DEVOPS

Plan

- Qu'est que c'est DevOps?
- But et objectifs

DevOps : Définition

- Le terme DevOps vient de la contraction des deux termes anglais : Development qui désigne les équipes de développement et Operations qui correspond à l'équipe en charge de l'exploitation des serveurs (les administrateurs systèmes et réseaux).

- DevOps est en fait un mouvement récent visant à rapprocher ces deux équipes et à aligner leurs objectifs sur les besoins de l'entreprise.

Cloisonnement et organisation en silo

- La plupart des services informatiques sont organisés en silo. Les équipes de développement, celles de la qualité, celles en charge de l'exploitation et celles qui s'occupent du support applicatif, sont séparées dans plusieurs services distincts.
- Ces différents services ne se parlent généralement pas ou uniquement lors des mises en production, ou quand un problème survient. Et bien entendu, ce n'est pas forcément le même interlocuteur à chaque fois.
- Ce type d'organisation ne facilite pas les échanges et la collaboration entre les équipes. Et pour couronner le tout, ces services sont éloignés physiquement.
  - Les développeurs sont proches du métier et l'exploitation proche des serveurs. Les équipes ne se rencontrent presque jamais physiquement.

Éloignement culturel

- En plus d'être cloisonnés, ces deux mondes sont totalement éloignés culturellement et ont des objectifs diamétralement opposés.
- Les développeurs cherchent à répondre rapidement aux besoins et demandes d'évolutions du métier. Pour y parvenir, ils s'appuient sur les méthodes agiles. Cela se traduit par des livraisons et des demandes de mise en production beaucoup plus rapprochées.
- De l'autre côté, l'équipe d'exploitation doit assurer au système d'information de l'entreprise un environnement stable, sécurisé et pérenne dans le temps.

Environnement de développement différent de celui de production

- Il n'est pas toujours possible pour des raisons de coût et de mobilisation de ressources d'avoir des environnements (test, dev, recette, production) tous identiques.
- Sur des infrastructures complexes, les développeurs ne peuvent, au mieux, tester leur application dans un environnement proche de la production qu'à partir de la recette.
- Avant d'arriver à cet environnement ils testent son application dans un environnement très différent avec :
  - OS différents
  - Serveurs et versions différentes
  - Configuration différente : pas de cluster ni de load-balancer
  - Règles de sécurité différentes
  - Ces différences se traduisent généralement par des mises en production douloureuses.

Les pratiques et outils DevOps

- Comme pour les méthodes agiles, qui ont rapproché les développeurs et le métier, le mouvement DevOps vise à rapprocher les équipes de développement et d'exploitation.
- Pour cela il s'appuie sur plusieurs processus, des bonnes pratiques et des outils.

Organisation

- L'organisation est le sujet le plus complexe et sensible à mettre en œuvre. Mais c'est un prérequis fondamental pour réussir la mise en place des pratiques DevOps.
- Tout d'abord il faut casser les silos pour fluidifier la communication entre les équipes. Et quoi de mieux pour faciliter cette communication que de regrouper les deux disciplines au sein d'une même équipe et de les rapprocher physiquement. Idéalement dans un même bureau.
- Au-delà de partager un même bureau, il est intéressant de partager les mêmes outils. Il faut que les développeurs et l'exploitation comprennent les contraintes de chacun, et qu'ils recherchent ensemble de façon constructive des solutions à leurs problèmes. L'équipe doit avoir un objectif commun et non des objectifs différents et opposés. Sans quoi les bénéfices des pratiques DevOps seront vite limités.

Continuous Integration

- L'intégration continue consiste à compiler, tester et livrer une application régulièrement. C'est un sujet qui est normalement aujourd'hui bien maîtrisé par les équipes de développement.

