INGENIERIE LOGICIELLE

COMMENT CONSTRUIRE UN LOGICIEL ?

François ANDRE

Fonctionnalités à mettre en place

1. Configurer le poste de développement
2. Gérer les dépendances
3. Organiser le code
4. Editer / Compiler le code
5. Collaborer
6. Tester / Valider
7. Construire / Livrer
8. Installer
9. Gérer le projet

Remarque préliminaire

Orienté Java … mais pas que

Les pratiques présentées ici sont majoritairement issues du monde Java.
Toutefois:

- Certaines sont agnostiques (Ex Git)
- Certaines ont été transposées d'un écosystème à l'autre

CONFIGURER LE POSTE DE DEVELOPPEMENT

Objectif: standardiser les postes de développements avec tous leurs composants (SDK, éditeur, SGBD,...) afin

- de simplifier la préparation des postes pour un stagiaire, un nouvel arrivant sur le projet.
- de basculer d'un environnement de projet à l'autre

CONFIGURER LE POSTE DE DEV.

CLONEZILLA

Réaliser des images standards des différents types de postes afin de permettre une remise à zéro des postes rapides

Avantages :
- Performances

Inconvénients :
- Peu évolutif
- Peu portable (OS, machine)
- Peu partageable (au sein d'une communauté)

VMWARE

Réaliser des images sous forme de machine virtuelles

Avantages :
- Possibilité d'avoir des OS différents
- Bascules entre environnements

Inconvénients :
- Performances

DOCKER

Réaliser des images sous forme de micro containers

Avantages :
- Bascules entre les environnements rapide
- Partageable (communauté)
- Proche de la production

Inconvénients :
- Linux, Windows X Pro ou cloud.

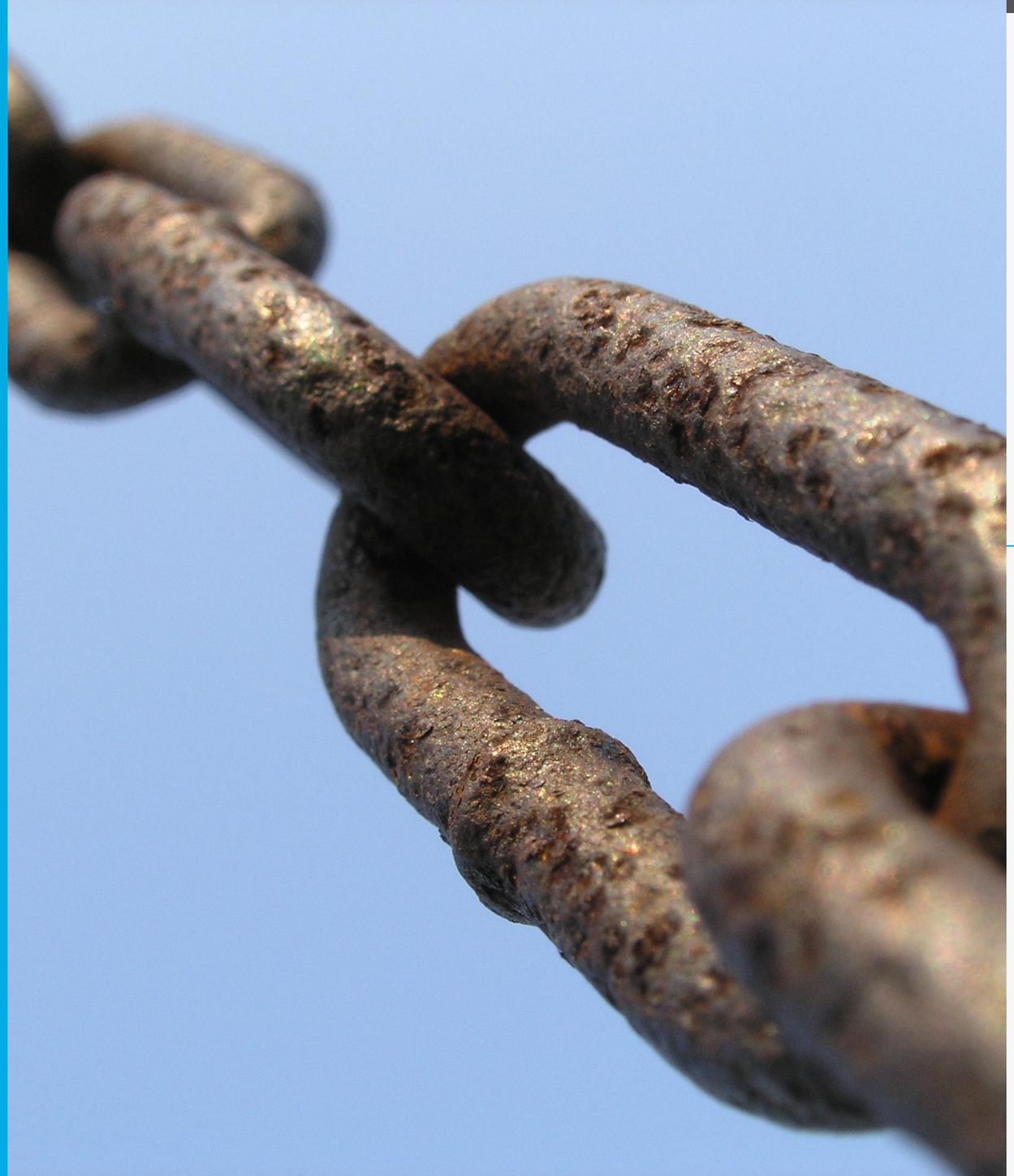
https://www.grafikart.fr/tutoriels/docker-stack-web-635

OUTILS

CLONEZILLA

VMWARE

DOCKER



GERER LES DEPENDANCES

Objectif: Les logiciels dépendent d'autres librairies dans des versions précises. Celles-ci dépendent ont également des dépendances. Dès qu'un projet atteint une certaine complexité l'outillage de la gestion des dépendances est nécessaire.

