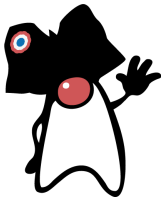
Production logicielle, outils et pratiques

JDEV2017 - T4 / Johan.Moreau - at - gmail.com|ircad.fr

Johan Moreau

IRCAD/IHU

6 juillet 2017



Johan Moreau

@ : Johan.Moreau sur ircad.fr/gmail.com

D.S.I de l'Institut de Recherche contre les Cancers de l'Appareil Digestif Responsable des développements dans l'équipe R&D
`http://www.ircad.fr/` - `http://www.ihu-strasbourg.eu/`

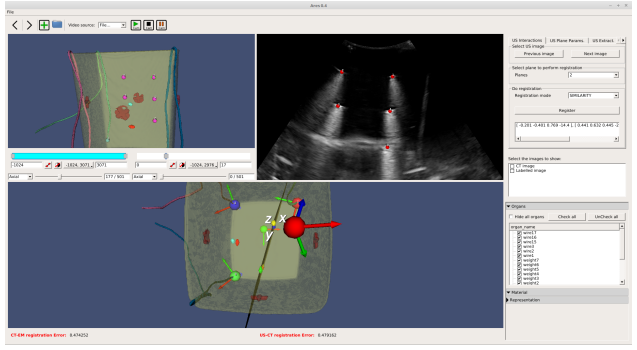
Membre du bureau des associations Clusir-Est et Elsass-JUG
Participation à divers projets opensource :
nagios-i18n, kosmos-i18n, SConspiracy, FW4SPL, RMLL, ...
Membre HackingHealth et Réserve Citoyenne

Enseignements : Génie logiciel/POO, SSI/SSR, ...

IRCAD/IHU c'est quoi ?

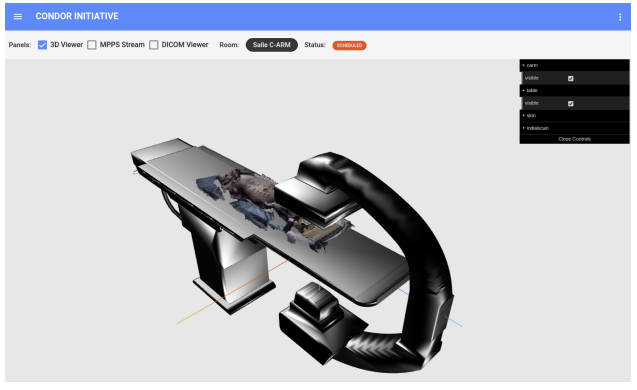
Le métier :

- Institut de recherche (voir posters)
- Chirurgie guidée par l'image



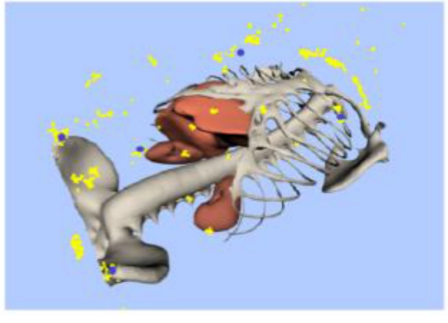
Les apps :

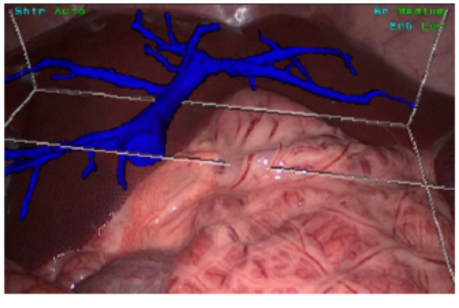
- Applications lourdes (2M loc/15M loc)
- Applications microservices



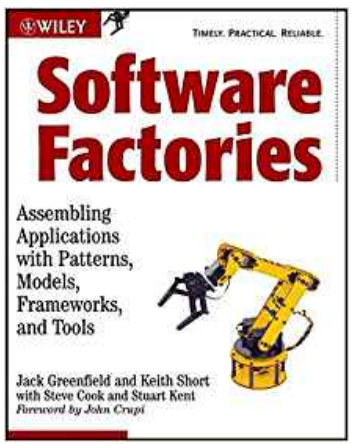
Les dev et les ops :

