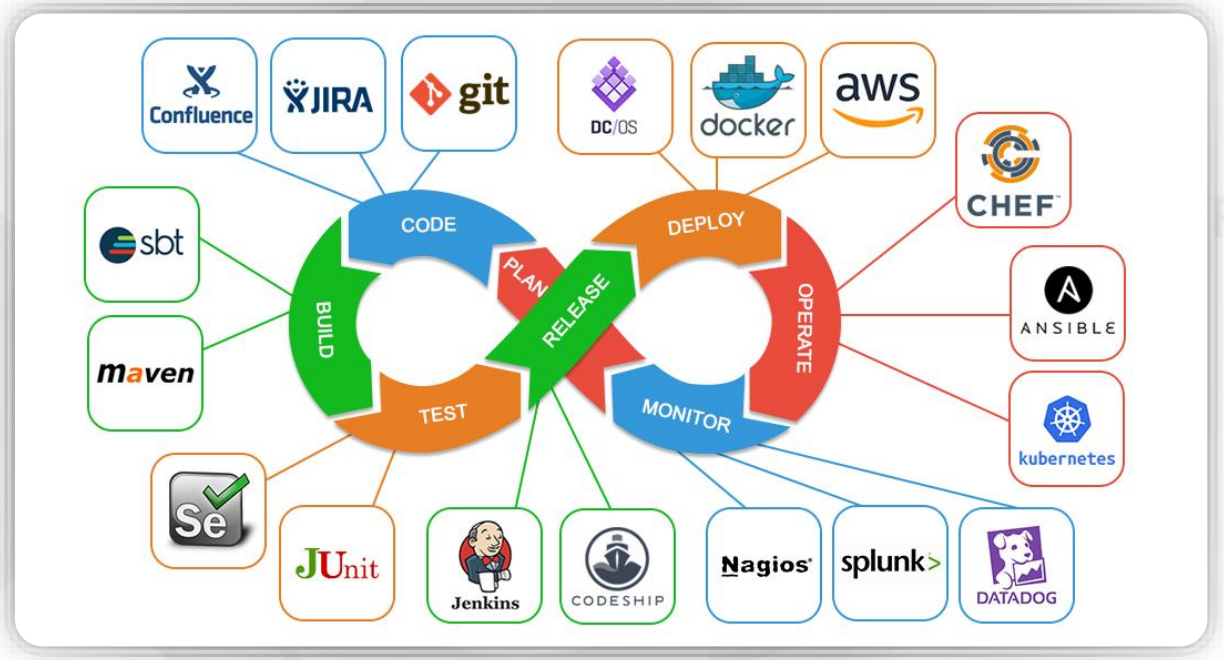
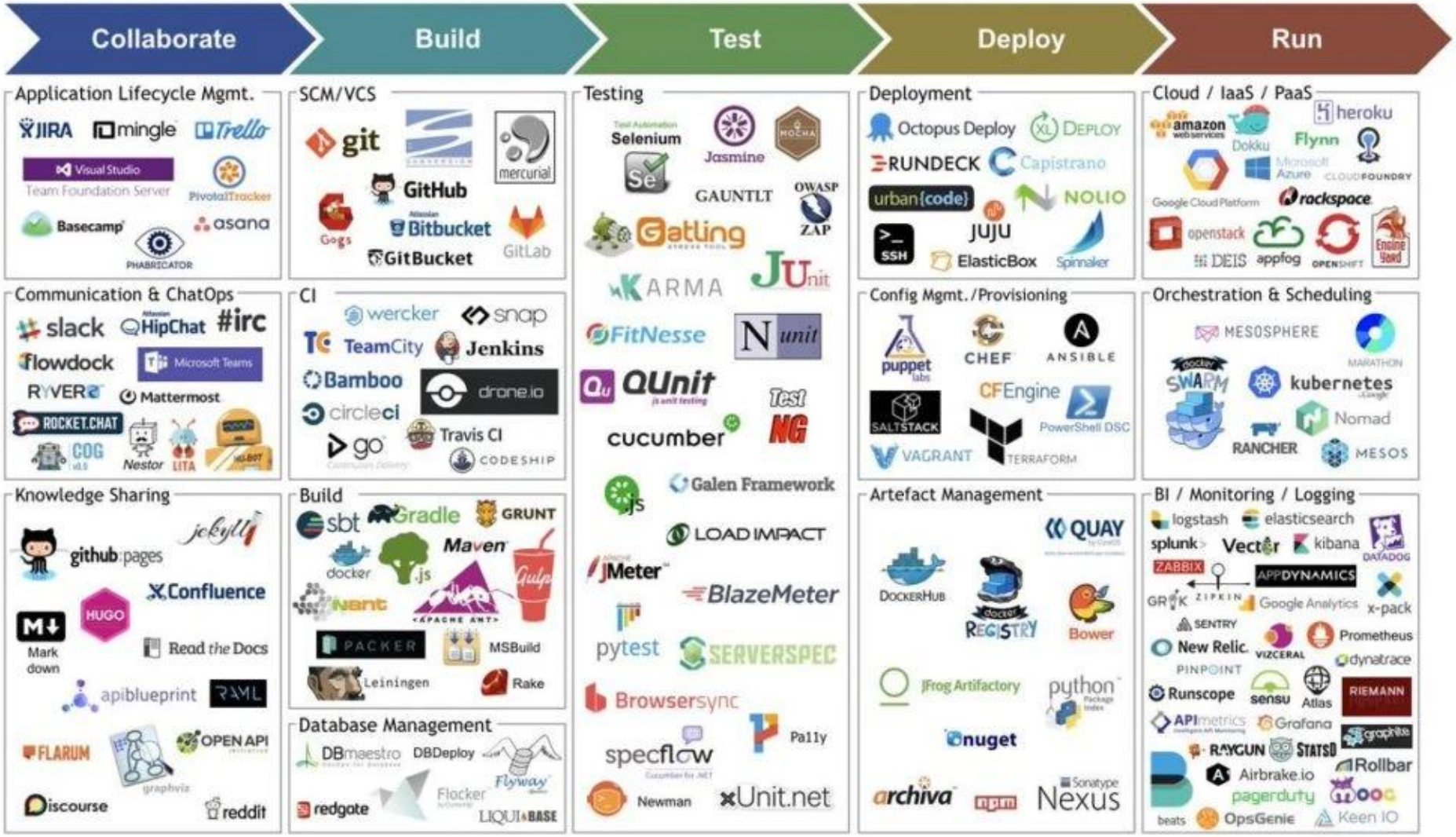
Afpa
se former, avancer

Intégration CONTINUE (CI)
Déploiement CONTINU (CD)

DE QUOI ALLONS-NOUS PARLER ?

- La notion de DEVOPS.
- CI/CD : Continuous Integration / Continuous Delivery
- Intégration continue / Continuous Integration (CI)
- Le Déploiement continu / Continuous Delivery (CD)
- Les environnements de déploiement.
- Plan de tests et Plan de déploiement
- les 12 facteurs
- Les outils à notre disposition
- Cloud Computing




Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
SCM/VCS
git
mercurial
GitHub
Bitbucket
Gogs
GitBucket
GitLab
Testing
Selenium
Jasmine
MOCHA
GAUNTLT
OWASP ZAP
Gatling
KARMA
JUnit
FitNesse
Nunit
QUnit
Test NG
cucumber
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
Pally
specflow
Newman
xUnit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
SSH
juju
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
heroku
Dokku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG
Nestor
LITA
HUBOT
CI
wercker
snap
TeamCity
Jenkins
Bamboo
drone.io
circleci
Travis CI
go
CODESHIP
Config Mgmt./Provisioning
puppet labs
CHEF
ANSIBLE
SALTSTACK
CFEngine
PowerShell DSC
VAGRANT
TERRAFORM
Orchestration & Scheduling
MESOSPHERE
MARATHON
docker SWARM
kubernetes
Nomad
RANCHER
MESOS
Knowledge Sharing
github pages
jekyll
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
OPEN API
graphviz
Discourse
reddit
Build
sbt
Gradle
GRUNT
docker
Maven
.js
Gulp
Nant
ANT
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE
Artefact Management
QUAY
DockerHub
docker REGISTRY
Bower
JFrog Artifactory
python
nuget
archiva
npm
Sonatype Nexus
BI / Monitoring / Logging
logstash
elasticsearch
splunk
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic
VIZCERAL
Prometheus
dynatrace
PINPOINT
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Airbrake.io
Rollbar
pagerduty
Moog
beats
OpsGenie
Keen IO

