



*Version expérimentale
En cours de validation*



RÉSUMÉ THÉORIQUE – FILIÈRE DÉVELOPPEMENT DIGITAL OPTION WEB FULL STACK

Elaboré par :

Oussama HAIJ *Formateur au CMC NADOR*

Mohamed GOUMIH *Formateur au CMC AGADIR*

M110 - ADOPTER L'APPROCHE AGILE



100 heures



Equipe de rédaction et de lecture

Equipe de rédaction :

M. Haij Oussama : Formateur en développement digital option Web Full Stack

M. Goumih Mohamed : Formateur en développement digital option Applications Mobile

Equipe de lecture :

Mme Laouija Soukaina : Formatrice Animatrice au CDC Digital & IA



SOMMAIRE

1. Connaître les fondamentaux de la gestion de projet

Découvrir les Concepts de gestion de projet
Découvrir les différentes méthodes de gestion de projet

2. Planifier un projet

Analyser le cahier des charges
Préparer le projet

3. Adopter l'approche Agile dans gestion de projet

Appréhender la méthodologie Agile Scrum
Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

4. Mettre en œuvre des outils de gestion de versions et de mesure de la qualité du code

Manipuler les outils de gestion de versions (Git/Gitlab)
Manipuler l'outil de mesure de la qualité du code (SonarQube)

5. Mettre en œuvre les outils de la chaîne du DevOps

Introduire la chaîne DevOps
Mettre en place la CI/CD avec Gitlab

MODALITÉS PÉDAGOGIQUES



1

LE GUIDE DE SOUTIEN
Il contient le résumé théorique et le manuel des travaux pratiques



2

LA VERSION PDF
Une version PDF est mise en ligne sur l'espace apprenant et formateur de la plateforme WebForce Life



3

DES CONTENUS TÉLÉCHARGEABLES
Les fiches de résumés ou des exercices sont téléchargeables sur WebForce Life



4

DU CONTENU INTERACTIF
Vous disposez de contenus interactifs sous forme d'exercices et de cours à utiliser sur WebForce Life



5

DES RESSOURCES EN LIGNES
Les ressources sont consultables en synchrone et en asynchrone pour s'adapter au rythme de l'apprentissage



PARTIE 1

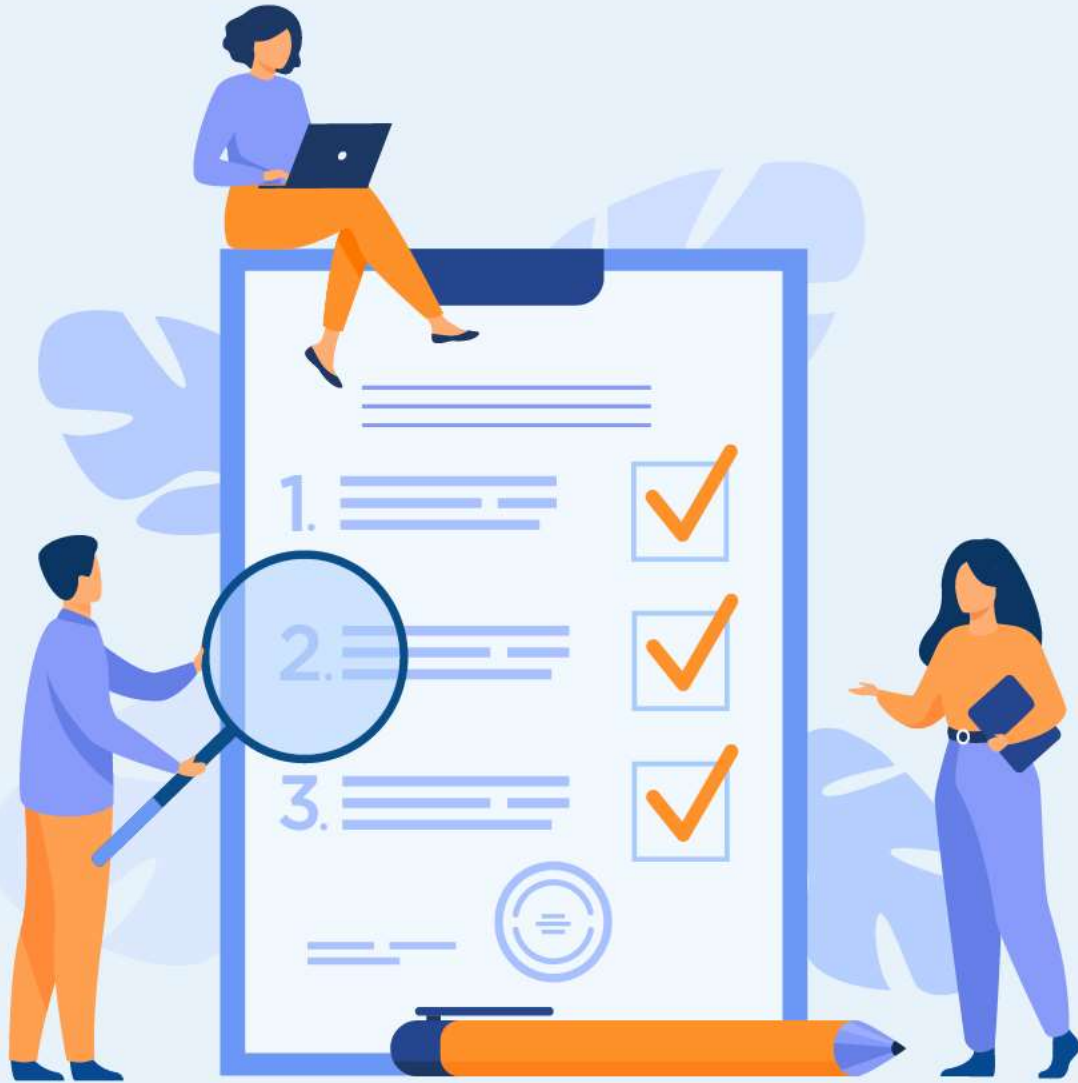
Connaître les fondamentaux de la gestion de projet

