



Introduction au Métier DevOps Cloud

1

Objectifs du cours

- ❑ Savoir les origines de Devops
- ❑ Découvrir les objectifs et les principes de Devops
- ❑ Comprendre les pratiques Devops
- ❑ Découvrir des outils Devops
- ❑ Comprendre les principes du cloud computing
- ❑ Découvrir les principaux acteurs de cloud
- ❑ Découvrir les solutions Devops offertes par les fournisseurs cloud

Plan



Origines de
DevOps



DevOps

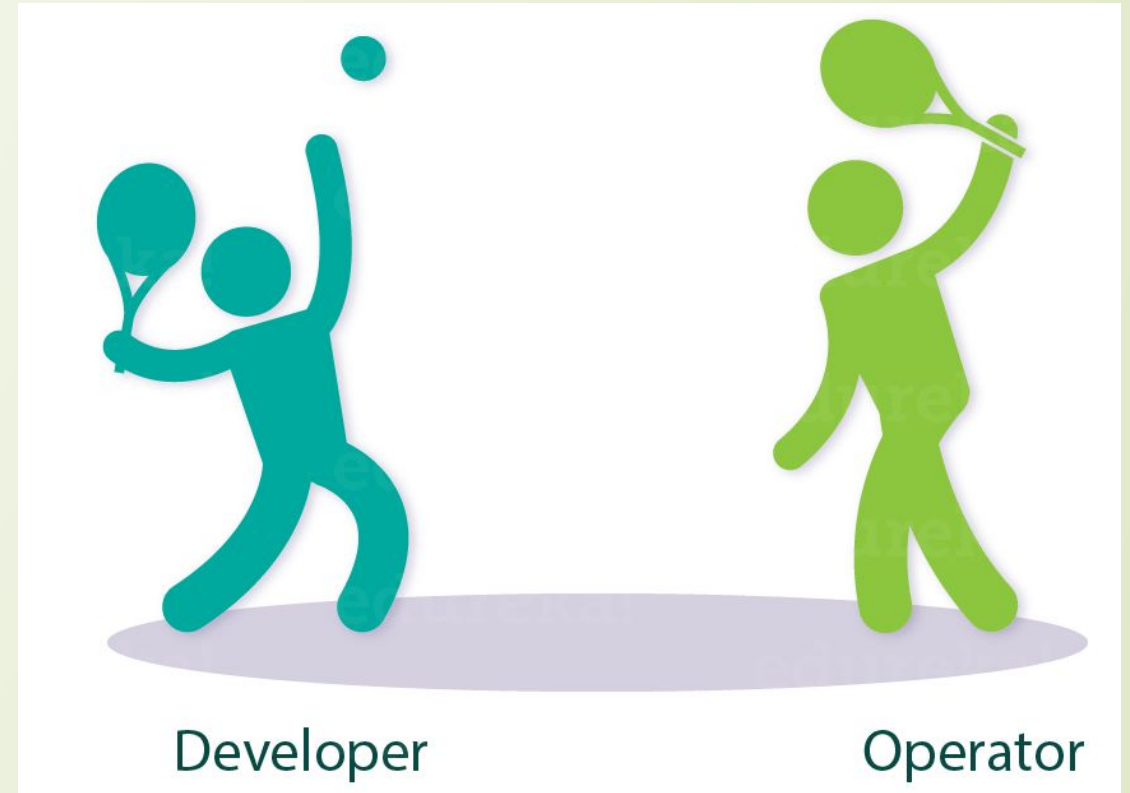


Cloud



Devops et Cloud

Origines de DevOps



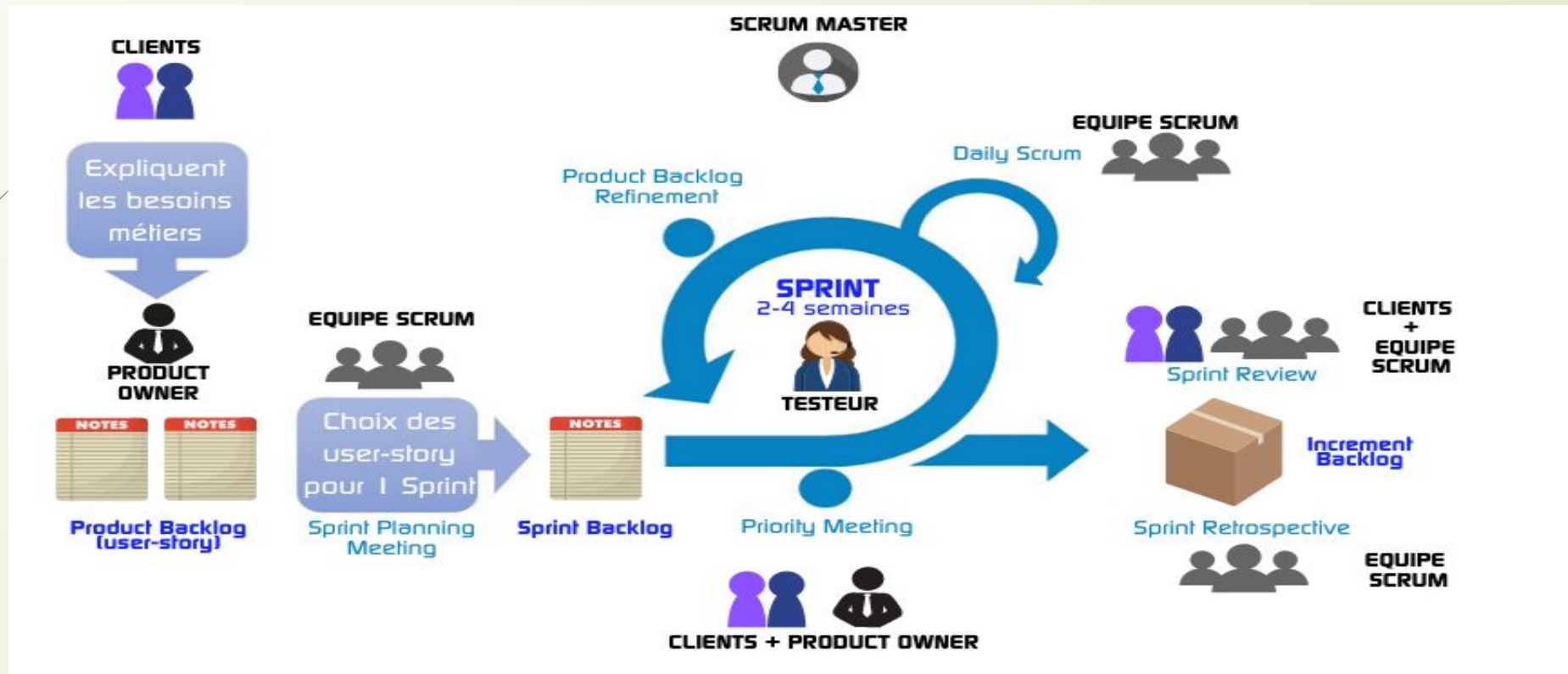
Origines de Devops

Processus de développement

- Modèles en cascade
 - Code and Fix
 - Waterfall model
 - Cycle en V
- Modèles évolutifs/incrémentaux
- Modèle en spirale
- Modèles agiles
 - XP
 - Scrum

Origines de Devops

Scrum



Origines de Devops Agile

□ Avantages

- Intégration du client dans le processus de développement
- Deadlines intégrées
- Visibilité continue
- Focus et flexibilité

Origines de Devops Agile

- ❑ On n'a pas atteint l'idéal !!!
 - ❑ Le cercle de communication n'a pas intégré les « Ops »
 - ❑ Concentration seulement sur les changements/incréments
 - ❑ Buts conflictuels entre l'équipe de développement et celle d'exploitation

Origines de Devops

Equipe « Dev »

- Scrum Master
- Product Owners
- Développeurs
- Testeurs

Origines de Devops

Equipe « Ops »

- Administrateur système
- Administrateur BD
- Ingénieur sécurité
- Administrateur Réseaux

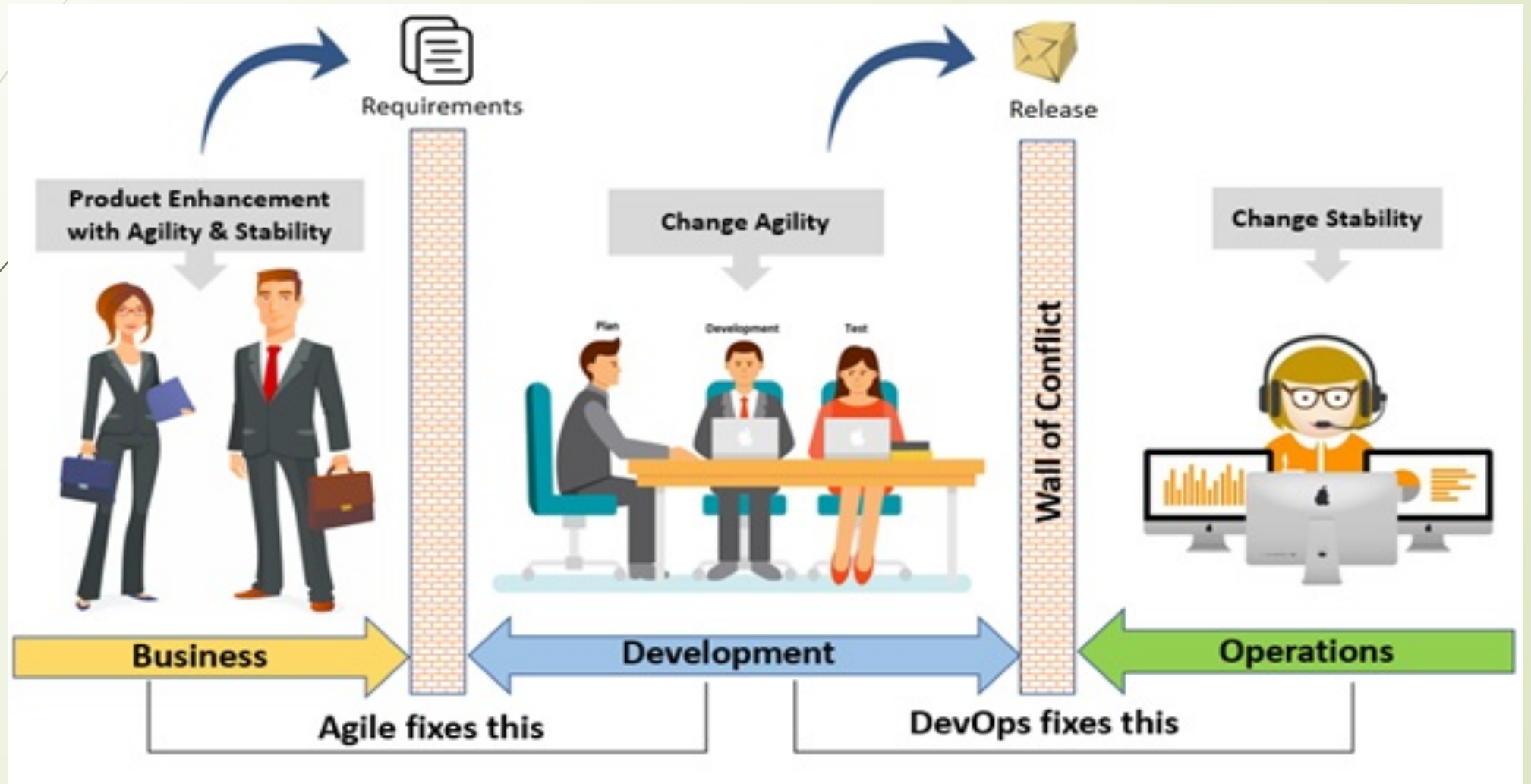
Origines de Devops

Problématique



Origines de Devops

Problématique



Origines de Devops

Les débuts...

□ Regarder

□ <https://www.youtube.com/watch?v=o7-IuYS0iSE>

DevOps



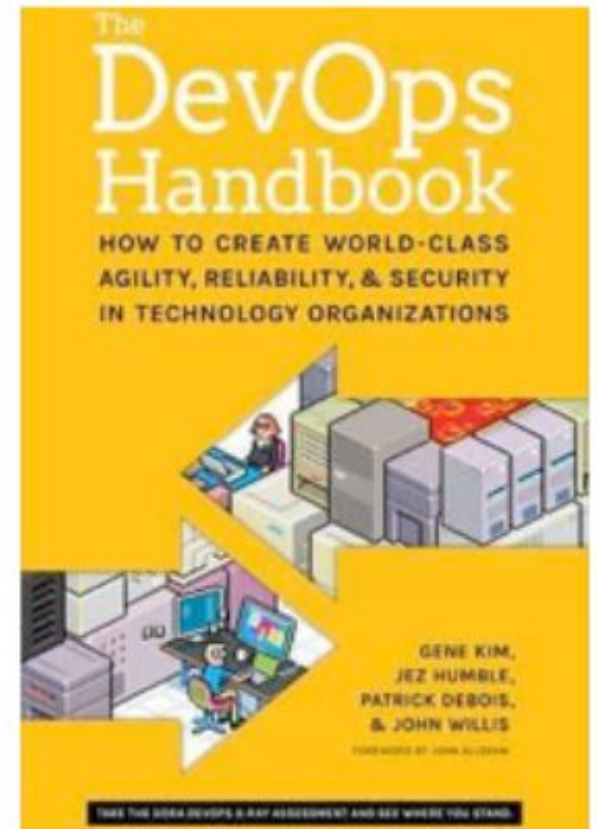
Devops

Définition

- *« Le devops — est un mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops), notamment l'administration système. » [Wikipédia]*
- *« "devops" est un terme issu de la contraction des mots anglais "development" (développement) et "operations" (exploitation). » [DevOps.fr]*

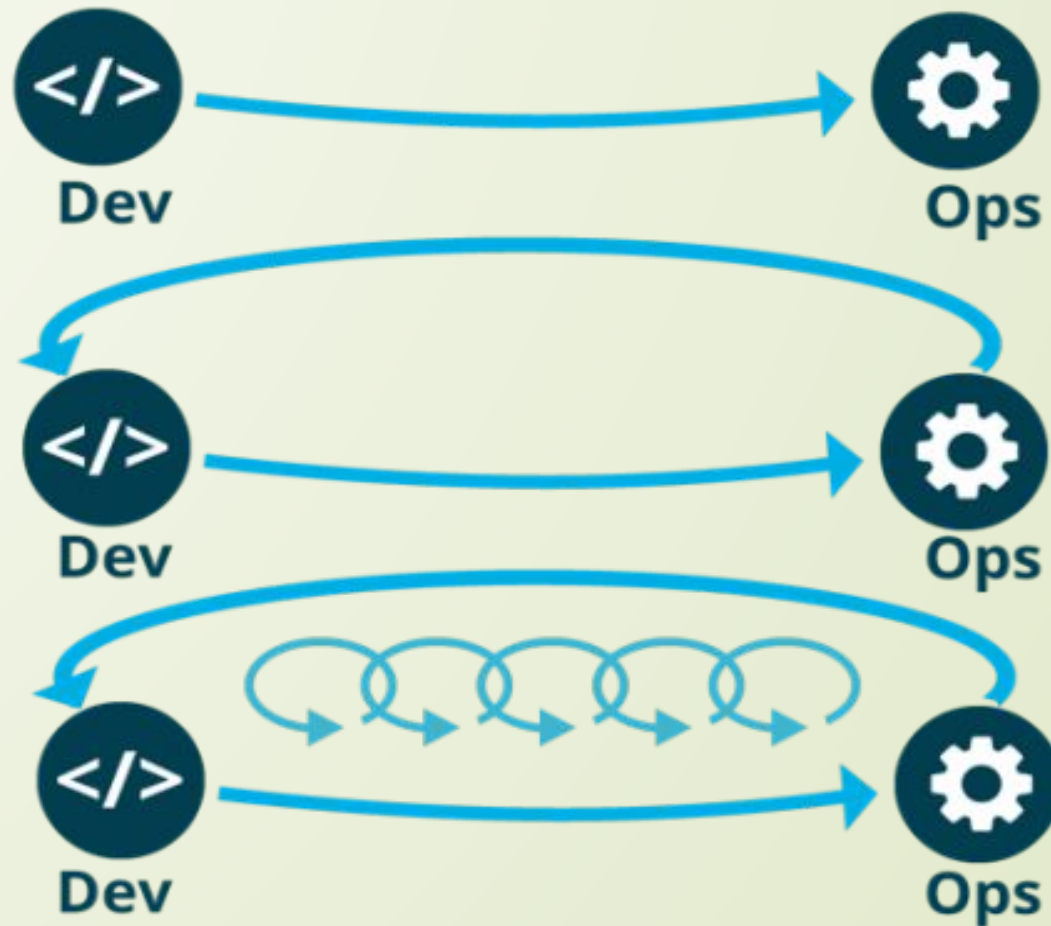
Devops Définition

“Imagine a world where product owners, Development, QA, IT Operations and Infosec work together, not only to help each other, but also to ensure that the overall organization succeeds. By working towards a common goal, they enable the fast flow of planned work into production, while achieving world-class stability, reliability, availability and security.”



