

ATELIER

Orchestration de Conteneurs avec Roboconf

Mercredi 5 Juillet 2017

Licence : CC BY 3.0



- Quelques slides d'introduction
- Atelier en 5 parties
 - ✓ Découverte de Roboconf
 - ✓ Découverte de Apache Storm
 - ✓ Storm pour Roboconf, version 1
 - ✓ Scalabilité horizontale avec Roboconf
 - ✓ Storm pour Roboconf, version 2
- Conclusion

A Propos...

- **Travaille sur les problématiques de déploiement adaptatif**
 - Cloud / Conteneurs
 - Projet Roboconf
- **Dernières Missions**
 - Études stratégiques / techniques sur les solutions de cloud
 - Mise en production d'un cluster Kubernetes
- **Développeur Java / JS**
 - Roboconf s'appuie sur OSGi et AngularJS
- **Anciennement**
 - Consultant SOA / ESB
 - Committer OW2 (Petals ESB)
 - Committer Eclipse (Eclipse STP / SOA)

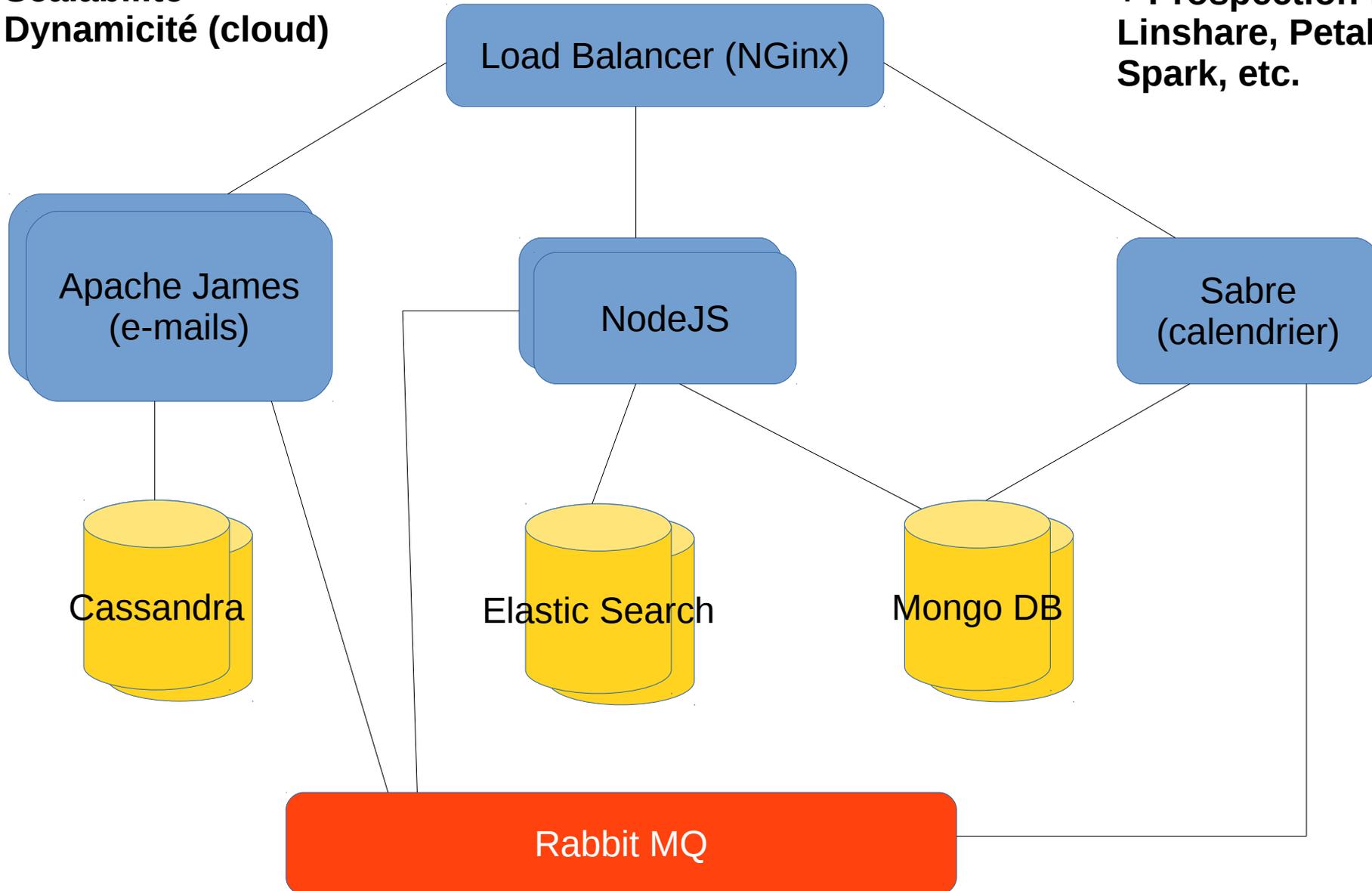
Contexte Linagora

- Linagora : éditeur de solutions logicielles open source