- 2*15 devs (10 projets, 50 apps)
- 2 devs (10 apps labo/IT)
- 3+2 ops






- Présenter la thématique, donner des pointeurs sur les ateliers/GT
- Donner des pistes sur la littérature mais pas rentrer dans les détails
- Donner 1 point de vue, chaque logiciel est différent et chaque manière de le construire (avec des humains dans le processus) est différent



Les difficultés :

- Augmentations des besoins numériques (donc ↗ des services)
- Fréquence de mises à disposition ultra rapide
- Inteconnexions plus fortes entre les services
- Difficultés de maintenir la documentation d'exploitation
- Multiples environnements (dev, tests, production, ...)
- Cas complexes des clusters, de la HA, des workflow de dev, ...

Les difficultés :

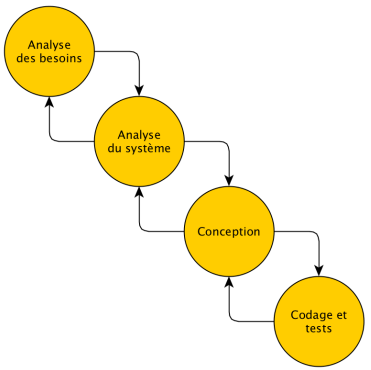
- Augmentations des besoins numériques (donc ↗ des services)
- Fréquence de mises à disposition ultra rapide
- Inteconnexions plus fortes entre les services
- Difficultés de maintenir la documentation d'exploitation
- Multiples environnements (dev, tests, production, ...)
- Cas complexes des clusters, de la HA, des workflow de dev, ...

Réponses partielles :

- Agilité : meilleure production/vélocité
- DevOps : meilleure exploitation
- Anciens outils toujours plus robustes
- Nouveaux outils liés à ces tendances

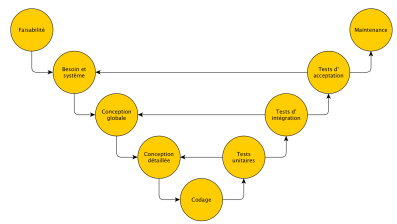
Méthodes linéaires :

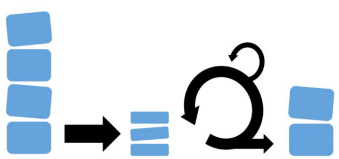
- Adaptées à un cahier des charges fixé/stable



Les principes de l'agilité :

- Cycle en V trop long, cycle rapide [1]
- Beaucoup de principes :
  - livraisons très régulières pour retours clients,
  - proximité du client (meilleure communication),
  - mode équipe, le code est à tout le monde,
  - gestion du code, construction simple, ...
- Méthodologies : Scrum [2], XP, ...
- Outils : DVCS, intégration continue, ...
- *Simple sur le papier, long à mettre en place ...*

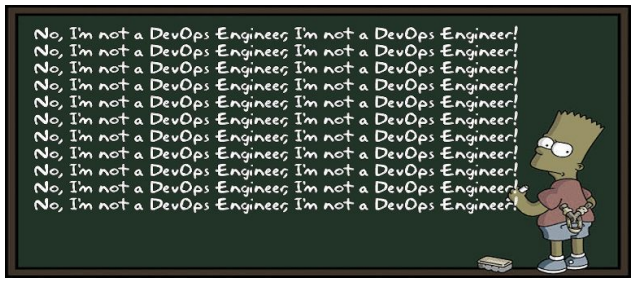




[1] http://eureka.sbs.ox.ac.uk/897/1/WP_2011_08_15.pdf
[2] Ken Schwaber, Jeff Sutherland

Les principes du DevOps :

- Les exploitants ont des conditions différentes des développeurs :
  - les faire travailler ensemble !
- Mises en exploitation très régulières et rapides [3] [4]
- Très présent chez les leaders :
  - Amazon - toutes les 11,6s [5],
  - Etsy - 30fois/j [6],
  - Flickr - 25fois/j