Devops

Les 3 chemins



Devops

Les 3 chemins

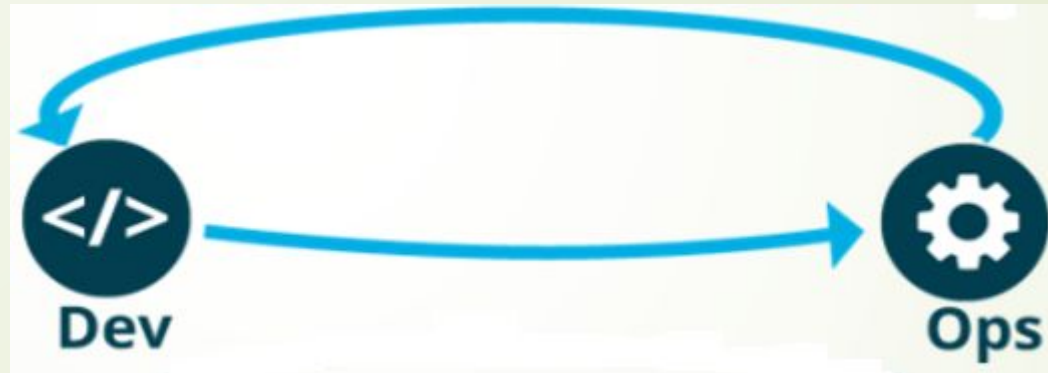
- ❑ Objectif du premier chemin : Comprendre et maximiser le flux de travail
 - ❑ Comprendre le flux de travail
 - ❑ Maximiser le flux de travail en éliminant les contraintes
 - ❑ Ne jamais omettre un défaut connu
 - ❑ Ne jamais accepter une optimisation locale qui génèrera une dégradation globale



Devops

Les 3 chemins

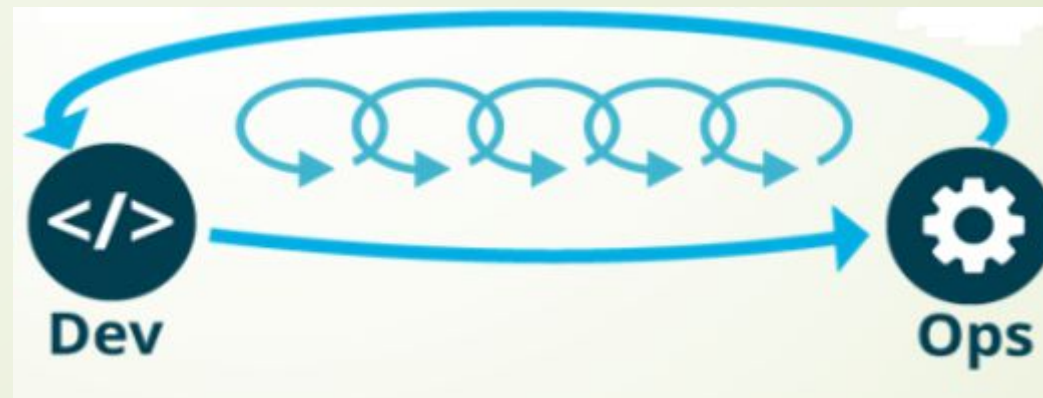
- Objectif du second chemin : Avoir du feedback coté « Ops »
 - Qui doit être bref et rapide



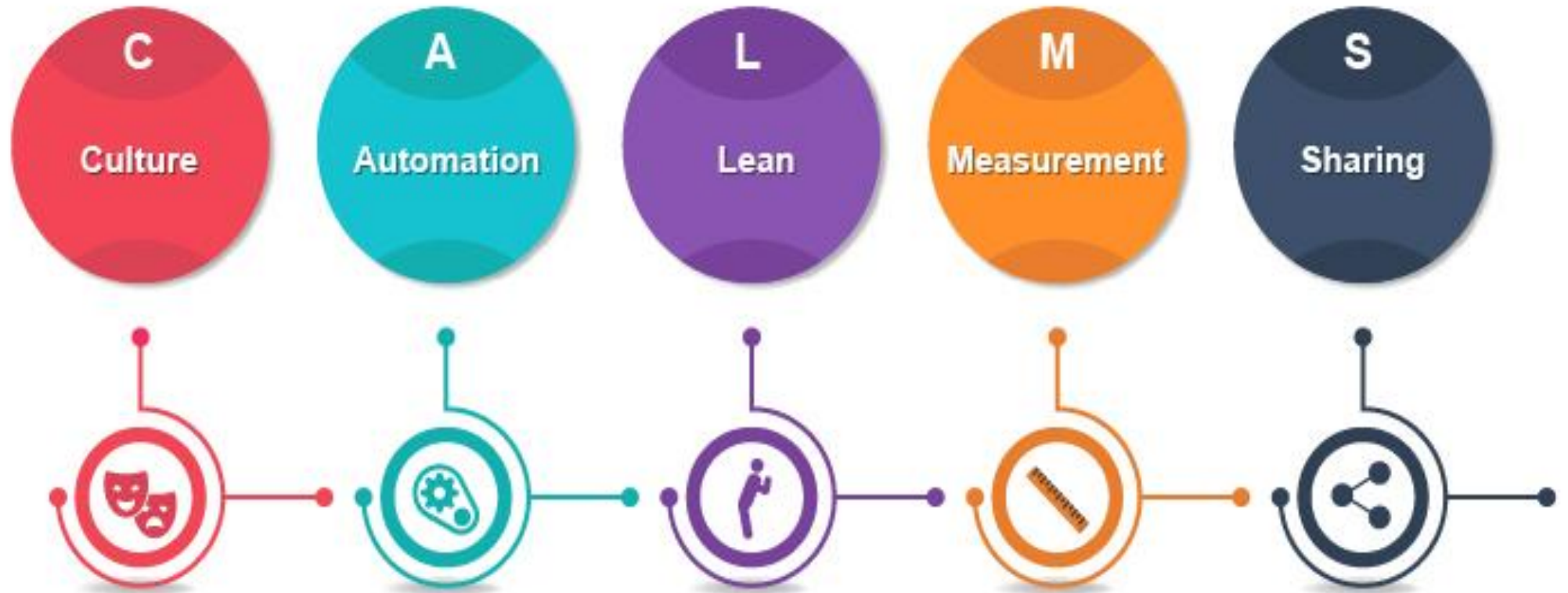
Devops

Les 3 chemins

- ❑ Objectif du troisième chemin : On a le droit d'apprendre tout au long du processus de développement
 - ❑ Nouveaux langages
 - ❑ Nouvelles bibliothèques



Devops Principles



Devops Culture

- Relative aux personnes et au processus relatifs à Devops
 - Avoir la culture de communication et de collaboration
 - Avoir l'esprit d'automatisation
 - Avoir l'esprit d'apprentissage

Devops

Automatisation

- Consiste à utiliser des technologies pour réaliser des tâches, avec une intervention humaine réduite, afin de faciliter les flux de feedback entre les équipes d'exploitation et de développement pour accélérer le déploiement en production des mises à jour itératives apportées aux applications.

Devops

Lean

- ❑ créé par Toyota au Japon et introduit au sein de ses usines dans les années 1970
- ❑ La participation de l'ensemble des employés d'une entreprise à la lutte contre le gaspillage en éliminant toute les activités non rentables de l'entreprise.
- ❑ Augmenter la productivité tout en améliorant les conditions de travail.
- ❑ Sources de gaspillage :
 - ❑ Surproduction, Défauts
 - ❑ Compétences
 - ❑ Communication et formation
 - ❑ Coûts
 - ❑ Stocks
 - ❑ Délais
 - ❑ Transports

Devops

Mesures

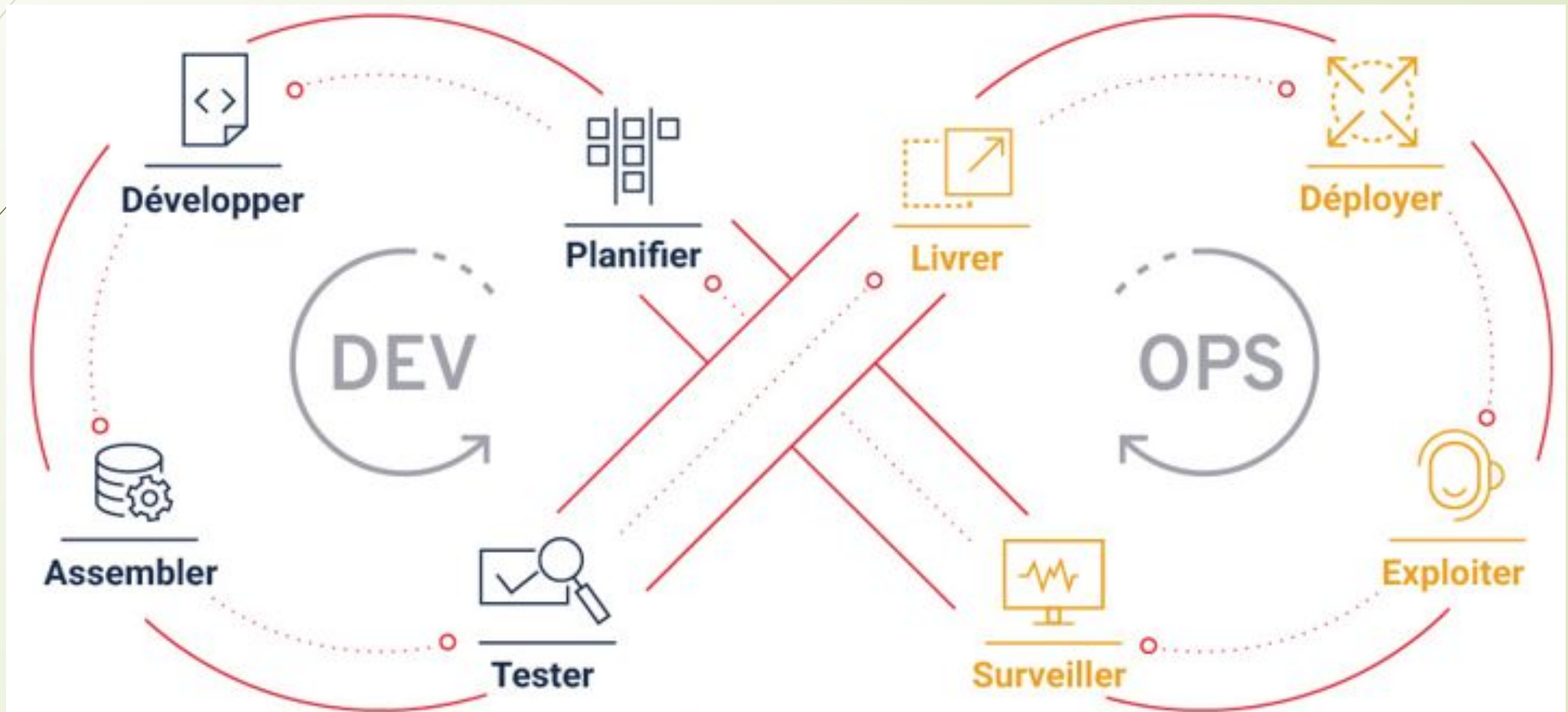
- Ce qu'on doit mesurer et enregistrer :
 - Projets terminés et fréquence de publication
 - Pourcentage de déploiements réussis / infructueux
 - Temps moyen de récupération (MTTR)
 - Lead time (du développement au déploiement)
 - Lead time (du développement au déploiement)
 - Fréquence des nouveaux déploiements
 - Qualité du code

Devops

Partage


















- Coding dojos
- Conférences internes
- Brown bag meetings
- eXtreme Programming
- <vos idées ici>

Devops Chaine



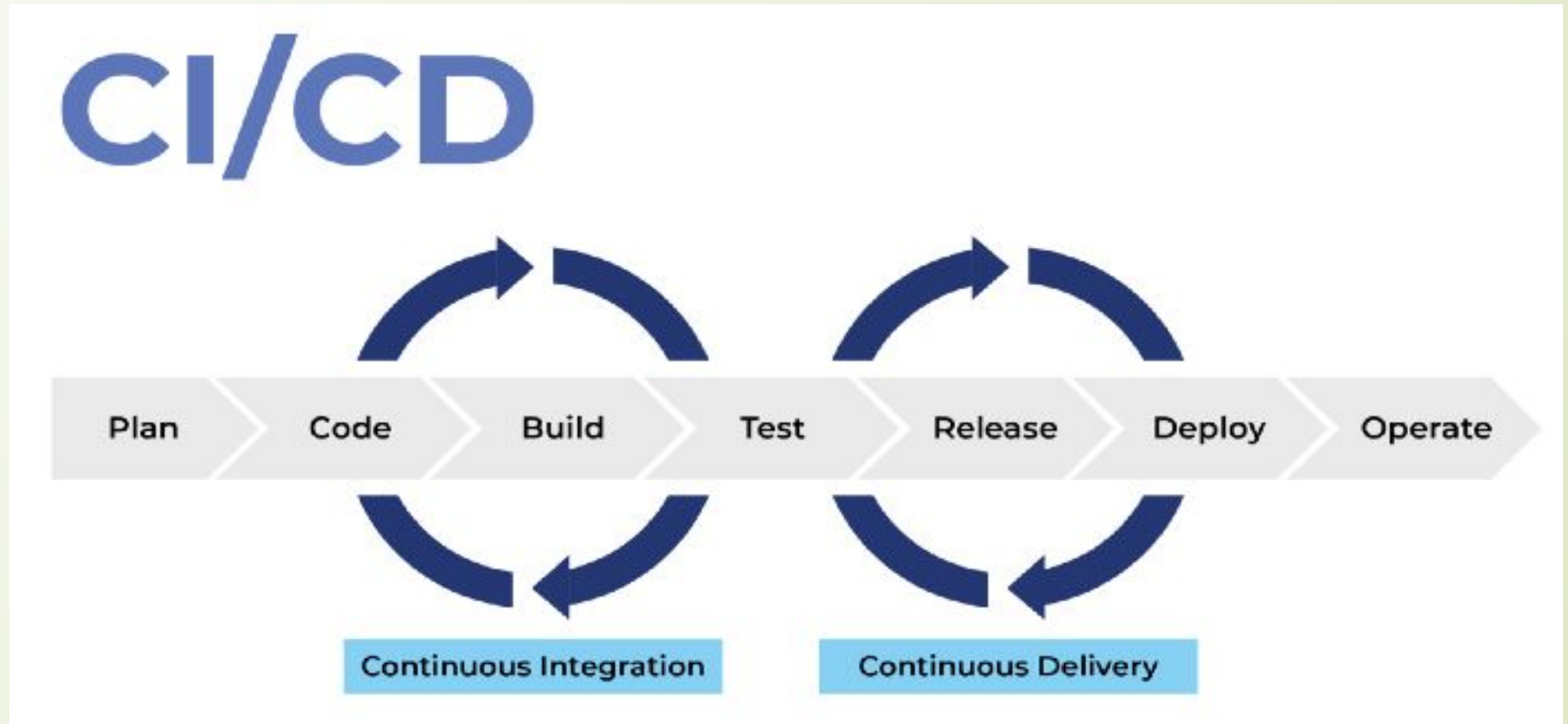
Devops

Catégories des Outils

 AiOps/Analytics	 Continuous Integration	 Security
 Artifact/Package Management	 Database Management	 Serverless/PaaS
 Cloud	 Deployment	 Source Control Management
 Collaboration	 Enterprise Agile Planning	 Testing
 Configuration Automation	 Issue Tracking/ITSM	 Value Stream Management
 Containers	 Release Management	

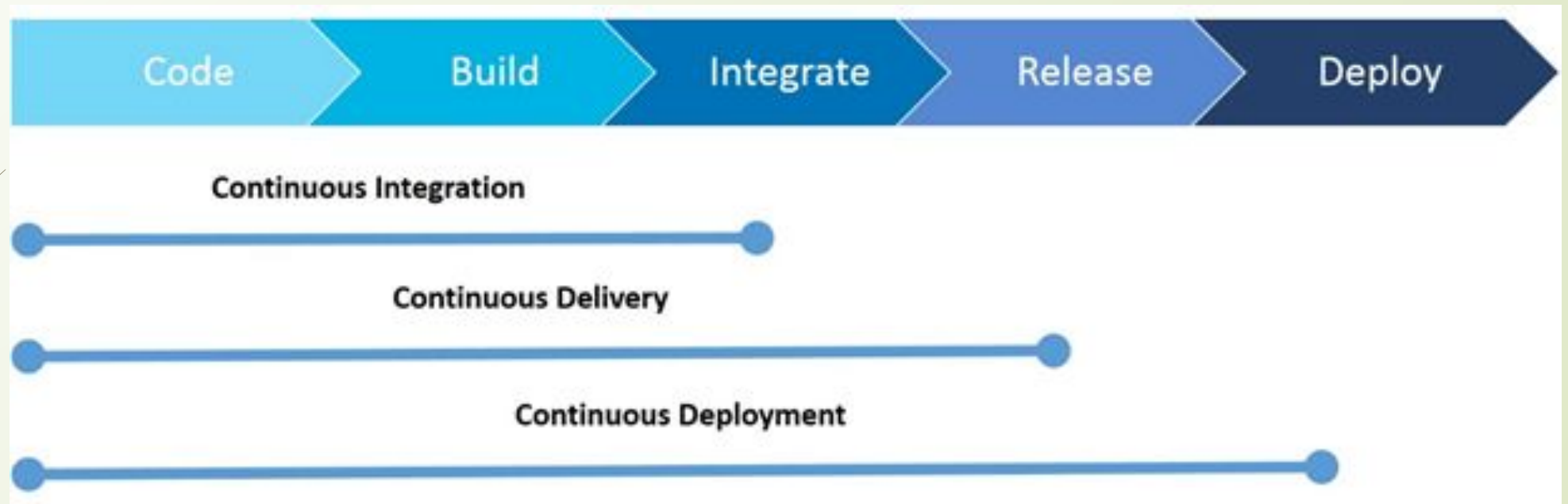
Devops

CI/CD



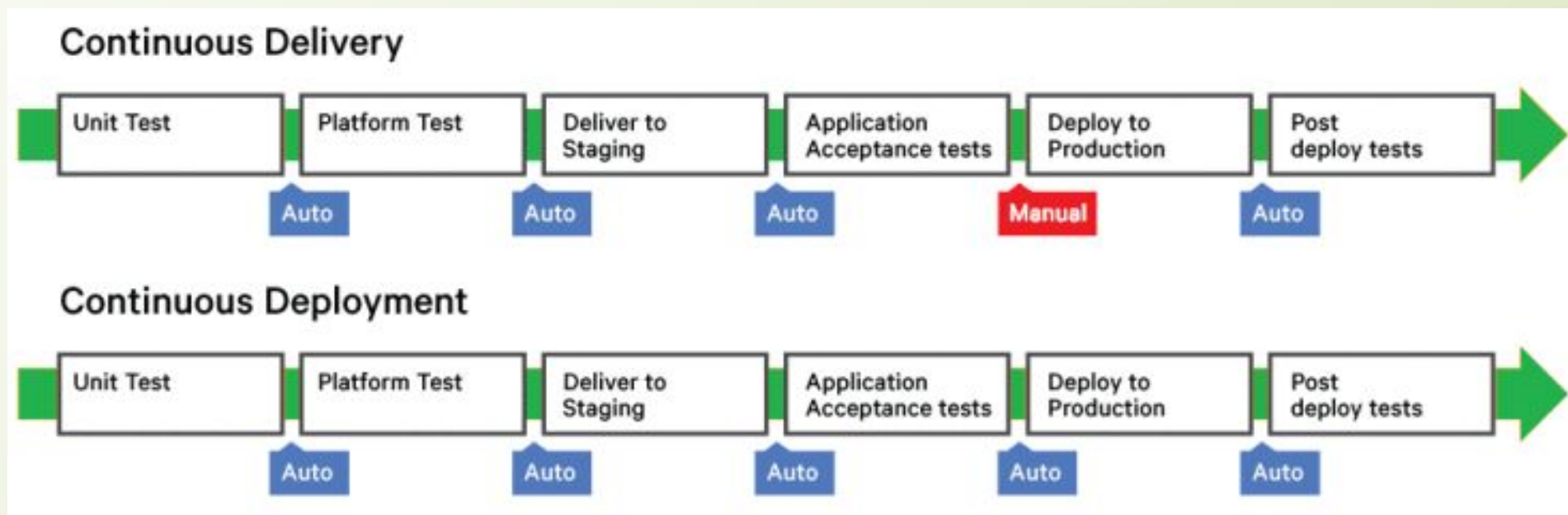
Devops

Différence CDelilvery/CDeploymentment



Devops

Différence CDelilvery/CDeployment



Devops

CI, Comment?