- En résumé, les pratiques de base pour faire de l'intégration continue sont :
  - Gérer dans un repository le code source type SVN, Git, Mercurial.
  - Compiler et construire l'application de façon automatique
  - Écrire des tests unitaires et les exécuter régulièrement ou lors de la compilation
  - Construire et packager l'application à intervalle régulier (toutes les nuits) ou déclencher la construction après chaque commit. Et informer l'équipe de développement si un problème apparaît

Continuous Delivery

- Le Continuous Delivery ou Livraison continue, est une pratique visant à automatiser le déploiement d'une application sur différents environnements tout au long de son cycle de vie : Dev, Integration, Recette, Production.
- Pour y parvenir il est nécessaire d'avoir une application correctement packagée et que les différents environnements soient le plus proche possible de celui de production.
- L'application doit pouvoir être déployée sur les différents environnements sans être modifiée. Exit donc les fichiers de configuration spécifiques à chaque environnement nécessitant une recompilation de l'application.
- Même si le déploiement est 100 % automatisé, la mise en production peut nécessiter une validation manuelle.

Continuous Deployment

- Le Continuous Deployment est la pratique qui vient après celle du Continuous Delivery.
- La différence principale avec le Continuous Delivery c'est que les livrables sont déployés automatiquement jusqu'en production.
- Les géants du web font cela plusieurs fois par jour sans que leurs utilisateurs ne s'en rendent compte.
- Le Continuous Delivery doit d'abord être mis en œuvre avant de pouvoir faire du Continuous Deployment. Si le Continuous Delivery peut être mis en œuvre simplement et rapidement pour toutes les applications, le Continuous Deployment est difficilement applicable tout le temps. Cette pratique nécessite d'avoir une grande confiance dans les modifications qui sont envoyées en production. Il faut pour cela réussir à automatiser les tests d'intégration.

Infrastructure as Code

- L'infrastructure as Code consiste à définir son infrastructure sous forme de code.
- Pour assurer une plus grande stabilité, on applique les mêmes pratiques que pour le développement à l'infrastructure :
  - Tests unitaires
  - Intégration continue
  - L'objectif est d'avoir un environnement identique du dev à la production.
  - En dev, on utilise des VM lancées sur le poste du développeur. On y installe les mêmes outils que pour la production.

Amélioration continue

- Tout automatiser est une bonne pratique, mais automatiser et répéter inlassablement les mêmes erreurs n'en est définitivement pas une.
- La culture DevOps implique d'améliorer continuellement ses processus et ses outils. La première étape consiste à récolter régulièrement des feedbacks et des indicateurs. Puis sur la base de ces informations, formaliser, définir puis suivre un plan d'amélioration.
- Attention toutefois à ne pas prendre des indicateurs qui risqueraient de faire entrer en conflit les développeurs et l'exploitation.
  - Les indicateurs à prendre en compte doivent être ceux permettant de réduire le Time to Market.

Intégration Continue

Intégration Continue : Wikipedia

L'intégration continue est un ensemble de pratiques utilisées en génie logicielle consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

Intégration Continue : Pratique

- Etre confiant qu'un logiciel fonctionne correctement et efficacement
  - Améliorer la qualité du code
  - Vérifier le comportement du code avec les tests unitaires
  - Déployer un système complet et valider les fonctionnalités

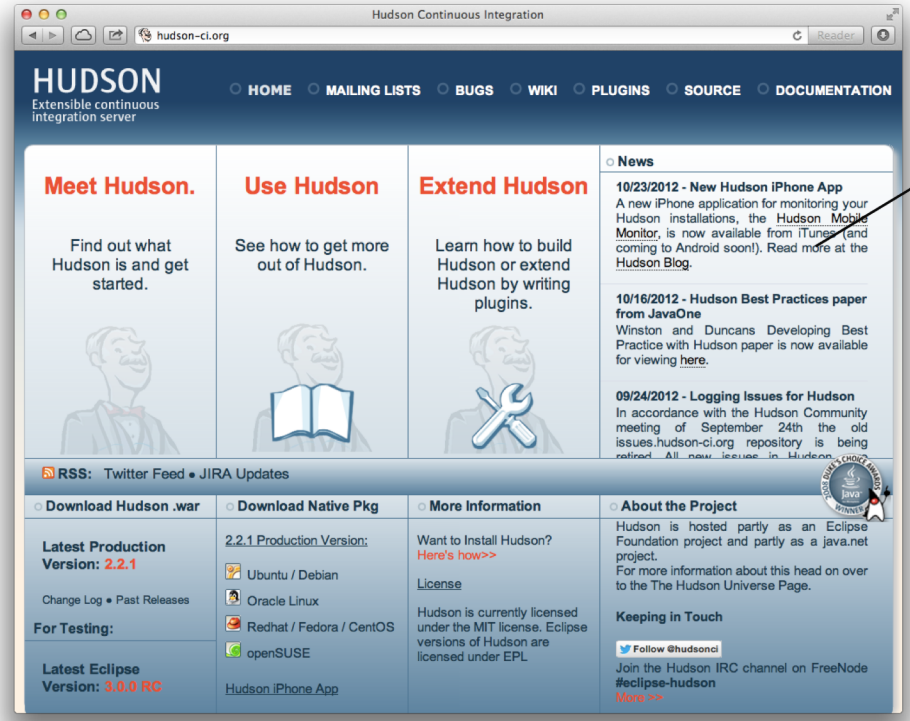
- Toutes ces propriétés pour chaque changement du code!

