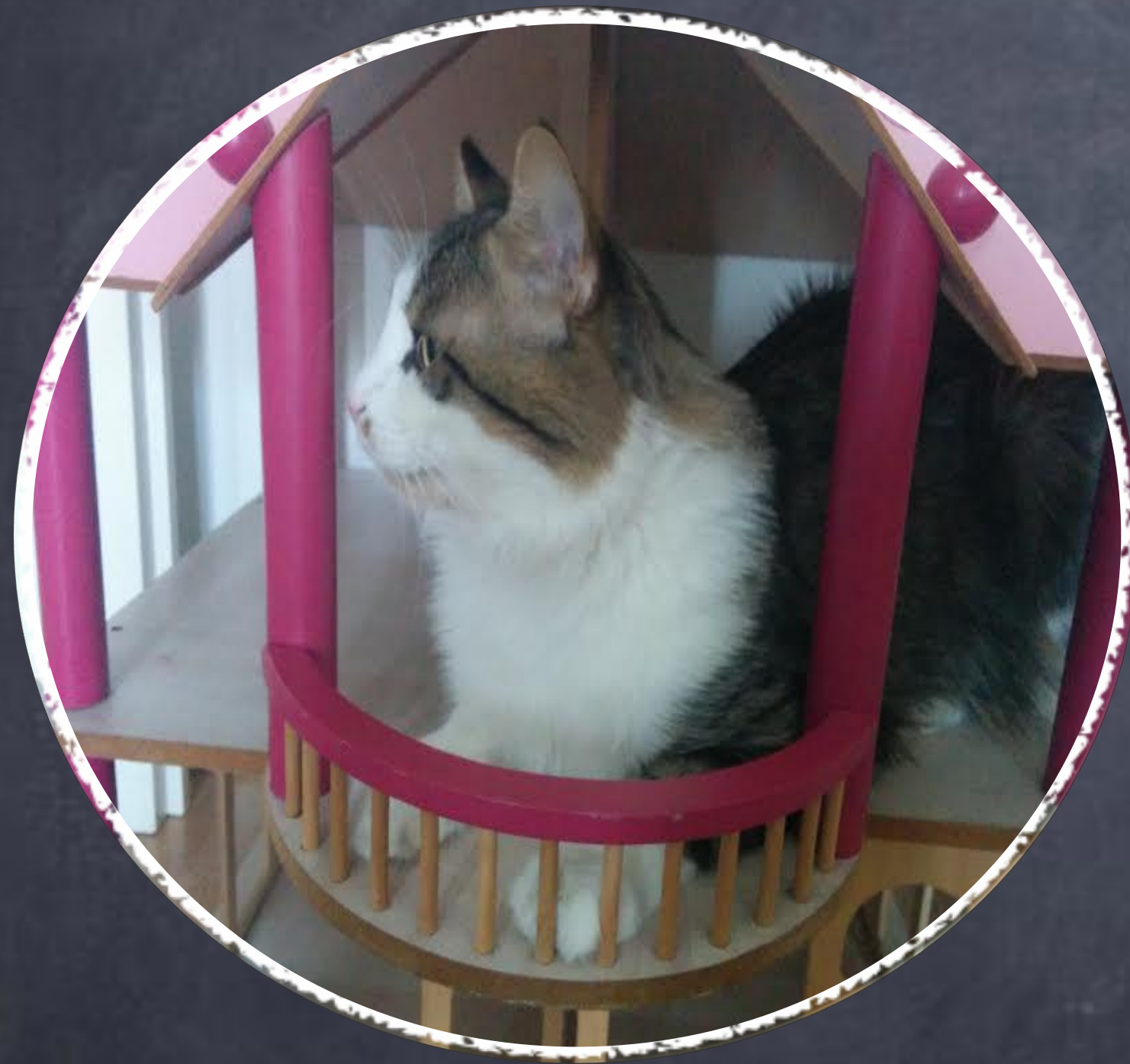
💡 Configuration

```
<group>
  <name>dev</name>
  <password>dev-pass</password>
</group>
<network>
  <join>
    <aws enabled="true">
      <access-key>FOO</access-key>
      <secret-key>BAR</secret-key>
      <region>us-east-1</region>
      <security-group-name>sec</security-group-name>
    </aws>
  </join>
</network>
```



STRUCTURES DE DONNEES

Les (moins) classiques...