- Un **serveur de configuration des versions logicielles (SCM)** qui va permettre aux développeurs de travailler sur leur poste tout en centralisant au fur et à mesure les modifications effectuées et gérer ainsi toutes les évolutions ; dès qu'un nouveau code est jugé prêt, il est "poussé" vers le serveur d'intégration. Les outils SCM les plus répandus sont actuellement **Git (et Github) et Subversion (SVM)**, idéalement dans un mode collaboratif.
- Un **serveur d'intégration (CI)** qui vient récupérer le code et va réaliser une série de tests et, si le résultat est au vert, l'intégrer dans une nouvelle branche du code qui sera poussée à la fois sur le serveur SCM et vers la recette. Les outils phares sont ici **Jenkins, CircleCI, CodeShip ou TravisCI**.
- Ainsi les développeurs peuvent se concentrer sur leurs actions de conception/développement. Ils disposent d'un serveur pour gérer leurs sources tout en travaillant de manière décentralisée et, dès qu'un lot de code est prêt, il est traité par le serveur de CI sans nécessité de planification et en réduisant les tâches manuelles au minimum.

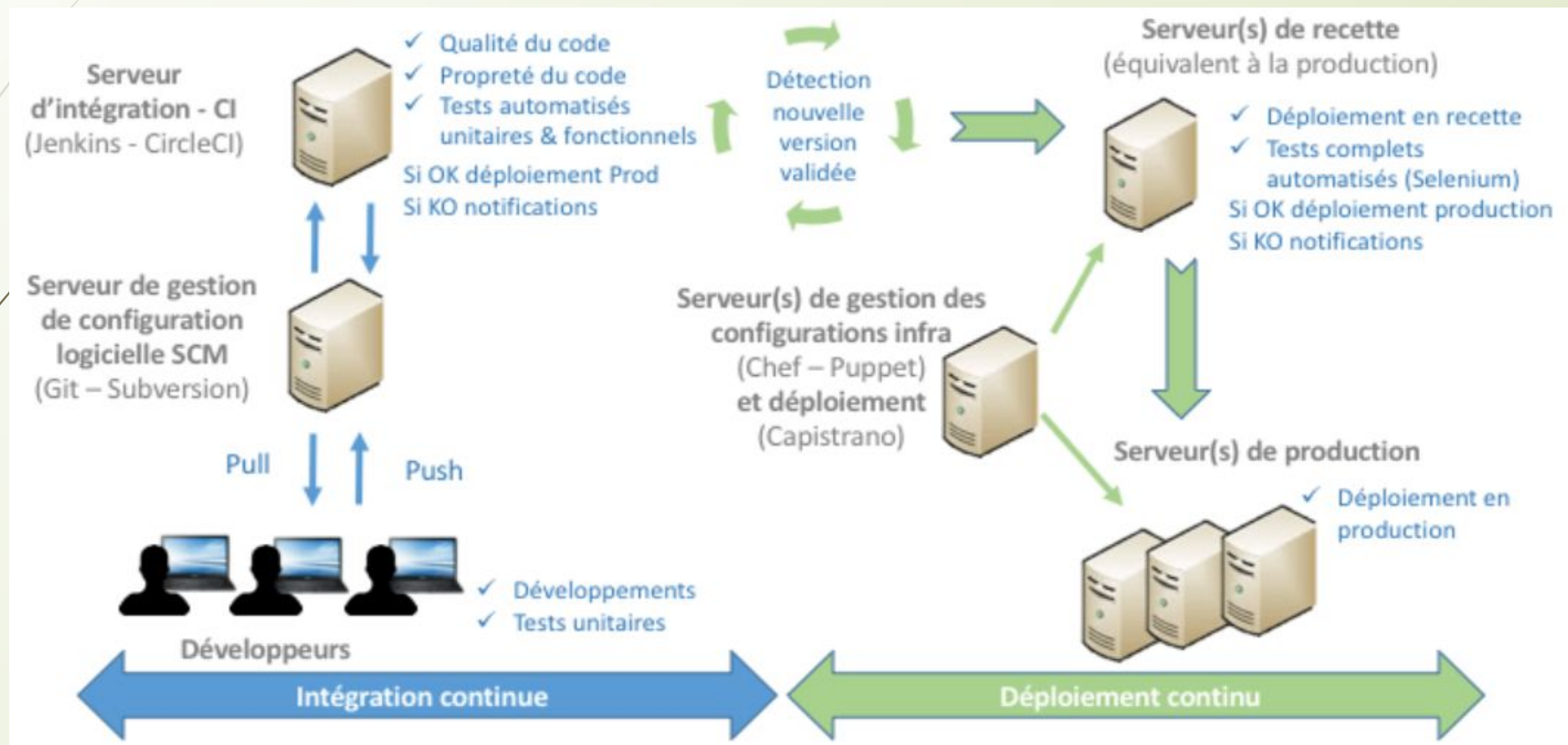
Devops

CD Comment?

- Un **serveur de configuration d'infrastructures** sur lequel seront stockées les modèles de configuration des serveurs (production et recette), ainsi tout nouveau serveur provisionné l'est bien selon le modèle en cours. Les outils utilisés sont **Chef, Puppet, SaltStalk, ou Ansible**.
- Une **automatisation maximale des tests** de recette, via un outil comme **Selenium**, qui permet d'enregistrer et dérouler des scénarios complets de test.
- Un **serveur de déploiement** qui va orchestrer les opérations de déploiement sur le serveur de recette, déclencher les tests, puis lancer le passage en production si ces derniers sont positifs, Parmi les outils d'orchestration fréquemment utilisés pour les applications web, on peut citer **Spinnaker**.

Devops

CI/CD



Devops

Rôle des « Ops »

- Avant Devops:
 - Réaliser les opérations de déploiement,
- Avec Devops :
 - Créer des modèles de configurations
 - Créer des chaînes d'automatisation,
 - Mettre à jour ces chaines
 - Superviser

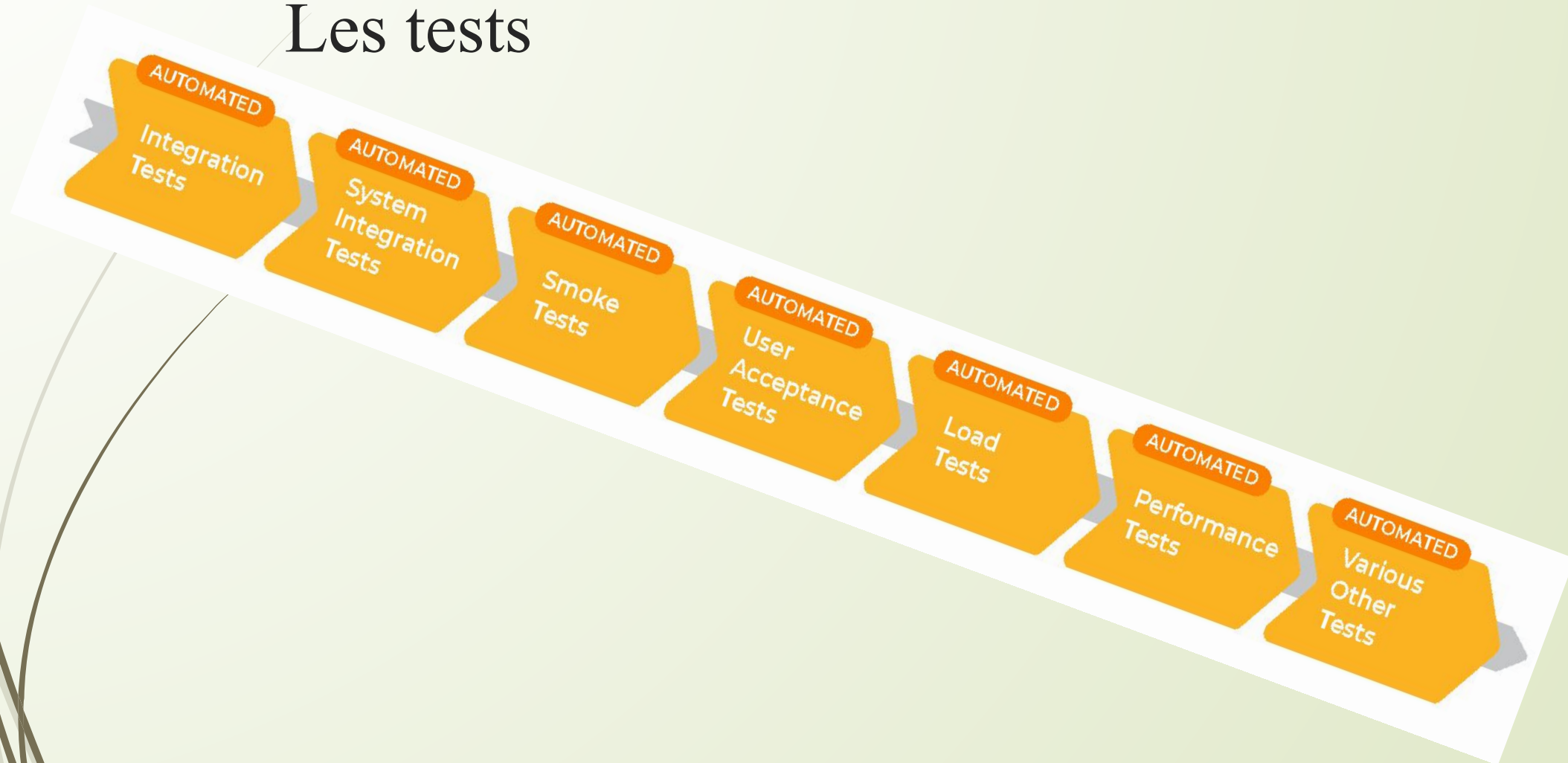
Devops

Outils CI

- Jenkins
- Travis CI
- Appveyor
- Circle CI
- Gitlab CI

Devops

Les tests



Devops

Outils de test

- Junit
- Selenium

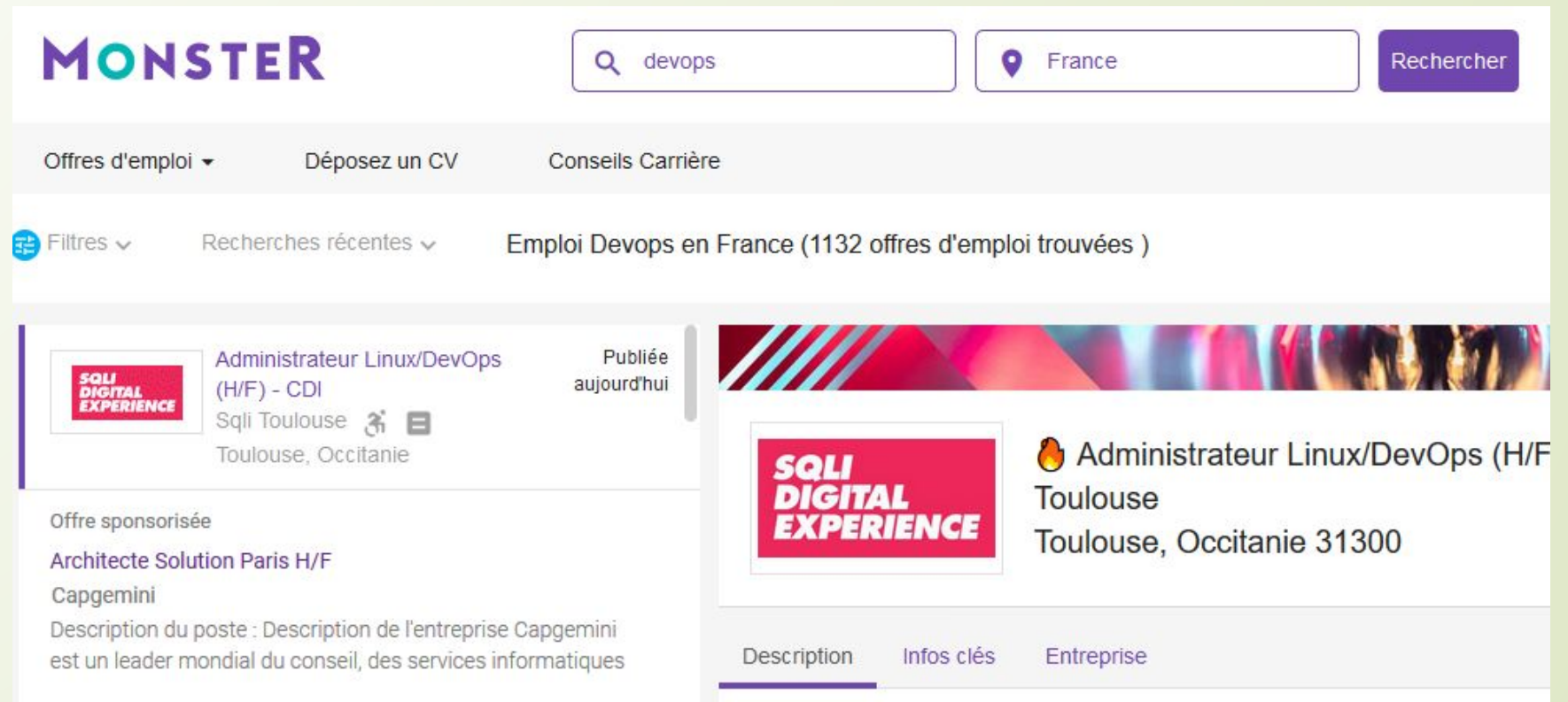
Devops

Outils de Collaboration

- Slack
- Microsoft Teams
- Mattermost
- Stack Overflow

Devops

Le marché de travail



The screenshot displays the Monster job portal interface. At the top, the Monster logo is on the left, and a search bar contains the text 'devops'. To the right of the search bar is a location filter set to 'France' and a 'Rechercher' button. Below the search bar, there are navigation links: 'Offres d'emploi', 'Déposez un CV', and 'Conseils Carrière'. A secondary navigation bar includes 'Filtres', 'Recherches récentes', and a summary 'Emploi Devops en France (1132 offres d'emploi trouvées)'. The main content area shows two job listings. The first listing is for 'Administrateur Linux/DevOps (H/F) - CDI' at 'Sql Toulouse' in 'Toulouse, Occitanie', published 'aujourd'hui'. It features a 'SQL DIGITAL EXPERIENCE' logo and is marked as an 'Offre sponsorisée'. The second listing is for 'Administrateur Linux/DevOps (H/F)' at 'Toulouse, Occitanie 31300', also featuring the 'SQL DIGITAL EXPERIENCE' logo. Below the listings, there are tabs for 'Description', 'Infos clés', and 'Entreprise'.

MONSTER

devops France Rechercher

Offres d'emploi Déposez un CV Conseils Carrière

Filtres Recherches récentes Emploi Devops en France (1132 offres d'emploi trouvées)

SQL DIGITAL EXPERIENCE Administrateur Linux/DevOps (H/F) - CDI
Sql Toulouse Toulouse, Occitanie
Publiée aujourd'hui

Offre sponsorisée
Architecte Solution Paris H/F
Capgemini
Description du poste : Description de l'entreprise Capgemini est un leader mondial du conseil, des services informatiques

SQL DIGITAL EXPERIENCE Administrateur Linux/DevOps (H/F)
Toulouse
Toulouse, Occitanie 31300

Description Infos clés Entreprise

Cloud Computing

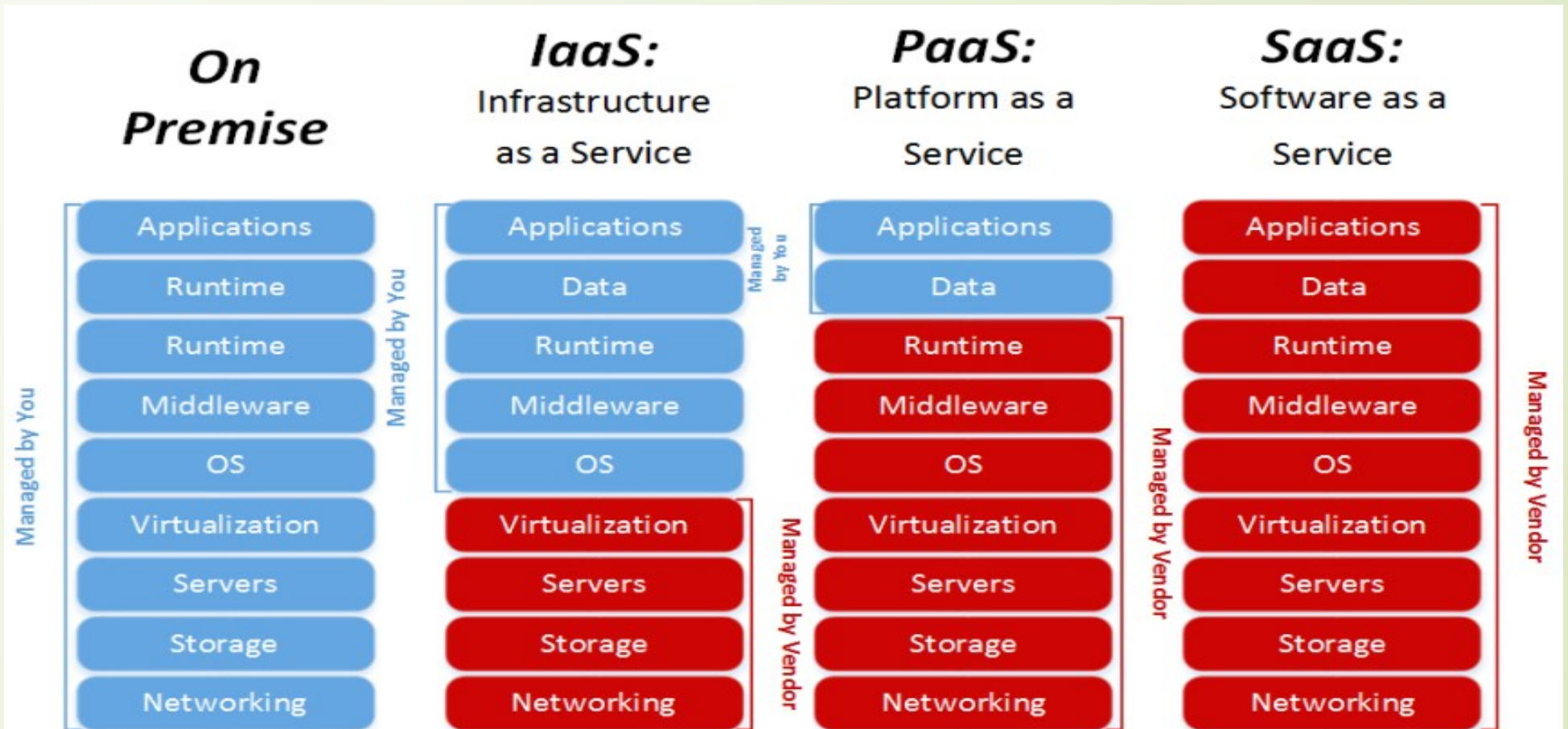


Cloud

Définition

- Le cloud computing ou informatique en nuage est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée. L'ordinateur de bureau ou portable, le téléphone mobile, la tablette tactile et autres objets connectés deviennent des points d'accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le cloud se caractérise également par sa souplesse qui permet aux fournisseurs d'adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs. [futura-sciences]

Cloud Services



Cloud

Avantage d'utilisation

- C'est un écosystème permettant d'améliorer
 - La réduction des coûts
 - L'évolutivité/Scalabilité
 - La sécurité
 - La fiabilité
 - La continuité des activités

Cloud

Déploiement

- ❑ **Le cloud public** : les ressources informatiques de l'entreprise sont stockées sur un serveur mutualisé, autrement dit partagé entre plusieurs clients, et accessibles par Internet. Ces serveurs sont partitionnés de manière à interdire les fuites de données.
- ❑ **Le cloud privé** : comme son nom l'indique, il est dédié à un seul utilisateur. L'avantage du cloud privé est son important niveau de sécurité, renforcé par une connexion VPN. Le cloud privé est administré par l'entreprise elle-même ou un prestataire de services.
- ❑ **Le cloud hybride** : l'entreprise utilise à la fois le cloud privé et le cloud public pour mettre en œuvre certaines activités. Par exemple, le cloud public est utilisé par les collaborateurs pour les tâches opérationnelles, tandis que le cloud privé sert à héberger le site web e-commerce de l'entreprise ou ses données financières, pour réduire le risque de piratage.