Les outils Automatisés

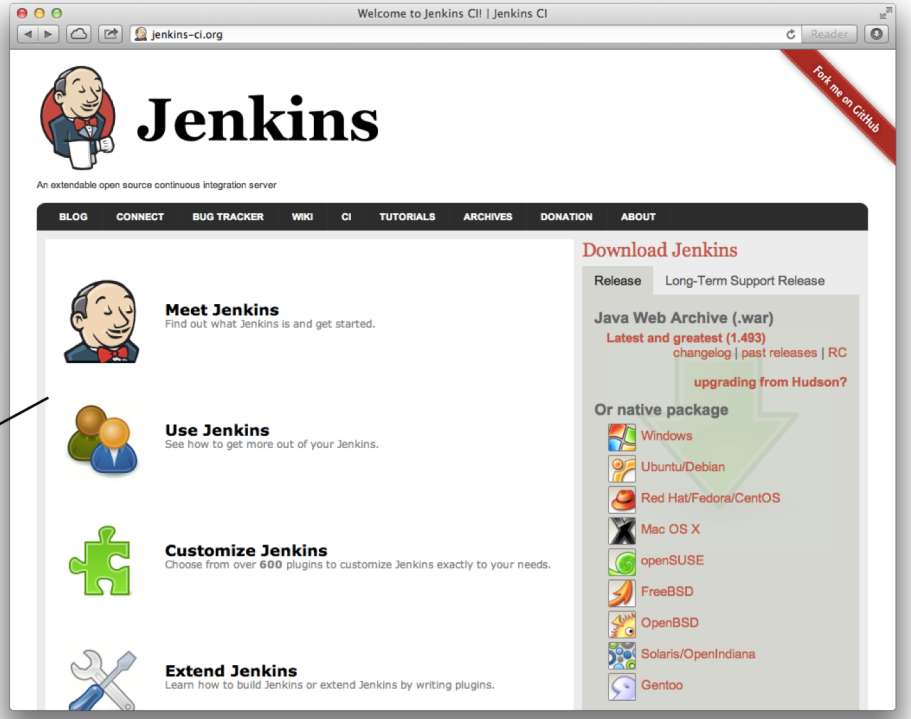
- L'intégration continue est possible avec un minimum d'effort humain grâce à des outils automatisés

- Open Source
  - Hudson, Jenkins, Travis-CI, CruiseControl, …
- Commerciaux
  - Bamboo, TeamCity, …