DEVOPS

LA NOTION DE DEVOPS

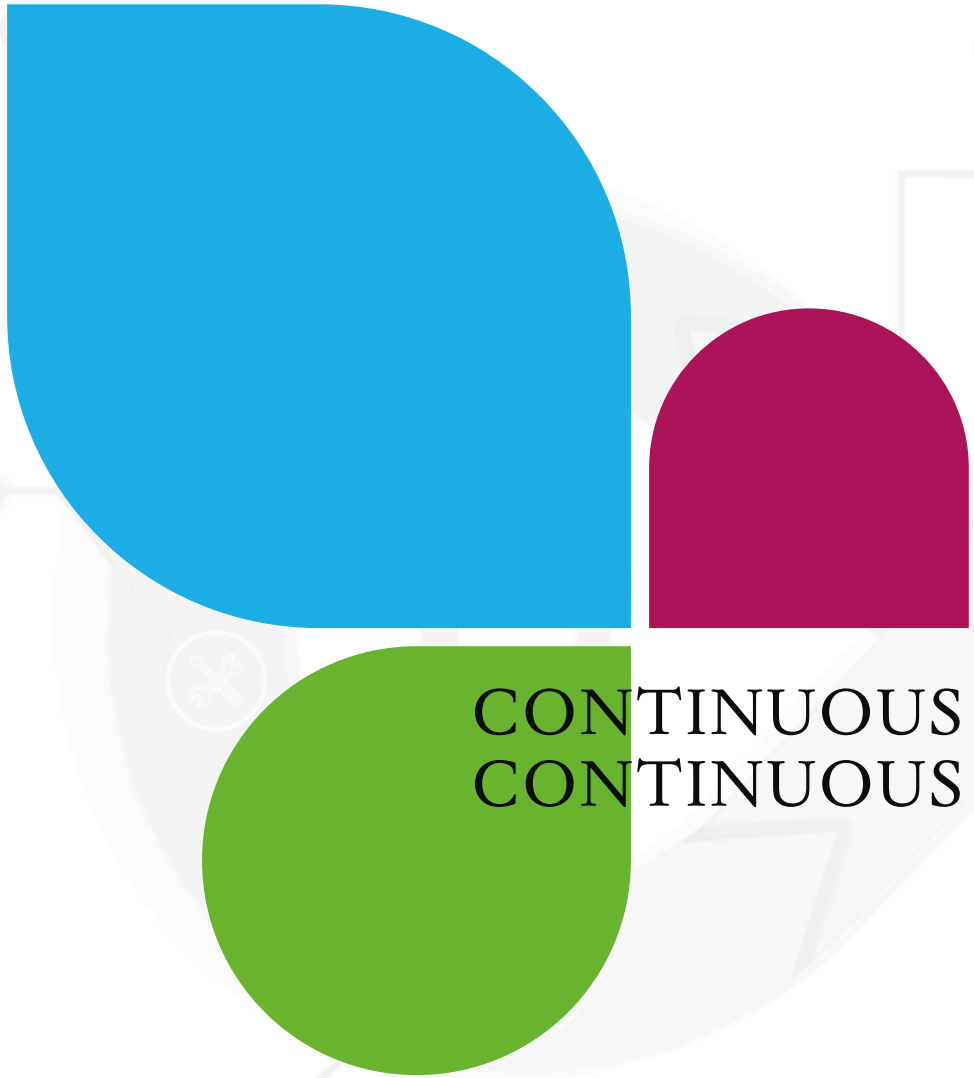
LA NOTION DE DEVOPS

DevOps : contraction de Développeur (Dev) et de Opérationnel (Ops) :

- Les Devs produisent de nouvelles fonctionnalités.
- Les Ops s'assurent de la fiabilité du site ou de l'application.

DevOps consiste à faire communiquer et collaborer les développeurs et les Ops sur les projets :

- Gagner du temps sur la résolution des problèmes.
- Lancer des nouvelles "features" plus rapidement.
- Réduire les risques grâce à l'automatisation des processus. (phases de tests par exemple).
- Augmenter la satisfaction de vos clients. (plus on livre plus il est satisfait).



CONTINUOUS INTEGRATION - CONTINUOUS DELIVERY CI/CD

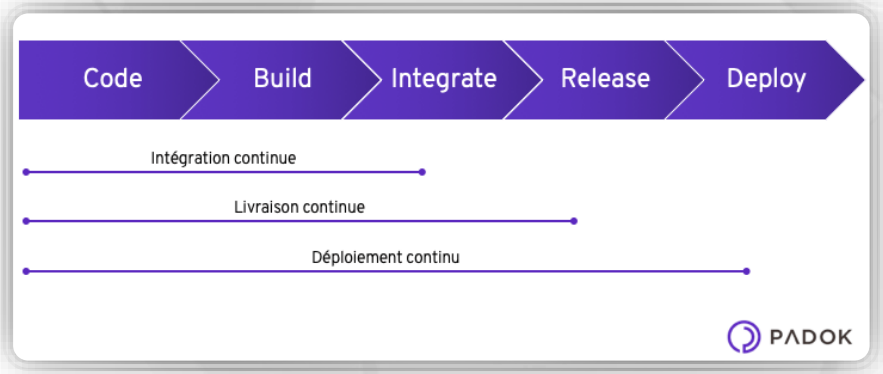
DÉFINITION

Le modèle CI/CD, qui désigne l'intégration et la distribution a pour objectif de rationaliser et d'accélérer le cycle de vie de développement des logiciels.

- La pratique de l'intégration continue (CI) consiste à intégrer automatiquement et régulièrement les modifications de code dans un référentiel de code source partagé.
- La distribution et/ou le déploiement continus (CD) désigne quant à eux un processus en deux volets qui englobe l'intégration, les tests et la distribution des modifications apportées au code.
- La distribution continue se limite au déploiement automatique dans les environnements de production, alors que le déploiement continu publie automatiquement les mises à jour dans ces environnements.

CONTINUOUS INTEGRATION CI

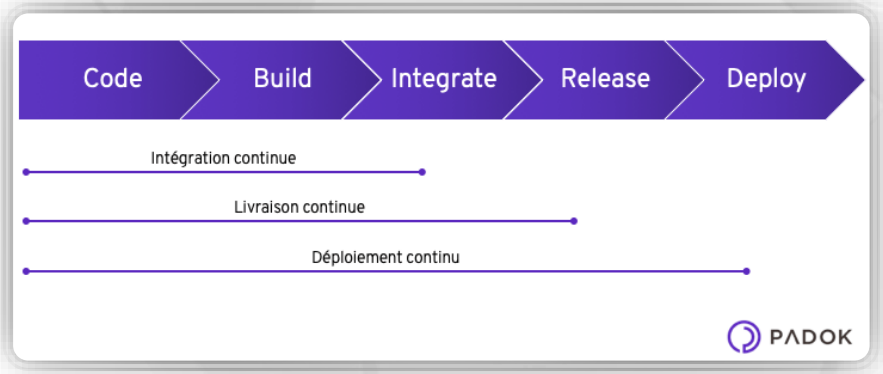
La phase CI pour Continuous Integration est la 1ère phase de la démarche CI/CD.



- Manière d'utiliser le concept de DevOps
- CI : Phases de tests automatisés intégrées au flux de déploiement.
  - Ensemble de pratiques en génie logicielle consistant :
    - Quotidiennement, mettre son code en commun à chaque nouvelle fonctionnalité
    - Ne pas simplement envoyer son code mais le TESTER avant toute mise en commun !!
    - Tests unitaires, tests d'intégrations et tests systèmes
    - Idéalement dans un environnement de tests au plus proche de l'environnement de production.

CONTINUOUS DELIVERY CD

La phase CD pour Continuous Delivery est la suite de la phase CI.



- Suite de l'intégration continue
- Déploiement continu :
  - Une fois les tests validés sur les environnements de DEV, SIT et UAT, cela consiste à automatiser les actions de déploiements.
  - Avantages :
    - Les déploiements sont moins fastidieux.
    - La plupart des tâches dont les tests sont automatisés.
    - Fréquence des mises en production sont augmentées.
- Livraison continue :
  - Toutes les étapes du déploiement continu sauf la mise en production
    - Permet de garder la main sur la dernière étape par les équipes IT et MOA.

LE DÉPLOIEMENT

MISE EN PRODUCTION

OBJECTIF ET ENJEUX

Objectif :

- Mettre en production l'application validée lors de la recette.

Enjeux :

- Mettre à disposition une solution stable et conforme à l'attendu.
- Accompagner les utilisateurs dans leur montée en compétence et dans l'appropriation de l'outil.

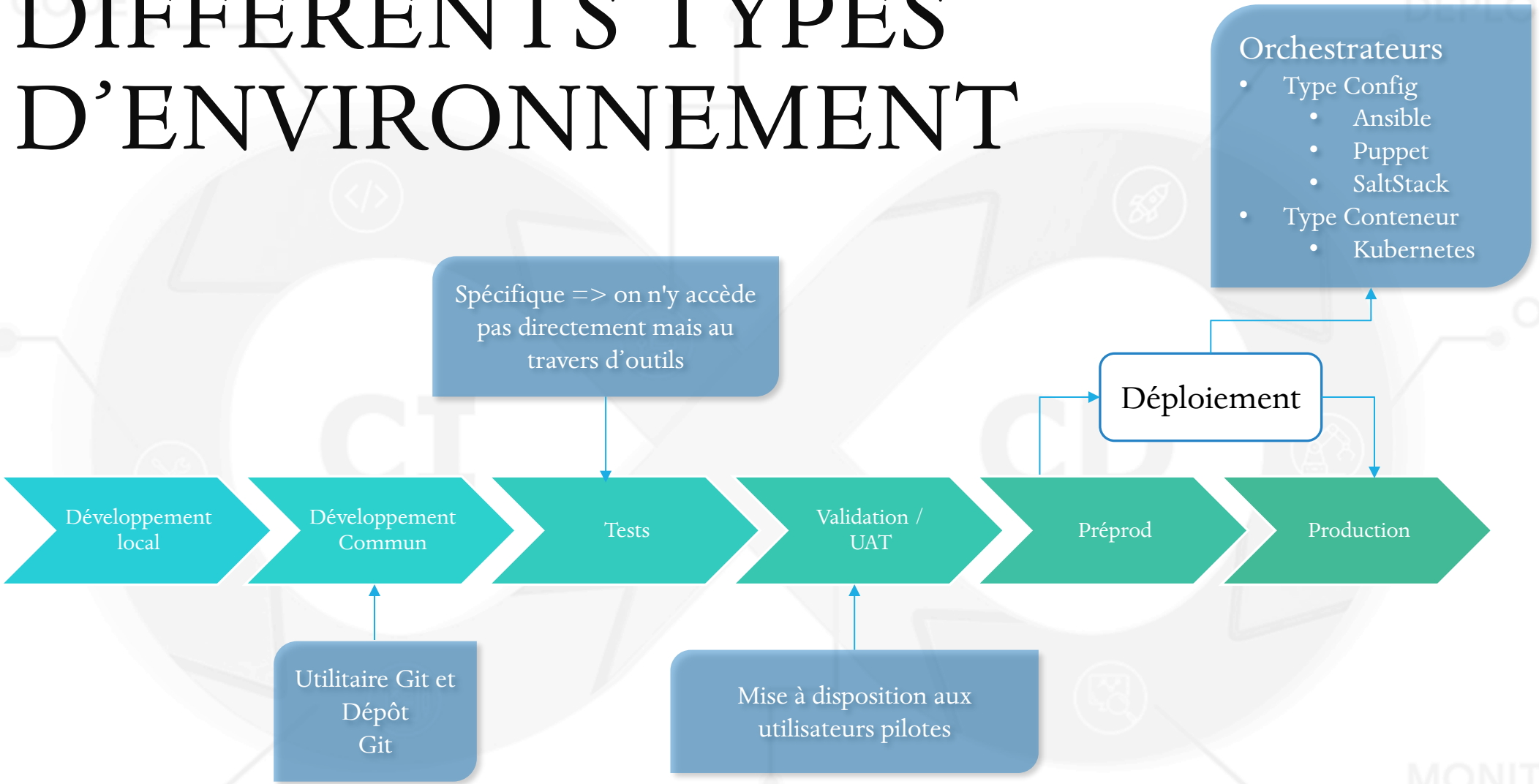
Quand parles-t-on de déploiement ?