Cloud

Recettes et prévisions

	2018	2019	2020	2021	2022
Cloud Business Process Services (BPaaS)	41.7	43.7	46.9	50.2	53.8
Cloud Application Infrastructure Services (PaaS)	26.4	32.2	39.7	48.3	58.0
Cloud Application Services (SaaS)	85.7	99.5	116.0	133.0	151.1
Cloud Management and Security Services	10.5	12.0	13.8	15.7	17.6
Cloud System Infrastructure Services (IaaS)	32.4	40.3	50.0	61.3	74.1
Marché total	196.7	227.8	266.4	308.5	354.6

Cloud Les Leaders

Figure 1. Magic Quadrant for Cloud Infrastructure and Platform Services



Cloud Comapratif

Fournisseur	Chiffre d'affaire	Points forts	Faiblesses
Amazon Web Services	<ul style="list-style-type: none">• 25,7 milliards de dollars en 2018	<ul style="list-style-type: none">• Un vaste ensemble d'outils qui continue de croître de manière exponentielle• Les fonctionnalités offertes par Amazon Web Services sont incomparables.	<ul style="list-style-type: none">• Orientation quasi unique vers le cloud public• La complexité de la tarification et des coûts
Microsoft Azure	<ul style="list-style-type: none">• 23,2 milliards de dollars en 2018	<ul style="list-style-type: none">• Microsoft Azure dispose d'une infrastructure cloud très performante qui est très orientée vers et pour les entreprises, l'écosystème Windows et son cloud hybride est sa force.	<ul style="list-style-type: none">• L'assistance technique, la documentation disponible, et l'offre de formation
Google Cloud Platform	<ul style="list-style-type: none">• 6,8 milliards de dollars en 2018	<ul style="list-style-type: none">• Le Deep Learning, l'intelligence artificielle, le machine Learning, l'analyse de donnée et l'engagement en faveur de l'open source sont ses points forts	<ul style="list-style-type: none">• Son infrastructure n'est pas encore développée comme celles d'AWS et d'Azure.

Cloud Virtualisation

- Types de virtualisation
 - Virtualisation des serveurs
 - Virtualisation des systèmes d'exploitation
 - Virtualisation des postes de travail
 - Virtualisation des applications
 - Virtualisation du stockage
 - Virtualisation de réseau

Cloud Virtualisation

- Hyperviseurs

- [Hyper-V](#) · [VMware ESX](#) ([vSphere](#) et [vCloud](#)) · [Xen](#) · [KVM](#) · [Oracle VM](#)

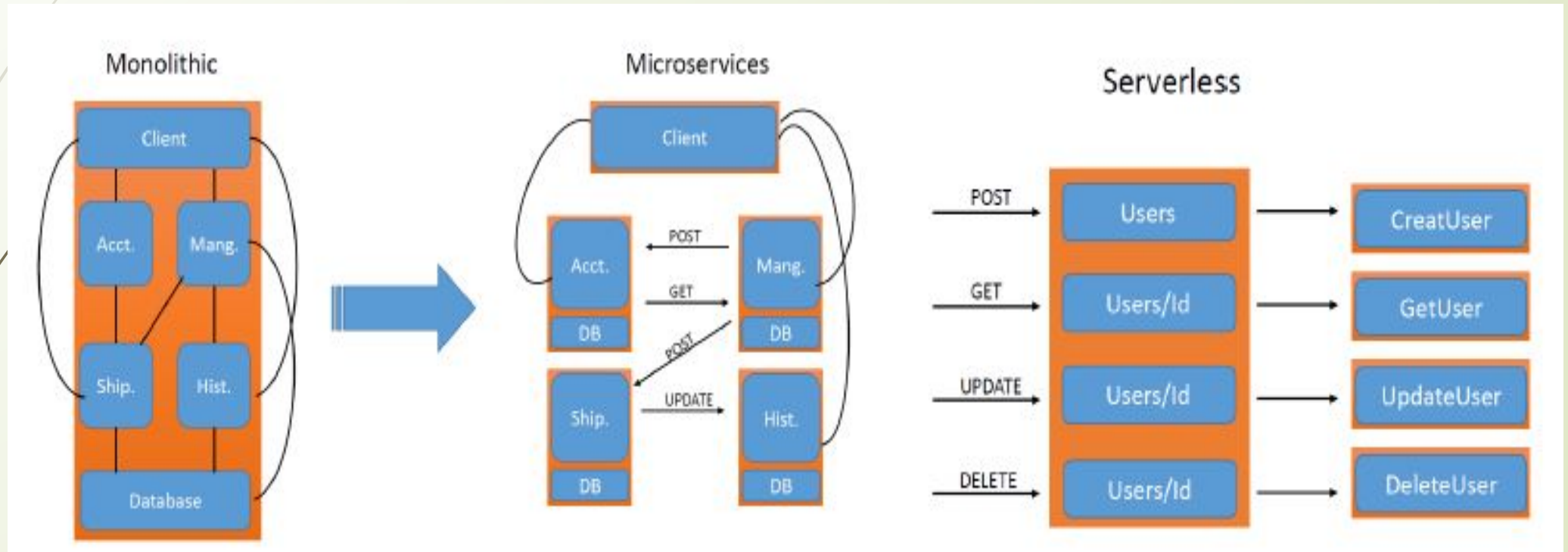
Devops et Cloud

Serverless

- ❑ Aucun serveur à provisionner
- ❑ Mise à l'échelle à l'usage
- ❑ Vous ne payez pas quand votre application attend
- ❑ Disponibilité et tolérance de panne intégrée

Devops et Cloud

Serverless et Microservices



Devops & Cloud



Devops et Cloud AWS



Hébergement Git privé AWS CodeCommit

AWS CodeCommit est un service de [contrôle de source](#) entièrement géré, qui permet aux entreprises d'héberger facilement des référentiels Git privés sécurisés et hautement scalables. Vous pouvez utiliser CodeCommit pour stocker tous les éléments que vous souhaitez en toute sécurité, du code source aux fichiers binaires. En outre, cet outil fonctionne parfaitement avec les outils Git existants. [En savoir plus »](#)

Devops et Cloud AWS



Flux de travail de communiqué
logiciel
AWS CodePipeline



Création et test de code
AWS CodeBuild



Automatisation du
déploiement
AWS CodeDeploy



Projets CI/CD unifiés
AWS CodeStar

Devops et Cloud AWS



Mise en service d'infrastructure avec des templates

AWS CloudFormation

AWS CloudFormation permet aux développeurs et aux administrateurs système de créer et de gérer facilement un ensemble de ressources AWS liées entre elles, de les mettre en service et de les actualiser de manière ordonnée et prévisible. Vous pouvez utiliser les exemples de modèle d'AWS CloudFormation ou créer les vôtres.

Devops et Cloud Azure



Application Insights

Full observability into your applications, infrastructure, and network



Azure Artifacts

Create, host, and share packages with your team



Azure Boards

Plan, track, and discuss work across your teams



Azure DevOps

Services for teams to share code, track work, and ship software



Azure DevTest Labs

Quickly create environments using reusable templates and artifacts



Azure Monitor

Full observability into your applications, infrastructure, and network



Azure Pipelines

Continuously build, test, and deploy to any platform and cloud



Azure Repos

Get unlimited, cloud-hosted private Git repos for your project



Azure Test Plans

Test and ship with confidence with a manual and exploratory testing toolkit

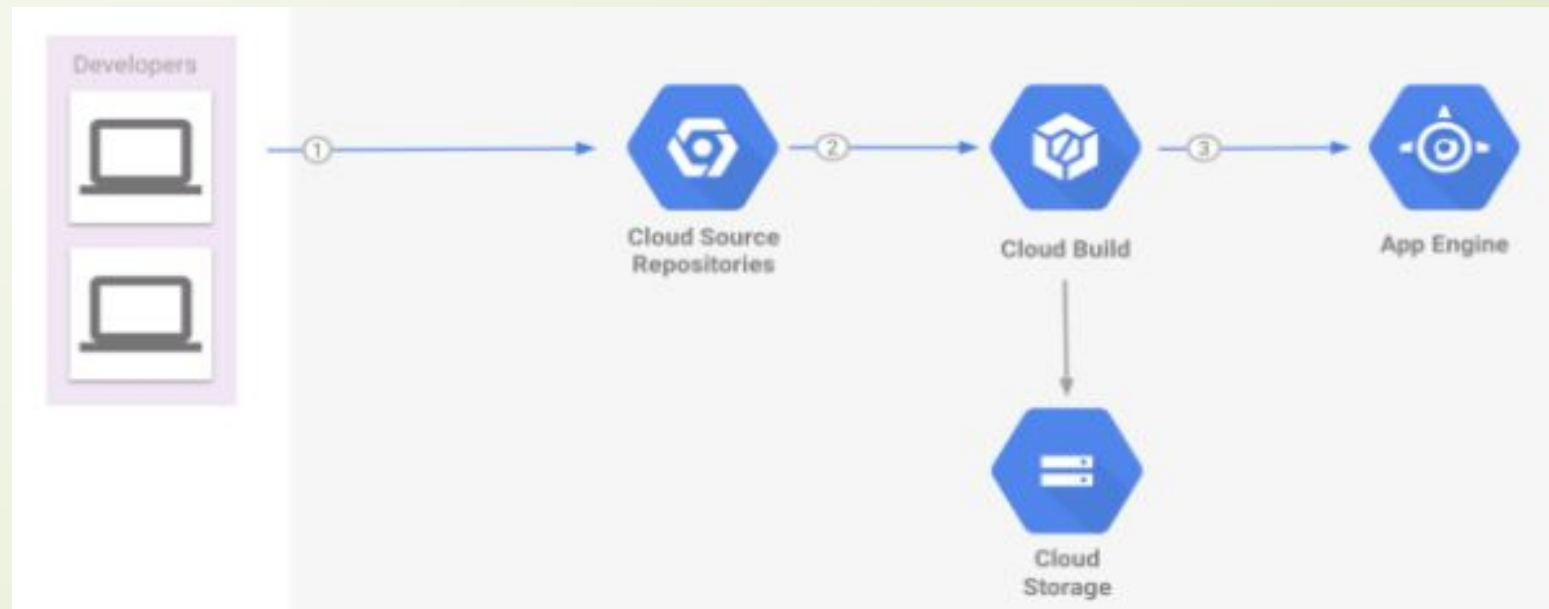


GitHub Actions for Azure

Build, test and deploy any app from GitHub to Azure

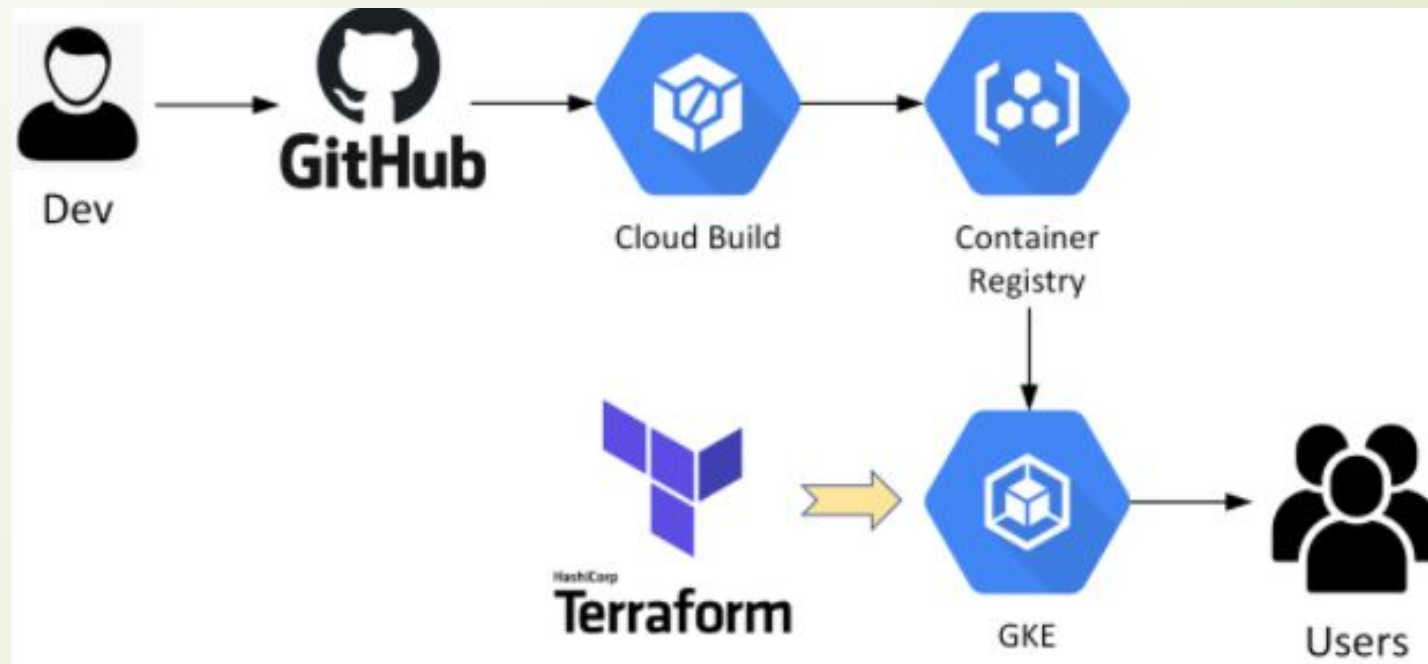
Devops et Cloud google

- ❑ Cloud Source Repositories → source
- ❑ Cloud Build → Build et test
- ❑ Container Registry & cloud Storage → Artifact Storage
- ❑ App Engine → Delivery



Devops et Cloud google

□ Google Kubernetes Engine



Fin de la journée

Rappel sur les thématiques abordées

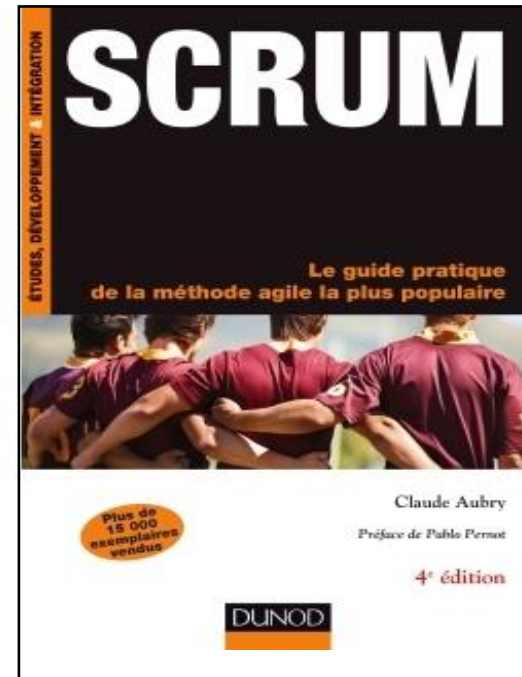
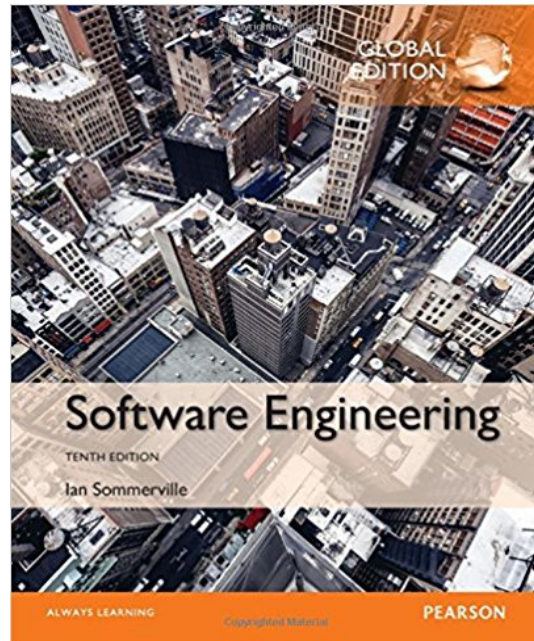
- Agile et motivation pour Devops
- Principes de Devops
- Les chemins de Devops
- CI/CD
- Clouds et principes
- Les acteurs dominant cloud

A photograph of a bright blue sky with several white, fluffy clouds. The clouds are scattered, with a large, prominent one in the upper center and several smaller ones below it. The text is overlaid on the bottom right of the image.