GERER LES DEPENDANCES

X.Y.Z-SNAPHOT

Les différents développements sont étiquetées avec un numéro de version composé

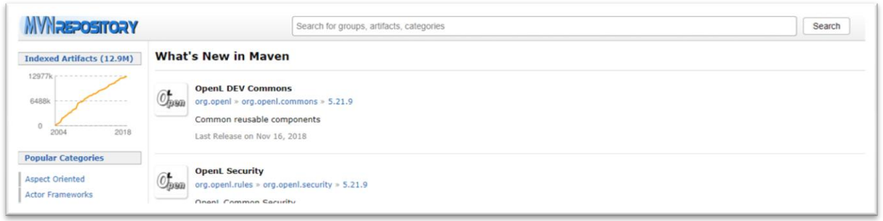
- **X** : numéro de version **majeur**
- **Y** : numéro de version **mineur**
- **Z** : numéro de **correction**
- **SNAPSHOT** : **version de travail** dont la cible est X.Y.Z (qui est la *RELEASE*)

MAVEN

Maven est un outil Java permettant entre autre la gestion des dépendances.

Il repose sur des dépôt (*repository*) de librairies Java mis en place aux niveaux suivants :

- **Poste de développement**
- **Entreprise**
- **Internet**



OUTILS

MAVEN

OUTILS SIMILAIRES

COMPOSER (PHP)

BOWER (JavaScript)

NPM (Node.js)

ORGANISER LE CODE

Objectif: Le code doit être physiquement structuré de manière claire afin que chaque participant puisse rapidement savoir où intervenir

"Any fool can write code that a computer can understand.
Good programmers write code that humans can understand."

Martin Fowler

ORGANISER LE CODE (1/2)

LIBRAIRIES

Les projets sont découpés en librairies reflétant leur rôle afin de permettre la mutualisation et la compréhension:

- *-commons
- *-core
- *-webapp
- ...

REPERTOIRES

Maven impose une organisation sous forme de répertoires permettant de différencier

- le code principal
- le code de test
- les ressources (fichiers de configuration)
- ...

PACKAGES

Au sein de cette hiérarchie, Java structure le code le code sous forme de package.

Les frameworks précisent également cette classification.

Les nommages de certains packages sont unifiés

- *.client
- *.server
- *.domain
- *.view
- ...

DESIGN PATTERNS

L'utilisation de design patterns – notamment MVC et IOC – amène un découpage des responsabilités permettant évolution et maintenance.

ORGANISER LE CODE (2/2)

NORMES DE CODAGE

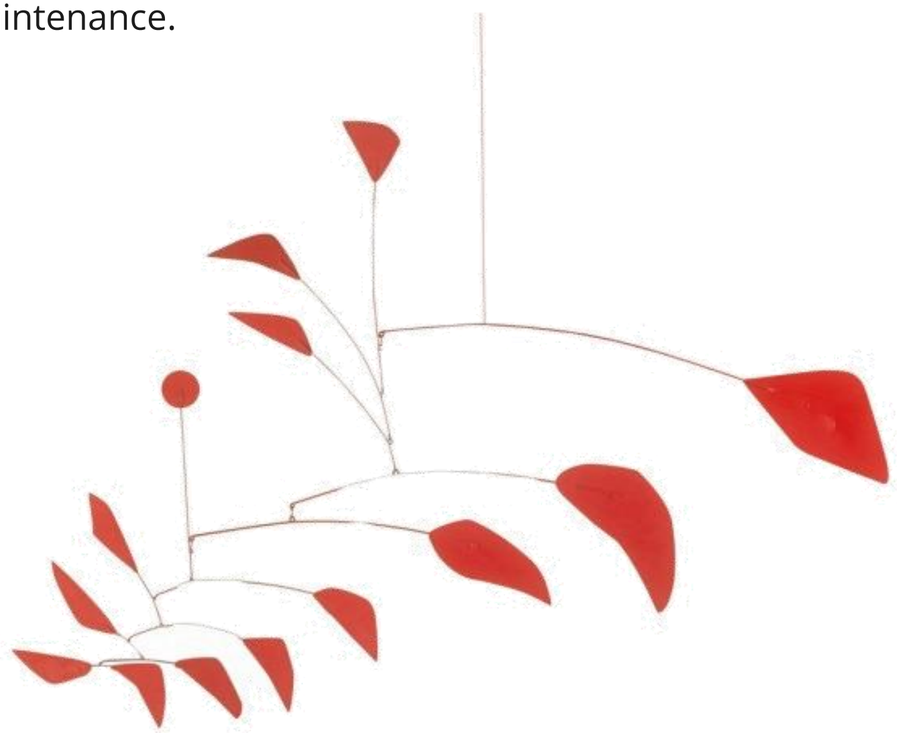
Les normes de codages définissent les standards pour les noms des classes, des champs et des méthodes.