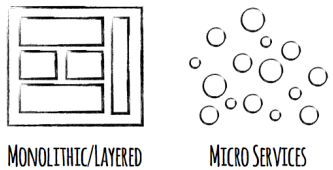
- Méthodologies : Kanban, ...
- Outils : VM, conteneurs, gestion de configuration, ...
- *Etape complexe : bon alignement Devs et Ops, + agilité côté Devs !*

[3] https://puppetlabs.com/wp-content/uploads/2013/03/2013-state-of-devops-report.pdf
[4] http://pages.zeroturnaround.com/RebelLabs-AllReportLanders_DevopsProductivityReport.html
[5] http://assets.en.oreilly.com/1/event/60/Velocity%20Culture%20Presentation.pdf
[6] https://codeascraft.com/2012/03/13/making-it-virtually-easy-to-deploy-on-day-one/
Image from https://3ovyg21t17l11k49tk1oma21-wpengine.netdna-ssl.com/wp-content/uploads/2014/03/bart-simpson-generator-1.jpg

Architectures :

Johan Moreau
GT2

- Monolitique(client lourd/léger/hybride) SOA/BPM
- **Microservices REST** [8] **:**
  - Minimisation afin de gagner en maintenabilité
  - Définition d'un service la plus claire possible
  - Ancien (Minix vs Linux [9]) mais arrivée de REST
  - *Impose un design/réflexion très particulier*

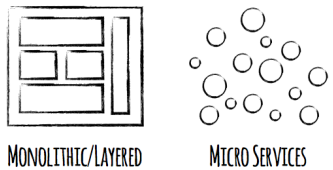


[8] http://en.wikipedia.org/wiki/Microservices
[9] http://en.wikipedia.org/wiki/Tanenbaum-Torvalds_debate
Image from https://lh4.googleusercontent.com/-X1VVNh7EDRM/VIck4odR7yI/AAAAAAAAA9w/pOd7adeT0E8/w600-h282-no/micro-service-architecture.png
[11] SDI ou Infra-as-Code :http://en.wikipedia.org/wiki/Software_Defined_Infrastructure

Architectures :

Johan Moreau
GT2

- Monolitique(client lourd/léger/hybride) SOA/BPM
- **Microservices REST** [8] **:**
  - Minimisation afin de gagner en maintenabilité
  - Définition d'un service la plus claire possible
  - Ancien (Minix vs Linux [9]) mais arrivée de REST
  - *Impose un design/réflexion très particulier*



Infrastructure définie par le logiciel [11] :

- Pour gagner en productivité, automatisation d'infrastructures complexes (stockage, réseau, sécurité)
- Outils et API de plus en plus présentes
- TDI : Test-Driven-Infrastructure (suite du TDD)
- *Projet transversal imposant des connaissances larges*

[8] http://en.wikipedia.org/wiki/Microservices
[9] http://en.wikipedia.org/wiki/Tanenbaum-Torvalds_debate
Image from https://lh4.googleusercontent.com/-X1VVNh7EDRM/VIck4odR7yI/AAAAAAAAA9w/pOd7adeT0E8/w600-h282-no/micro-service-architecture.png
[11] SDI ou Infra-as-Code :http://en.wikipedia.org/wiki/Software_Defined_Infrastructure

Ensemble d'outils fonctionnant ensemble pour développer dans un/des langage(s) :

- Editeur de texte spécialisé, intégration :
  - compilateur, interpréteur, débogueur, générateur de doc, outil de construction, gestionnaire de versions, ...

Ensemble d'outils fonctionnant ensemble pour développer dans un/des langage(s) :

- Editeur de texte spécialisé, intégration :
  - compilateur, interpréteur, débogueur, générateur de doc, outil de construction, gestionnaire de versions, ...