- Open PaaS : plate-forme pour la collaboration des organisations
 - Messagerie électronique
 - Messagerie instantanée
 - Conférence vidéo
 - Agenda
 - Collaboration transversale (gestion de groupes)
- LE projet R&D de Linagora
 - Conçu et structuré pour le passage à l'échelle
 - Axes de dynamicit 
 - Impact sur le choix des technologies

Open PaaS : un bref aperçu (simplifié)

Scalabilité
Dynamicité (cloud)

+ Prospection :
Linshare, Petals ESB,
Spark, etc.



Contexte Général

- **Des solutions diverses**

- Cloud : services à la demande, bouleversement des équipes
- Conteneurs : changement de paradigme, approche micro-services
- Des remises en cause (ex : no-SQL)

- **Avec des limitations**

- Cloud : protection des données
- Conteneurs : dangereux sur certains briques logicielles (BD)

- **Risques**

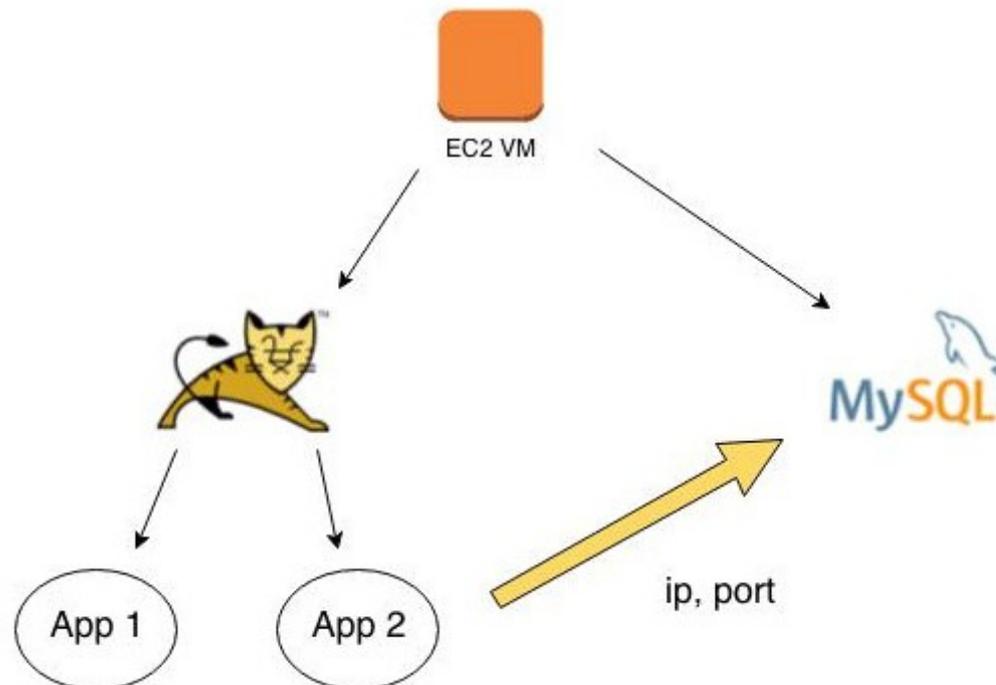
- Faire des choix pour un projet donné
- Faire des choix cohérents au niveau du SI