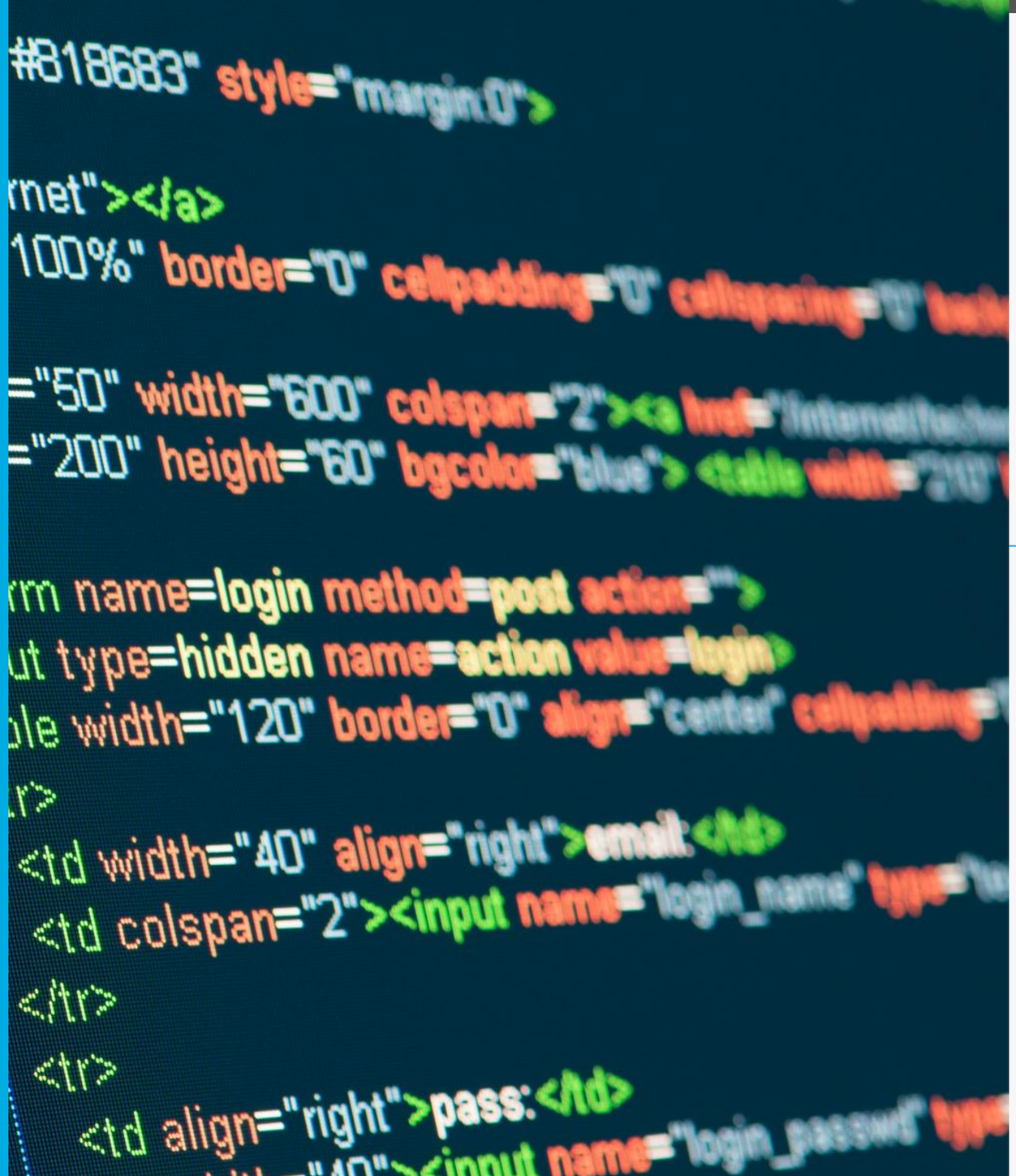
Ils facilitent la compréhension du code par des autres développeurs, des architectes, …

https://google.github.io/styleguide/javaguide.html

DESIGN PATTERNS

L'utilisation de design patterns – notamment MVC et IOC – amène un découpage des responsabilités permettant évolution et maintenance.





EDITER / COMPILER

Objectif: Utiliser au maximum une interface complète et extensible permettant de gérer efficacement l'édition du code