Dans ce module, vous allez :

- Découvrir les Concepts de gestion de projet
- Découvrir les différentes méthodes de gestion de projet



? heures



CHAPITRE 1

Découvrir les Concepts de gestion de projet

Ce que vous allez apprendre dans ce chapitre :

- Concepts de gestion de projet
- Parties prenantes de projet
- Principaux rôles dans un projet informatique
- Caractéristiques de base d'un projet
- Contraintes dans la gestion d'un projet



? heures

CHAPITRE 1

Découvrir les Concepts de gestion de projet

- 1. Concepts de gestion de projet**
2. Parties prenantes de projet
3. Principaux rôles dans un projet informatique
4. Caractéristiques de base d'un projet
5. Contraintes dans la gestion d'un projet



01 - Découvrir les Concepts de gestion de projet

Concepts de gestion de projet

Définitions

- **Un projet** est un effort ponctuel et coordonné pour atteindre un objectif unique, incluant une dose d'incertitude quant à sa réalisation (**petit Larousse**). On appelle un projet c'est l'ensemble des actions à entreprendre afin de répondre à un besoin défini dans des délais fixés (un début et une fin). Le projet mobilise des ressources identifiées (humaines et matérielles) durant sa réalisation, celui-ci possède également un coût et fait donc l'objet d'une budgétisation de moyens.
- **La gestion de projet** est une action temporaire avec un début et une fin, qui mobilise des **ressources** identifiées (**humaines, matérielles, équipements, matières premières, informationnelles et financières**) durant sa réalisation, visant à organiser de bout en bout le bon déroulement d'un projet.
- **Une ressource** est un élément nécessaire à la réalisation d'une tâche ou d'un projet. Une ressource peut être une personne, une équipe, un outil, de la trésorerie ou du temps. La plupart des projets nécessite de nombreuses ressources différentes pour se dérouler. Les ressources doivent être estimées et affectées avant le début du projet. Leur mauvaise planification peut entraîner un manque pendant le projet, des retards sur certaines échéances ou même la livraison finale du projet.
- **La conduite d'un projet** débouche sur un produit, un service, une nouvelle organisation, etc. Cette finalité, appelée "**livrable**", est le résultat tangible d'une production réelle, appréhendable, mesurable attendue par le client final. Un projet peut, bien sûr, avoir plusieurs livrables.
- Toutefois, cette notion ne se limite pas à l'aboutissement du projet. Les réalisations intermédiaires (documents de travail, budgets, etc.) sont aussi des **livrables**.
- **Une charte de projet** est un document formel, généralement court, qui décrit votre projet dans son intégralité, y compris les objectifs, la manière dont il sera réalisé et les parties prenantes.



CHAPITRE 1

Découvrir les Concepts de gestion de projet

1. Concepts de gestion de projet
- 2. Parties prenantes de projet**
3. Principaux rôles dans un projet informatique
4. Caractéristiques de base d'un projet
5. Contraintes dans la gestion d'un projet



01 - Découvrir les Concepts de gestion de projet

Parties prenantes de projet

Pour atteindre les objectifs fixés, il est indispensable d'identifier les parties prenantes d'un projet, puis d'analyser leurs attentes et besoins et enfin déclencher, le cas échéant, les actions de communication adaptées.

Définition des parties prenantes d'un projet

- Il s'agit de l'ensemble des personnes et des organisations qui ont quelque chose à voir avec le projet. **Soit elles sont directement impliquées** dans la conduite des opérations, **soit elles sont impactées par la problématique de départ, par le choix ou la mise en œuvre des solutions.** Certaines parties prenantes peuvent exercer une influence à différents niveaux.
- Ces acteurs clés se situent aussi bien en interne - à tout niveau de la hiérarchie de l'entreprise - qu'en externe (un fournisseur concerné par de nouvelles méthodes d'approvisionnement d'un client, etc.).

Acteurs externe:

Les clients	Les fournisseurs	Les diverses communautés d'utilisateurs, de fans, etc.	Les organismes privés	Les investisseurs et partenaires financiers
<ul style="list-style-type: none"> • les premiers concernés en externe, impactés directement si leur rôle s'inscrit dans l'utilisation du produit ou service livré par le projet - ou indirectement (par exemple dans le cas d'un projet d'organisation destiné à améliorer la qualité d'un processus) . 	<ul style="list-style-type: none"> • de matière, de prestation, de main d'œuvre les organismes publics : dans le cas où votre projet doit s'inscrire dans un cadre juridique précis. 	<ul style="list-style-type: none"> • Présentes sur les réseaux sociaux. 	<ul style="list-style-type: none"> • ce sont les associations diverses, ONG... 	<ul style="list-style-type: none"> • ils ont des exigences de rentabilité et de sécurisation des ressources.