Roboconf

- **Ambition : gestion semi-automatisée de plates-formes logicielles**
 - Adaptation des topologies applicatives (déploiement, reconfiguration)
 - Procédures automatisées (ex : planification, migrations)
 - Supervision couplée à un moteur de règles (réparation, élasticité)
- **Principes**
 - Agnostique par rapport aux technologies, coexistence, ouverture
 - Rôles développeurs / opérationnels dans une même solution
- **Né d'un partenariat laboratoire public / entreprise**
 - Linagora d'un côté, Université Grenoble-Alpes de l'autre
 - Industrialisation d'un prototype / Transfert de compétences
 - Équipes co-localisées à Grenoble

- Roboconf gère des applications
- Une application contient...
 - ... des méta-données
 - ... un graphe de composant logiciels (relations)
 - ... des « recettes » pour gérer le cycle de vie des composants
 - ... une description de la topologie de l'application

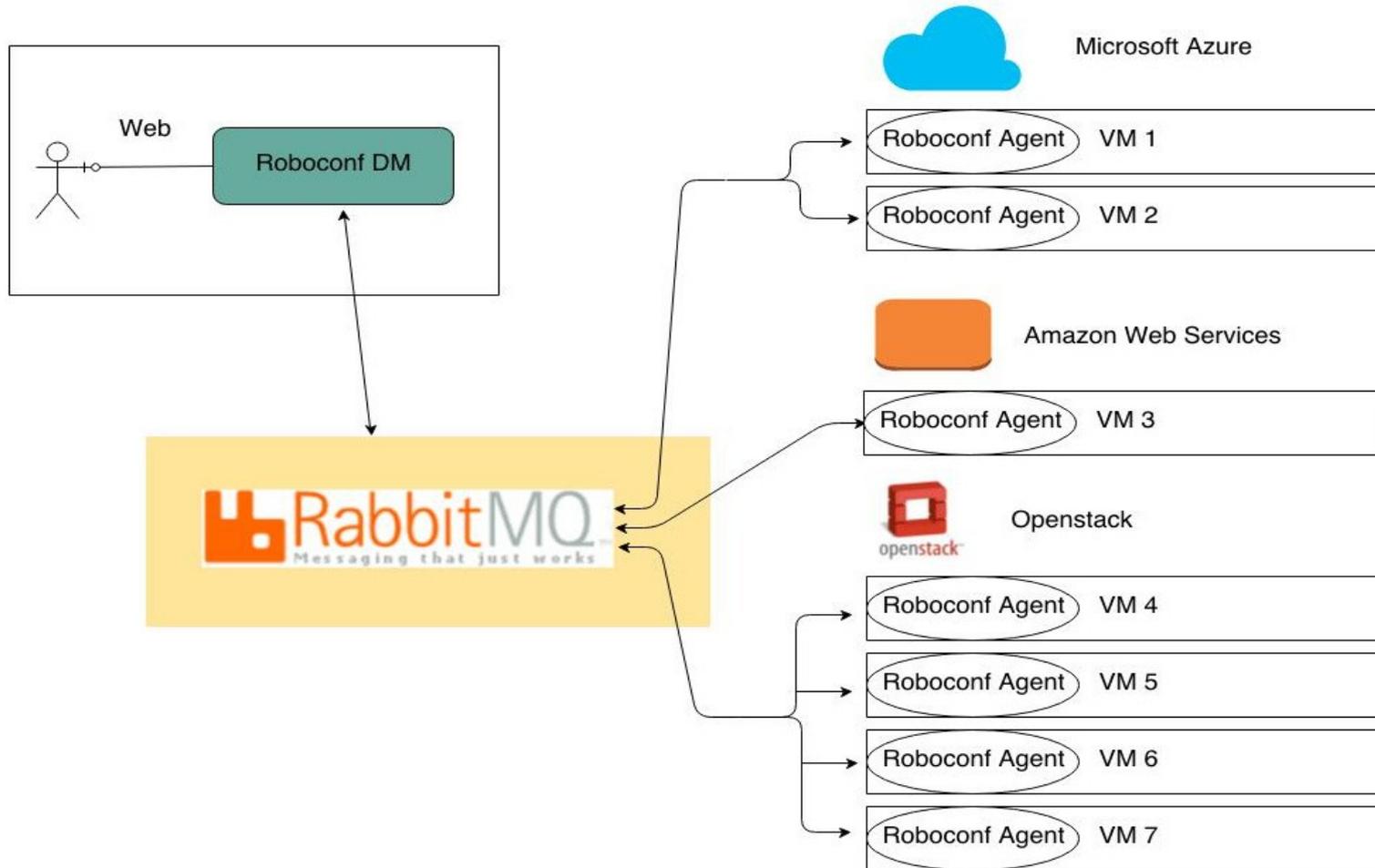
Gestion des dépendances
inspirée d'OSGi.