EDITER/COMPILER

IDE

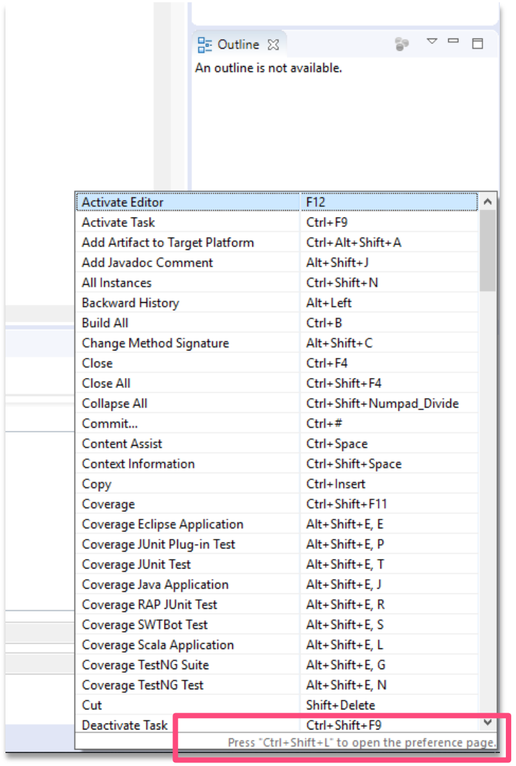
Les IDE par leur extensibilité et l'intégration d'outil permettent d'augmenter la productivité. (Ex: Plugin Git, Editeur spécialisé)

MUTUALISER

Utiliser un outil commun au sein de l'équipe permet une meilleure diffusion des connaissances.

OUTILS

ECLIPSE



CTRL+SHIFT+L

COLLABORER

Objectif: Pouvoir travailler à plusieurs – à l'intérieur ou à l'extérieur de l'équipe - sans compromettre le travail des autres développeurs.

COLLABORER

GIT

Git est un SCM qui permet de collaborer de manière très souple, par exemple centralisée ou non.

Il se distingue de ses prédécesseurs par une gestion efficace – c.à.d. rapide – des branches.

GITFLOW

GitFlow propose une surcouche d'organisation efficace basée sur les branches :

- Deux branches pérennes – master et develop – correspondent à la notion de snapshot et release.
- Chaque fonctionnalité est développée dans une branche temporaire puis fusionnée dans develop.
- Les livraisons sont effectuées à partir de master
- Les anomalies en productions sont corrigées sur des branches temporaires

Il est intégré aux principaux IDE.

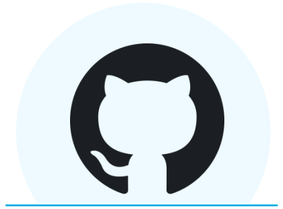
GITHUB/GITLAB

Les implémentations *sociales* ont mis en place des mécanismes complémentaires (*pull-request, merge-request*) permettant la modération de proposition de code.