**Merci
pour
votre Attention**

Les processus agiles - SCRUM

Sources



Les processus dits agiles

Le manifeste agile -Valeurs

- Les individus et leurs interactions plus que les processus et les outils
- Des logiciels opérationnels plus qu'une documentation exhaustive
- La collaboration avec les clients plus que la négociation contractuelle
- L'adaptation au changement plus que le suivi d'un plan

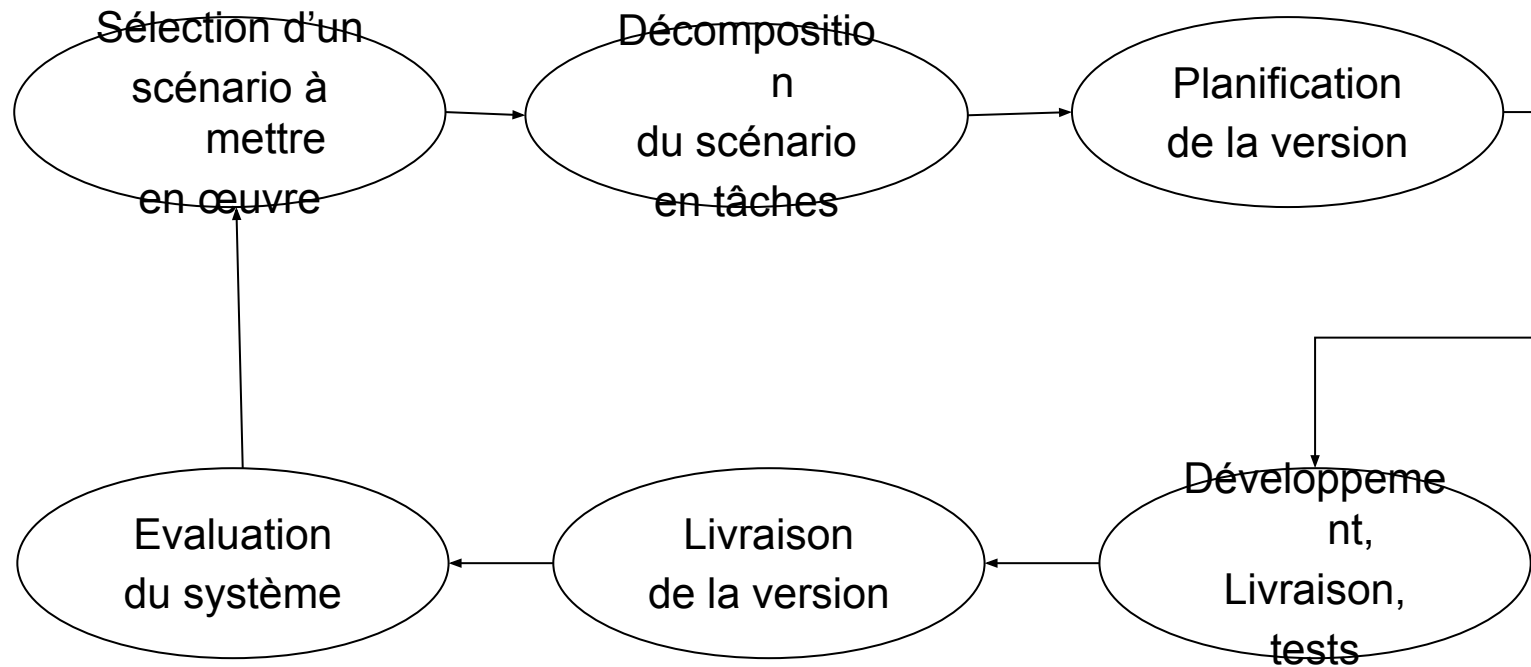
Le manifeste agile -Principes

- Satisfaction des clients
- Accepter le changement du besoin
- Livraison fréquentes
- Implication du client
- Motivation des équipes
- Le dialogue face à face
- Opérationnel sinon rien
- Rythme soutenable
- Excellence technique
- La simplicité
- Equipes auto-organisées
- Amélioration continue

Principes

- Ce sont des méthodes incrémentales
- Les activités de spécification, de conception et d'implantation sont entrelacées; on travaille avec des incréments de petite taille
- Le système est développé comme une succession de versions qui correspondent à l'ajout des incréments
- On prototype rapidement les interfaces du système pour avoir un retour rapide des utilisateurs

« Extreme programming »



XP – Terminology (a)

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

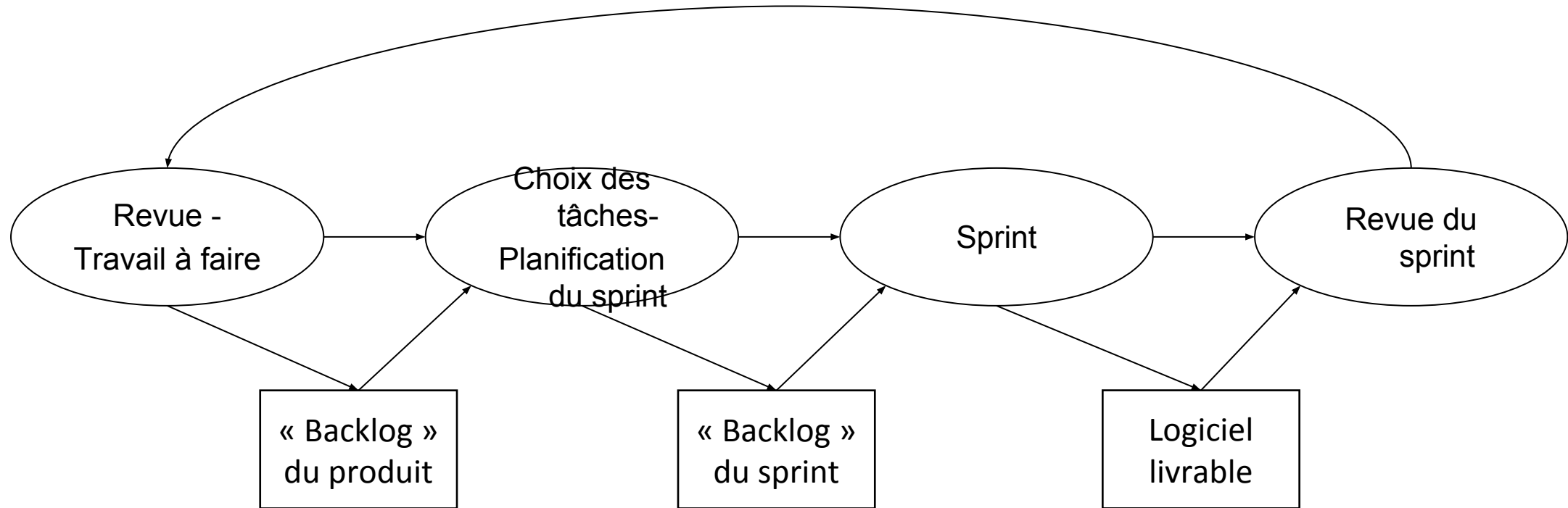
XP – Terminology (b)

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Scrum

- Scrum est une méthode agile qui se concentre sur la gestion itérative de projet en exploitant les propriétés créatives des membres de l'équipe de développement.
- On peut distinguer trois phases :
 - La phase initiale au cours de laquelle les fonctionnalités du système sont listées et une architecture logicielle générale est définie
 - Suit une série de "sprints", chaque sprint correspondant à un incrément du système
 - La phase de terminaison du projet développe les derniers artefacts (manuel d'utilisation ...) et tire les leçons apprises durant le développement.

Cycle de vie d'un sprint SCRUM



Scrum terminology (a)

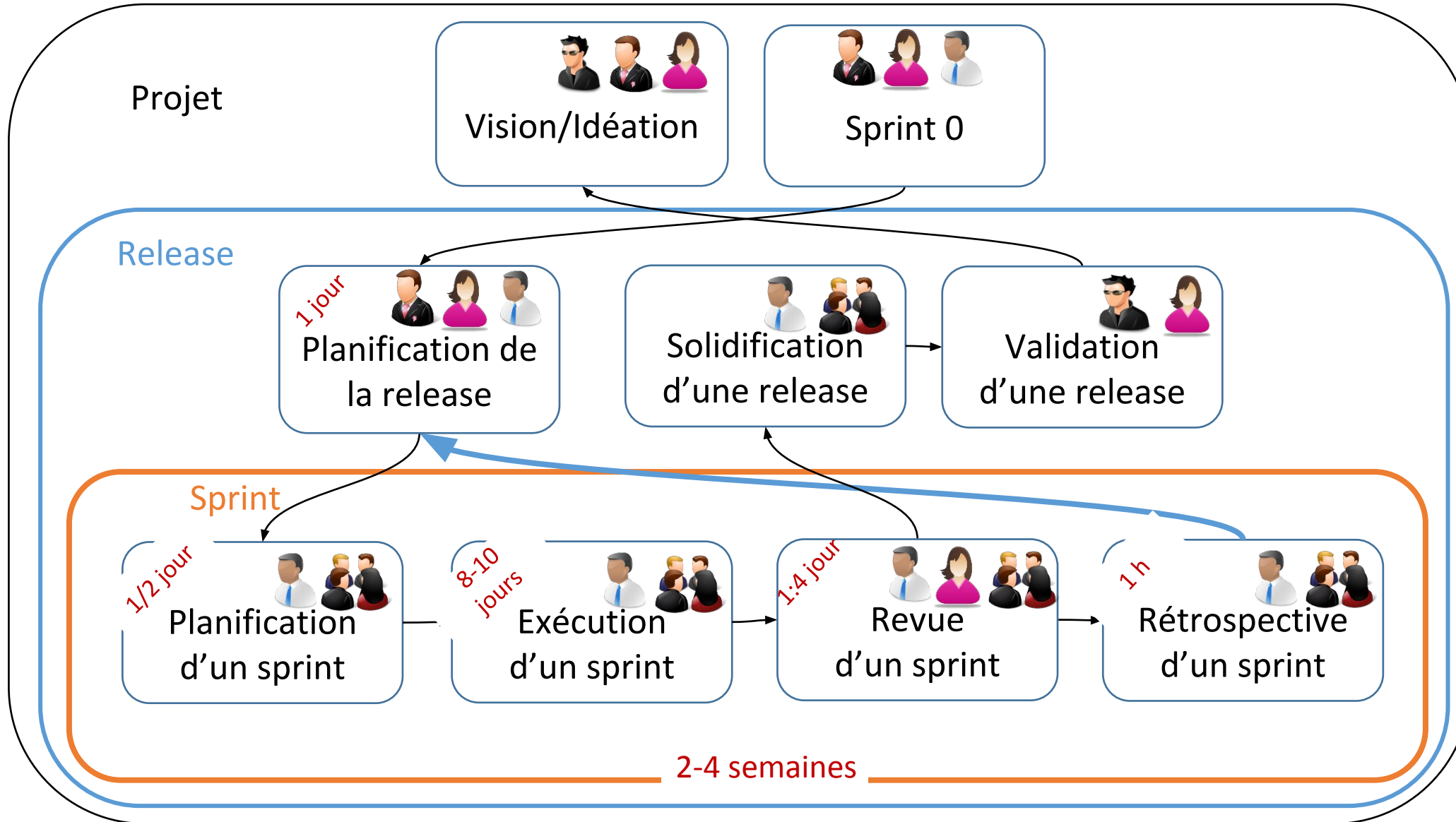
Scrum term	Definition
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

Scrum terminology (b)

Scrum term	Definition
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
ScrumMaster	The ScrumMaster is responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the ScrumMaster should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.

Le processus logiciel SCRUM

Processus logiciel Scrum



Client- User



Manager



Product owner

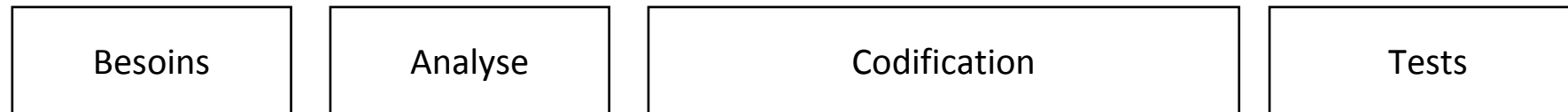


Scrum master

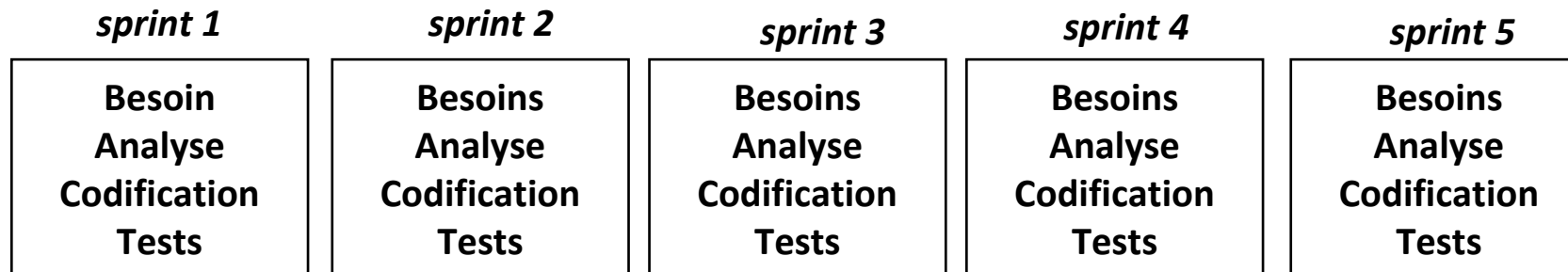


Développeur

Processus logiciel SCRUM (*sprints* et *releases*)



Cycle séquentiel



Cycle SCRUM

Processus logiciel SCRUM

- Une phase d'idéation
- Une succession de sprints de taille fixe
- Chaque *sprint* produit un logiciel fonctionnel
- Chaque *sprint* réalise toutes les activités de développement logiciel
- Les *sprints* sont groupés en *Releases* (de taille fixe)

Release SCRUM vs. Release traditionnelle

- Traditionnellement, une *release* (livrable) correspond à une nouvelle version d'un logiciel
 - Incrément fonctionnel (une *release* correspond à la réalisation d'un objectif fonctionnel)
 - Evolution technique
- SCRUM
 - Une *release* :
 - correspond à la livraison d'une *feature* (fonctionnalité)
 - travail réalisé pendant une période de temps fixe qui comprend plusieurs *sprints*
 - pour faire coïncider ces deux objectifs, on décompose/regroupe les *features*
 - Mais une nouvelle version du produit (livrable) peut-être produite à la fin de n'importe quel sprint

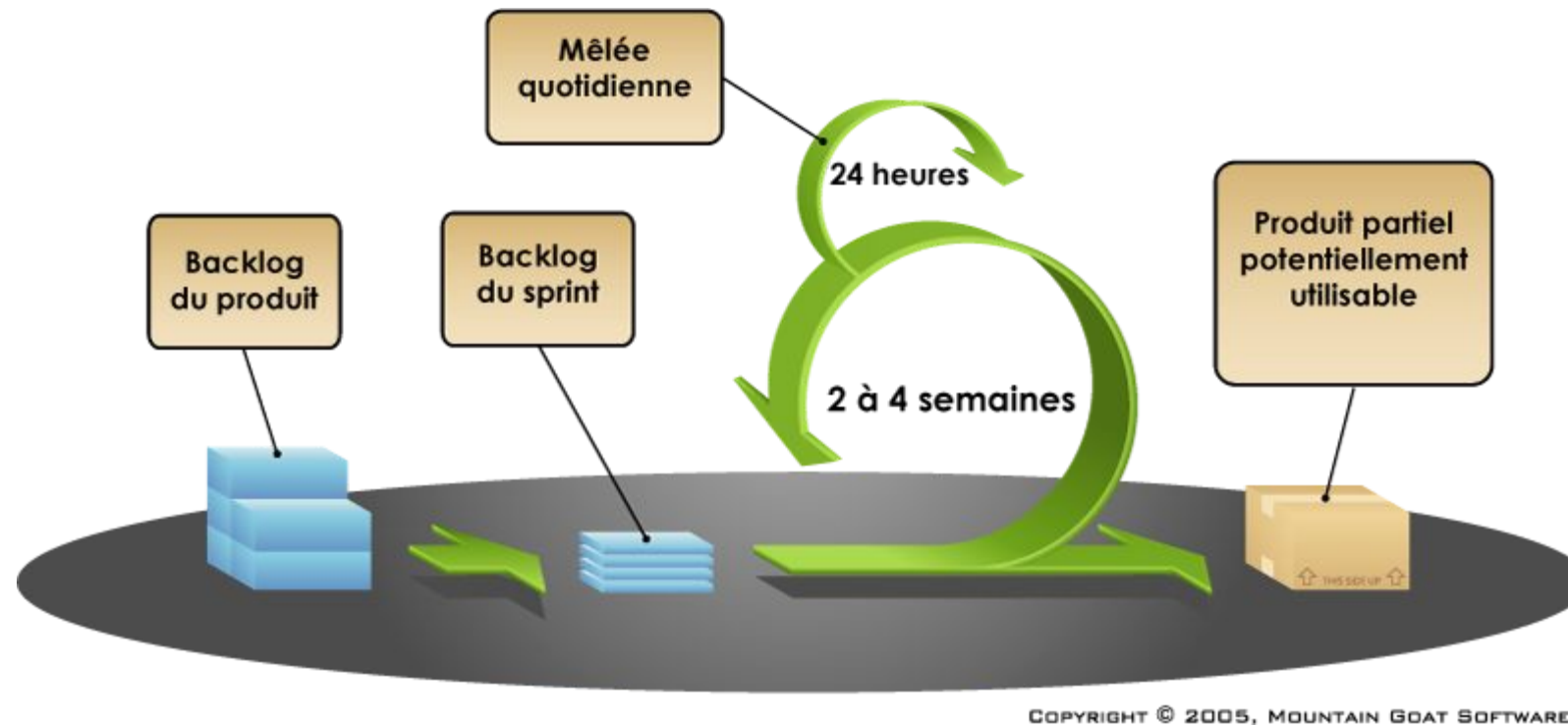
Périodes d'une *Release*

- La période avant le premier *sprint* de développement, appelée « sprint zéro » : planification de la *release* en plusieurs *sprints*
- La période des *sprints* (de développement)
- La période après le dernier *sprint* et avant la fin de la release

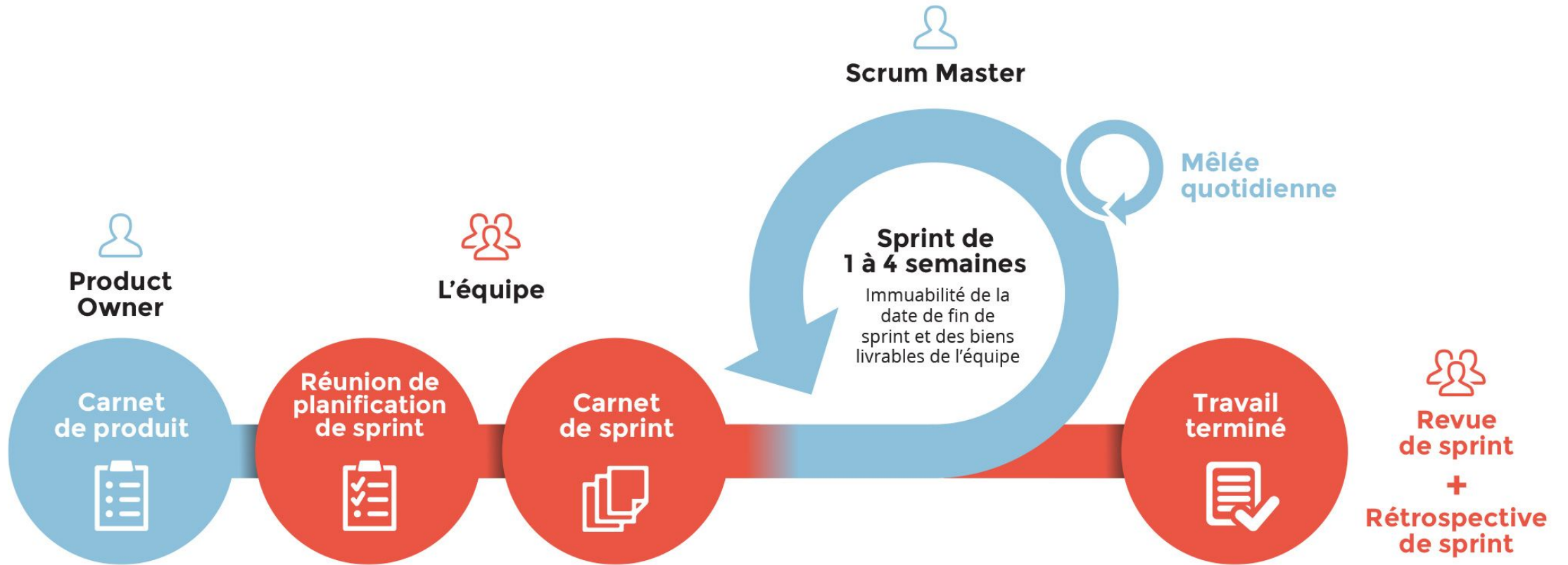
Sprint

- Chaque *release* est décomposée en *sprints* au cours de l'activité de planification de la *release* (la première fois « *sprint* zéro »)
- Un *sprint* est une période de développement de taille fixe (en moyenne 2 à 4 semaines)
 - Durée fixe, équipe stable
- Un *sprint* se décompose en un ensemble de tâches
- Mêlée journalière : réunion d'équipe pour faire le point sur le travail réalisé depuis le début du sprint et le travail à réaliser avant la fin du sprint
- Chaque *sprint* termine par :
 - une revue du produit
 - une rétrospective sur le processus

