
# http://www.roboconf.net
# https://github.com/roboconf

*Déploiement et reconfiguration dynamique*
*pour le développeur et l'exploitant*

Licence : Apache 2.0
(c) Linagora / Université Joseph Fourier

RMLL 2014 / Pierre-Yves Gibello - pygibello@linagora.com

Présentation sous licence CreativeCommons BY-SA

# Un partenariat

- Linagora
  - Migrer ses solutions vers le cloud
- UJF (Université Joseph Fourier, Grenoble)
  - Recherches middleware (déploiement intelligent, IoT...) + cloud (big data...)
- Equipe commune
  - Dans les locaux UJF, engagement sur 2 ans
- Besoins convergents
  - plateforme de déploiement cloud de niveau industriel, servant de socle.

# Deployer...

- Des applications complexes
  - Incluant des briques patrimoniales ("legacy")
- Sur des IaaS multiples
  - et divers types/tailles de systèmes (du serveur à l'embarqué)

# ... S'adapter ...

- Elasticité
  - Ajout / suppression de noeuds (ex. adaptation à la charge)
- Adaptation des flux
  - Load-balancing, optimisation (ex. co-localisation)

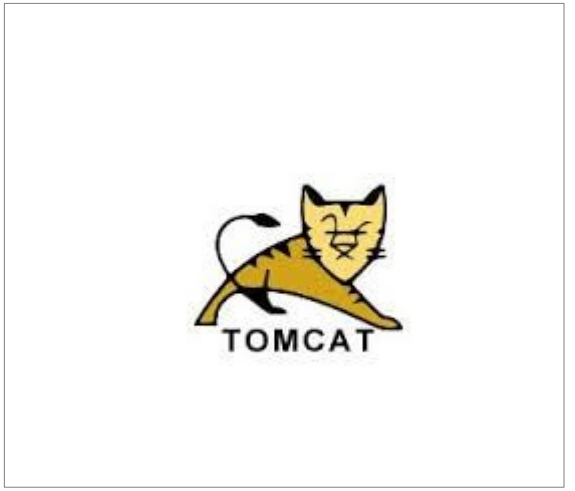
# = Gérer un cycle de vie

- Résoudre les dépendances au runtime
  - Où ? (ex. où envoyer les données, ip, port, etc...)
  - Quoi ? (ex. database credentials...)
- Mettre à jour les configurations, redémarrer les services...
  - Sur chaque noeud, et cohérence globale
  - A chaud (autant que possible)

# Le composant

- Applicatif (Apache, Tomcat...) ou VM (OpenStack, Amazon...)
- Fournit sa configuration et ses recettes de déploiement
  - ex. scripts bash ou puppet, templates de fichiers de configuration...
- Définit ses relations avec d'autres composants
  - contenance ou dépendance runtime
- L'ensemble des composants est un graphe

# Composants : exemple


```
# Apache Load Balancer
Apache {
        alias: Apache Load Balancer;
        installer: puppet;
        imports: Tomcat.portAJP, Tomcat.ip;
}
```

```
# Tomcat
Tomcat {
        alias: Tomcat;
        installer: puppet;
        exports: ip, portAJP = 8009;
        children: Webapp;
}
```

+ Scripts cycle de vie : deploy.pp, start.pp, stop.pp, undeploy.pp
+ Script événementiel dépendances : update.pp
+ Fichiers ou templates de config.

# L'application (modèle)

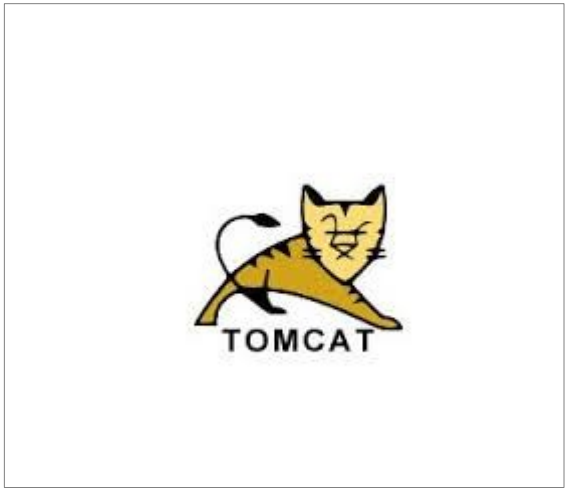
- Ensemble d'instances de composants, déployées dans des conteneurs
  - Conteneur racine = VM
  - Une instance peut en contenir d'autres (ex. serveur d'applications : tomcat avec webapps, etc...)
- Dépendances runtime résolues à chaud
  - Variables de configuration échangées par messagerie asynchrone entre instances dépendantes

# Application : exemple


VM #1

```
# A VM with Apache + load balancer
instanceof VM_EC2 {
        name: Apache VM;

        instanceof Apache {
                name: Apache;
        }
}
```




VM #2

```
# A VM with Tomcat + webapp
instanceof VM_EC2 {
        name: Tomcat VM;

        instanceof Tomcat {
                name: Tomcat1;
                instanceof Webapp {
                        name: DemoApp;
                }
        }
}
```