https://blog.zenika.com/2017/01/24/pull-request-demystifiel

OUTILS

GIT



GITUB

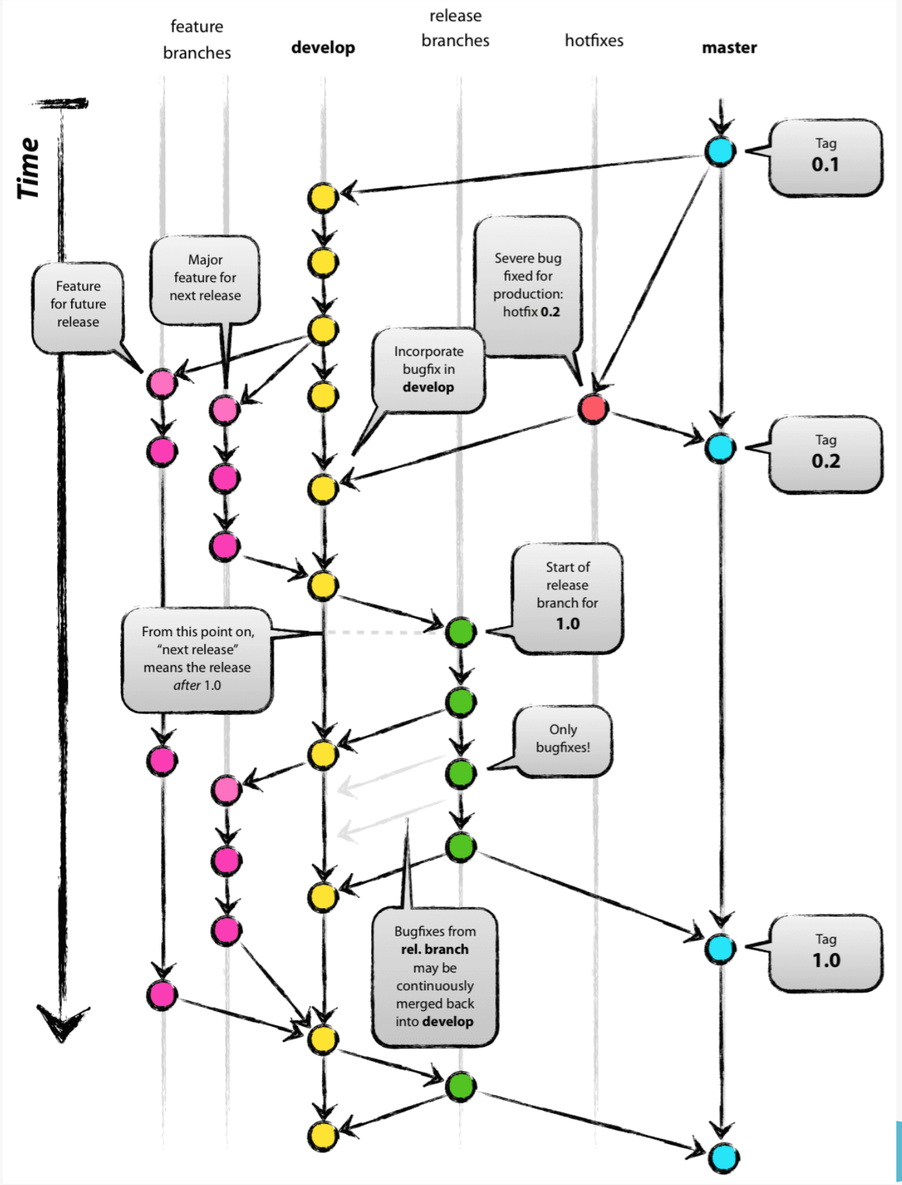
GITLAB

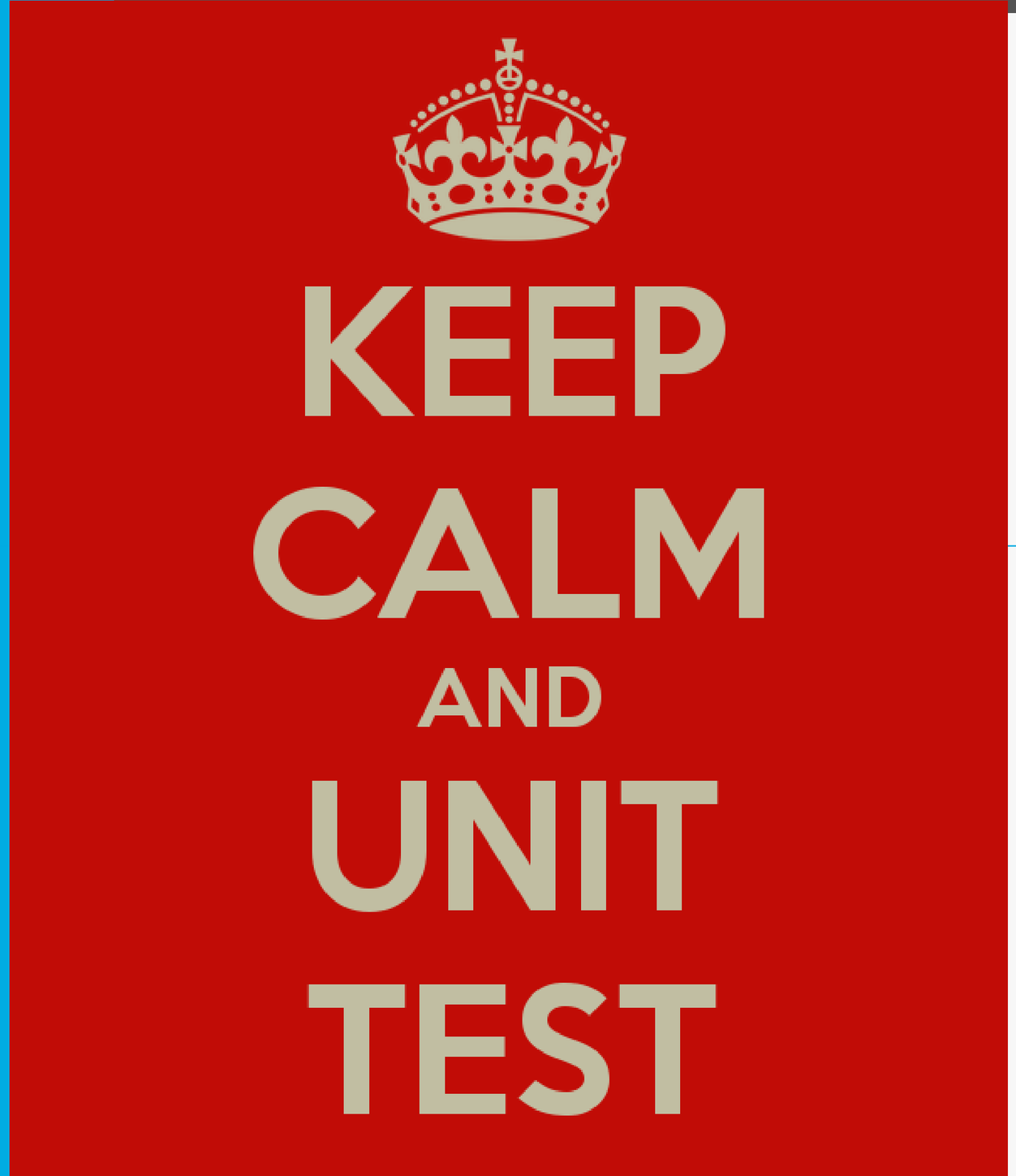
COLLABORER

GitFlow
https://nvie.com/posts/a-successful-git-branching-model/

GitHub
https://guides.github.com/introduction/flow/index.html

GitLab
https://docs.gitlab.com/ee/workflow/gitlab_flow.html





TESTER / VALIDER

Objectif: Automatiser un maximum de test afin de

- limiter les erreurs/régressions lors des livraisons/relivraisons de fonctionnalités de l'application
- limiter les campagnes de tests manuels

TESTER / VALIDER

TESTS UNITAIRES

Les livraisons itératives empêchent des campagnes manuelles. La mise en place de T.U. permet une meilleure structuration du code.

- **JUnit** : permet de définir simplement des tests
- **Maven** identifie explicitement les parties de code/ressources qui concernent les tests
- **Spring** : permet de définir des environnements différents pour chaque test.