Sprint



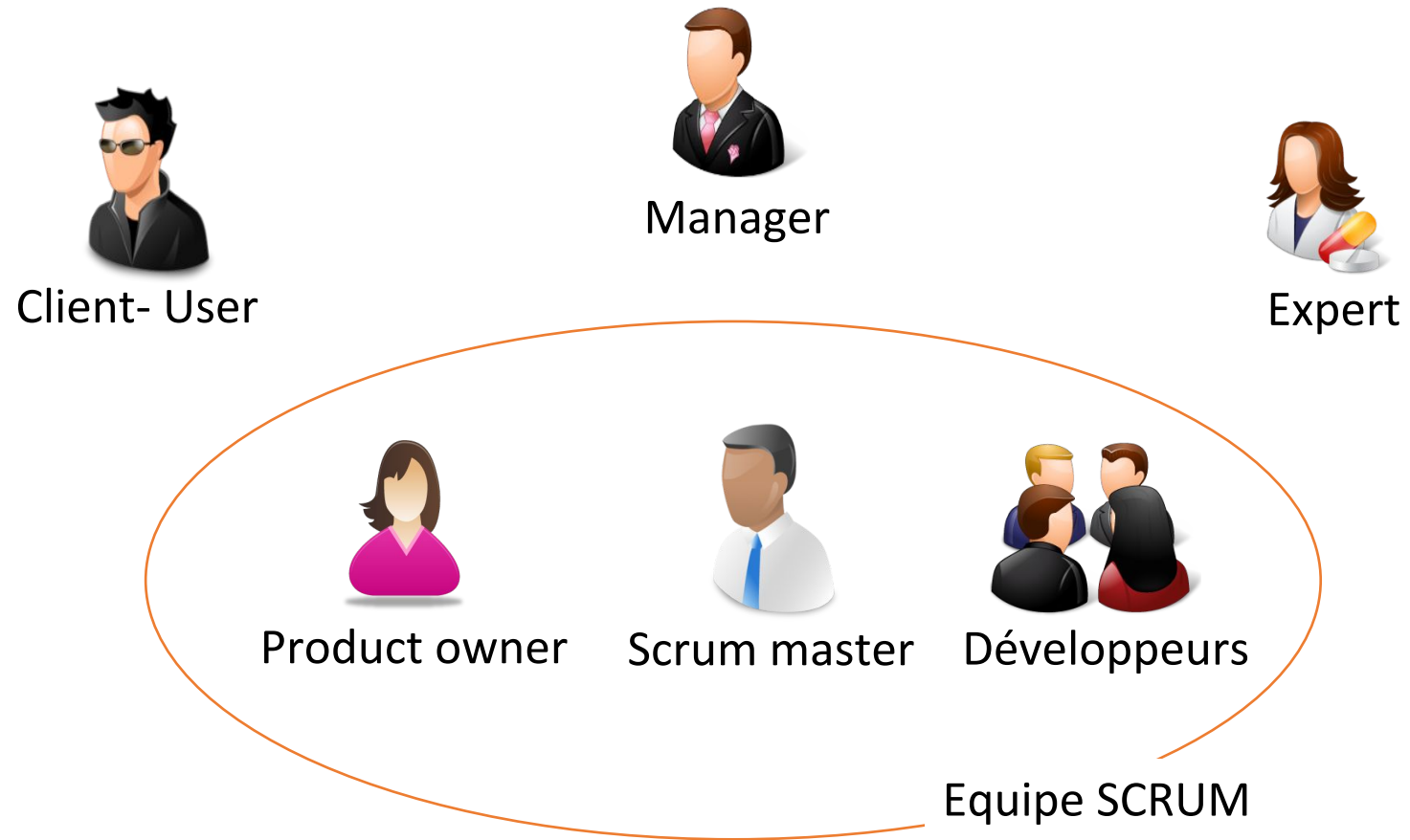
Backlog: carnet, liste ordonnée de « choses » à faire
(*user stories*, tâches)



Les acteurs

Des créateurs de valeur

Rôles



L'équipe

- Composition:
 - 1 Product Owner
 - 1 Scrum Master
 - 2 à 7 développeurs
 - Le product owner et le Scrum master peuvent aussi prendre le rôle de développeur
- Principes
 - Auto-organisation
 - Pluridisciplinarité
 - Stabilité
 - Valeurs communes

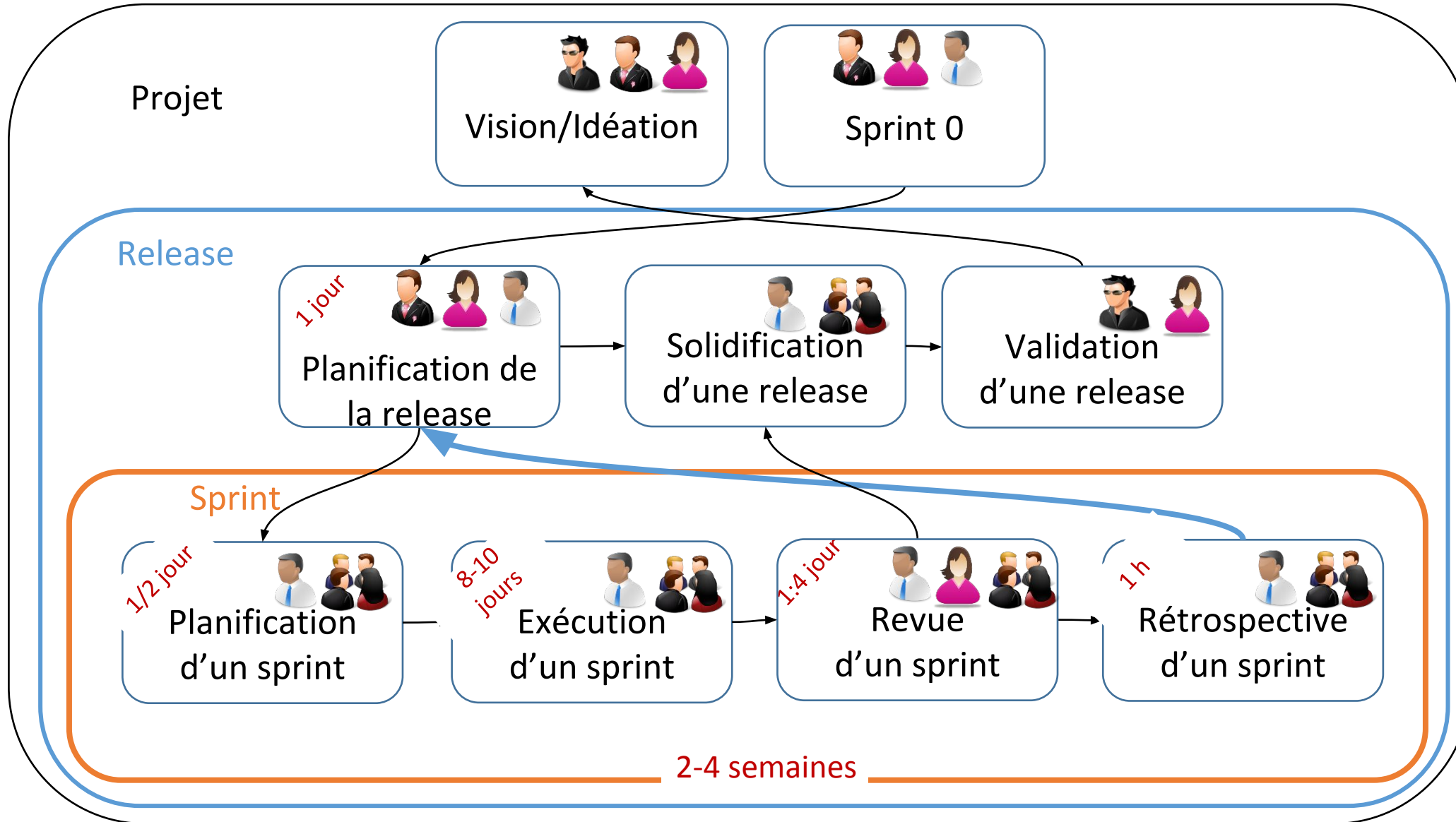
Product owner

- Responsabilités
 - Fait partager la vision globale du produit
 - Gère le backlog du produit (liste ordonnée des « choses » à faire)
 - Définit les priorités
 - Accepte ou rejette les *Releases* (livrables)

Scrum master

- Responsabilités
 - n'est pas « le chef », mais un facilitateur
 - Motive l'équipe
 - Fait appliquer les bonnes pratiques de Scrum
 - Gère les obstacles

Processus logiciel Scrum




Client- User


Manager


Product owner


Scrum master


Développeur

Les objets de SCRUM

Feature, Story, Tâche, Backlog

Feature

- Une *feature* (*fonctionnalité*) est un service ou une fonctionnalité du produit à développer
- Elle se décompose en *stories* (histoires) de tailles différentes.
On distingue :
 - les *stories* complexes (épiques) qui seront affinées en *stories* plus simples
 - Les *stories* atomiques qui ne se décomposent pas et sont réalisées dans les *sprints*

Feature (exemple)

un site pour propriétaires d'animaux domestiques
Des *features* (fonctionnalités, chapites ...)



<https://pablopernot.fr/2017/01/cartographie-plan-action/>

Workflow d'une *feature*

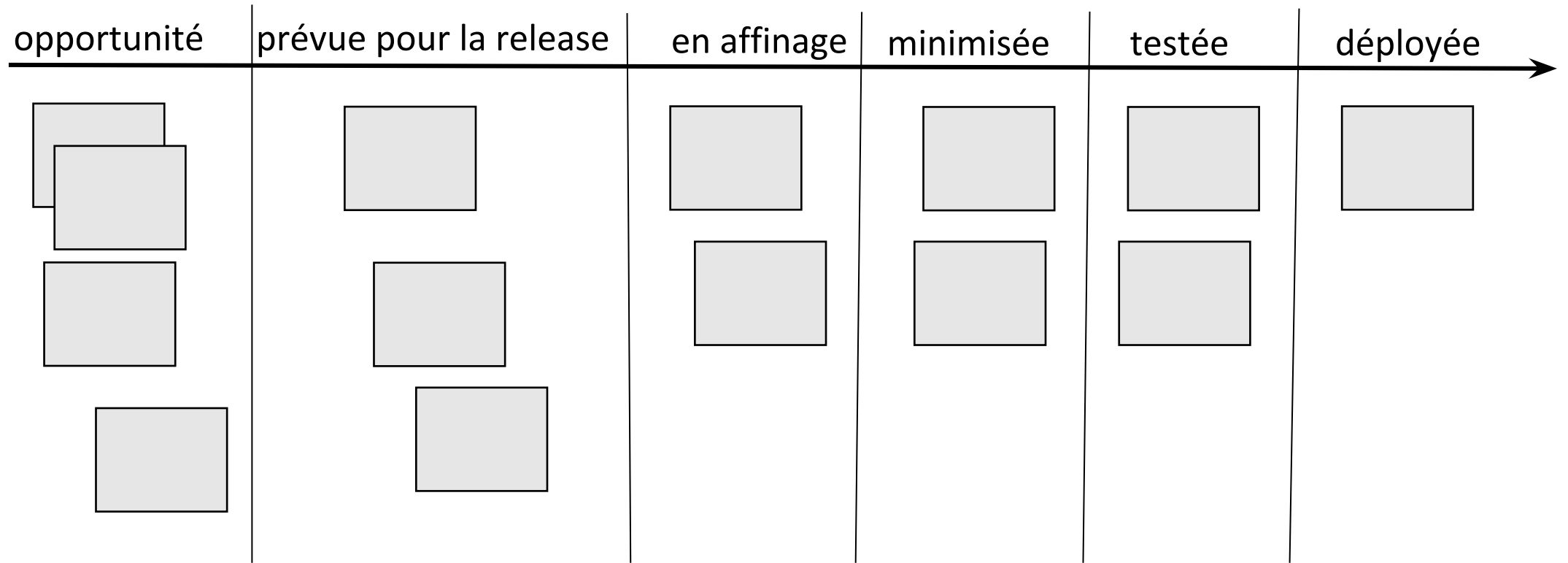


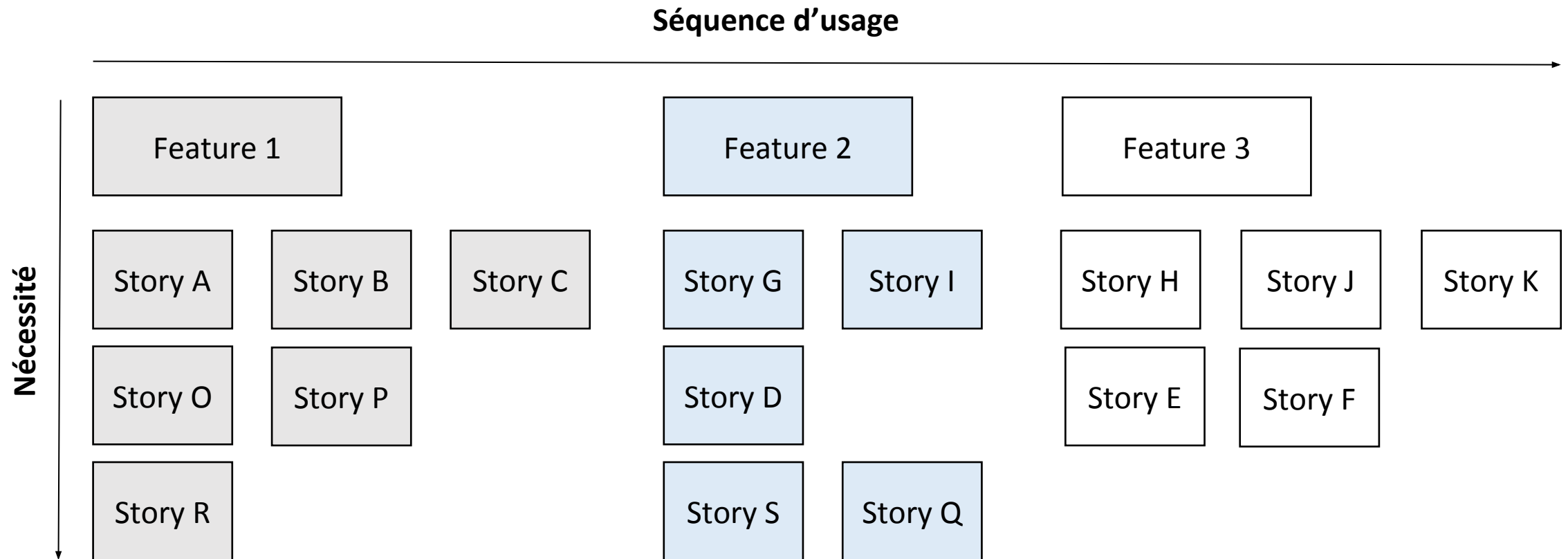
Tableau des *features*

Workflow d'une *feature*

- Opportunité : idée de feature qui présente une opportunité pour le produit
- Prévues pour la release : l'étude d'opportunité a abouti, et la feature est prévue pour la release en cours
- En affinage : initialement sous forme d'épique, est décomposée en stories
- Minimisée : *Minimal Marketable Feature*
- Testée : ... avec des stories finies
- Déployée : utilisable par des utilisateurs qui peuvent fournir du feedback

Story map :

décomposition d'une *feature* en *stories*



Exemple

POSSE- SSEUR	ANIMAL	BALADE	GARDIEN- NAGE	CONSEIL EXPERTISE	PUBLICITE	BOUTIQUE
Données Basiques	Données Basiques	Liste simple	Liste simple	fiches conseils vétérinaires	Encart pub	Catalogue
Données Avancées	Pédigree	Liste départs sur carte geoloc	Créer/mod gardiennage	fiches conseils éleveurs	Compteurs / stats	Panier
Ajout Média	Ajout média	Liste partici- pants	Note organisateur	Forum	Mailing	Paiement CB
Messagerie Tchat	Données avancées	Offrir/créer balade	Comment -aires	Tchat expert	Profiling	Paiement Paypal
Archivage		Visu parcours sur carte	Ajout média		Invitation événement	Paiement 3 fois
géoloc- alisation		Note organisateur				Coupon promo
		Comment -aires				Promo avancée
		Notifications				Visu export/ facture

<https://pablopernot.fr/2017/01/cartographie-plan-action/>)

Example :



Story

- Une *story* est une exigence du système à développer, formulée en une ou deux phrases dans le langage de l'utilisateur.
- Les *Stories* émergent au cours d'ateliers de travail menés avec le Métier, le Client et/ou les Utilisateurs.
- On distingue :
 - *Story* fonctionnelle
 - *Story* technique
 - Correction de bug
 - Remboursement de la « dette technique »