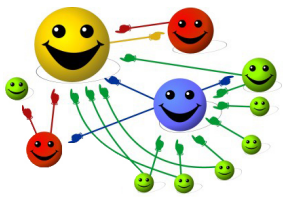
Nombreux outils avec approche différente :

- console (open) : vim, emacs, nano, ...
- IHM (open) : eclipse, pydev, xemacs, gedit, atom.io, ...
- IHM : VS/code, xcode, SublimeText, WebStorm, ...
- online : codeanywhere, eclipse/che, OrionHub, Jupyter, ...

Convergence avec l'environnement serveur (CI, gestion conteneurs, ...)

Automatiser l'ensemble des actions contribuant, à partir de données sources, à la production d'un logiciel [12] [13] [14] :

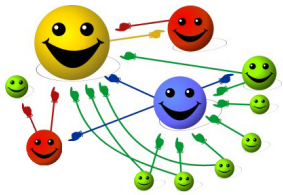
- Automake, CMake, Ants, SCons, Gradle, Grunt, ...
- *Dockerfile*



[12] http ://en.wikipedia.org/wiki/Software_build
[13] http ://en.wikipedia.org/wiki/Build_automation
[14] https ://projet-plume.org/files/ENVOL2010-ConstructionApplications_0.pdf
[15] https ://en.wikipedia.org/wiki/Dependency_hell
[16] Type git-flow ou "features branching"

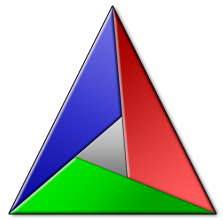
Automatiser l'ensemble des actions contribuant, à partir de données sources, à la production d'un logiciel [12] [13] [14] :

- Automake, CMake, Ants, SCons, Gradle, Grunt, ...
- *Dockerfile*



Gestion des dépendances :

- Nombre de dépendances croissant [15] , multi-branches [16]
- Mise à jour des dépendances



Outils :

- Autoconf, CPAN, Maven, npm, pypy, conan.io, ...
- apt/rpm, pkgsrc, chocolatey, homebrew, ...
- Docker registry, Actifactory, Nexus, Archiva

[12] http ://en.wikipedia.org/wiki/Software_build
[13] http ://en.wikipedia.org/wiki/Build_automation
[14] https ://projet-plume.org/files/ENVOL2010-ConstructionApplications_0.pdf
[15] https ://en.wikipedia.org/wiki/Dependency_hell
[16] Type git-flow ou "features branching"

M. Andujar/R. Duvignau
Ateliers A5,A7

Tester pour ne plus avoir peur du code :

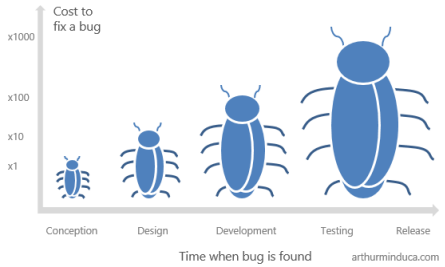
- Avoir plus de certitudes sur le code (doc)
- Etre capable de le faire évoluer (refactoring)
- Eléments nécessaires suivant les projets :
  - Projets communautaires
  - Projets dispositifs médicaux(ISO13485, FDA/CE), CMMi, ou +
- CleanCode [17] : xUnit, Selenium, SoapUI, ...
- Vision/Couverture : sonar, squash, JMeter, ...

[17] Software Craftsmanship
Image from https://arthurminduca.files.wordpress.com/2014/03/bugs.png
Image from https://www.yeast-id.org/wp-content/uploads/2016/03/debug.png

M. Andujar/R. Duvignau
Ateliers A5,A7

Tester pour ne plus avoir peur du code :

- Avoir plus de certitudes sur le code (doc)
- Etre capable de le faire évoluer (refactoring)
- Eléments nécessaires suivant les projets :
  - Projets communautaires
  - Projets dispositifs médicaux(ISO13485, FDA/CE), CMMi, ou +
- CleanCode [17] : xUnit, Selenium, SoapUI, ...
- Vision/Couverture : sonar, squash, JMeter, ...



A la recheche du bug perdu :

- Debogage : gdb, valgrind (memcheck), VTune, ...
- Analyse : gprof, Intel Advisor, Linux Trace, memprof, ...