- On parle de déploiement informatique pour la mise en œuvre d'un nouveau système, un logiciel.
- C'est également le déploiement d'un réseau d'entreprise, d'un serveur de messagerie, d'outils collaboratifs, ou d'une application métier.
- Si vous souhaitez déployer le dernier système/logiciel/progiciel pour avoir plus de fonctionnalité, votre projet s'inscrit dans le cadre d'un déploiement informatique.

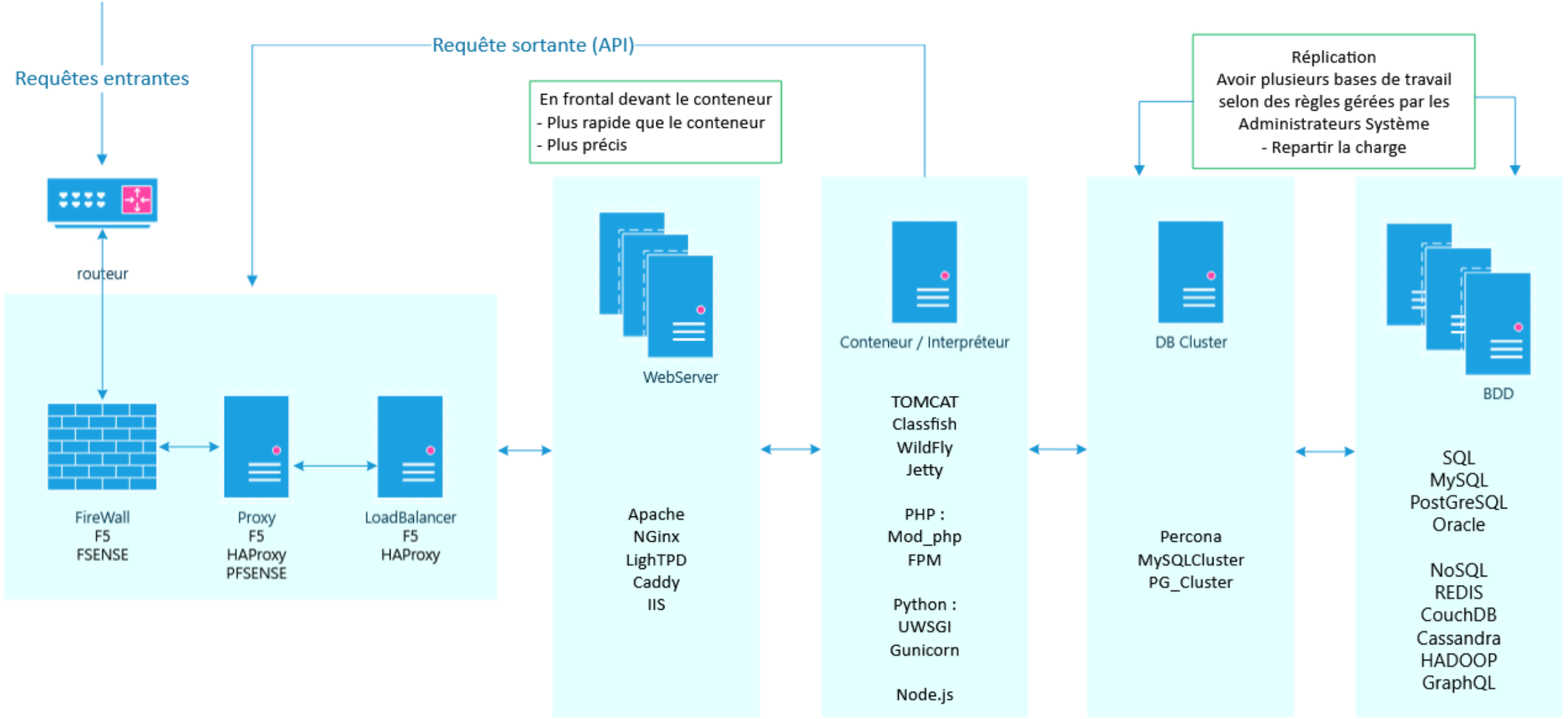
LES ENVIRONNEMENTS

LES DIFFÉRENTS TYPES D'ENVIRONNEMENT

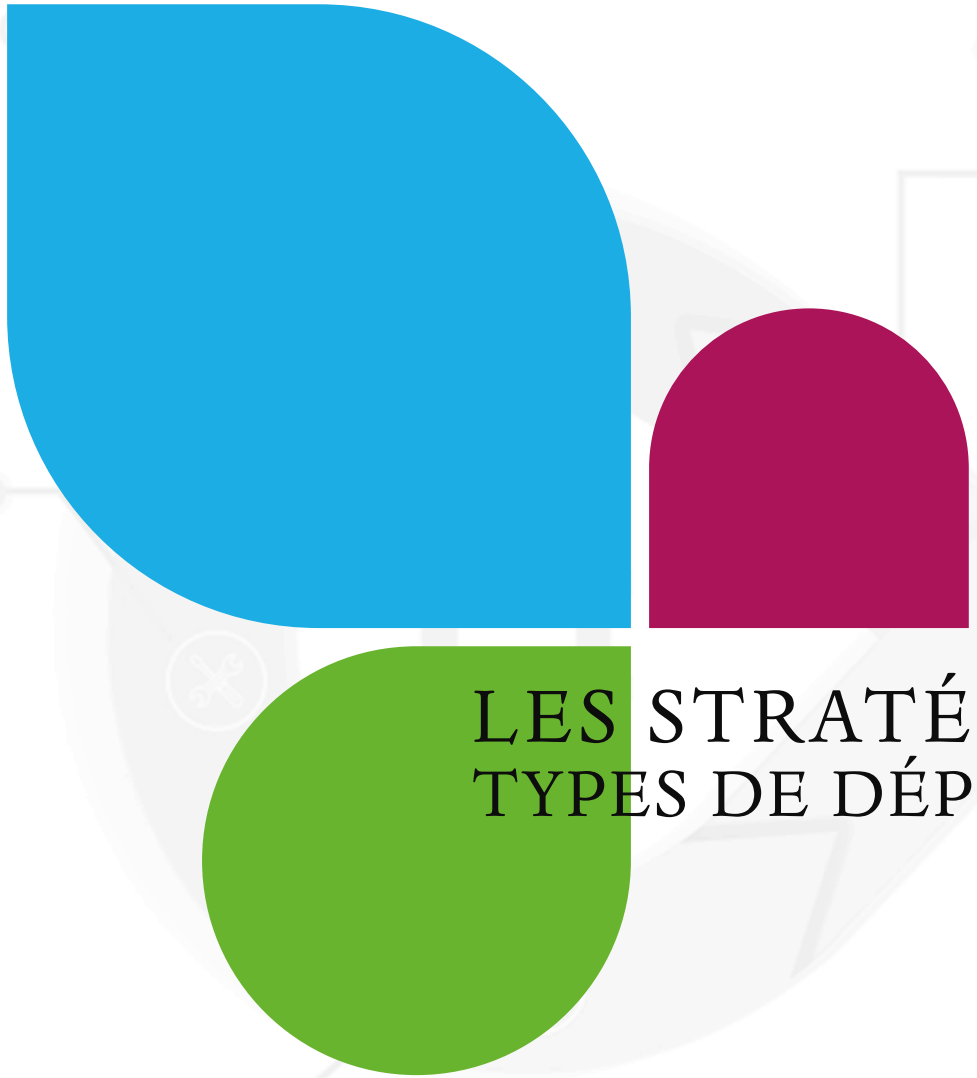
DIFFÉRENTS TYPES D'ENVIRONNEMENT



ENVIRONNEMENT DE PRODUCTION



Chacun de ses éléments peut donc se trouver dans un conteneur Docker, mais aussi des VM ou des machines physiques



LES STRATÉGIES DE DÉPLOIEMENT
TYPES DE DÉPLOIEMENT

TYPE DE DÉPLOIEMENT

Il existe plusieurs types de déploiement.

- Chacun comporte des avantages et des inconvénients.

Les services informatiques doivent les comparer pour trouver la meilleure technique à utiliser selon l'application prise en charge.

Le déploiement de base : c'est le plus simple. Il permet de mettre à jour tous les environnements cibles de manière simultanée, sans avoir de stratégie ou de processus.