BACKOFFICE FIRST

Le front office est toujours difficile à tester.

- **Selenium:** permet d'effectuer des tests de navigation WEB et de les intégrer également. Il est indépendant du langage sous-jacent.

Le back office se prête plus facilement aux tests.

- Il est nécessaire d'identifier des jeux de tests pertinents (volume, contenu,...).
- Chaque erreur détecter doit entrainer la mise en place d'au moins un test unitaire

AMELIORER

Sonar permet d'estimer la qualité du code et ainsi améliorer ses pratiques. Exemple: **duplication du code**, métriques, erreurs classiques...

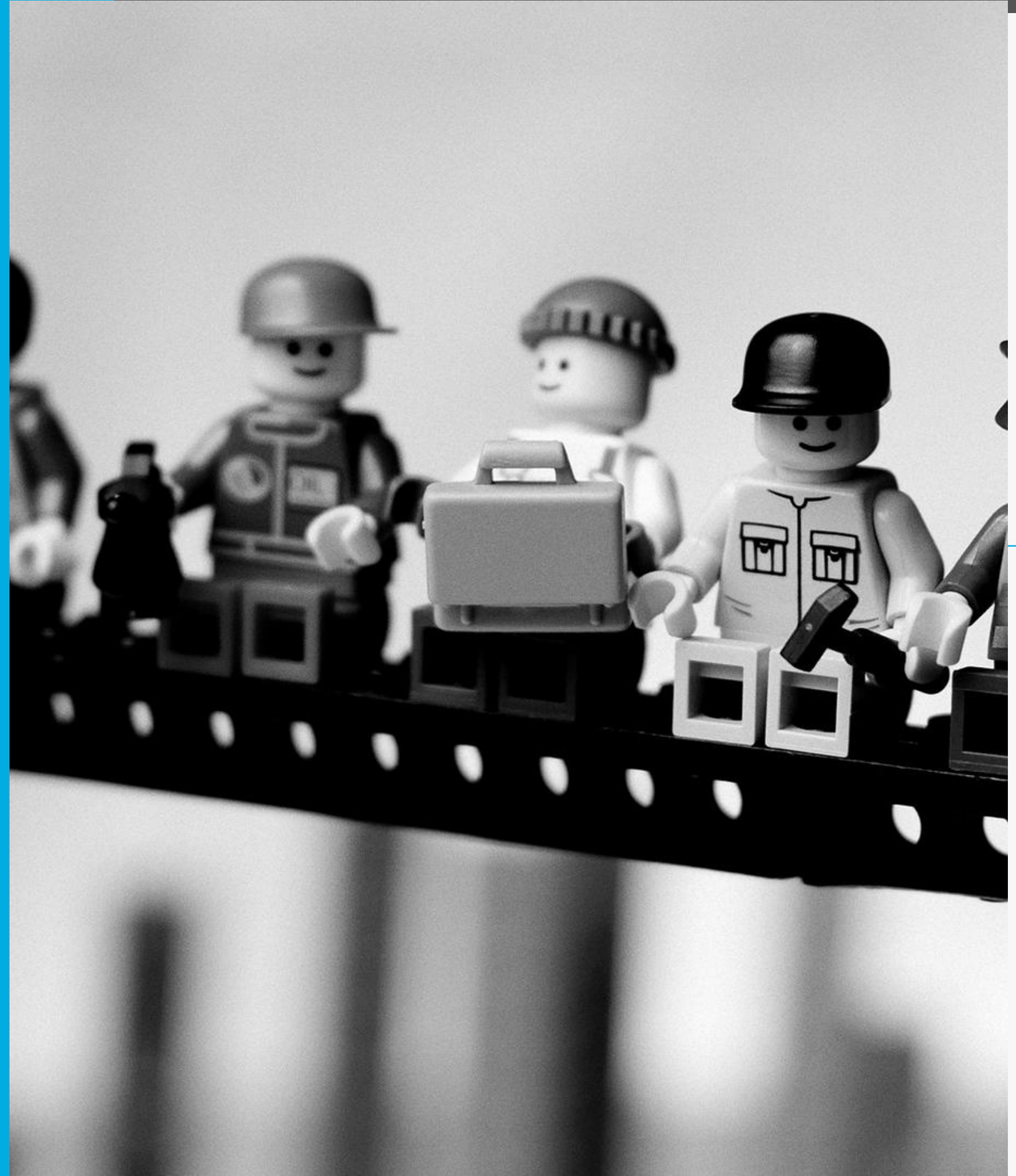
OUTILS

JUNIT

SELENIUM

SONAR

https://www.pluralsight.com/guides/getting-started-with-page-object-pattern-for-your-selenium-tests



CONSTRUIRE

Objectif: Pouvoir construire les livrables de manière fiable, régulière et indépendante du poste du développeur. La construction doit pouvoir cibler plusieurs environnements distincts

CONSTRUIRE / LIVRER

FIABILITE

Maven et Ant permet de gérer automatiquement des constructions complexes.

ENVIRONNEMENTS

Maven, Spring et les autres couches d'abastraction (JPA, utilisation des DAO…) permettent de définir des architectures différentes en fonction des environnements (production, recette) ou des tests en cours.