01 - Découvrir les Concepts de gestion de projet

Parties prenantes de projet

Acteurs interne:

Le commanditaire (ou demandeur, ou encore client interne)	Le sponsor du projet	Les utilisateurs, les services impactés	La direction	L'équipe projet	Les services supports impliqués	Les actionnaires	Les autres experts	Les syndicats et représentants du personnel
<ul style="list-style-type: none"> • C'est le premier concerné par le projet. 	<ul style="list-style-type: none"> • le parrain du projet, le responsable du projet. 	<ul style="list-style-type: none"> • ceux qui sont concernés directement par les livrables (par exemple la force de vente pour un nouveau logiciel de CRM). Les chefs de service comme les collaborateurs sont à prendre en compte. Ces utilisateurs finaux tiennent une place centrale dans le projet, car ils utiliseront directement le service, la nouvelle organisation ou le produit issu du projet. 	<ul style="list-style-type: none"> • représente le pouvoir décisionnel et de contrôle ultime. 	<ul style="list-style-type: none"> • comprenant le chef de projet ainsi que les autres membres de l'équipe. 	<ul style="list-style-type: none"> • la comptabilité, la logistique, les ressources humaines, l'informatique... qui apportent leur support dans le cadre des travaux d'analyse et de conception de solutions. 	<ul style="list-style-type: none"> • ils sont en attente d'un résultat, d'une performance valorisant leur patrimoine. 	<ul style="list-style-type: none"> • apportant leurs conseils ponctuellement (directeurs fonctionnels...). 	<ul style="list-style-type: none"> • ces parties prenantes sont vigilantes quant à la défense des intérêts des salariés.

CHAPITRE 1

Découvrir les Concepts de gestion de projet

1. Concepts de gestion de projet
2. Parties prenantes de projet
- 3. Principaux rôles dans un projet informatique**
4. Caractéristiques de base d'un projet
5. Contraintes dans la gestion d'un projet



01 - Découvrir les Concepts de gestion de projet

Principaux rôles dans un projet informatique

Qu'est-ce qu'un Chef de projet informatique ?

- Expert en informatique, le chef de projet informatique (**CPI**) peut également être appelé chef de projet intégrateur, chef de projet applicatif, Project manager ou responsable de domaine. Il a sous son égide plusieurs techniciens et ingénieurs qui ont chacun un rôle spécifique dans le traitement des demandes de clients particuliers.
- Garant de l'état d'avancement d'un projet informatique, le **CPI** doit ajuster les évolutions et les besoins y afférents si cela s'avère nécessaire. Il doit également tenir compte des contraintes en termes de financement et de délais. Il se doit donc de posséder de multiples capacités regroupant des compétences techniques et managériales à la fois.

Quel est son rôle ?

Rattaché à un directeur des systèmes d'information ou à un directeur des études, le chef de projet informatique est à la tête d'un ou plusieurs services dans l'entreprise.

Accompagné par son équipe, le **CPI** a pour rôle principal de concevoir et d'intégrer un logiciel ou une solution informatique spécifique. Ses tâches sont multiples et couvrent l'ensemble de toutes les étapes du projet depuis le **brief client** (Le brief client, aussi appelé cahier des charges) jusqu'à la réception par ce dernier ainsi qu'au suivi et maintenance.

La personne doit être capable de solutionner les différentes problématiques qui risqueraient de nuire au projet. Elle doit avoir un esprit créatif pour pouvoir améliorer et sublimer sa création. Elle est amenée à trouver des idées innovantes ainsi que des stratégies optimales qui seront bénéfiques pour l'entreprise. Le chef de projet informatique est notamment spécialisé dans un langage informatique particulier.

01 - Découvrir les Concepts de gestion de projet

Principaux rôles dans un projet informatique

Matrice d'assignation des responsabilités

- La matrice RACI est une matrice d'attribution des responsabilités servant à décrire la participation des divers rôles, à remplir les tâches ou livrables pour un projet ou processus.
- Elle est utile pour clarifier les rôles et responsabilités dans des projets et des processus transversaux ou, d'une manière plus générale, dans un département ou service, afin d'avoir une vision claire de la répartition des tâches.
- Il s'agit donc de donner à chaque membre de l'équipe un niveau de responsabilité en fonction des tâches du projet.

 Responsable	 Accountable	 Consulted	 Informed
<p><i>Celui qui réalise la tâche</i></p> <p>Qui ? Personne qui va exécuter la tâche : elle en est responsable.</p> <p>Sa mission : Réaliser la tâche qui lui a été attribué</p>	<p><i>Celui qui approuve la tâche</i></p> <p>Qui ? Personne qui va approuver la tâche : elle en est l'autorité.</p> <p>Sa mission : Veiller à l'exécution correcte de la tâche réalisée par le(s)</p>	<p><i>Celui qui est consulté</i></p> <p>Qui ? Personne qui va être consultée dans l'exécution de la tâche : elle est consultée.</p> <p>Sa mission : Contribuer avec des conseils et</p>	<p><i>Celui qui doit être informé</i></p> <p>Qui ? Personne qui sera informée lorsque la tâche est finie : elle est informée.</p> <p>Sa mission : Être tenue à jour sur les progrès</p>

Figure 1 : Explication sous format de tableau des 4 principales responsabilités du RACI : Responsable, Accountable, Consulted et Informed

CHAPITRE 1

Découvrir les Concepts de gestion de projet

1. Concepts de gestion de projet
2. Parties prenantes de projet
3. Principaux rôles dans un projet informatique
- 4. Caractéristiques de base d'un projet**
5. Contraintes dans la gestion d'un projet



01 - Découvrir les Concepts de gestion de projet

Caractéristiques de base d'un projet

Un projet peut se définir comme un ensemble d'actions mises en œuvre pour atteindre un but précis, afin de répondre à un besoin spécifique. **Il se caractérise par :**