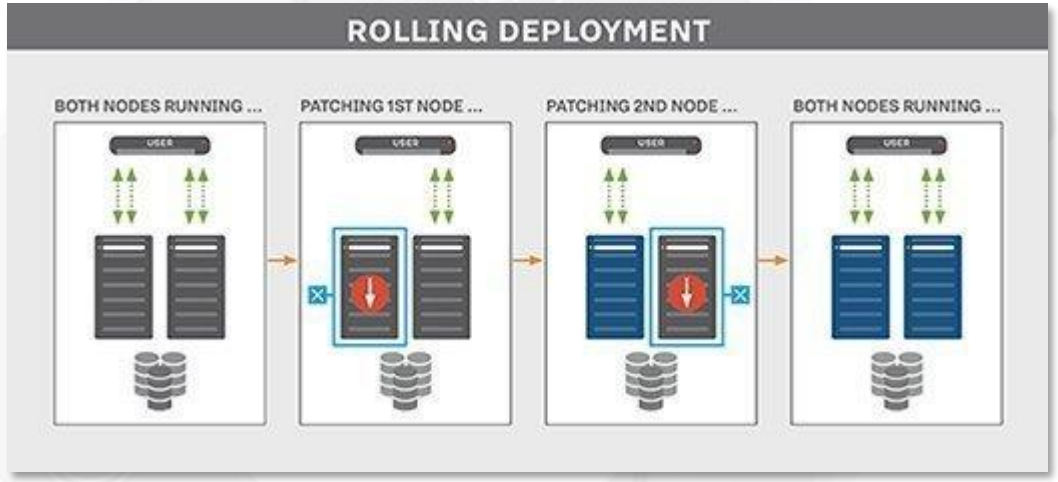
- C'est aussi le type de déploiement le plus risqué, car les logiciels ne sont pas déployés de façon contrôlée et lente.

Référence : https://www.lemagit.fr/conseil/DevOps-les-trois-types-de-deploiement-les-plus-populaires

TYPE DE DÉPLOIEMENT

Le déploiement progressif : ici, les applications logicielles sont mises à jour lentement et vont progressivement remplacer l'ancien logiciel.

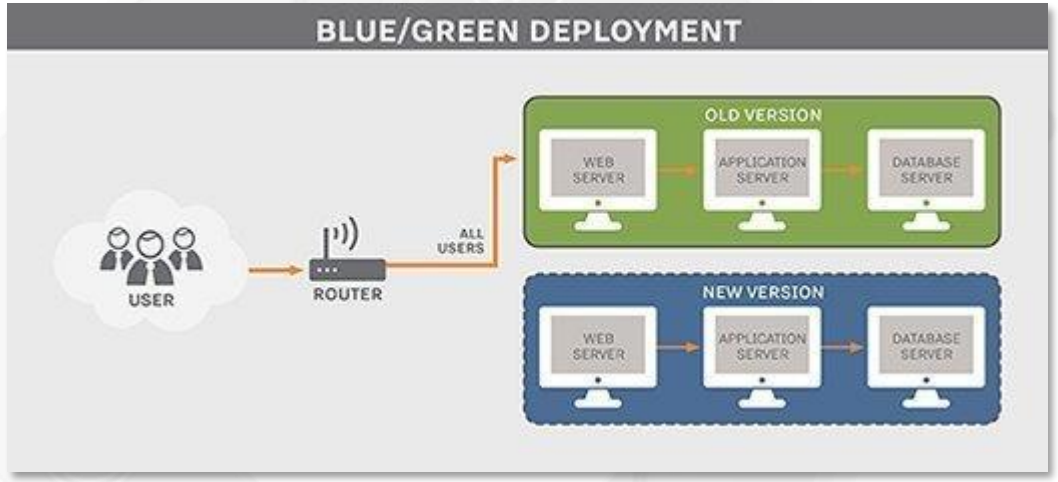
- Il possède un certain risque, car l'application originale n'est pas préservée.



TYPE DE DÉPLOIEMENT

Le déploiement bleu-vert : celui-ci permet de préserver l'ancien environnement tout en déployant le nouveau en simultanée.

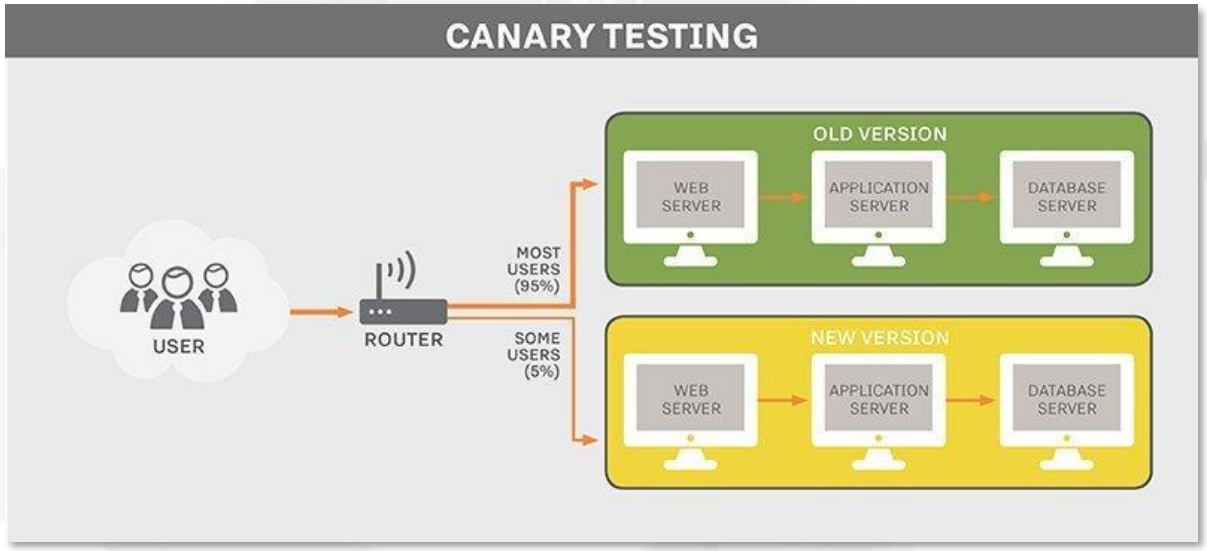
- Une fois l'application déployée, s'il y a un problème, il est possible de rediriger le trafic vers l'ancien pour qu'il fonctionne de manière optimale. Dès que le nouvel environnement est fonctionnel et ne présente pas de faille, il convient de mettre fin à l'ancien environnement.



TYPE DE DÉPLOIEMENT

Le déploiement canari : il consiste à déployer une application par sous-ensemble. Au début, il est destiné à un petit groupe de personnes, puis il est déployé de manière incrémentielle au travers de versions progressives.

- Ce type de déploiement permet d'avoir des retours d'expérience anticipés d'utilisateurs et d'identifier les bugs afin de les supprimer pour la version finale.
- Le déploiement canari est intéressant pour les applications qui possèdent un groupe identifiable d'utilisateurs et non général.



PLAN DE DÉPLOIEMENT

DÉFINIR LA MÉTHODE DE DÉPLOIEMENT

PLAN DE DÉPLOIEMENT

Un plan de déploiement se décompose en plusieurs phases.

PLAN DE DÉPLOIEMENT

Avant toute chose, il convient de déterminer la méthode de déploiement à utiliser en fonction du projet.

Il faut se poser diverses questions.

- Ces questions permettent de définir la meilleure manière d'aborder le déploiement et donc de choisir entre l'un des types de déploiement existants.

- Quel système, logiciel ou application faut-il déployer ?
- Quels sont les besoins de l'entreprise ?
- Combien d'utilisateurs finaux y aura-t-il ?
- Quels sont les risques du déploiement ?
- Quel outil est utilisé pour suivre le déploiement ?
- Qui fera partie de l'équipe de déploiement ?
- Quand le déploiement sera-t-il effectué ?
- Quand et comment les données existantes du système migreront-elles ?
- Le déploiement doit-il être simultané ou effectué de manière lente ?
- Combien de temps le système existant continuera-t-il d'exister en parallèle du système cible ?
- Quelle sera l'assistance fournie le jour suivant le déploiement ?
- Comment les problèmes sont-ils identifiés, suivis et résolus ?

PLAN DE DÉPLOIEMENT

Mise en place de la méthode

Planifier le déploiement

- Déterminer les fondements du plan de déploiement : les objectifs visés, les indicateurs de performance et les moyens utilisés pour atteindre les objectifs fixés. Chaque plan de déploiement s'adapte à la structure et aux pratiques de l'entreprise.

Identifier l'équipe de déploiement

- Pour mener le projet à bien de la phase d'idée jusqu'à la production, une équipe doit être constituée avec tous les talents nécessaires pour mettre en œuvre la méthode de déploiement choisie.
- L'équipe de projet comprend aussi bien l'équipe informatique, que l'équipe de développement et les responsables des opérations. Il ne faut pas non plus négliger le fait de collaborer avec les fournisseurs.

Faire des tests

- Lors du déploiement, nombreux sont les facteurs qui peuvent venir le perturber. La meilleure chose à faire pour s'assurer que cela n'arrive pas est de mettre en place un environnement de test.
- Les tests permettent de détecter les potentiels problèmes et de vérifier que les éléments clés du projet fonctionnent correctement.
- C'est aussi un bon moyen de s'assurer de la bonne migration des données de l'ancien système vers le nouveau système.

PLAN DE DÉPLOIEMENT

Mise en place de la méthode

Note : Prévoir une stratégie de communication

La communication est la clé du succès dans un projet de déploiement informatique. Une stratégie de communication efficace doit être mise en place pour assurer que toutes les parties prenantes sont informées de l'avancement du projet, des changements apportés et des défis rencontrés. Celle-ci doit comprendre des réunions régulières, des rapports d'étape et l'utilisation de canaux de communication clairs et accessibles. Une bonne communication permet de maintenir tout le monde aligné et de réagir rapidement aux imprévus.