- Map : Imap (Thread-Safe)
- Queue
- Set
- List
- Topic
- IdGenerator
- Isemaphore / IAtomicLong

INTEGRATION

Quelques exemples et statistiques

Exemples

- Second level cache pour Hibernate (hibernate \geq 3.3)
- Web Session Replication (Servlet \geq 2.4)
- Intégration Spring (Spring \geq 2.5)

Administration

- API pragmatique pour les structures de données et clusters
- JMX pour les structures de données et clusters
- Management Center (plutôt complet)
- Services REST pour les structures de données et clusters



SECURITE

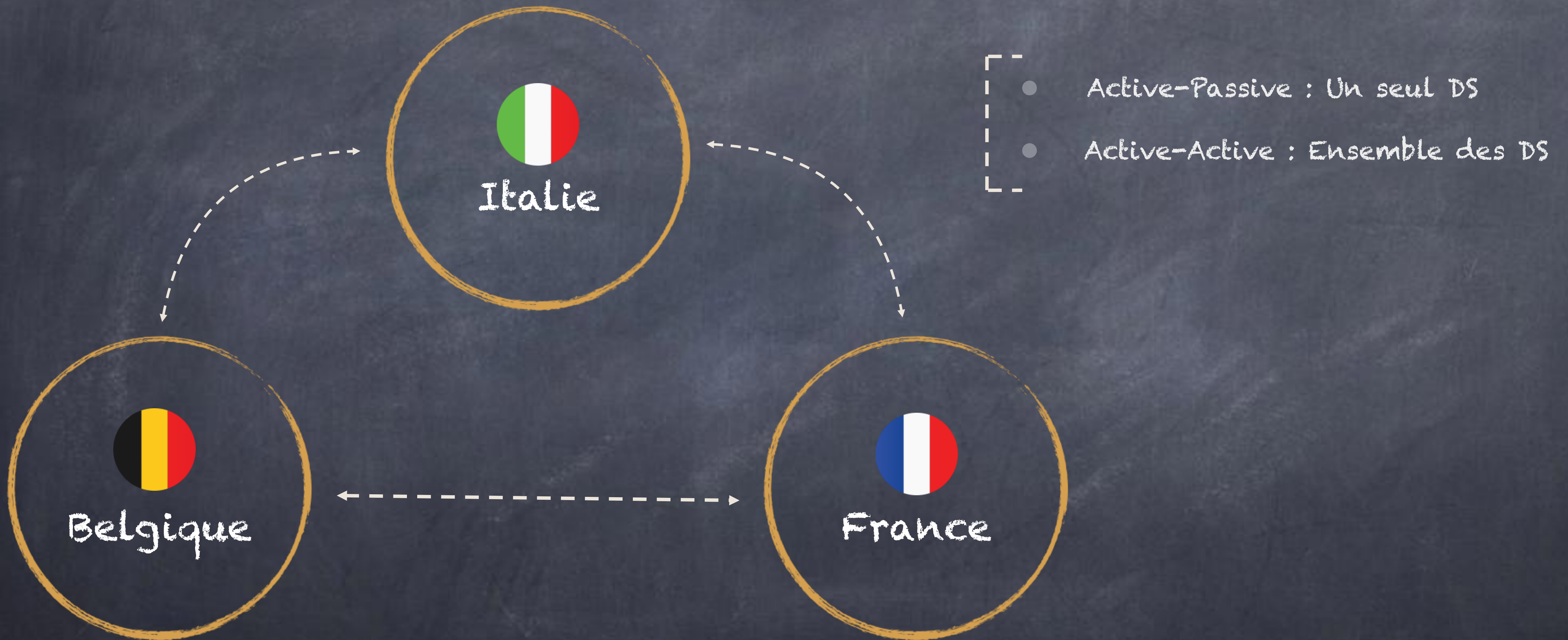
Généralement pour la version Enterprise



- SSL & cryptographie
- JAAS (Java Standard Security)
- Login / Mot de passe (Classique mais non sécurisé)
- ...

REPLICATION WAN

Utilisation dans plusieurs centres de données



DEMONSTRATION

Monitoring du réseau V'Lille

Quoi ?

Connaitre en « temps réel »
le nombre de vélos ou
d'attaches le réseau V'Lille

Un cache ?