- Chaque projet doit comporter **des objectifs** clairement définis qui permettent la satisfaction d'un besoin spécifique et particulier.
- **Une limite dans le temps** : il a un début et une fin, marquée par l'atteinte de l'objectif .
- **Une activité** est une action qui transforme les ressources (main d'oeuvre, connaissances, l'équipement, les matières premières, le temps) en résultats attendus dans un délai de temps spécifié.
- Parfois, une activité est suffisante pour obtenir les résultats souhaités, mais souvent il faut passer par toute une série d'activités. Lorsque vous devez passer par la même série d'activités ou des tâches à chaque fois que vous voulez obtenir un résultat, on peut parler d'un processus. Dans le cadre logique, chaque résultat dépend d'une ou de plusieurs activités ou processus.
- **Les ressources** (les intrants) sont les choses qui se transforment en résultats (tangibles). Lorsque nous parlons des ressources, nous pensons généralement à l'argent, le personnel, le matériel ou l'équipement. Mais il ya d'autres choses qui sont nécessaires pour un projet: le temps, les connaissances et le savoir faire, l'espace, l'infrastructure , la communication (accès à l'information) et ainsi de suite.
- **Les résultats attendus** se créent à la suite des activités du projet. Ensemble, les résultats mènent à la réalisation de l'objectif spécifique du projet. L'objectif spécifique est la situation que vous espérez atteindre lorsque le projet est terminé. Les résultats sont les biens, les services et ainsi de suite que vous souhaitez créer, au cours du projet. En tant que tel, l'achèvement des résultats est - en principe - entièrement sous votre contrôle.
- A ce niveau, la logique du projet est le plus fort: vous investissez des moyens (ressources) pour faire les activités et les activités mèneront à leur tour aux résultats concrets

CHAPITRE 1

Découvrir les Concepts de gestion de projet

1. Concepts de gestion de projet
2. Parties prenantes de projet
3. Principaux rôles dans un projet informatique
4. Caractéristiques de base d'un projet
5. **Contraintes dans la gestion d'un projet**



01 - Découvrir les Concepts de gestion de projet

Contraintes dans la gestion d'un projet

Définition : Les contraintes de projet sont les limites générales d'un projet, notamment les délais, les coûts et les risques. Il est important d'identifier les contraintes d'un projet, car elles ont des répercussions sur les performances de ce dernier.

❖ Contraintes de délais :

- Fenêtre temporelle à l'intérieur de laquelle le projet doit être réalisé
- **Contrainte externe absolue :** contraintes externes au projet qui s'imposent à tous. Si elle n'est pas respectée, le projet n'a plus de sens

Exemple : un salon ou une manifestation sportive à une date donnée, une clôture de compte, le passage à l'an 2000.

❖ Contraintes dues aux clients

- Contrainte externe « fixe » : Elle est souvent contractuelle, généralement moins forte que la contrainte externe absolue, elle est souvent assortie d'une pénalité de retard. Contrainte externe « variable ». Elle concerne la réalisation d'une partie du projet qui est liée à un événement dont la date n'est pas absolument fixe.