Hudson & Jenkins

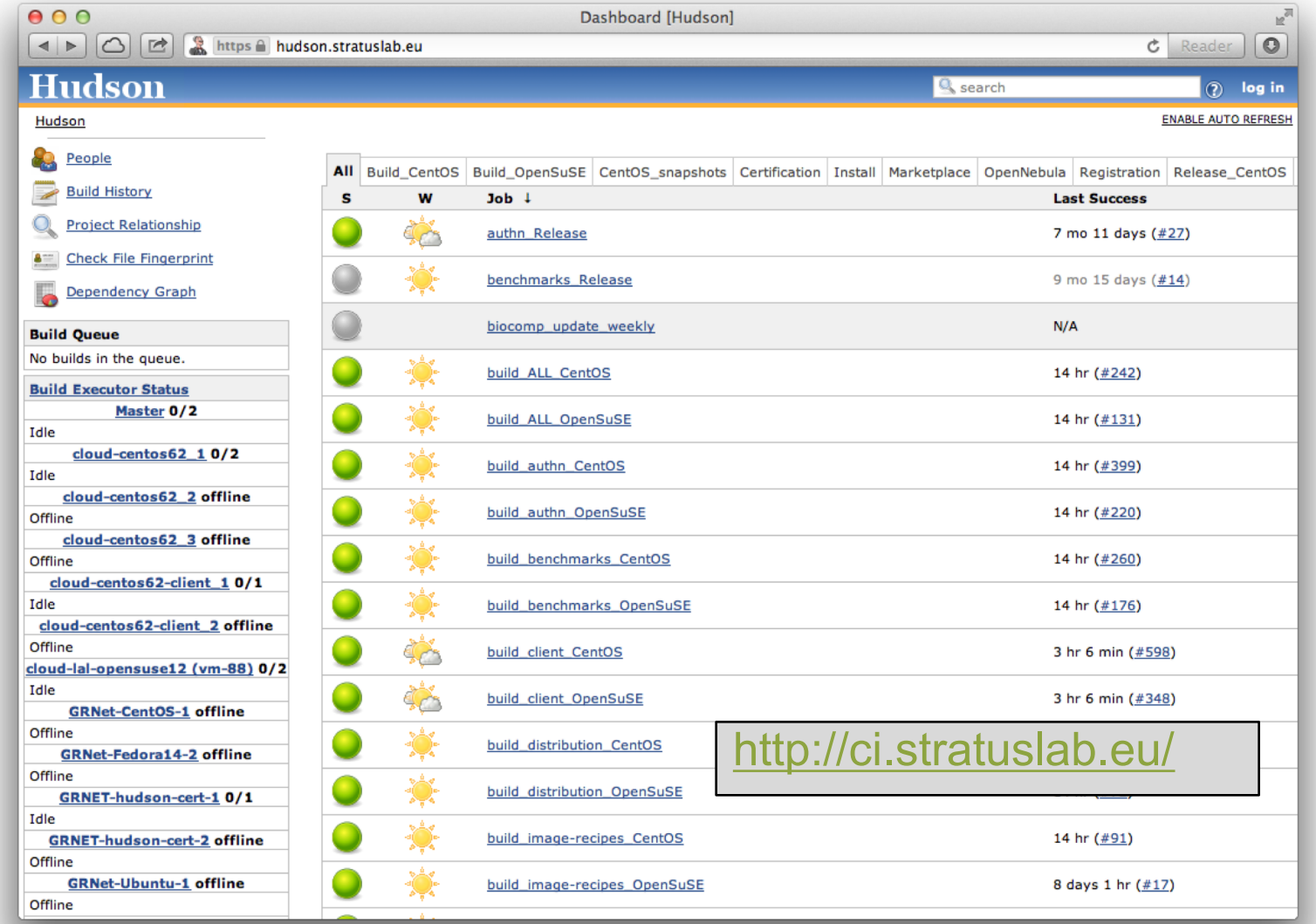


Hudson
http://hudson-ci.org



Jenkins
http://jenkins-ci.org

StratusLab


Dashboard [Hudson]
https hudson.stratuslab.eu
Reader
Hudson
search
log in
Hudson
ENABLE AUTO REFRESH
People
Build History
Project Relationship
Check File Fingerprint
Dependency Graph
Build Queue
No builds in the queue.
Build Executor Status
Master 0/2
Idle
cloud-centos62_1 0/2
Idle
cloud-centos62_2 offline
Offline
cloud-centos62_3 offline
Offline
cloud-centos62-client_1 0/1
Idle
cloud-centos62-client_2 offline
Offline
cloud-lal-opensuse12 (vm-88) 0/2
Idle
GRNet-CentOS-1 offline
Offline
GRNet-Fedora14-2 offline
Offline
GRNET-hudson-cert-1 0/1
Idle
GRNET-hudson-cert-2 offline
Offline
GRNet-Ubuntu-1 offline
Offline
All
Build_CentOS
Build_OpenSuSE
CentOS_snapshots
Certification
Install
Marketplace
OpenNebula
Registration
Release_CentOS
S
W
Job
Last Success
authn_Release
7 mo 11 days (#27)
benchmarks_Release
9 mo 15 days (#14)
biocomp_update_weekly
N/A
build_ALL_CentOS
14 hr (#242)
build_ALL_OpenSuSE
14 hr (#131)
build_authn_CentOS
14 hr (#399)
build_authn_OpenSuSE
14 hr (#220)
build_benchmarks_CentOS
14 hr (#260)
build_benchmarks_OpenSuSE
14 hr (#176)
build_client_CentOS
3 hr 6 min (#598)
build_client_OpenSuSE
3 hr 6 min (#348)
build_distribution_CentOS
build_distribution_OpenSuSE
build_image-recipes_CentOS
14 hr (#91)
build_image-recipes_OpenSuSE
8 days 1 hr (#17)
http://ci.stratuslab.eu/

OpenStack

- Openstack aussi utilise Jenkins.

http://ci.openstack.org/jenkins.html

Testing Continu

- Exécuter les tests unitaires sur plusieurs OS après chaque commit