[17] Software Craftsmanship
Image from https://arthurminduca.files.wordpress.com/2014/03/bugs.png
Image from https://www.yeast-id.org/wp-content/uploads/2016/03/debug.png

Alexandre Ancel
Présentation/Atelier AP3

[D]VCS, un élément clé dans la réflexion :

- Manipulation de fichiers 'texte' [20] , donc gestion réaliste possible
- Fichiers légers pouvant utiliser les fonctions des DVCS (branches, staging, ..)
- Bonnes pratiques compatibles (tout en texte, tout est suivi, ...)
- Intégration continue dans les 2 mondes (devops) et entre les outils

[20] Possibilités binaire du côté de git-lfs ou hg-largefiles

Alexandre Ancel
Présentation/Atelier AP3

[D]VCS, un élément clé dans la réflexion :

- Manipulation de fichiers 'texte' [20] , donc gestion réaliste possible
- Fichiers légers pouvant utiliser les fonctions des DVCS (branches, staging, ..)
- Bonnes pratiques compatibles (tout en texte, tout est suivi, ...)
- Intégration continue dans les 2 mondes (devops) et entre les outils

Des artéfacts intéressants :

- Versions journalières ("nightly build")
- Déploiements automatiques/continus
- Génération auto. de l'historique (release notes)

Les leaders :

- git, TFS, mercurial, svn, Bazaar, Fossil, ...

[20] Possibilités binaire du côté de git-lfs ou hg-largefiles

Rien de bien nouveau :

- Hébergement des dépôts, de quelques artefacts, forums, listes, suivi des bugs, gestionnaire de tâches, ...
- En mode public : Sourceforge, Savannah, ...
- En mode hébergé : Trac, Redmine, Tuleap, FusionForge, Sourcesup ...

Image from https://upload.wikimedia.org/wikipedia/commons/thumb/1/18/GitLab_Logo.svg/2000px-GitLab_Logo.svg.png
Image from https://cdn.meme.am/instances/500x/62024625/please-guy-please-merge-my-pull-request.jpg

Claire Mouton
Présentation/Atelier A1/GT1

Rien de bien nouveau :

- Hébergement des dépôts, de quelques artefacts, forums, listes, suivi des bugs, gestionnaire de tâches, ...
- En mode public : Sourceforge, Savannah, ...
- En mode hébergé : Trac, Redmine, Tuleap, FusionForge, Sourcesup ...

Et pourtant, une nouvelle vague :

- En mode public : GitHub, Bitbucket, GitLab, ...
- En mode hébergé : GitLab, Rhodecode, Gogs, ...
- Réseau social, discussions autour des problèmes, PR/MR, revue de code, hébergement de sites statiques, ...

Image from https://upload.wikimedia.org/wikipedia/commons/thumb/1/18/GitLab_Logo.svg/2000px-GitLab_Logo.svg.png
Image from https://cdn.meme.am/instances/500x/62024625/please-guy-please-merge-my-pull-request.jpg

Intégration continue [23] [24] :

Frédéric Woelffel
Présentation/Atelier AP2/A4

- Ensemble de pratiques vérifiant de manière automatique et régulière que chaque modification de code n'engendre pas de régression

Les leaders :

- Jenkins, Travis-ci, Gitlab-ci, CDash, ...

[23] http://martinfowler.com/articles/continuousIntegration.html
[24] http://fr.wikipedia.org/wiki/Int%C3%A9gration_continue
[25] http://static.googleusercontent.com/media/research.google.com/de//pubs/archive/42184.pdf
Image from http://vichargrave.com/wp-content/uploads/2013/02/Hadoop-Development.png
[27] http://www.sciencemag.org/content/334/6060/1226

Intégration continue [23] [24] :

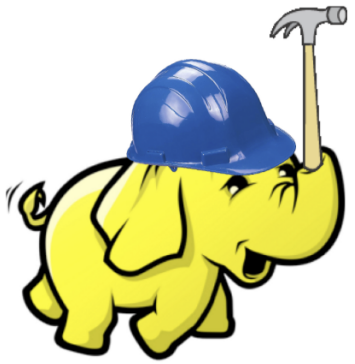
Frédéric Woelffel
Présentation/Atelier AP2/A4

- Ensemble de pratiques vérifiant de manière automatique et régulière que chaque modification de code n'engendre pas de régression

Les leaders :

- Jenkins, Travis-ci, Gitlab-ci, CDash, ...