AUTOMATISATION

Jenkins permet de lancer la construction automatique des projets Maven lorsque le code est modifié sur le serveur Git.

Si les test unitaires sont validés, les librairies générées sont mises à disposition sur le dépôt d'entreprise.

OUTILS

MAVEN

JENKINS

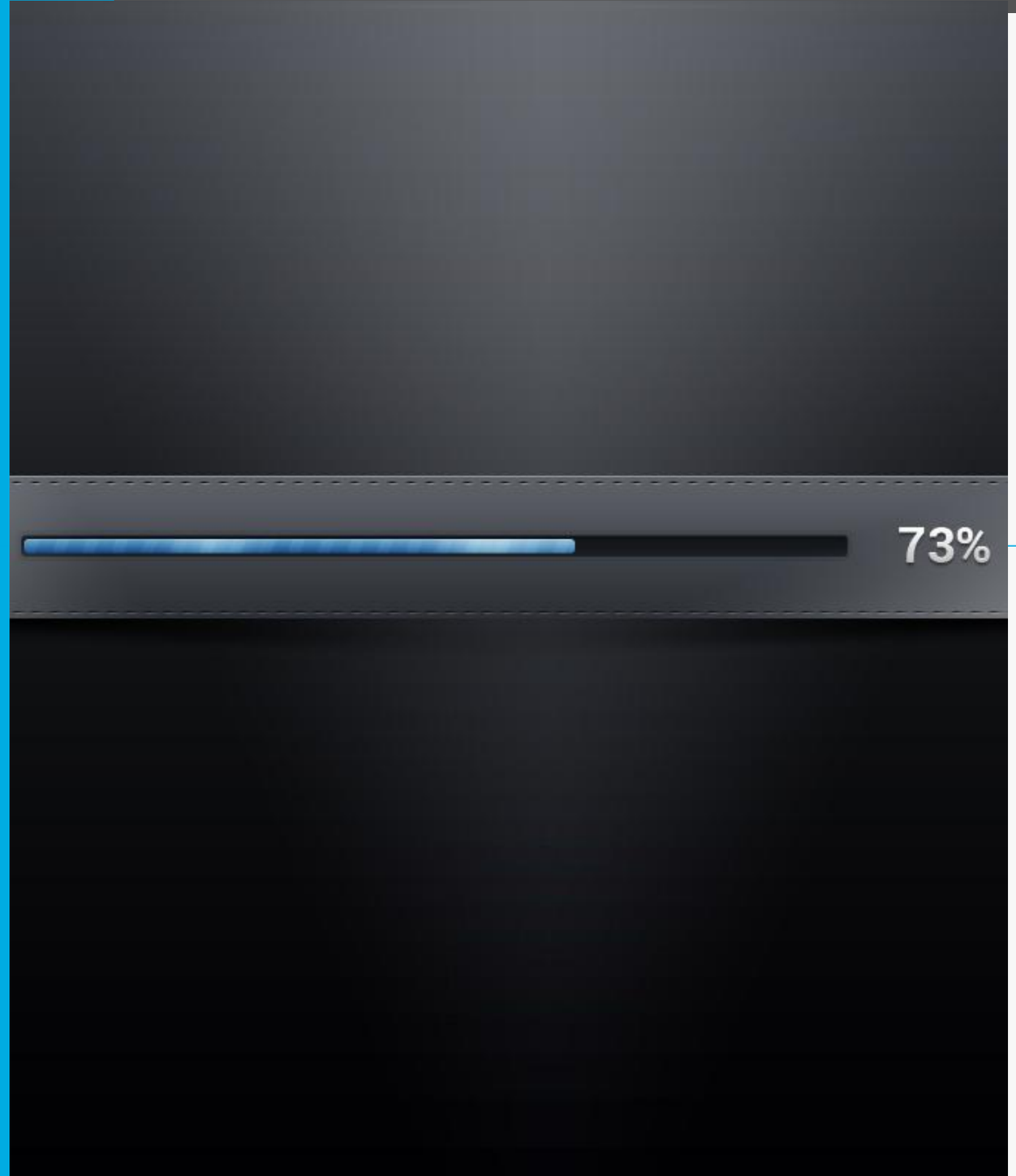
ANT

OUTILS SIMILAIRES

TRAVIS
(Github)

GRUNT
(JavaScript)
Mais il y aussi
Gulp, WebPack…

AWS Code
Pipeline



INSTALLER

Objectif: Simplifier la mise en place des applications dans leur environnement de production

INSTALLER

STANDARD

Pour les applications web JEE, la norme .war permet de simplifier l'installation et la réinstallation d'une application sur un serveur.

DEPLOIEMENT

Maven permet la mise en place automatique des applications en mode commande.

JPA permet la création automatiques des tables des bases de données.

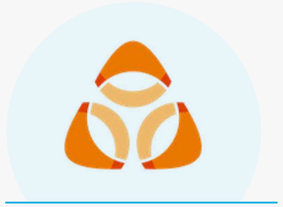
La librairie Liquibase permet l'exécution de scripts SQL au déploiement de l'aplication

QUE DEPLOYER ?

- Une nouvelle version d'une librairie
- Un application web Java
- Une application mobile
- Une application composée (incluant une base de données, ...): Microservice,...
- Tout un système d'information

Le choix de l'outil sera différent.