  - Bonne intégration avec les gestionnaires du code : git, svn, cvs, …
  - Définir une matrice d'environnements différents : OS, java, …
  - Les jobs peuvent être déclenchés automatiquement avec notification des résultats
  - Support excellent pour maven
  - Bonne intégration avec LDAP

Contrôle de Qualité

- Intégré avec les tests unitaires et automatisés au maximum
- FindBugs – analyseur statique pour Java
  - Traité comme un test unitaire pour les composants écrit en Java
  - Détecte les problèmes de codage fréquents et forme les développeurs non-experts en Java
  - Les « code reviews » peuvent se concentrer sur les problèmes de design et d'architecture

Intégration Continue

- Créer tous les composants nécessaires pour un système complet
  - Construire les packages (CentOS, OpenSuSE, …)
  - Déployer un cloud test openstack avec ces packages
  - Faire des tests fonctionnels dans le cloud test
- Jenkins peut gérer les dépendances entre les jobs quasi-automatiquement
- Peut stocker les artifacts dans Jenkins

Gestion de Releases

- Faire toutes les actions pour une release finale
  - Le tagging du code est fait par les jobs jenkins
  - Génération des packages finaux
  - Création des dépôts YUM
  - Installation et vérification du système dans une infrastructure pour la certification

Points Forts

- Jenkins est stable
- Simple à déployer et à configurer
- Milliers de plugins pour automatiser les tâches
- Très bonne intégration avec les outils du build (Maven par exemple)
- Très bonne intégration avec les outils du test
- Historique du statut des jobs et un résumé des changements du code
- Facile d'ajouter de nouveaux jobs et éditer les jobs existants