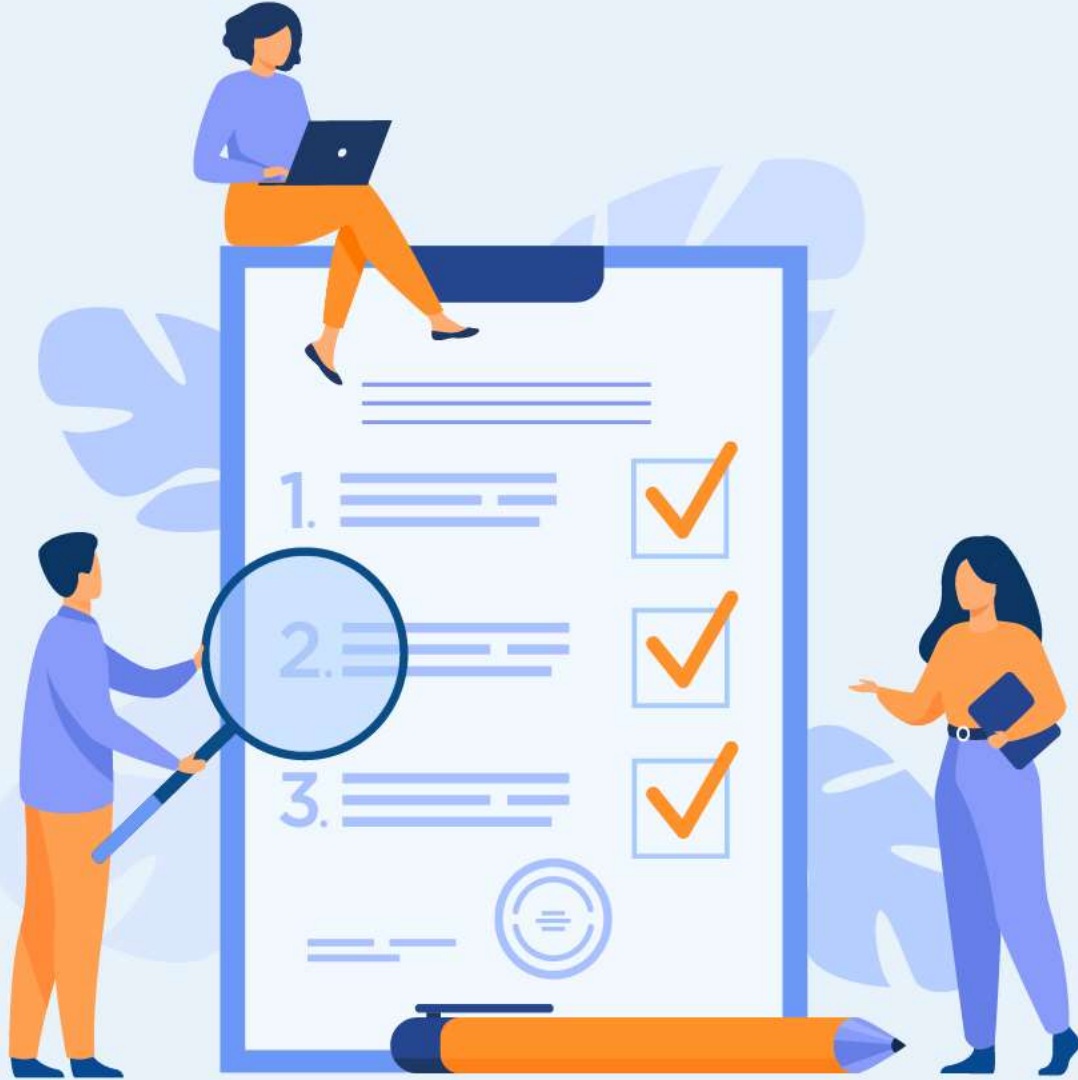
Exemple : les projets de sous-traitance

❖ Contraintes de coûts :

- Budget pour réaliser le projet
- Contrainte de rentabilité : Marge entre les rapports du projet et les coûts engagés pour sa réalisation
- Contrainte pour l'équilibre financier de l'entreprise

❖ Contraintes de qualité :

- Contraintes fortes, leur non-respect est susceptible de remettre en cause le projet lui-même. Par exemple, des impératifs légaux, de santé ou de sécurité publique.
- Des impératifs de nature commerciale, des engagements contractuels existent : le projet doit s'y conformer.
- La certification de l'entreprise dans un système d'assurance qualité faite qu'elle se doit de respecter certaines règles.



CHAPITRE 2

Découvrir les différentes méthodes de gestion de projet

Ce que vous allez apprendre dans ce chapitre :

- Méthodes prévisibles (cascades, V, Y)
- Méthodes imprévisibles (Agile)
- Cycle en V vs. Méthodes agiles



? heure

CHAPITRE 2

Découvrir les différentes méthodes de gestion de projet

1. Méthodes prévisibles (cascades, V, Y)
2. Méthodes imprévisibles (Agile)
3. Cycle en V vs. Méthodes agiles



02 - Découvrir les différentes méthodes de gestion de projet

Méthodes prévisibles (cascades, V, Y)

Définition

- Cette catégorie regroupe les méthodes reposant sur une organisation stricte du travail et sur un fonctionnement par étapes. Il n'y a ici aucune rétroactivité.
- Dès que les contours du projet sont définis avec le client, le chef de projet se charge tout seul de veiller à ce que chaque tâche soit accomplie au moment prévu et dans le respect des objectifs définis. C'est seulement lorsqu'une tâche est bien exécutée que la phase suivante est lancée.
- Ce type de management permet d'écartier tout risque en s'attachant strictement au respect des plans préalablement établis. Les trois méthodologies habituellement employées dans cette catégorie sont présentées ci-dessous.

Le modèle en cascade

- Le modèle en cascade, appelé **Waterfall** en anglais, tel qu'appliqué aux projets, est une approche linéaire et séquentielle des différentes phases et activités du projet nécessaires à la livraison du ou des livrables.

Exemple de modèle en cascade

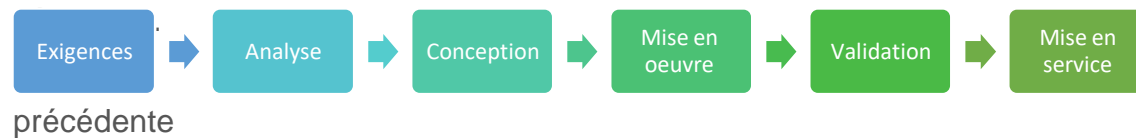


Figure 2 : Modèle en cascade générique

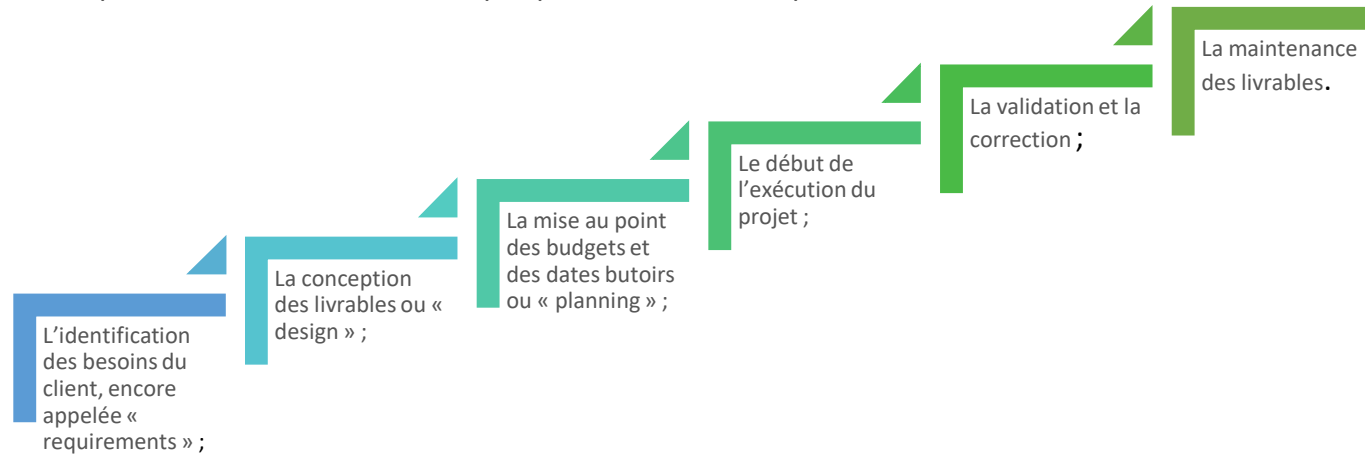
Remarque : Les étapes s'exécutent en

Chaque étape dépend de l'étape

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes prévisibles (cascades, V, Y)