Quelles difficultés avec l'intégration continue :



- se battre pour le 1er job est motivant, après ... [25]
- évolutions OS dans baremetal/VM ?
- comment tester l'évolution des esclaves ?
- gérer la combinatoire croissante des jobs ?

Principes de l'infrastruture définie par le code et de l'infrastructure reproductible correcte - même approche que la recherche reproductible [27]

[23] http://martinfowler.com/articles/continuousIntegration.html
[24] http://fr.wikipedia.org/wiki/Int%C3%A9gration_continue
[25] http://static.googleusercontent.com/media/research.google.com/de//pubs/archive/42184.pdf
Image from http://vichargrave.com/wp-content/uploads/2013/02/Hadoop-Development.png
[27] http://www.sciencemag.org/content/334/6060/1226

Les mêmes idées que la virtualisation, mais sans virtualisation :

- Agnostique sur le contenu et le transporteur
- Isolation et automatisation
- Principe d'infrastructure consistante et répétable
- Peu d'overhead par rapport à une VM ! [28]

En gros, un super chroot (ou un jails BSD plus sympa) : un des points forts de Solaris depuis plusieurs années. Techno existante aussi chez Google depuis longtemps. Rien de neuf, mais pourtant ...

Certains parlent de virtualisation "niveau OS" ou "légère", isolation applicative

[28] http://fr.slideshare.net/BodenRussell/kvm-and-docker-lxc-benchmarking-with-openstack
Image from https://msopentech.com/wp-content/uploads/DockerUbuntu.png

Pour la livraison continue [30] :

Maurice Poncet
Présentation

- les artefacts de l'intégration continue sont envoyés vers un environnement de test ou de préparation hors production,
- des tests sont réalisés
- à la dernière étape, le développeur approuve.

[30] techtrends xebia devops

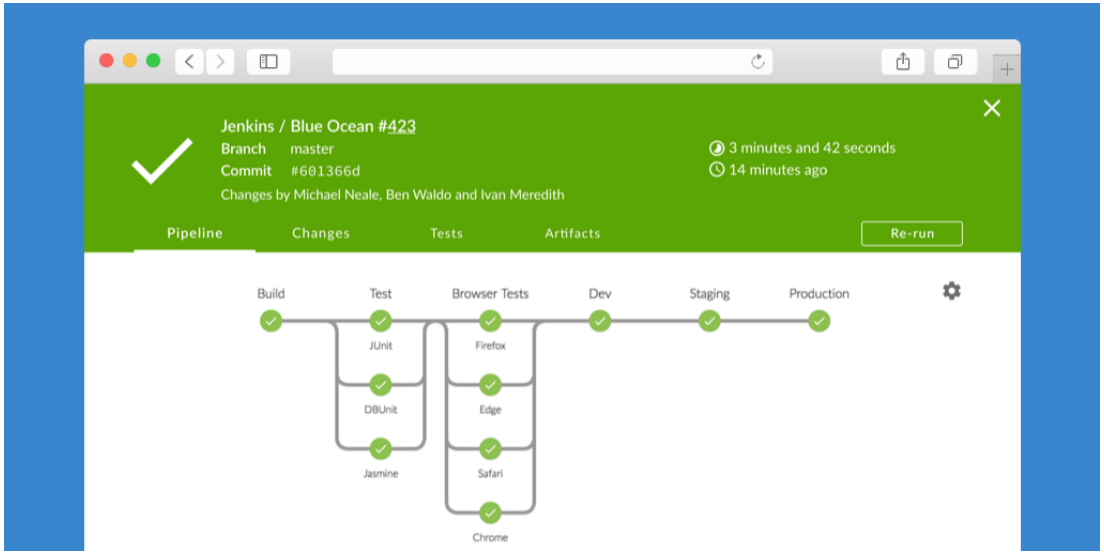
Pour la livraison continue [30] :

Maurice Poncet
Présentation

- les artefacts de l'intégration continue sont envoyés vers un environnement de test ou de préparation hors production,
- des tests sont réalisés
- à la dernière étape, le développeur approuve.