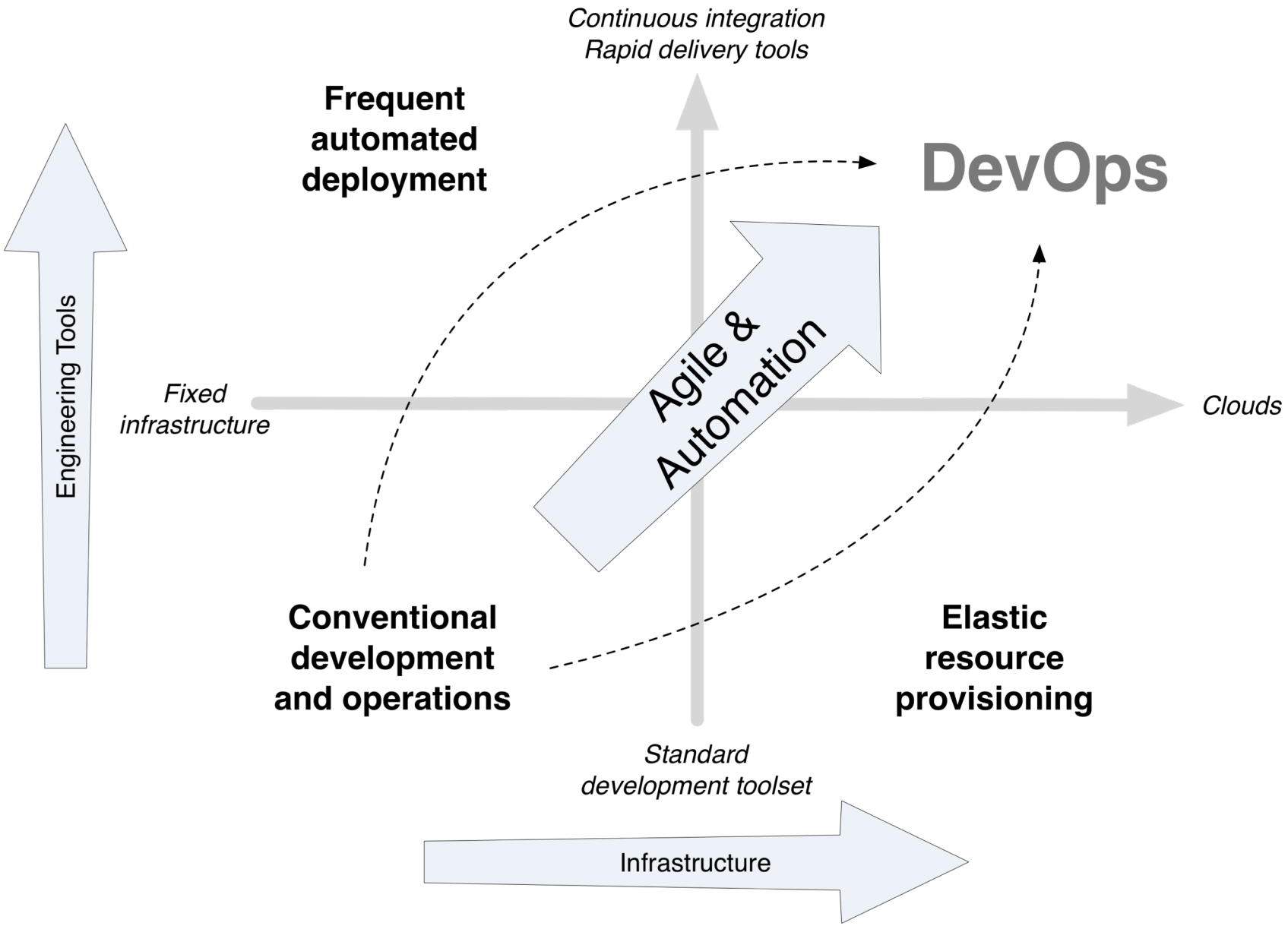
Pas Complètement Automatique

- Ça prend du temps pour maintenir les jobs dans Jenkins et ajouter des nouveaux
- Doit avoir quelqu'un(e) qui surveille les résultats et force les gens à corriger leurs bugs
- Ça prend du temps pour créer et maintenir les machines avec le(s) bon(s) environnement(s) pour le build et le test

Résumé

- Tous les projets de développement doivent utiliser un système de l'intégration continue
  - Ça valide les changements du code rapidement
  - Permet la validation dans plusieurs environnements
  - Facilite les discussions des problèmes/changements
- Jenkins
  - Bon candidat pour vos projets : facile à déployer mais robuste et complet
  - Suffisamment flexible pour automatiser les tests unitaires, contrôles de qualité, l'intégration des composants, et la gestion des releases

Vers DevOps…


Continuous integration
Rapid delivery tools
Frequent automated deployment
DevOps
Engineering Tools
Fixed infrastructure
Agile & Automation
Clouds
Conventional development and operations
Elastic resource provisioning
Standard development toolset
Infrastructure

Buts et Objectifs : Devops CloudPlatform