OUTILS

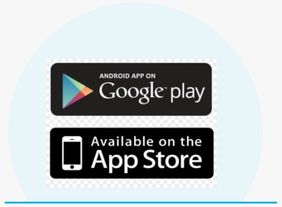


OSGI

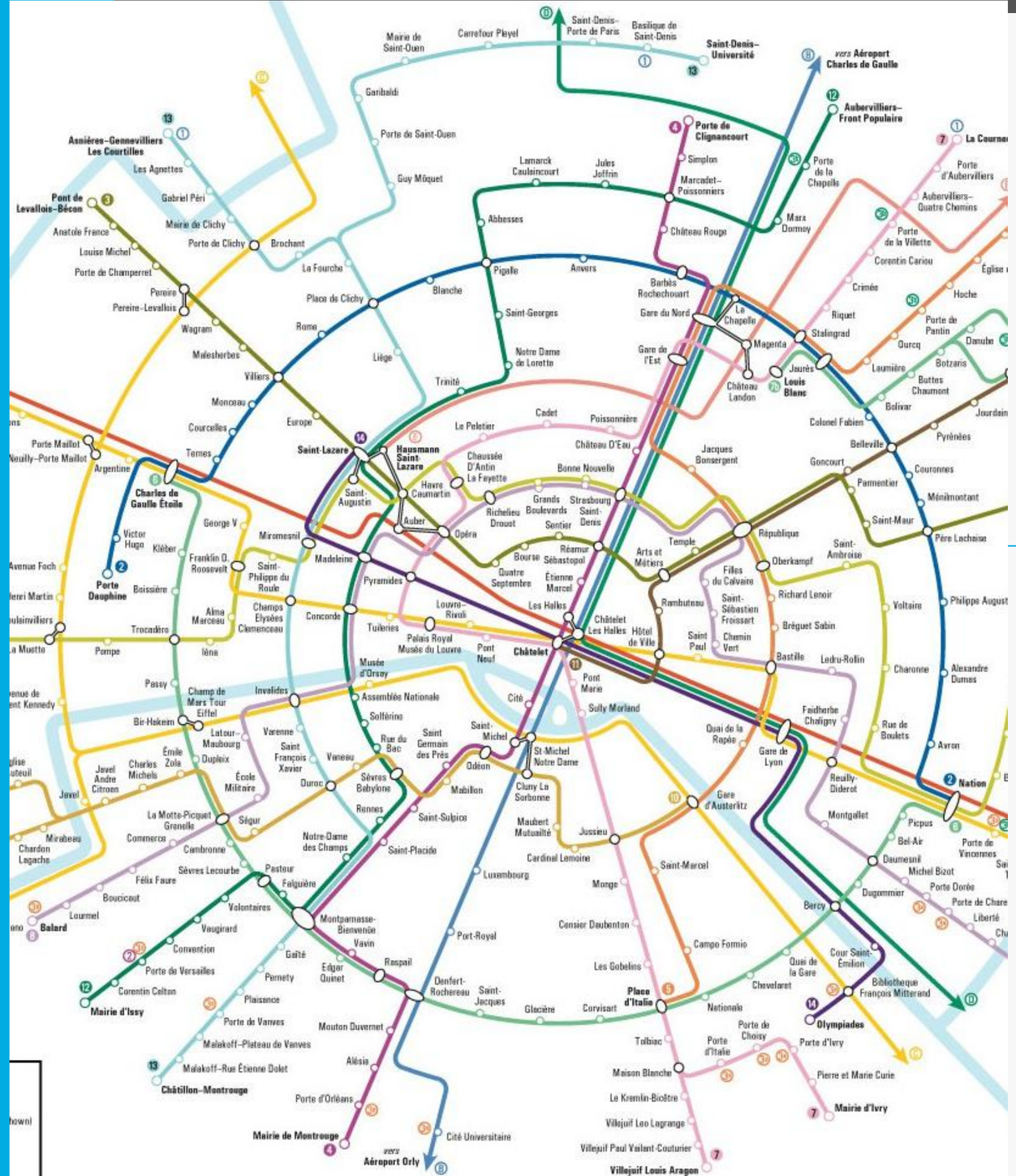
TOMCAT

DOCKER /
DOCKER COMPOSE

AWS
CLOUDFORMATION



APP STORE



GERER LE PROJET

Objectif: mettre en place les outils permettant de suivre le déroulement du développement ainsi que les anomalies rencontrées

GERER LE PROJET

KANBAN

Un kanban est un outil permettant de mettre en place un suivi simple des tâches à effectuer.

Il s'adapte aux pratiques de chaque projet mais est particulièrement adapté aux cycle itératifs.

www.youtube.com/watch?v=OCpK3nf0oZ4

ANOMALIES

Tout projet doit disposer d'un outil permettant de suivre les différentes anomalies détectées ou les demandes d'évolution proposées. A chaque ticket est associé un cycle de vie indiquant entre autre la version de correction.

L'outil doit être personnalisable pour s'adapter aux spécificités d'un projet

La plupart des forges en ligne disposent d'outils de ce genre.

OUTILS

TRELLO

MANTIS BT

Merci

Des questions?

• • •

Contenu – Francois ANDRE

Modèle PPT – Jun Akizaki (http://thepopp.com)