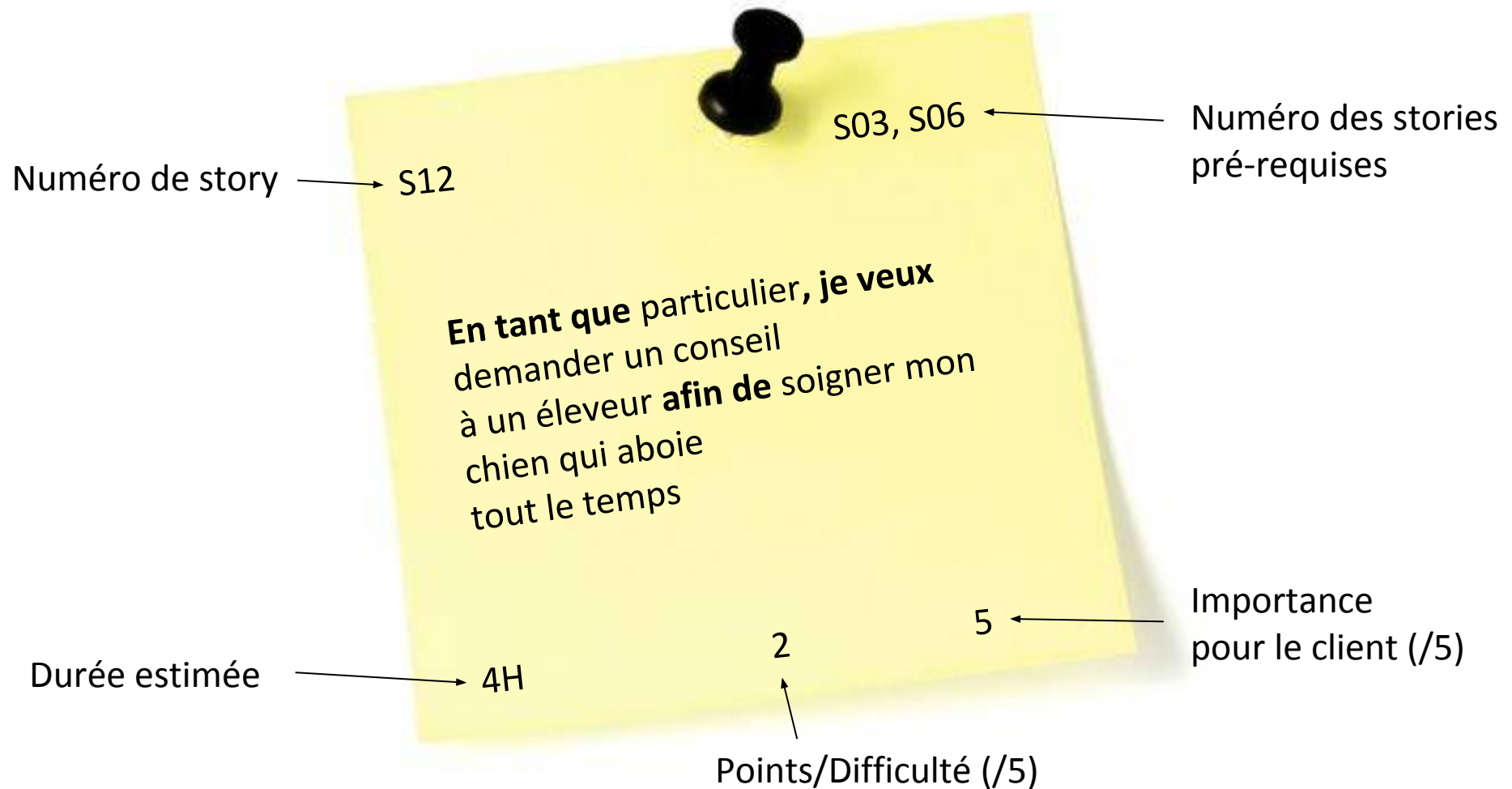
Story

- Les 3C
 - Carte : l'histoire est courte et sa description tient sur une carte (demi-page)
 - Conversation : l'histoire est définie avec les gens du métier
 - Confirmation : l'histoire est confirmée par des tests d'acceptation rédigés au même moment que celle-ci
- Workflow de la *story*
 - Idée d'une *story* rédigée sur une **C**arte (1/2 feuille)
 - **C**onversation dirigée par le *product owner* qui inclut les gens du métier
 - L'équipe apporte sa **C**onfirmation que la *story* est prête
 - L'équipe réalise la story
 - Le *product owner* apporte sa **C**onfirmation que la story est finie

Description type d'une story

- Plan type
 - **En tant que** <acteur>, **je veux** <un but> [**afin de** <une justification>]
 - **En tant que** client, **je veux** pouvoir accéder au site de ma banque **afin de** gérer mon compte sur Internet
- Priorité
- Nombre de points
 - Représente le niveau de difficulté intrinsèque, en fonction de la taille et de la valeur métier
 - Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la réutilisation possible de composants ...)
- Conditions d'acceptation
 - **Etant donné** <le contexte> **quand je** <événement> **alors** <résultat>
 - **Etant donné** que je suis sur la page de connexion et que j'ai entré un login et un mot de passe dans le formulaire et que le login et le mot de passe correspondent à un utilisateur enregistré, **quand je** clique sur le bouton "Se connecter" **alors** j'arrive sur la page d'accueil du site.

Post-it de story



Estimation des points d'une story

- Difficulté intrinsèque
 - Taille et valeur métier
 - Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la ré-utilisation de composants ...)
- Planning Poker (source Wikipedia)
 - Les participants s'installent autour d'une table, placés de façon que tout le monde puisse se voir.
 - Le responsable de produit explique à l'équipe un scénario utilisateur ([user story](#)).
 - Les participants posent des questions au responsable de produit, discutent du périmètre du scénario, évoquent les conditions de satisfaction qui permettront de le considérer comme "terminé".
 - Chacun des participants évalue la complexité de ce scénario, choisit la carte qui correspond à son estimation et la dépose, face vers le bas, sur la table devant lui.
 - Au signal du facilitateur, les cartes sont retournées en même temps.
 - S'il n'y a pas unanimité, la discussion reprend.
 - On répète le processus d'estimation jusqu'à l'obtention de l'unanimité.
 - Une procédure optimisée consiste, après la première "donne", de demander aux deux acteurs ayant produit les évaluations extrêmes d'expliquer leurs points de vue respectifs. Ces explications achevées et comprises de tous, une nouvelle estimation est produite et c'est alors la moyenne arithmétique de ces estimations qui est prise en compte.

Tâche

- A l'exécution, une story se décompose en tâches

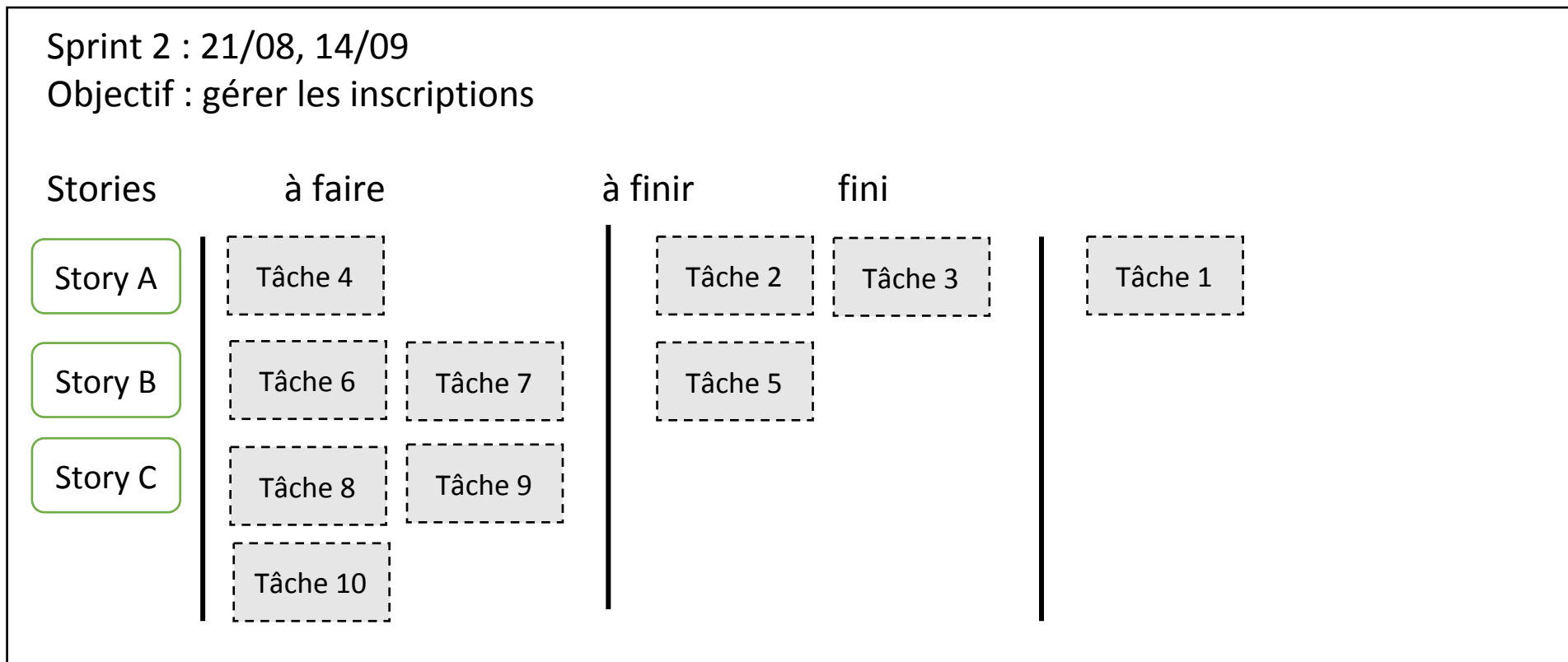
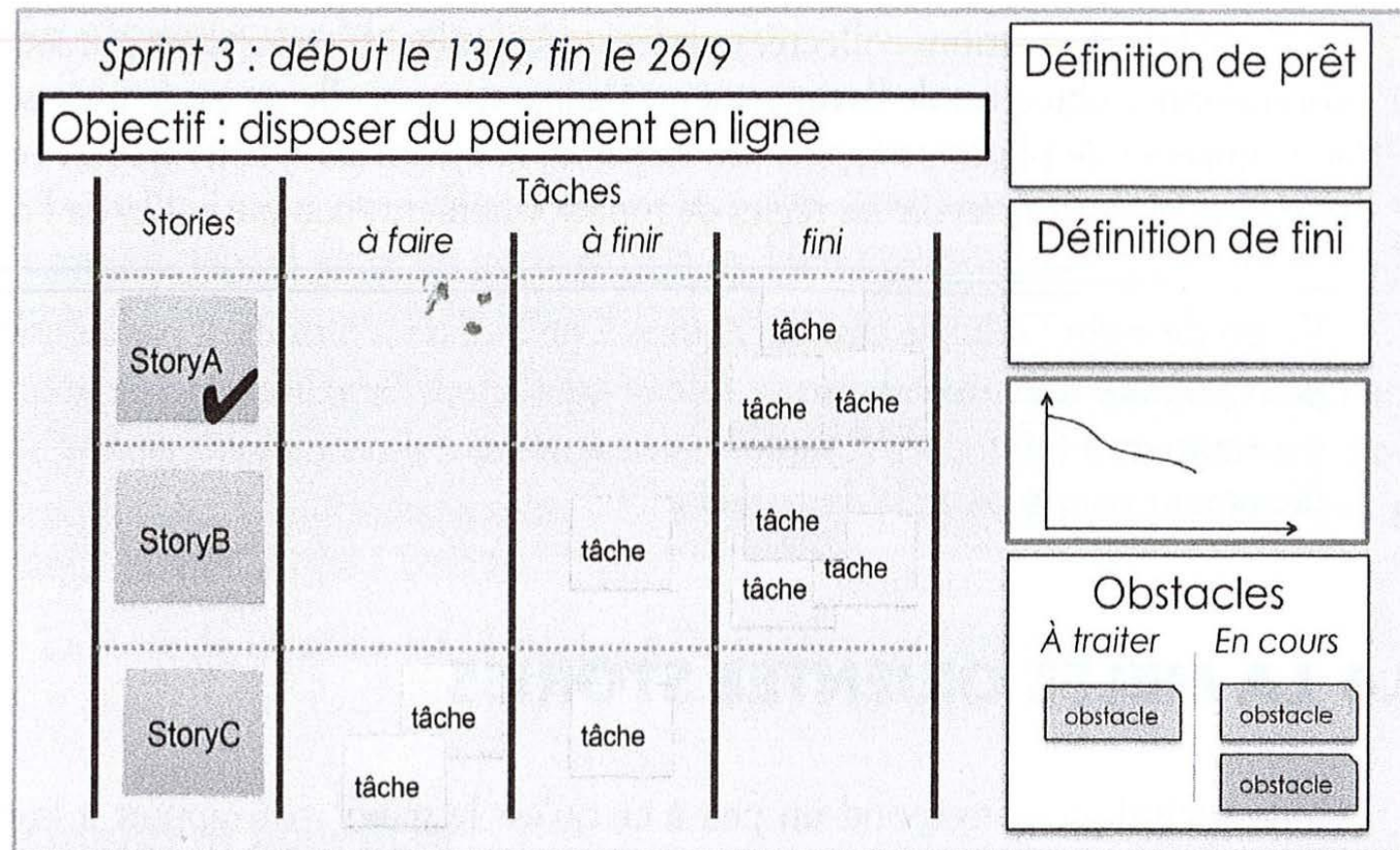


Tableau de la Story

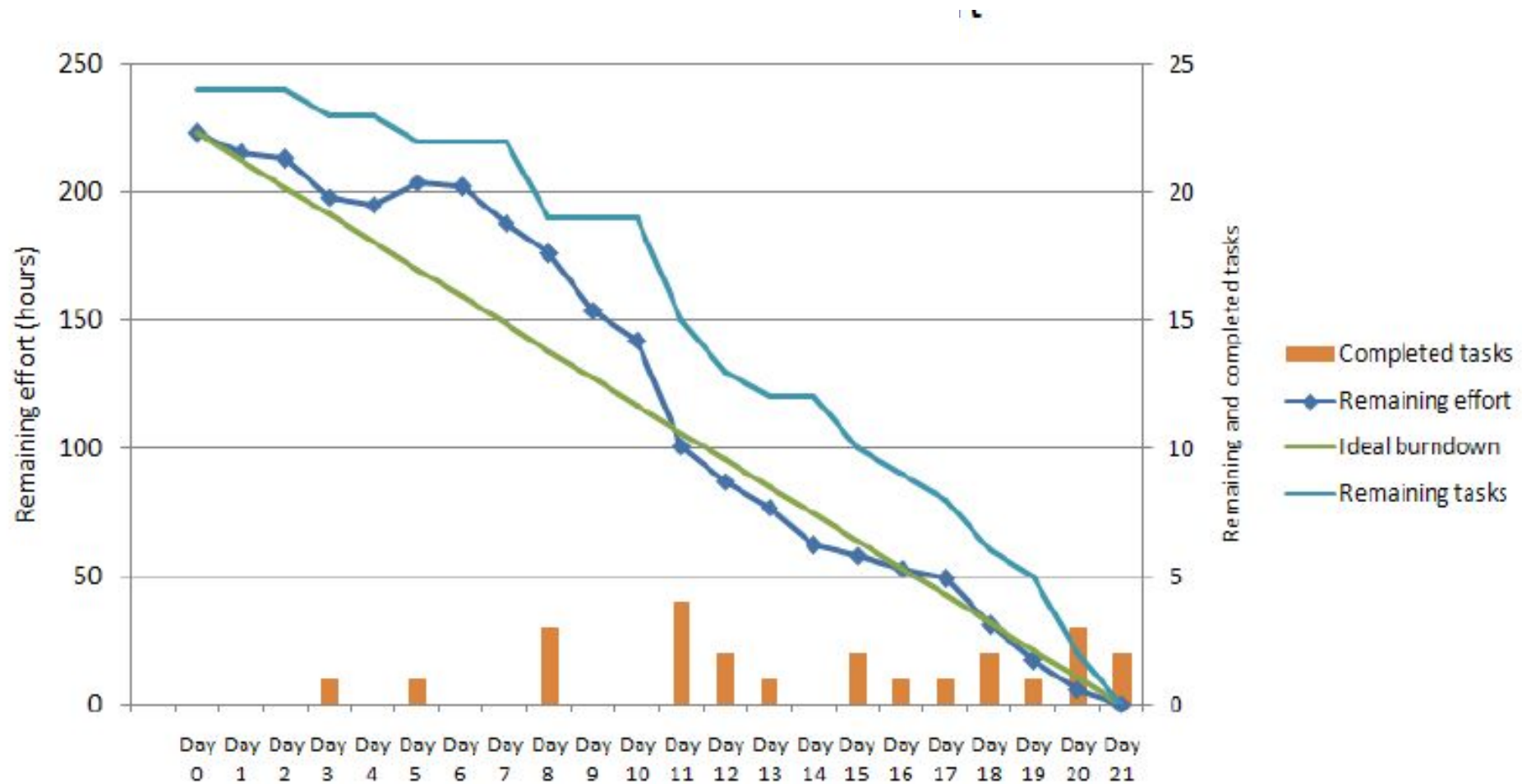
- Le tableau de la *story* décrit son état.



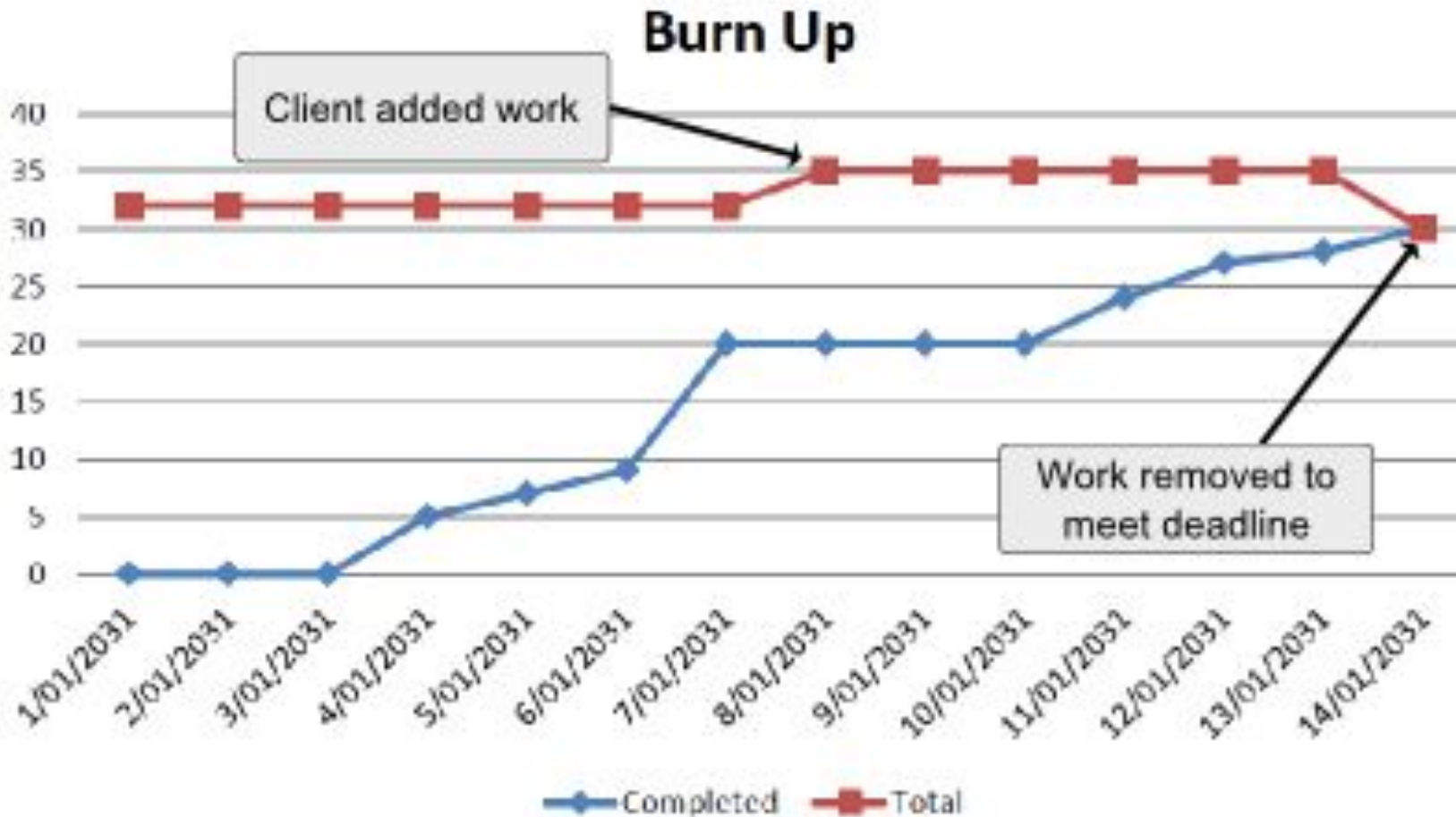
Indicateurs

- Sprint
 - *Burndown de sprint* (orienté reste à faire)
 - *Burnup de sprint* (orienté ce qui a été déjà fait)
- Release
 - *Burndown de release*
 - *Burnup de release*
- Equipe
 - Vitesse (capacité de l'équipe)
 - Suivi des obstacles
 - Perturbations exogènes ou endogènes qui perturbent le bon déroulement du sprint
 - Peut de générer de nouvelles tâches (dette) et/ou leur réorganisation