Architecture

- **Trois briques logicielles**

- Le « deployment manager » (DM : brique d'administration)
- Des agents, localisés sur les machines (répartition du travail)
- Une messagerie, avec plusieurs implémentations



- **Ubuntu Juju**

- = Système de résolution des dépendances
- ≠ Juju restreint à Ubuntu
- ≠ Pas de gestion automatisée

- **Cloudify**

- = Système de résolution des dépendances (TOSCA)
- ≠ Pas ou peu de gestion automatisée

- **Puppet**

- = Architecture à agents, capacité de déploiement
- ≠ Pas de moteur de règles pour gestion automatisée

- **Ansible**

- = Capacités de déploiement
- ≠ Scripting ++, pas de gestion automatisée

- **Kubernetes**

- = Solution riche pour les conteneurs (Docker, Rocket...)
- ≠ Limité aux conteneurs
- ≠ Contraintes fortes sur les types d'applicatifs

- **La suite Docker (engine, swarm, machine...)**

- = Solution riche pour Docker et même le cloud (Docker machine)
- ≠ Peu de gestion automatisée encore (/ Kubernetes)

- **Openshift (PaaS)**

- = Capacités de déploiement / Reconfiguration / Gestion auto.
- Version 3 basée sur Kubernetes (K8s)
- Rajout de fonctionnalités pour le cloud et le cluster K8s
- ≠ Approche usine logicielle, mêmes limitations que K8s

- **Cloud Foundry (PaaS)**

- = Capacités de déploiement
- Solution basée sur des cartouches (extensions)
- Approche usine logicielle, plus haut niveau
- ≠ Complexe à maîtriser

Atelier

- Voir...

- <http://roboconf.net/fr/guide-utilisateur/tutoriel-apache-storm-et-docker-1.html>

- Objectifs

- Comprendre le fonctionnement de Roboconf
- Déploiement d'une pile « Big Data » (Apache Storm) dockerisée
- L'étendre pour pouvoir la déployer sur un cloud
- => déploiement local ou réparti dans environnement dynamique

- Prérequis

- Docker installé
- Maven installé (plus pratique)

Conclusion

- **Découverte**

- Roboconf
- Apache Storm

- **Atelier**

- Application Docker prête pour le cloud
- Cas d'usage atteignable avec Docker Swarm ou Kubernetes...
- ... avec quelques étapes manuelles en plus

- **Aller plus loin...**

- Intégrer dans cette application des briques non-dockerisées
- Exemple : cluster de base de données

Merci pour votre attention !

Avez-vous des questions ?