Créer un calendrier de déploiement

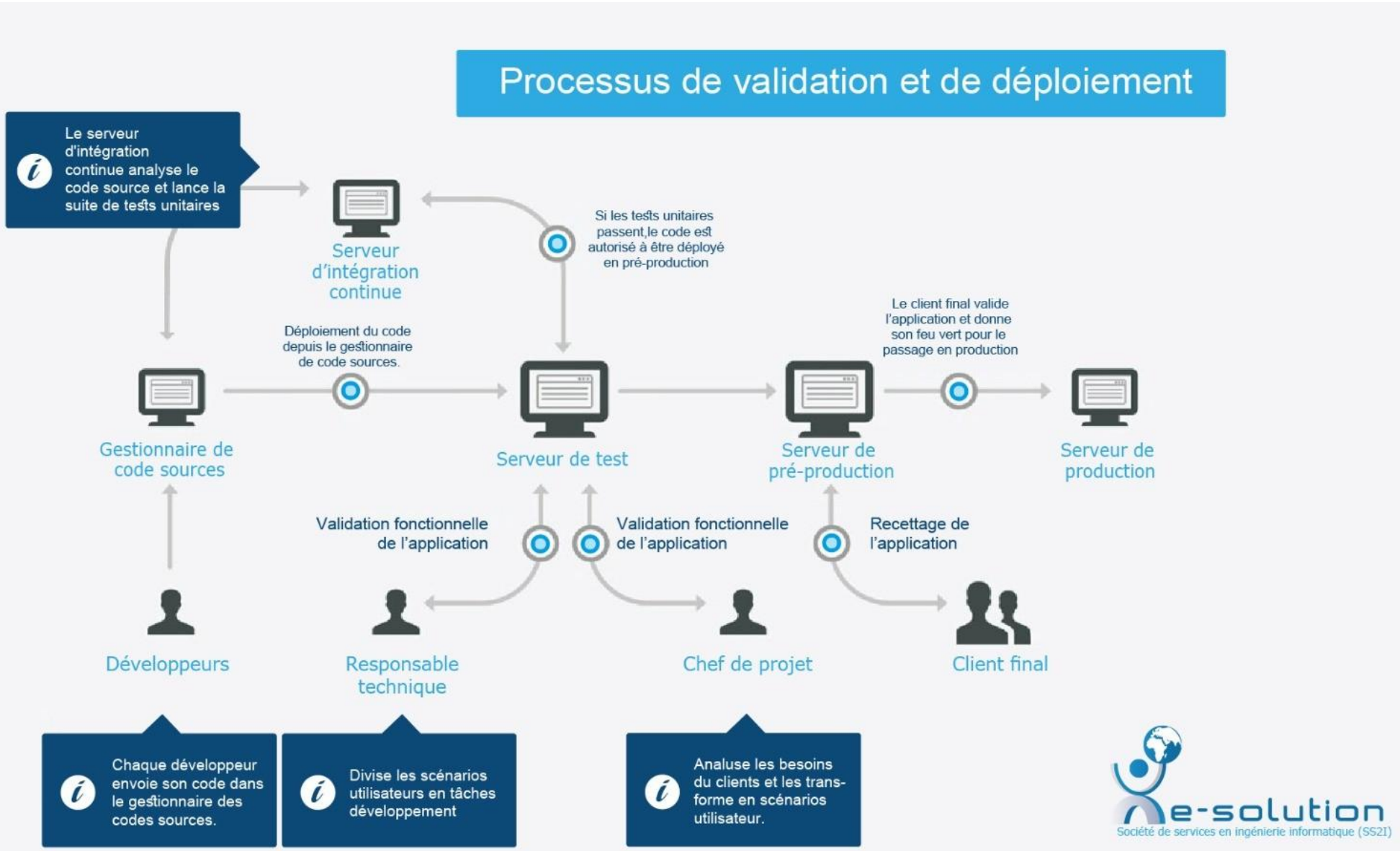
- Pour un plan de déploiement efficace, il est conseillé de le diviser en tâches.
- Cela permet de faciliter le travail, de le rendre plus gérable tout en optimisant la productivité.
- Ces tâches doivent être planifiées dans un calendrier. Celui-ci peut être créé à l'aide d'un logiciel automatisé ou par les membres de l'équipe.
- Ce calendrier regroupe les échéances de chacune des tâches et leur attribution. La répartition des activités de déploiement est clairement indiquée pour simplifier le processus de déploiement et éviter les problèmes.

Déployer

- Il s'agit de déployer définitivement le logiciel ou l'application sur les terminaux.
- Il est utile d'avertir les collaborateurs et les utilisateurs du déploiement afin de renforcer la coordination tout au long du processus et d'être au plus près des attentes utilisateurs.
- Il est aussi conseillé de prévoir une formation pour les utilisateurs afin qu'ils utilisent le système de manière optimale. Celle-ci est d'ailleurs souvent oubliée dans le plan de déploiement.

Assurer un suivi continu

- Une fois l'application, le système ou le logiciel déployé, il est recommandé de surveiller le déploiement et de l'optimiser avec les corrections nécessaires en cas d'erreur.


Processus de validation et de déploiement
Le serveur d'intégration continue analyse le code source et lance la suite de tests unitaires
Serveur d'intégration continue
Si les tests unitaires passent,le code est autorisé à être déployé en pré-production
Déploiement du code depuis le gestionnaire de code sources.
Le client final valide l'application et donne son feu vert pour le passage en production
Gestionnaire de code sources
Serveur de test
Serveur de pré-production
Serveur de production
Validation fonctionnelle de l'application
Validation fonctionnelle de l'application
Recettage de l'application
Développeurs
Responsable technique
Chef de projet
Client final
Chaque développeur envoie son code dans le gestionnaire des codes sources.
Divise les scénarios utilisateurs en tâches développement
Analuse les besoins du clients et les trans-forme en scénarios utilisateur.
e-solution
Société de services en ingénierie informatique (SS2I)

Plan de déploiement d'applications logicielles avec livrables

Phase	Deliverable	Responsibility	End Date
Pre-Deployment	• Deployment Planning Meeting • Your Text Here	Project Manager	01/06/2021
Pre-Deployment	• Distribute Project Plan • Your Text Here	Project Manager	01/06/2021
Pre-Deployment	• Review Project Schedule • Your Text Here	Project Manager	01/06/2021
Identify Phase	• Identify Deliverable • Your Text Here	Deployment Manager	01/06/2021
Identify Phase	• Identify Deliverable • Your Text Here	System Integrator	01/06/2021
Deployment	• Test Deploy Codes • Your Text Here	Program Developer	01/06/2021
Deployment	• Achieve Successful Deployment • Your Text Here	Program Developer	01/06/2021
Testing	• Statistics and Analytics • Your Text Here	Project Manager	01/06/2021
Testing	• Report Any Bugs • Your Text Here	Program Developer	01/06/2021
Monitoring	• Track Website Performance • Your Text Here	System Integrator	01/06/2021
Monitoring	• Utilize Monitoring Software • Your Text Here	Support Engineer	01/06/2021
Monitoring	• Your Text Here • Your Text Here	Hardware Engineer	01/06/2021
Your Text Here	• Your Text Here • Your Text Here	Consultant	01/06/2021
Your Text Here	• Your Text Here • Your Text Here	Installation Manager	01/06/2021

This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Afpa

DGE

Rungis Volaille Informatique

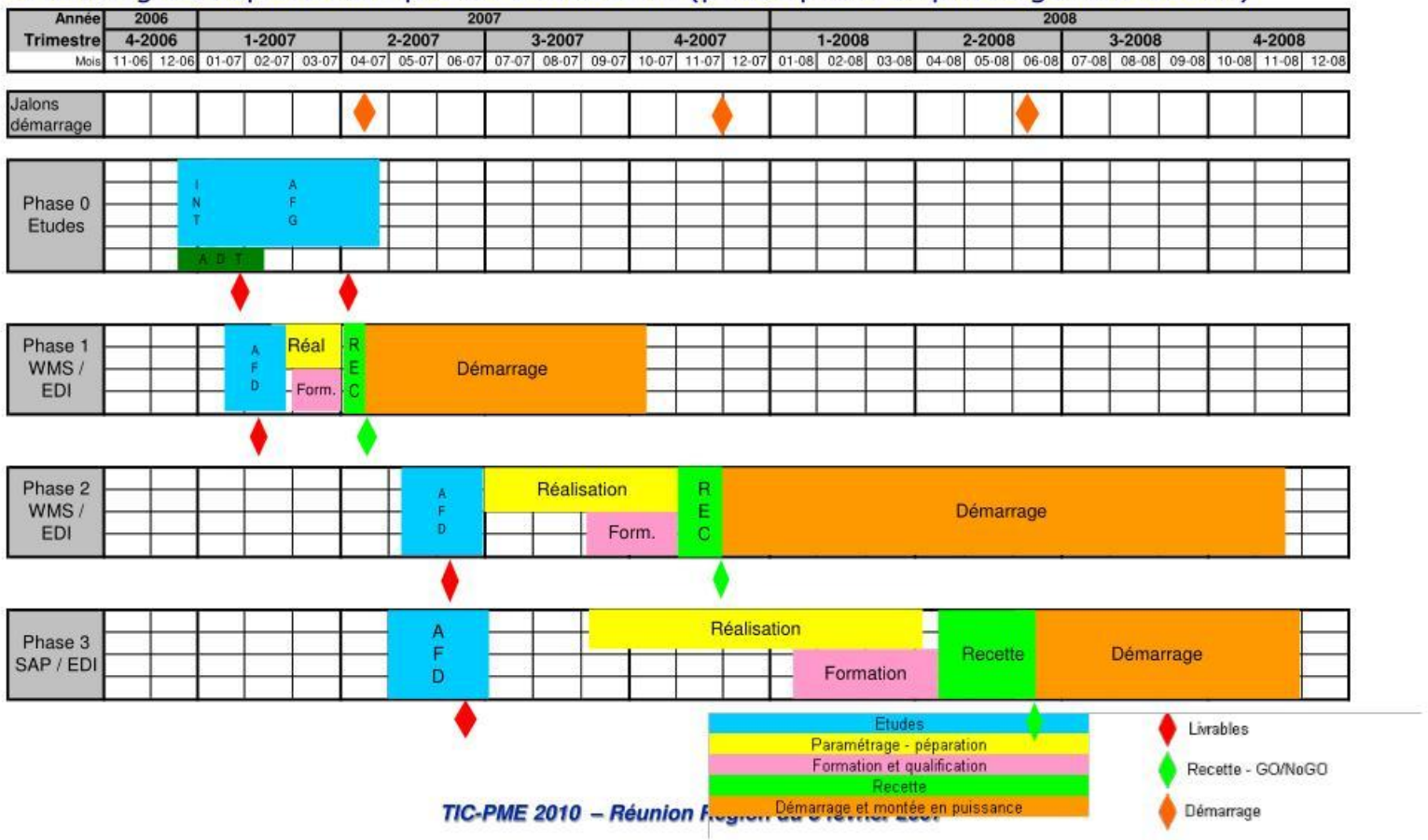
Rungis Volaille Plus

FENSCOPA

TIC PME Volo

Plan de déploiement

- Planning de déploiement prévu: début & fin (phase pilote et phase généralisation)



TIC-PME 2010 – Réunion Région

FACTEURS CLÉS DE SUCCÈS
LA RÉUSSITE D'UN DEPLOIEMENT

FACTEURS CLÉS DE SUCCÈS

Formaliser et communiquer le périmètre, l'organisation et le planning du déploiement.