Burndown graphe



Burnup graphe

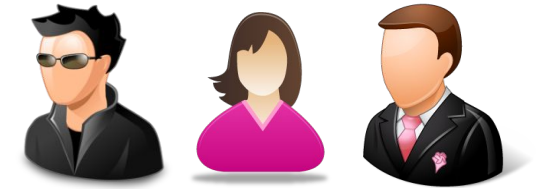


Vélocité

- Estime la capacité de l'équipe en nombre de points de stories par sprint
- Utilisé pour la planification de la release
- Affinée à la fin de chaque sprint
- Tendance à la stabilité

Les activités propres à SCRUM

Idéation



- Définition d'une vision commune
 - Identification des *features*
 - Impact mapping
 - Identification des parties prenantes
 - Acteurs
- Création d'un *backlog* de haut niveau du produit
 - Tableau ordonné des features
 - (éventuellement Story map haut niveau)

Impact mapping

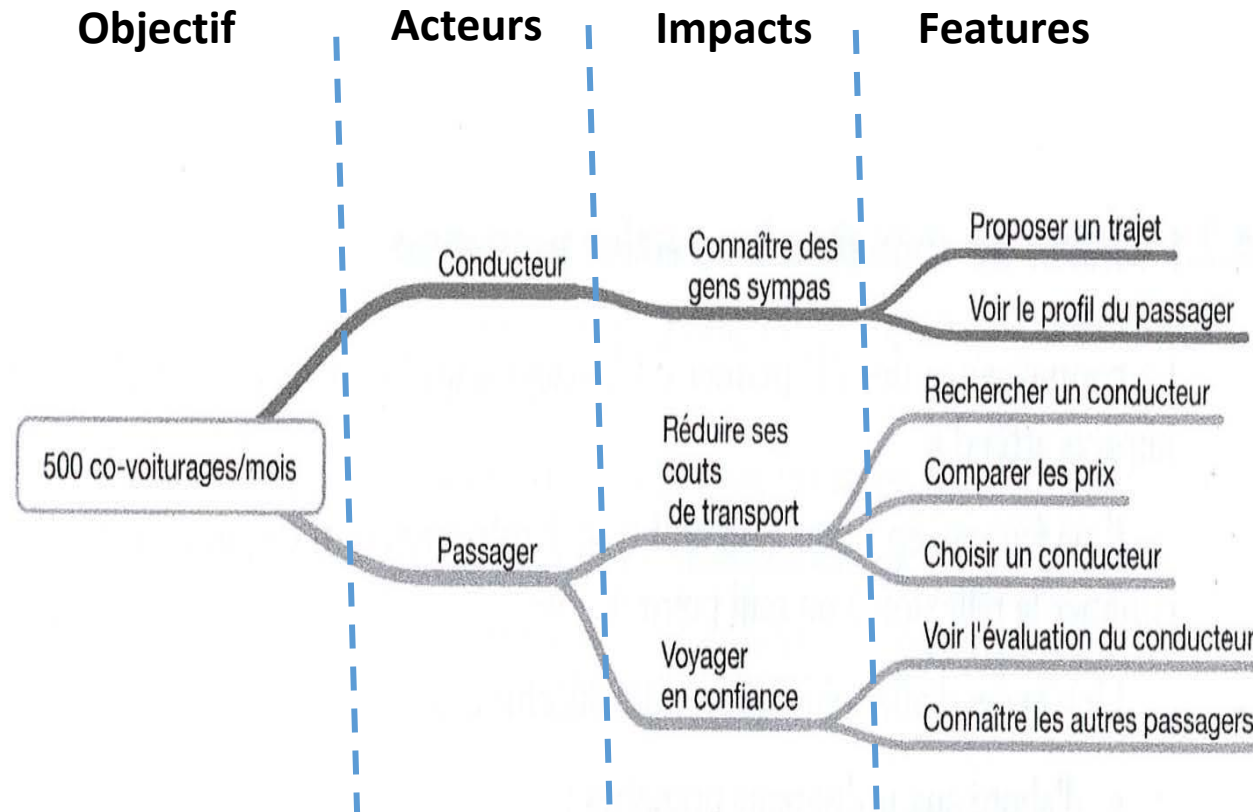


Figure 14.3 – Une carte qui donne la vision pour un site de co-voiturage

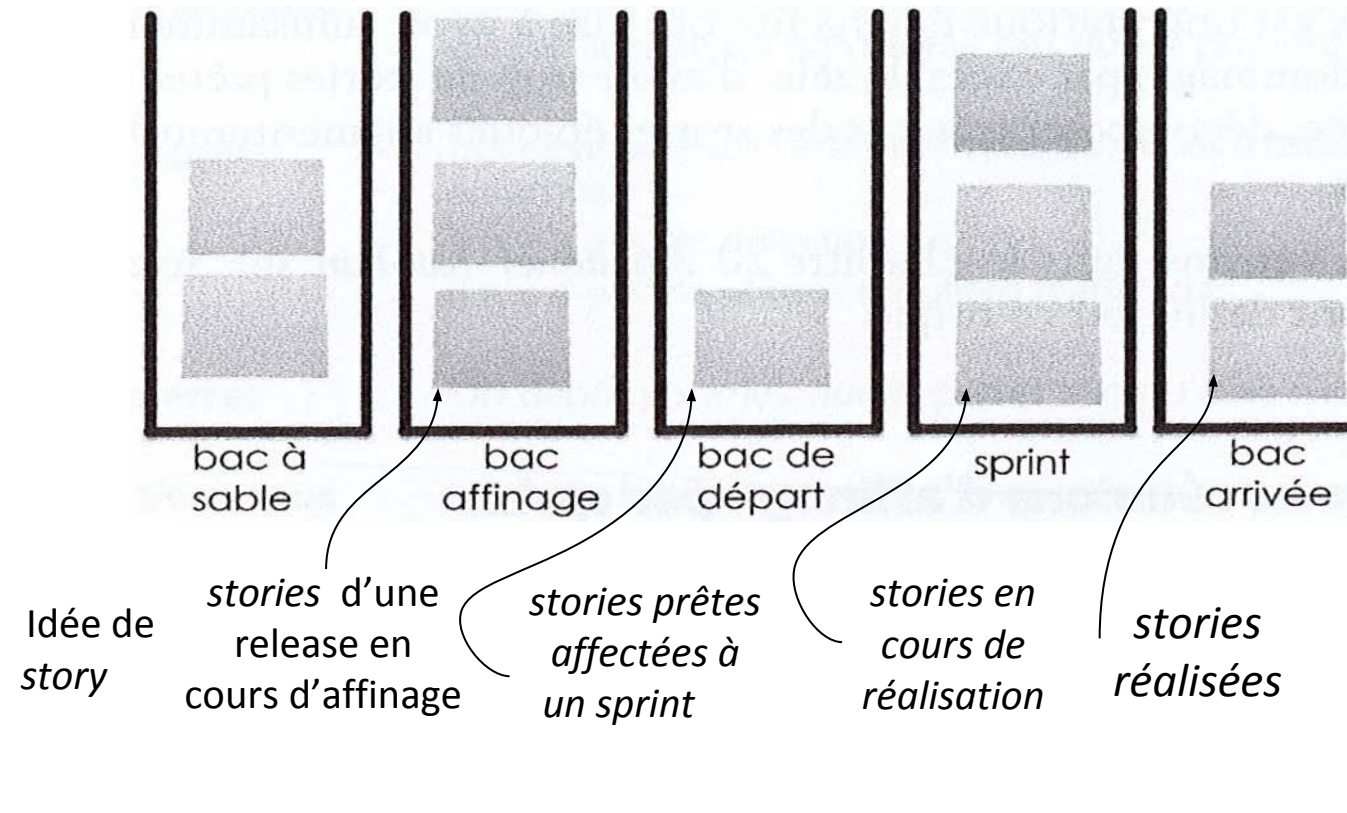
Sprint « Zéro »



- Affinage du backlog
 - *Story mapping* (feature → stories)(partiel)
 - Description des stories (description, conditions d'acceptation)
 - Ordonnancement des stories
 - Planification de la première *release*
 - Approvisionnement du bac d'affinage
 - Approvisionnement du bac de départ du sprint 1
- Peut durer plusieurs journées

Le *backlog* (carnet de route)

- Liste ordonnée des choses (stories) à faire
- En pratique, plusieurs (sous-)backlogs
 - *On peut distinguer le backlog de produit et le backlog de sprint*



Story mapping

- Ordonnancement des features
- Décomposition en stories
- (Organisation des releases)

Exemple

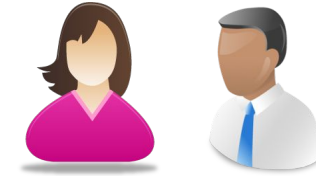
POSSE- SSEUR	ANIMAL	BALADE	GARDIEN- NAGE	CONSEIL EXPERTISE	PUBLICITE	BOUTIQUE
Données Basiques	Données Basiques	Liste simple	Liste simple	fiches conseils vétérinaires	Encart pub	Catalogue
Données Avancées	Pédigree	Liste départs sur carte geoloc	Créer/mod gardiennage	fiches conseils éleveurs	Compteurs / stats	Panier
Ajout Média	Ajout média	Liste partici- pants	Note organisateur	Forum	Mailing	Paiement CB
Messagerie Tchat	Données avancées	Offrir/créer balade	Comment -aires	Tchat expert	Profiling	Paiement Paypal
Archivage		Visu parcours sur carte	Ajout média		Invitation événement	Paiement 3 fois
géoloc- alisation		Note organisateur				Coupon promo
		Comment -aires				Promo avancée
		Notifications				Visu export/ facture

<https://pablopernot.fr/2017/01/cartographie-plan-action/>)

Example :

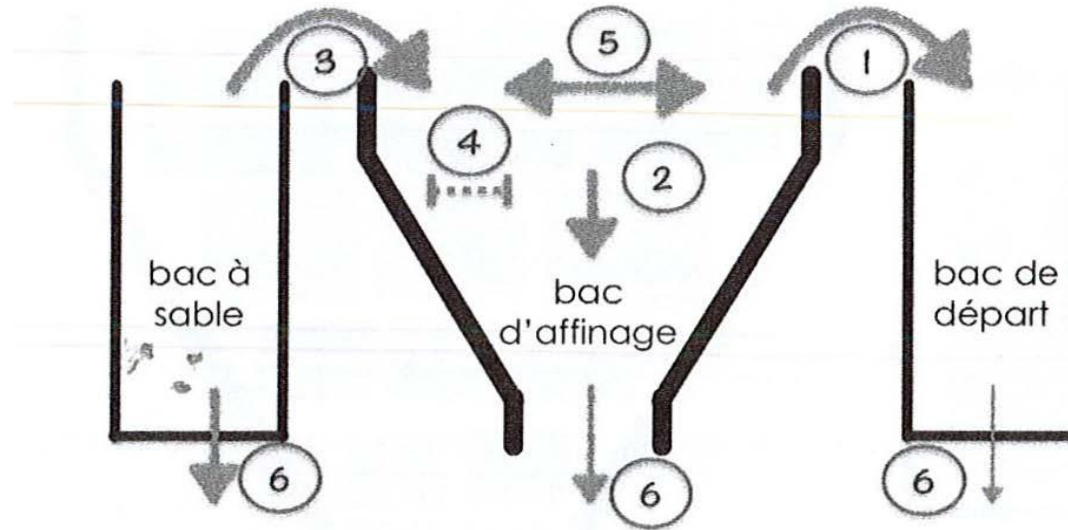
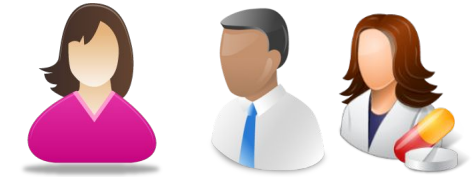


Planification des *releases*



- Affiner les risques, les incertitudes en fonction des retours de la revue et de la rétrospective du dernier sprint
- Ajuster la vélocité de l'équipe (capacité de travail de l'équipe en nombre de points de story)
- Affiner, (re-)planifier le(s) prochain(s) sprint(s)
 - Définition de prêt et fini
 - Nombre de points
- Affiner, (re-)planifier la future release

Affinage du backlog



1. Approvisionner le bac de départ en stories prêtes
2. Identifier les *epics* à décomposer en stories simples
3. Identifier les stories du bac à sable qui peuvent entrer dans le bac d'affinage
4. Evaluer les éléments du bac d'affinage
5. Réordonner le bas d'affinage
6. Purger les bacs



Planification d'un *Sprint*

- Confirmer les stories prêtes
 - Définition de prêt et fini
- Evaluer la nombre de points d'une story
 - Ponts de récit ou journée idéale (homme/jour)
- Organisation de l'essaimage (plusieurs stories en parallèle, répartition des ressources)
- Décomposition des stories en tâches
- Affectation des tâches aux développeurs

Exécution d'un *sprint*



- Conception, réalisation et test des *stories*
- Organisation, affectation des tâches
- Inclut les *sprints* journaliers

La mêlée quotidienne



- Courte : ~15mn
- Bilan: mise à jour du tableau de la story
 - Qu'est-ce que j'ai fait hier ?
 - Qu'est que je vais faire aujourd'hui ?
 - Quels sont les obstacles que j'ai rencontrés ?
- Objectif :
 - Rythmer le sprint (stories finies, prêtes)
 - Recenser les obstacles

Revue d'un *sprint*



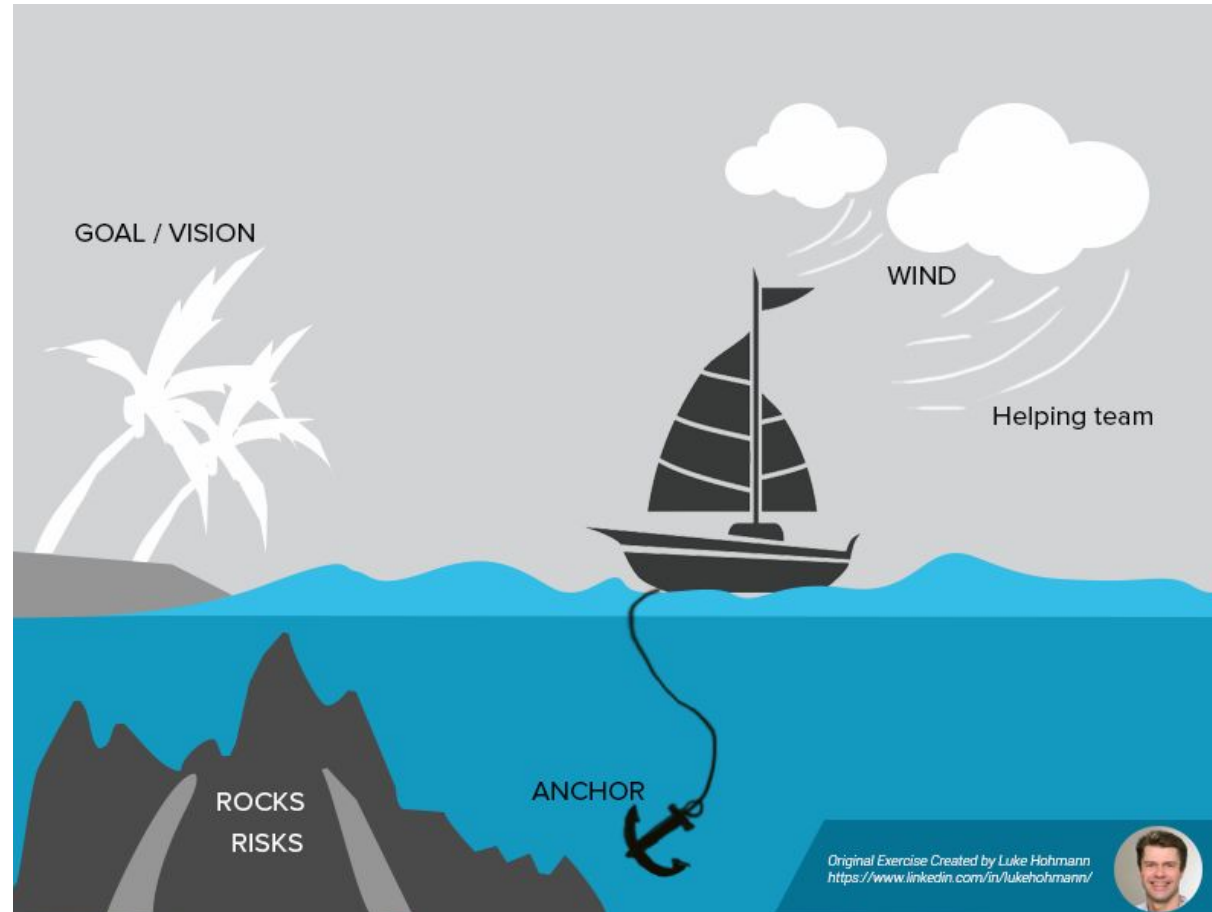
- Démonstration de chaque story finie
- Collecte du feedback
- Evaluation du niveau de réalisation de l'objectif
- Evaluation de l'impact du travail réalisé et décision d'une release (livraison) ou pas

Rétrospective d'un *sprint*



- Collecter les information sur le sprint passé par rapport à la pratique SCRUM
 - Ce qui c'est bien passé, moins bien passé
- Identifier les choses à améliorer
- Décider d'améliorer certaines choses
- Combinée à la revue, de courte durée

Rétrospective de sprint : « sailboat »



Rétrospective de sprint : « Starfish »



Solidification d'une *release*



- Tests de qualité de services (performances, coût, sécurité ...)
- Documentation
- ...

Validation d'une release



- Test d'usage avec le client, des utilisateurs

Outils

- Confluence
- Jira
- ...