Pour le déploiement continu :

- mise en production auto. sans nécessiter d'approbation explicite.



[30] techtrends xebia devops

Sur des architectures complexes, les tests sont proches de la production :

- déploiement multi-nodes,
- contraintes de répartition,
- changements de topologie des noeuds,
- réseaux privés
- gestion native des secrets,
- autoscaling de conteneurs,...

Les leaders :

- Docker Swarm, Kubernetes, Rancher, Mesos, Marathon,Nomad, Fleet, ...

Tests de charge/performances réelles :

- Gatling
- AppDynamics, NewRelics

Nombreuses solutions/techniques :

Johan Moreau
GT5

- Images disques (PXE, Ghost, ...)
- Scripts maisons (Python, Bash, PowerShell, ...)
- Logiciels de gestion de conf (CFEngine, Puppet, Chef, Ansible, ...)
- Logiciels orienté DC (fabric, terraform, ...)

[32] http://ryandlane.com/blog/2014/08/04/moving-away-from-puppet-saltstack-or-ansible/
[33] https://missingm.co/2013/06/ansible-and-salt-a-detailed-comparison/
[34] http://blog.xebia.fr/2014/07/18/test-driven-infrastructure-avec-chef-episode-2/
[35] http://jensrantil.github.io/salt-vs-ansible.html
[36] http://docs.saltstack.com/en/latest/topics/installation/ubuntu.html
[37] http://saltstarters.org
[38] http://docs.saltstack.com/en/latest/topics/tutorials/quickstart.html

Johan Moreau
GT5

Nombreuses solutions/techniques :

- Images disques (PXE, Ghost, ...)
- Scripts maisons (Python, Bash, PowerShell, ...)
- Logiciels de gestion de conf (CFEngine, Puppet, Chef, Ansible, ...)
- Logiciels orienté DC (fabric, terraform, ...)