- La **méthode Waterfall** repose sur une succession d'étapes prédéfinies. Ces étapes, au nombre de 6, sont les suivantes :



- Sur le modèle d'une cascade, c'est la fin d'une phase qui mène au démarrage de la suivante. Par ailleurs, il n'y a aucune possibilité de retour en arrière.
- **L'avantage de cette méthode** est que le planning à suivre est bien précis dès le départ. Le principal reproche fait à la méthode réside dans son manque de souplesse.
- **Les inconvénients de cette méthode** est que :
 - Les projets complexes ou à plusieurs niveaux ne peuvent que rarement être divisés en phases de projet clairement définies.
 - Une faible marge pour les ajustements du déroulement du projet en raison d'exigences modifiées.
 - L'utilisateur final est uniquement intégré dans le processus de production après la programmation.
 - Les erreurs sont parfois détectées uniquement à la fin du processus de développement.

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes prévisibles (cascades, V, Y)

Le cycle en Y

- La famille des “ **Unified Process** ” constitue une trame commune pour intégrer les meilleures pratiques de développement. Un processus UP (Processus Unifié) est itératif et incrémental, centré sur l’architecture, conduit par les exigences des utilisateurs, piloté par les risques et orienté composants.
- 2TUP : « **2 Track Unified Process** » propose un cycle de développement en Y, qui permet de décomposer le système d’information, suivant un axe fonctionnel et un axe technique, puis fusionner les résultats de ces deux branches formant ainsi la lettre Y.
- Le processus unifié combine les avantages de plusieurs approches, et en particulier une démarche structurée en phases avec une grande flexibilité au niveau des itérations.
- La principale critique est que la description détaillée des enchainements d'activité et des artefacts confère au PU(Processus Unifié) une certaine lourdeur et nécessite de ce fait une qualification élevée des membres de l'équipe projet (en particulier : maîtrise des approches itératives, connaissance approfondie d'UML, connaissance des enchainements d'activités et de leurs interdépendances).
- Le processus unifié laisse aussi une grande latitude d'appréciation pour l'adaptation des activités aux spécificités d'une entreprise. Cette flexibilité, combinée à la complexité du processus et à une grande liberté d'interprétation, peut donner lieu à des mises en œuvre très rigides du PU, alors que celui-ci possède en principe les caractéristiques d'une méthode agile.

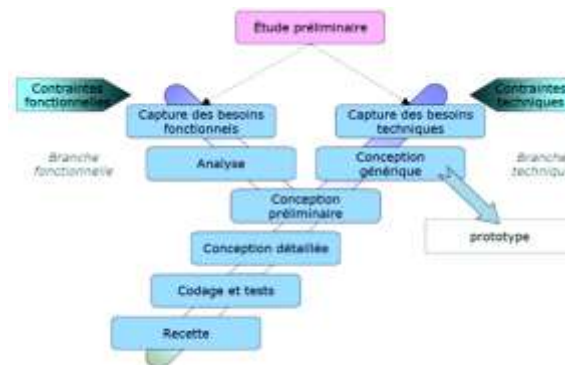


Figure 3 : Cycle en Y

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes prévisibles (cascades, V, Y)

Le cycle en V

- Le cycle en V est un modèle de gestion de projet qui implique toutes les étapes du cycle de vie d'un projet : conception, réalisation, validation.
- Le cycle en V en gestion de projet découle du modèle en cascade théorisé dans les années 1970, qui permet de représenter des processus de développement de manière linéaire et en phases successives.
- Ce mode de gestion de projet a été développé dans les années 1980 et appliqué au champ des projets industriels, puis étendu aux projets informatiques. Il a été remis en cause à partir du début des années 2000, sous l'effet de l'accélération des changements technologiques, favorisant davantage les méthodes dites « agiles ».
- La lettre V fait référence à la vision schématique de ce cycle, qui prend la forme d'un V : une phase descendante suivie d'une phase ascendante. Le cycle en V associe à chaque phase de réalisation une phase de validation, comme l'illustre le schéma ci-dessous :