1. Construire un planning réaliste
   - Un planning bien construit tient compte des délais incompressibles entre les différentes phases, des responsabilités des acteurs et comprend des marges suffisantes pour absorber les aléas.

2. Anticiper et gérer les risques
   - Chaque risque identifié doit être noté et une solution envisagée pour y palier.

3. Définir le rôle de chacun
   - Un Déploiement est un travail d'équipe, vous devez vous entourer des acteurs indispensables à sa réalisation et les coordonner.

4. Investir les temps de Formation
   - ENTRE 15 JOURS AVANT ET 15 JOURS APRÈS LE DÉPLOIEMENT.
   - Ce sont parfois les utilisateurs clés, formés avant la recette, qui formeront, à leur tour, les utilisateurs finaux

5. Réaliser un run à blanc
   - Le run à blanc consiste en une répétition des différentes étapes de mise en production de l'application qui permet de simuler le temps nécessaire et de planifier le déploiement pour respecter les délais et assurer la qualité de ce qui est mis en production.

6. Avoir le Go de la direction
   - Pour garantir le soutien de la direction lors de la phase de déploiement, il est primordial d'organiser une réunion du comité de pilotage juste après la fin de la recette, qui permettra de décider si l'organisation est prête à déployer le logiciel tel qu'il a été recetté.

LES 12 FACTEURS
CONVENTIONS

APPLICATIONS À 12 FACTEURS [12FACTOR.NET](12factor.net)

Ensemble de règles, de principes à suivre pour vos applications.

Les plus importants à appliquer sont :

1. Base de Code
   - une base = un dépôt versionné = application.
2. Dépendances
   - déclarer explicitement et isolé les dépendances. (pas de modifications). Utiliser des outils de gestionnaire de dépendances (Maven, npm composer…)
3. Configuration
   - stocker la configuration dans l'environnement. (par exemple .env de Symfony). L'objectif est de ne pas modifier le code en cas de changement et d'environnement.
4. Services externes
   - traiter les services externes comme des ressources attachées. (URL stocké dans la configuration pour la BDD). L'objectif est de ne pas modifier le code en cas de changement.
5. Assembler, publier, exécuter
   - séparer explicitement assemblage (Build), Publication (Release) et exécution (Runtime).
6. Parité dev/prod
   - garder les environnements de dev et de prod aussi proche que possible. Un développement peut prendre du temps, utilisé des outils différents que lors de sa mise en production. Il existe donc un fossé entre développement et déploiement.
     - Fossé temporel : temps entre le développement et la mise en production
     - Fossé des personnes : les développeurs, les testeurs, les devops
     - Fossé des outils : un développeur peut utiliser SQLite sur un OS particulier alors que le déploiement se fera sur un Oracle et Linux.

LES OUTILS DEVOPS

REVUE DES OUTILS DISPONIBLES

INTÉGRATION CONTINUE CONTINUOUS INTEGRATION (CI)

Des outils à disposition dont : GITLab, GITHub… Outils qui encourage à faire une intégration continue

- Une journée de code => 1 commit au moins pour sauvegarder le code.

- **GitLab CI** :
  - Composante de GitLab
  - En plus de l'intégration continue, GitLab offre un déploiement et une livraison continue
  - La configuration de GitLab CI s'effectue via un fichier YAML gitlab-ci.yml
    - Se positionne à la racine du projet
    - Des Template disponibles
    - Dés qu'on fait un push ou un merge request, les tests sont lancés.

- **GitHub Actions** permet de faire la même chose mais pour GitHub

JENKINS

JENKIS

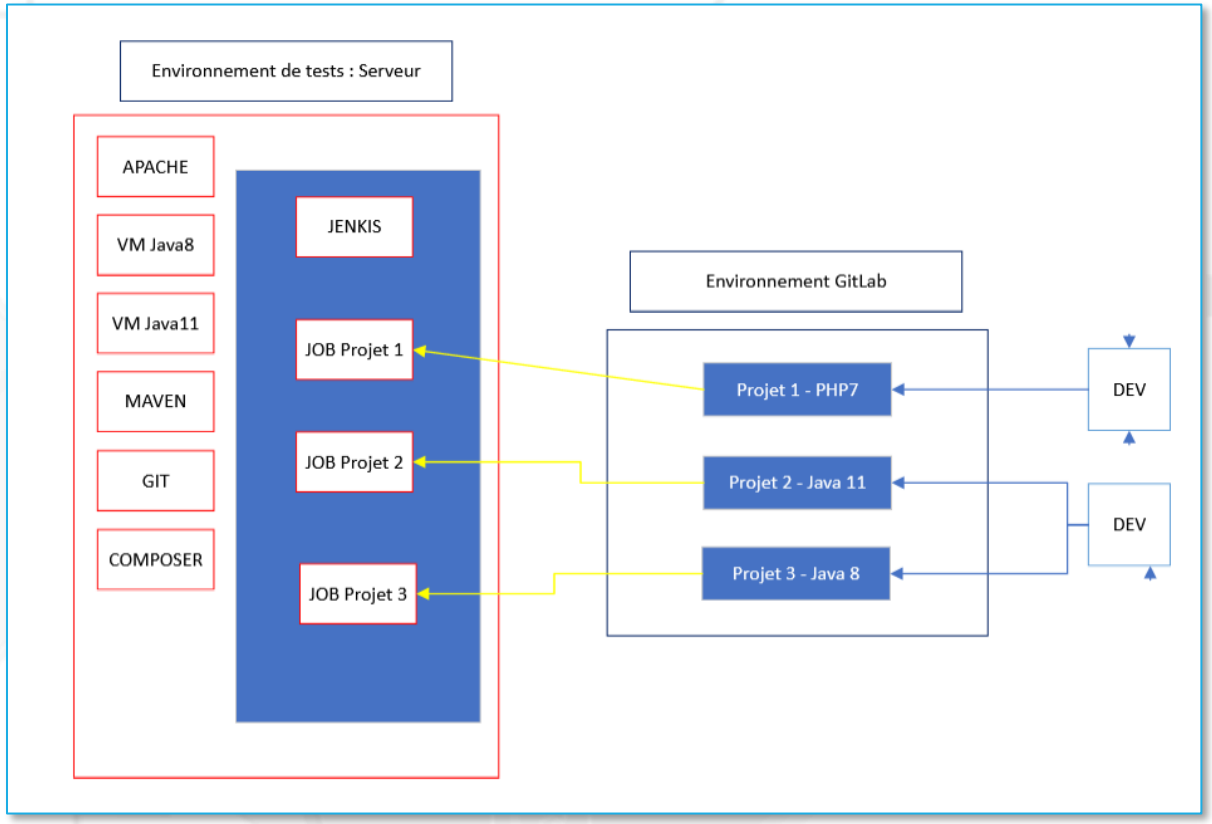
- Développé en Java et surtout fait pour Java au début.
- Outil d'intégration continue, facile à utiliser, simple, intuitif.
- Installé sur un serveur Tiers :
  - Paramétrage assez lourd mais permet beaucoup de choses.
  - De nombreux plugins disponibles PLUGINGS JENKINS
  - Permet d'avoir un environnement proche de la production.

FONCTIONNEMENT DE JENKIS

Voici comment se déroule généralement le fonctionnement de Jenkins.

1. Un développeur insère son morceau de code dans le répertoire du code source (GIT).
2. Jenkins, de son côté, vérifie régulièrement le répertoire pour détecter d'éventuels changements.
3. Lorsqu'un changement est détecté, Jenkins prépare un nouveau build.
   - Si le build rencontre une erreur, l'équipe concernée est notifiée.
   - Dans le cas contraire, le build est déployé sur le serveur test.
4. Une fois le test effectué, Jenkins génère un feedback et notifie les développeurs au sujet du build et des résultats du test.