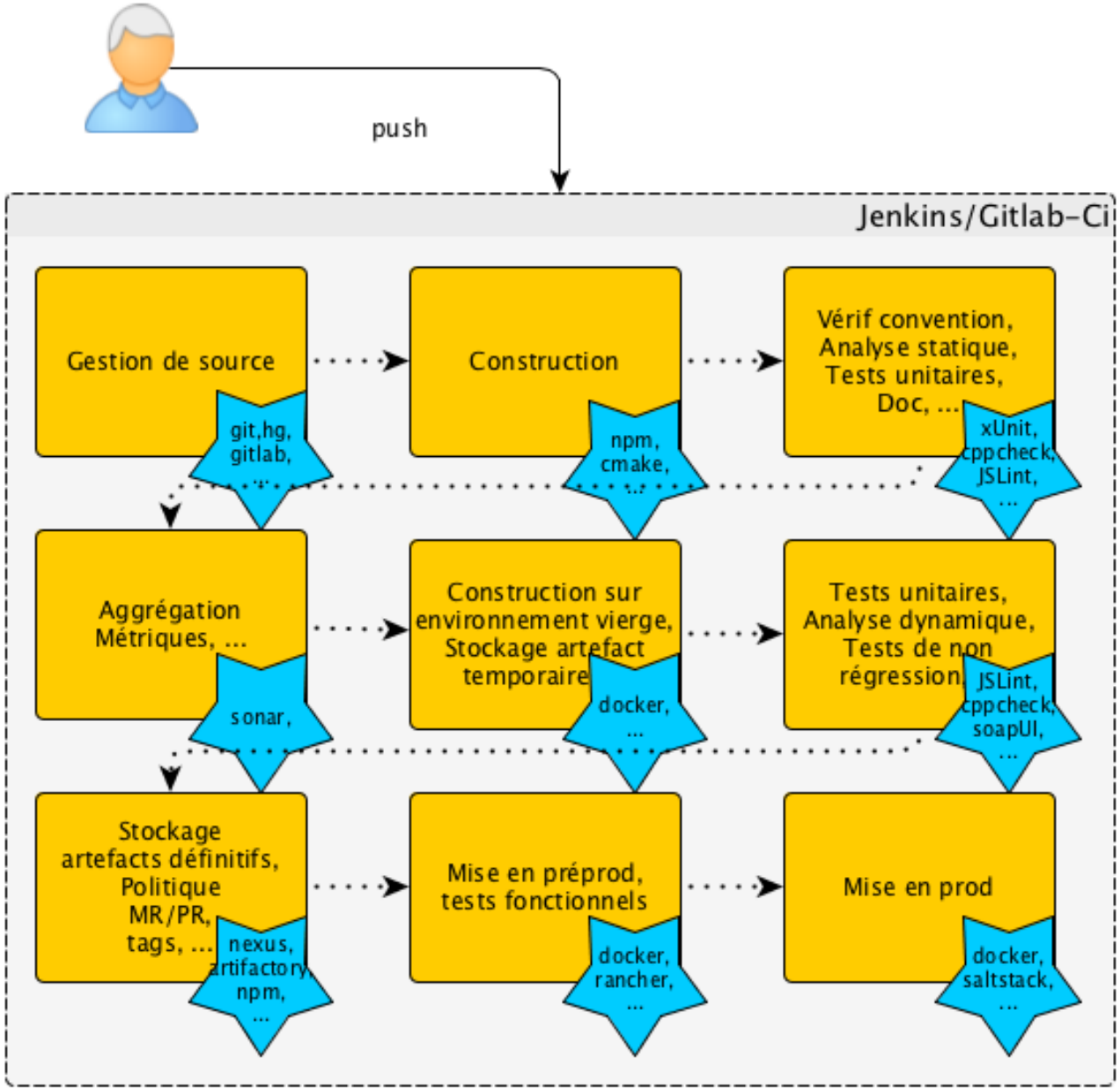
Le choix de Salt pour nous :

- Nouvelle génération d'outil (comparable à Ansible [32] [33] [34] [35])
- Multiplateforme (Linux, OSX, Windows), installation simple [36]
- Choix technologique : YAML, jinja2, 0MQ, AES256, Python
- Communauté importante [37] et bonne documentation [38]

---

[32] http://ryandlane.com/blog/2014/08/04/moving-away-from-puppet-saltstack-or-ansible/
[33] https://missingm.co/2013/06/ansible-and-salt-a-detailed-comparison/
[34] http://blog.xebia.fr/2014/07/18/test-driven-infrastructure-avec-chef-episode-2/
[35] http://jensrantil.github.io/salt-vs-ansible.html
[36] http://docs.saltstack.com/en/latest/topics/installation/ubuntu.html
[37] http://saltstarters.org
[38] http://docs.saltstack.com/en/latest/topics/tutorials/quickstart.html

Le schéma global



Johan Moreau
GT4

Les fondamentaux restent le nerf de la guerre :

- Processus de développement réfléchi et adapté (sprint, revue de code, ...)
- Gestion de l'arbre de sources et des artefacts ("Feature Branching", MR/PR, ...)
- Système de construction robuste

Johan Moreau
GT4

Les fondamentaux restent le nerf de la guerre :

- Processus de développement réfléchi et adapté (sprint, revue de code, ...)
- Gestion de l'arbre de sources et des artefacts ("Feature Branching", MR/PR, ...)
- Système de construction robuste

La suite (déjà là) :

- Sémantique dans le changelog (et utilisation de template) pour piloter l'arbre de source
- Règles intelligentes entre intégration continue et livraison continue
- Déploiement continu maîtrisé (supervision applicative pilotant le déploiement)

MERCI !

Questions ?

N'hésitez pas à m'envoyer vos remarques ou corrections sur johan.moreau sur gmail.com