Eviter de saturer L'API
V'Lille et accélérer les
temps d'affichage

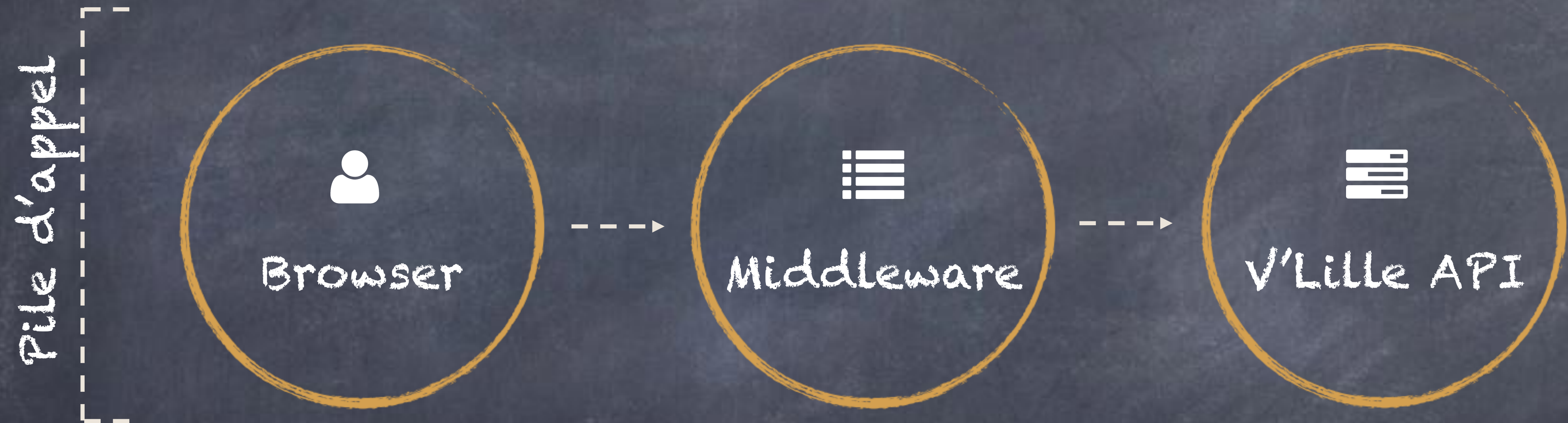
WebApp

Une application WEB :
Gradle / JAVA /
AngularJS / Docker



DEMONSTRATION

Description d'un appel



DEMONSTRATION

Description d'une instance



DOCKER

Présentation rapide de Docker



- Container léger
- S'adapte au serveur cible (Linux)
- Docker n'est pas une VM

DEMONSTRATION

Cluster

Container
10.24.202.133:4001

Container
10.24.202.133:4002

Container
10.24.202.133:4003

Container
10.24.202.133:4004

Container
10.24.202.133:4005

Container
10.24.202.133:4006

Container
10.24.202.133:4007

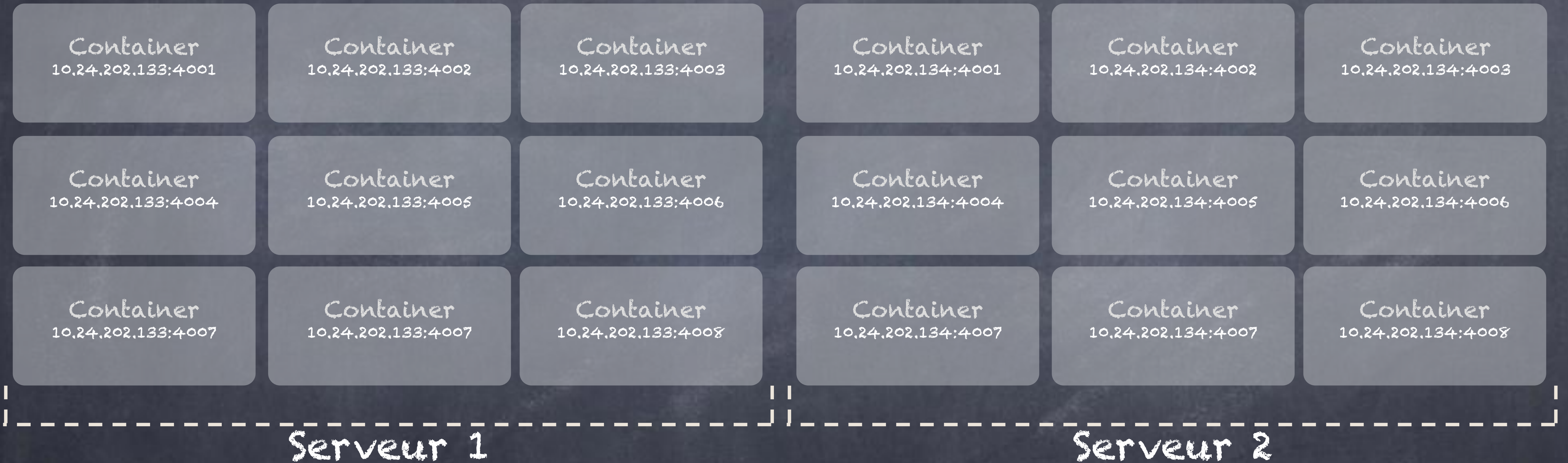
Container
10.24.202.133:4007

Container
10.24.202.133:4008

- Démarrage d'un container simple
- Souplesse pour suivre la charge / Tolérance à la panne
- Pas d'appel à l'API lors de l'intégration d'un container
- Transparent pour l'utilisateur

DEMONSTRATION

Cluster



DEMONSTRATION

Le matériel nécessaire est le suivant...



- Un ordinateur ou serveur de préférence sous Linux
- Une installation de Docker
- Une image de Docker avec un serveur Tomcat
- Un IDE ou de quoi coder quelques lignes de code
- Une installation de Gradle
- Un Kit de développement JAVA ou JDK
- Un peu de temps et de matière grise...



CODE SOURCE

<https://github.com/corentin59/hazelcast-with-docker>



QUESTIONS ?

@CooRentin_