DALL'AGNOL Tom
BUISSON Kévin
ROLLIN Antoine

09/04/2016
SI4
DevOps

DevOps – Mr. StataCookie

Artifactory & Jenkins

Brève description de l'objectif ici.

I. Architecture

Définir Jenkins et Artifactory et leur utilisation.

1. Artifactory

Définir les étapes de la configuration d'Artifactory. (Création d'un repo…)

2. Jenkins

Définir la configuration de Jenkins.

Définir la configuration des différents jobs.

II. Forces & Faiblesses

Dans cette partie nous aborderons les forces et faiblesses de l'architecture utilisée, comparée à la structure monolithique de départ.

1. Points forts

Modularité

- Seules les modules modifiés et les modules qui en dépendent vont être recompilés (Maven) => gain de temps (à développer)
- Pas besoin d'avoir le projet tout entier pour travailler puisque les dépendances sont téléchargées depuis Artifactory (à développer)
- Nous avons un projet fortement découpé. Plus le projet est découpé en modules, plus nous réduisons la récupération de ressources dont nous n'avons pas besoin. (à développer)

Fiabilité

- Lorsqu'un module est push alors qu'il ne compile pas ou qu'un des tests ne passe pas, le module n'est pas push vers Artifactory grâce à Jenkins. Les erreurs des uns n'empêchent donc pas les autres de travailler puisque Artifactory disposera et desservira toujours d'un module fonctionnel.

Gestion de la mémoire

- En raison de l'utilisation d'une machine virtuelle, nous disposons d'une capacité de mémoire limitée. C'est pourquoi nous avons configuré Jenkins de façon à ce qu'il ne garde que les dix derniers builds (dont le dernier build en succès).

Intégration continue

- Les tests d'intégrations sont effectués régulièrement pour assurer la cohésion entre les différents serveurs.
- Jenkins est configuré de façon à ce que le serveur dotNet soit lancé avant chaque exécution des tests d'intégration, et arrêté ensuite. (dire pourquoi est-ce mieux que s'il était lancé indéfiniment)

2. Points faibles

Modification non prise en compte

- Si nous modifions un module A et mettons un test d'un module B en erreur (A et B indépendant), le nouveau module A n'est pas push sur Artifactory à cause de l'erreur de B. (à développer)

Attente active

- Jenkins scrute le projet toute les minutes pour recompiler les modules modifiés. Si nous avons beaucoup de jobs qui font ça, la bande-passante est saturée.

Plus long

- Le téléchargement des dépendances rallonge le temps de build comparé au code monolithique (à développer)

Gestion de la mémoire

- Contrairement à Jenkins, Artifactory garde tous les builds, la mémoire est n'est donc pas nettoyée. (à developper)

Release ??

- Artifactory ne contient pas de release (cf Snapshot) (Kévin !!)