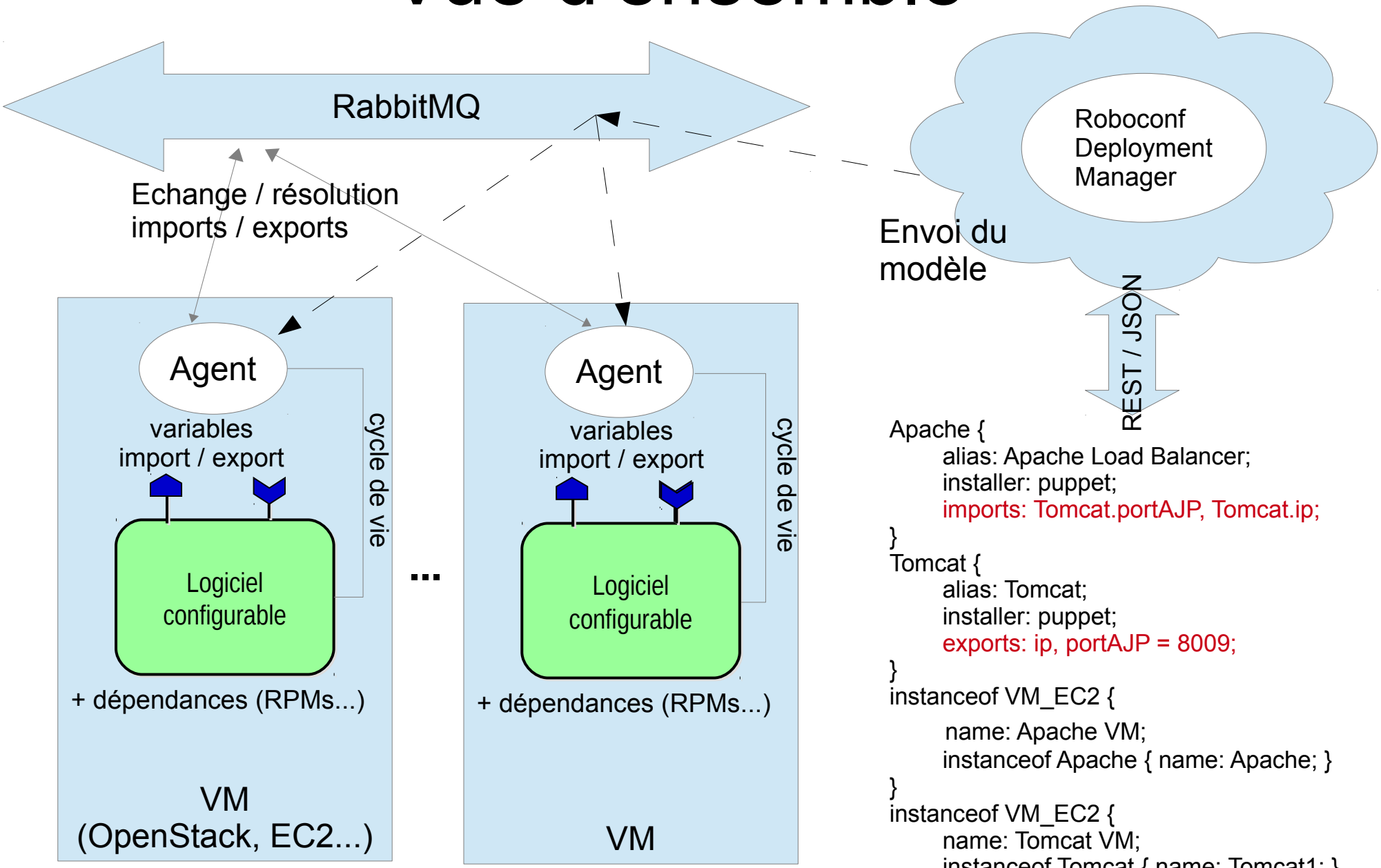
# Qui déploie ?

- Deployment manager
  - Application web
  - Services REST /JSON
- Déploie les VM, et y uploade les instances
  - La VM minimale comporte un agent roboconf
- Fault-tolerant
  - Persistant, restaure son état au redémarrage

# L'agent Roboconf

- Sur chaque VM (obligatoire !)
- Cycle de vie des instances applicatives
  - Scripts deploy / undeploy, start / stop
  - Sur injonction DM ou événement cycle de vie
- Echange les dépendances avec les autres agents
  - imports/exports, script update (événement)
  - les agents communiquent par messagerie asynchrone (RabbitMQ)
- Les instances ne démarrent que si toutes les dépendances sont résolues

# Vue d'ensemble




```
Apache {
        alias: Apache Load Balancer;
        installer: puppet;
        imports: Tomcat.portAJP, Tomcat.ip;
}
Tomcat {
        alias: Tomcat;
        installer: puppet;
        exports: ip, portAJP = 8009;
}
instanceof VM_EC2 {
        name: Apache VM;
        instanceof Apache { name: Apache; }
}
instanceof VM_EC2 {
        name: Tomcat VM;
        instanceof Tomcat { name: Tomcat1; }
}
```

# Divers...

- IaaS : OpenStack, EC2, Azure, Local (embarqué), VMWare (prototype).
  - A venir : Docker
- Installer : Puppet, bash
  - Envisageable : chef ?
- Roadmap 2014 / 15
  - OSGi bundles
  - Supervision / administration

# Demo !

Des questions ?