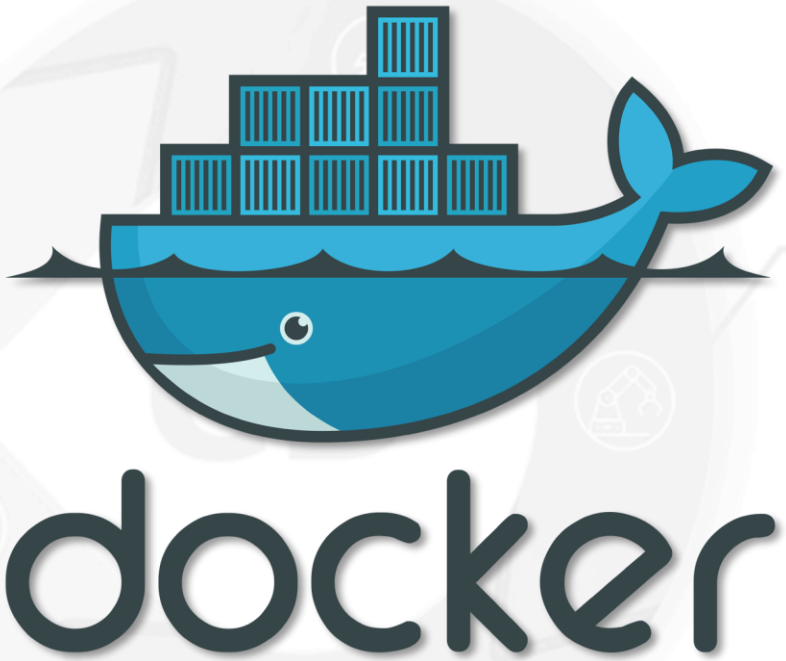
Jenkins peut faire un déploiement Bleu-Vert.
Le souci avec Jenkins est la gestion des versions !!



DOCKER

Docker est un logiciel libre permettant de lancer des applications dans des conteneurs logiciels.

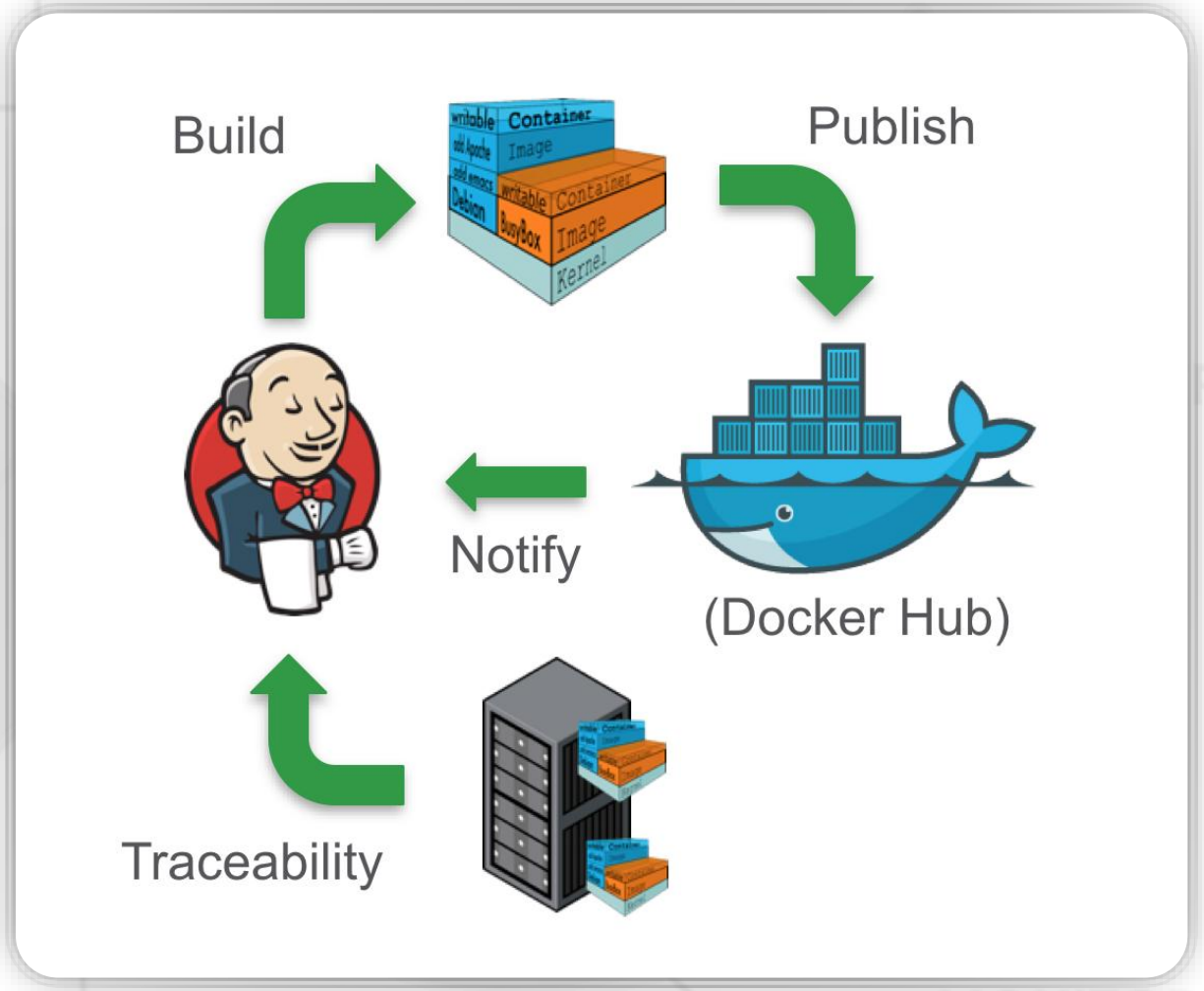
- Les conteneurs Docker contiennent une copie complète d'un système de fichiers Linux (/etc, /var, /usr, /tmp, /bin, etc…)
- Docker à un référenciel public fournissant des images prête à l'emploi : Docker Hub.
- un Dockerfile permet de configurer et de créer rapidement plusieurs images à partir d'une ou plusieurs images pour la rendre partageable plus facilement.
- Les conteneurs docker ont des environnements isolés. Ils vont se parler entre eux dans un environnement de réseau virtuel, isolé du reste du serveur.
- Docker Compose : outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs conteneurs comme un ensemble de services interconnectés



JENKINS + DOCKER

Avantages :

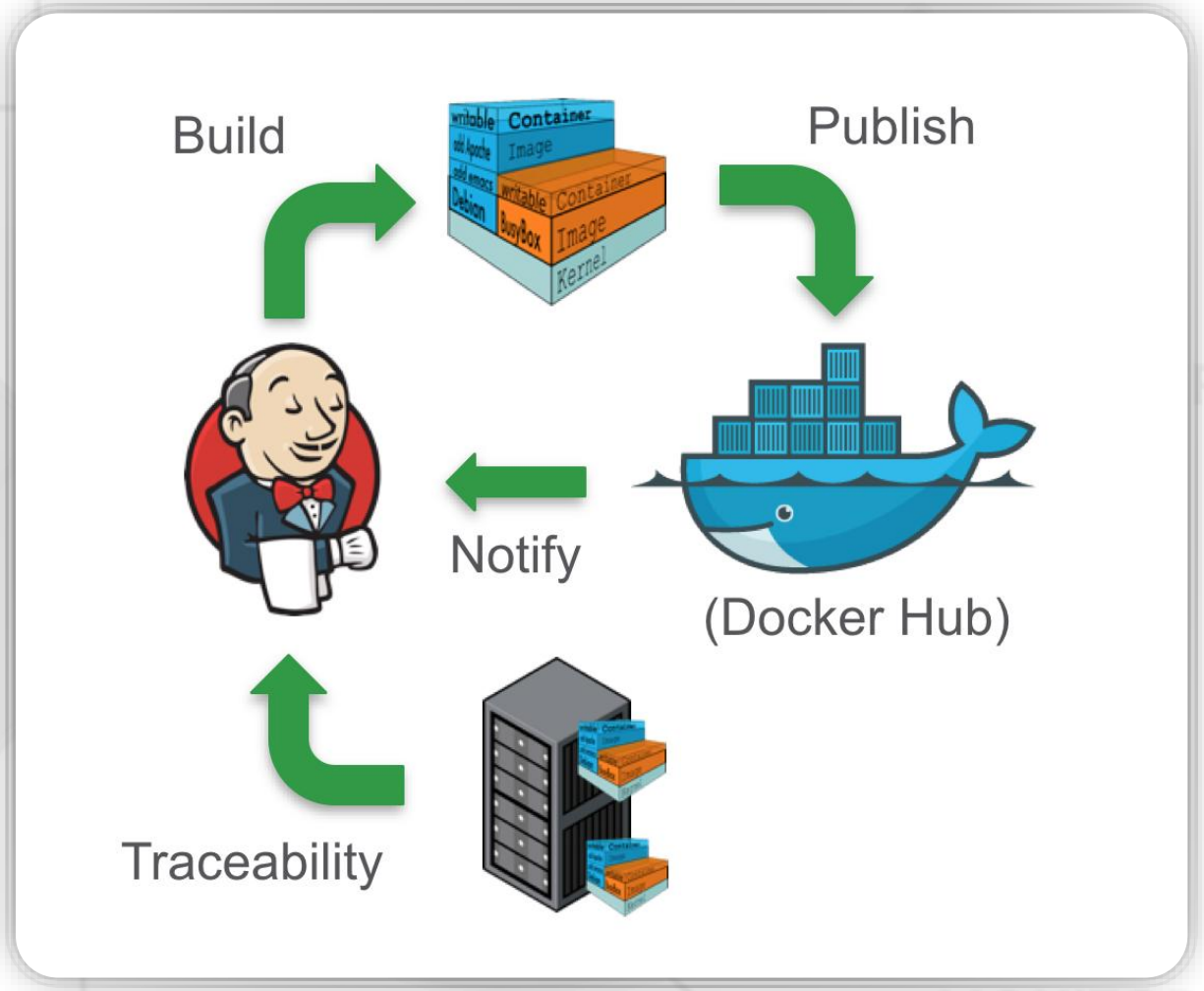
- Evite les conflits entre les différents environnements
- Les environnements virtuels isolés sont indépendants du reste de l'environnement du serveur
- Les développeurs décident de quel environnement ils ont besoin (docker file et docker compose). Ce sont des fichiers texte donc versionnables
- Dans le serveur de tests, on a juste à installer docker, Java et Jenkins. On peut même conteneuriser Jenkins
- C'est une infrastructure en tant que code (infrastructure as code) :
  - ensemble de mécanismes permettant de gérer, par des fichiers descripteurs ou des scripts, une infrastructure (informatique) virtuelle



JENKINS + DOCKER

Inconvénients :

- En production, plus compliqué d'avoir des conteneurs parce que la prod est accessible au public par le Web donc plus attaquable.
- Ce sont les développeurs qui donnent les images au docker daemon donc le l'administrateur système n'a plus la main sur les environnements présents sur son serveur de tests :
  - Le Docker daemon est un processus qui orchestre tous les environnements.
  - Il doit avoir les droits du user 'root' qui a tous les droits sur le serveur. Depuis un conteneur, on peut faire faire des choses au docker daemon et donc faire faire des choses au serveur.