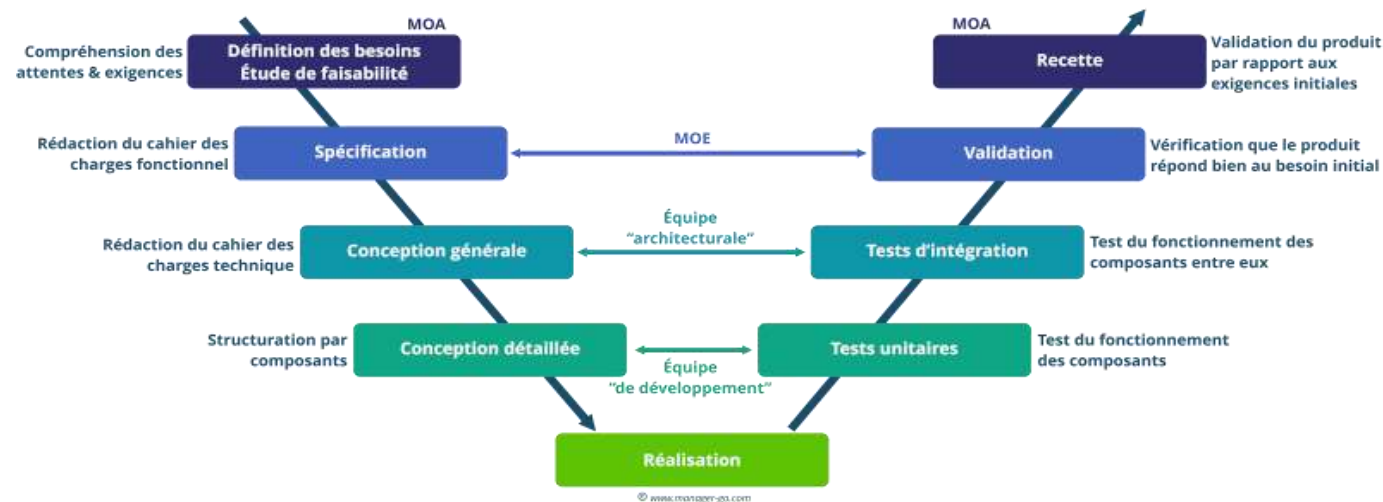


Figure 4 : Cycle en V

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes prévisibles (cascades, V, Y)



Avantages de cette méthodologie

- Le principal avantage du cycle en V est qu'il évite de revenir en arrière incessamment pour redéfinir les spécifications initiales, comme un cliquet. Chaque phase de conception demande la rédaction d'une documentation précise et exhaustive, où chaque point doit être validé par le produit final. Dès lors qu'une étape est validée, on ne revient pas en arrière et on passe à l'étape suivante sur une base solide ; c'est la principale force du cycle en V.
- De par son aspect à la fois rigoureux et intuitif, le cycle en V demeure un processus facile à mettre en œuvre. Le travail préalable de définition des spécifications en début de projet fait que, une fois lancé, l'ensemble des étapes est connu des collaborateurs, qui peuvent se repérer facilement dans la temporalité du projet et connaître la finalité de leurs tâches. De la même manière, les documentations nécessaires à chaque étape sont répliquables d'un projet sur l'autre dans leur structure (cahiers des charges, cahiers de test...).
- En général, le cycle en V est plus adapté aux structures multi sites, car il ne demande pas de réunions quotidiennes, mais seulement des réunions de pilotage actant le passage d'une phase à l'autre. Son aspect linéaire autorise donc une organisation géographique éclatée, où le côtoiement des collaborateurs n'est pas clé dans le processus.

Inconvénients

- L'inconvénient principal du cycle en V se résume en deux mots : l'effet tunnel. Après une phase de définition précise du produit auquel doit l'équipe doit aboutir, le projet est lancé dans un « tunnel » constitué des phases évoquées plus haut. Mais que faire si les spécifications initiales sont dépassées ? Si le besoin du client vient à changer, ou a été mal exprimé ? Le cycle en V supporte donc mal les changements, ce qui est à la fois sa force et sa principale faiblesse.
- Il offre ainsi moins de réactivité par rapport au contexte technologique et économique, aux demandes du client, aux événements inopinés ; la prise de risque s'en trouvera systématiquement limitée. L'effet tunnel est aussi induit par le travail conséquent de production de la documentation en début de projet, qui n'est plus rectifiable par la suite. Enfin, l'image du tunnel illustre le temps (parfois très) long qui sépare l'expression du besoin de la recette du produit final.

CHAPITRE 2

Découvrir les différentes méthodes de gestion de projet

1. Méthodes prévisibles (cascades, V, Y)
2. **Méthodes imprévisibles (Agile)**
3. Cycle en V vs. Méthodes agiles



02 - Découvrir les différentes méthodes de gestion de projet

Méthodes imprévisibles (Agile)

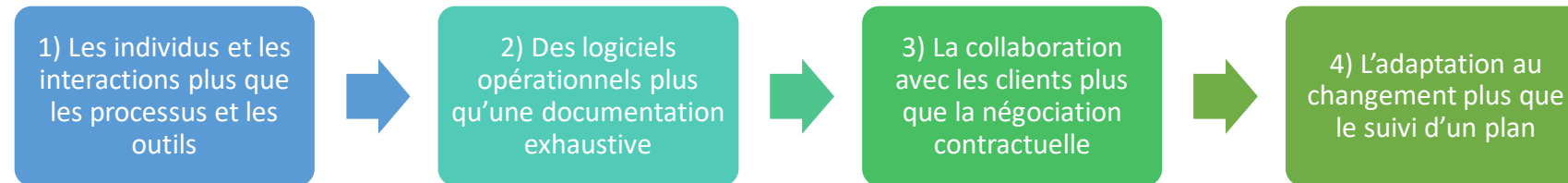


Signification d'Agile en gestion de projet

- Alors que les méthodes traditionnelles visent à traiter les différentes phases d'un projet d'une manière séquentielle (que l'on nomme aussi cycle de développement en cascade ou encore cycle en V), le principe des méthodes Agiles est de le découper en sous-parties (ou sous-projets) autonomes (on parle également de développement itératif).
- Les parties (itérations) forment le projet dans sa globalité.

Manifeste Agile, Les principes fondateurs

- Le **Manifeste Agile** est une déclaration rédigée par des experts en 2001 pour améliorer le développement de logiciels.
- Les 4 valeurs agiles :



- Le Manifeste définit 12 principes :

- 1 - La priorité n°1 est d'obtenir la satisfaction client au plus tôt par la livraison rapide et régulière de fonctionnalités attendues.
- 2 - Accepter les demandes de changement en cours de projet. Ce sont des opportunités pour donner plus de valeur au projet et coller aux vrais besoins des clients.
- 3 - Mettre en œuvre des livraisons rapides reposant sur des cycles courts (quelques semaines). Ces livrables doivent être opérationnels pour permettre des tests de validation des fonctionnalités attendues.