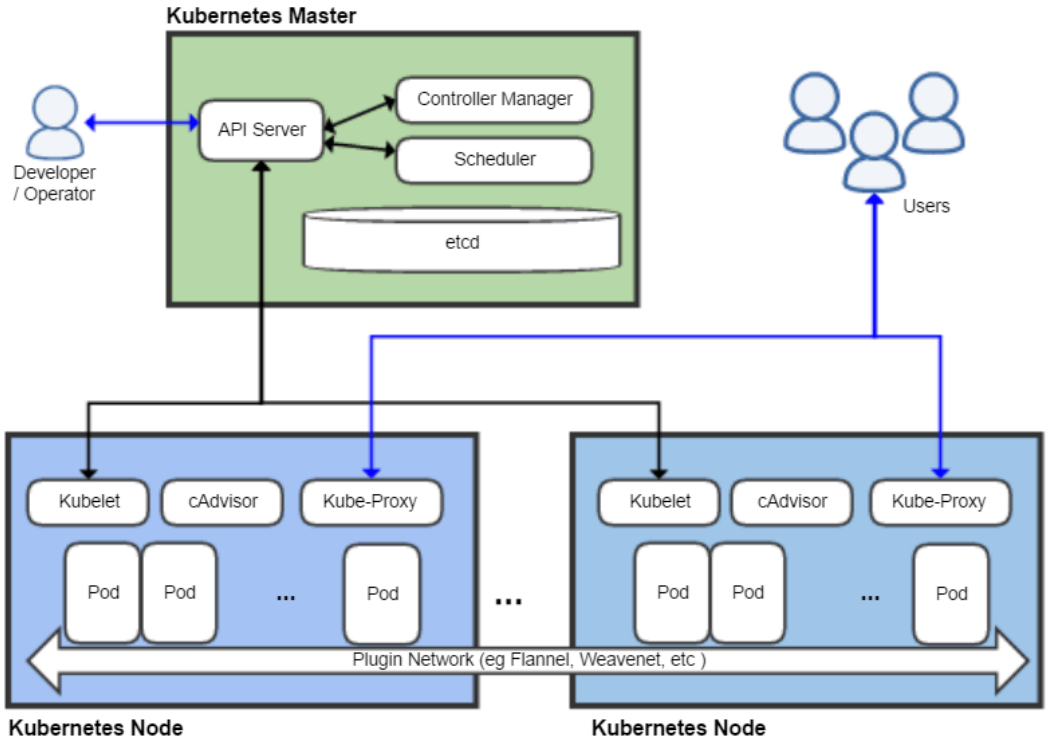
Ici pour des grosses gestions de containeurs DEVOPS

Kubernetes (communément appelé « K8s2 »)

système open source - Kubernetes est une plateforme d'orchestration de conteneurs Open Source qui automatise de nombreux processus manuels associés au déploiement, à la gestion et à la mise à l'échelle des applications conteneurisées.

- distribuer et ou à gérer des applications à grande échelle, qu'elles soient d'ancienne génération, conteneurisées ou cloud-native, ainsi que celles qui sont transformées en microservices dans différents environnements, y compris dans les clouds privés et les principaux clouds publics de fournisseurs tels que Amazon Web Services (AWS), Google Cloud, IBM Cloud et Microsoft Azure.

Souvent utilisé avec Docker - Conçu à l'origine par Google, puis offert à la Cloud Native Computing Foundation



CLOUD COMPUTING

DANS LES NUAGES…

LE CLOUD COMPUTING

Le cloud computing est un modèle de prestation de services informatiques dans lequel les ressources (serveurs, stockage, bases de données, réseaux, logiciels, analyses, etc.) sont fournies via Internet (cloud) pour offrir une innovation rapide, des ressources flexibles et une économie d'échelle.

Les principaux Cloud :

- AWS (Amazon Web Services)
- Microsoft Azure
- Google Cloud

Avantages du cloud computing

» «

- Les organisations n'ayant pas à assumer la charge et la responsabilité de la maintenance et du paiement de leur propre infrastructure, le temps nécessaire au développement et au déploiement des applications et des services est considérablement réduit.
- Les organisations peuvent se concentrer sur le développement d'applications plutôt que sur la maintenance manuelle laborieuse de l'infrastructure pour s'assurer que les logiciels et les outils sont sur les dernières versions.
- Grâce au cloud, les organisations deviennent plus rationalisées. Elles peuvent donc atteindre leurs clients plus rapidement, tester et déployer leurs applications plus rapidement, et stimuler davantage l'innovation.

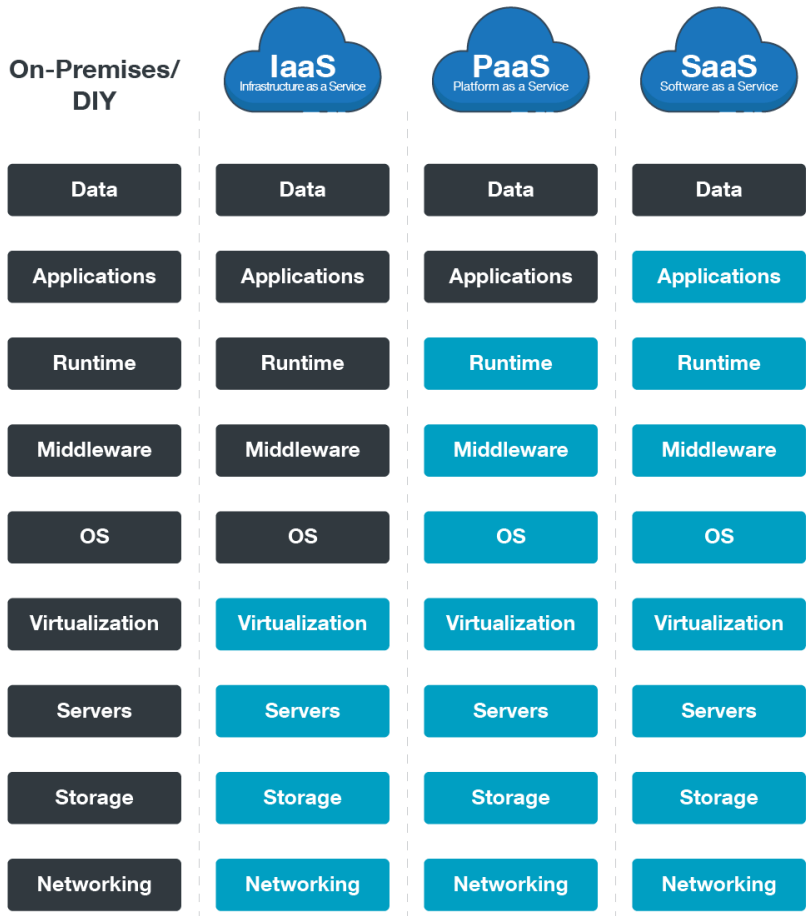
LE CLOUD COMPUTING

Types de cloud computing

Cloud public : ouverte au grand public ou à un grand groupe industriel. Existe dans les locaux du fournisseur de services cloud et est accessible via Internet

Cloud privé : dessert une seule organisation et est inaccessible pour les personnes extérieures à l'organisation. Peut exister dans les locaux du fournisseur de services cloud ou en tant que configurations de serveur exécutées physiquement dans les locaux de l'organisation

Cloud hybride : Une combinaison de modèles de cloud public et privé, associant la nature fiable du cloud privé et la capacité à la demande du cloud public. Idéal pour les entreprises qui fournissent des services ou proposent des produits



LE CLOUD COMPUTING

On-Promises

- Les services et infrastructures sont gérés localement, au sein de l'entreprise.
- L'entreprise est responsable de l'achat, de la maintenance et de la gestion de son propre matériel et logiciels.
- Offre un contrôle total sur les données et les systèmes, mais peut être coûteux et nécessiter des ressources importantes pour la gestion.

Infrastructure as a Service (IaaS)

- Fournit une infrastructure virtuelle, comme des serveurs, du stockage et des réseaux, via Internet.
- Les utilisateurs peuvent installer et gérer leurs propres systèmes d'exploitation, applications et données.
- *Exemples : Amazon Web Services (AWS) EC2, Microsoft Azure Virtual Machines, Google Compute Engine.*

LE CLOUD COMPUTING

Platform as a Service (PaaS)

- Offre une plateforme permettant aux développeurs de créer, tester et gérer des applications.
- Inclut des outils de développement, des environnements d'exécution et des bases de données.
- *Exemples : Google App Engine, Heroku, Microsoft Azure App Services.*

Software as a Service (SaaS)

- Fournit des applications logicielles via Internet, souvent sur un modèle d'abonnement.
- Les utilisateurs accèdent aux applications via un navigateur web sans avoir à gérer l'infrastructure sous-jacente.
- *Exemples : Google Workspace (G Suite), Microsoft Office 365, Salesforce, Gmail.*

MERCI !

Jérôme BOEBION
Concepteur Développeur d'Applications
Version 3 - révision 2025

Afpa
se former, avancer