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes imprévisibles (Agile)



- 4 - Coopération forte et continue entre les utilisateurs et le développement. A l'inverse des méthodes classiques où les rencontres entre les utilisateurs et la maîtrise d'oeuvre interviennent surtout en début et en fin de projet.
- 5 - Donner de l'autonomie à des personnes impliquées et leur faire confiance.
- 6 - Privilégier le face à face comme canal de communication entre les parties. Les interactions sont plus efficaces et plus riches. Tout va plus vite.
- 7 - L'important est d'avoir une application opérationnelle.
- 8 - Avancer avec un rythme constant compatible avec ce que peut produire l'ensemble des acteurs.
- 9 - Focus sur la qualité technique et la qualité de conception pour construire une base solide renforçant l'agilité.
- 10 - Rester simple dans les méthodes de travail : ne faire que ce qui est nécessaire.
- 11 - Une équipe qui s'organise elle-même produit de meilleurs résultats.
- 12 - En revoyant régulièrement ses pratiques, l'équipe adapte son comportement et ses outils pour être plus efficace.

Méthodes Agiles

- La méthodologie Agile se base sur une idée simple. Planifier la totalité de votre projet dans les moindres détails avant de le développer est contre-productif.
- Vous perdez du temps si vous organisez tous les aspects de votre projet en amont. Il est effectivement rare que tout se passe exactement comme prévu. Souvent, des aléas surviennent et vous forcent à revoir votre planification.
- La méthode Agile recommande de se fixer des objectifs à court terme. Le projet est donc divisé en plusieurs sous-projets. Une fois l'objectif atteint, on passe au suivant, et ce jusqu'à l'accomplissement de l'objectif final. Cette approche est plus flexible. Puisqu'il est impossible de tout prévoir et de tout anticiper, elle laisse la place aux imprévus et aux changements.

02 - Découvrir les différentes méthodes de gestion de projet

Méthodes imprévisibles (Agile)



Quelles sont les principales méthodes Agile ?

- Selon la méthode Agile à laquelle on se réfère, la démarche peut prendre différentes formes, et revêtir un vocabulaire spécifique.
 - **La méthode Scrum et son fonctionnement en sprints**
 - La plus célèbre des méthodologies de gestion de projets déclinées de la méthode Agile relève de la “**Scrum**”, autrement dit la “**mêlée**” dans le langage rugby. Le responsable de projet s’appelle ainsi le “**SCRUM Master**”.
 - Cette approche s’organise autour de cycles courts, qu’on appelle communément des itérations. En langage **Scrum**, une itération se nomme un “**sprint**”. À chaque nouveau sprint, l’équipe projet se rassemble pour lister les tâches à exécuter. Cette liste s’appelle le “**sprint backlog**”.
 - L’ensemble relève d’une logique de développement produit. C’est ce qui explique que la méthodologie **Scrum** se déploie autour d’acteurs spécifiques, comme le **Product Owner**. Des réunions **Scrum** ont d’ailleurs lieu quotidiennement. Il s’agit de courtes périodes d’échange, pendant lesquelles les membres de l’équipe projet communiquent sur leurs avancées et leurs difficultés.
 - **Les autres méthodologies d’inspiration Agile**
 - Si **Scrum** reste la méthode Agile la plus utilisée, elle entre en compétition avec la méthode **Kanban** pour ce qui relève du pilotage de projets dit “**mono-équipe**”.
 - L’approche **Kanban** trouve son origine dans le mot japonais pour “**panneau**”. Elle nous vient des procédures de production de Toyota, appliquées à l’univers de la programmation logiciel. Cette approche consiste à croiser des tâches avec leurs états d’avancement, au sein d’une matrice en colonnes.
 - Le “**Lean Development**” est une méthode proche de **Kanban**. Il s’en différencie seulement par deux objectifs : améliorer les apprentissages des participants et éviter le gaspillage de ressources.

CHAPITRE 2

Découvrir les différentes méthodes de gestion de projet

1. Méthodes prévisibles (cascades, V, Y)
2. Méthodes imprévisibles (Agile)
- 3. Cycle en V vs. Méthodes agiles**



02 - Découvrir les différentes méthodes de gestion de projet

Cycle en V vs. méthodes agiles



- De façon générale, l'on peut affirmer que le cycle en V se focalise sur le processus, tandis que les méthodes agiles privilégient le produit.
- Dans le cadre des méthodes agiles (**Scrum, XP, RAD, ...**), le projet s'affine par itérations, à travers la répétition d'un cycle d'opérations (le sprint dans le cadre de la méthode Scrum). Comme nous l'avons vu, le cycle en V définit l'intégralité du produit final dès les premières étapes, et ne laisse que peu de place à l'adaptation dans la suite du cycle.
- Ensuite, les méthodes agiles permettent d'élaborer le produit par incrémentation. On produit un peu plus à chaque fois, morceau par morceau, pour aboutir au résultat final. Le cycle en V concentre au contraire la réalisation de l'ensemble dans une seule phase, qui est intégralement conçue en amont et vérifiée en aval.
- Ce manque d'adaptation et de flexibilité du cycle en V a précisément conduit à l'émergence des méthodes agiles, en particulier dans le domaine du logiciel et du marketing, pour répondre aux changements de plus en plus rapides des technologies et des demandes des consommateurs.



PARTIE 2

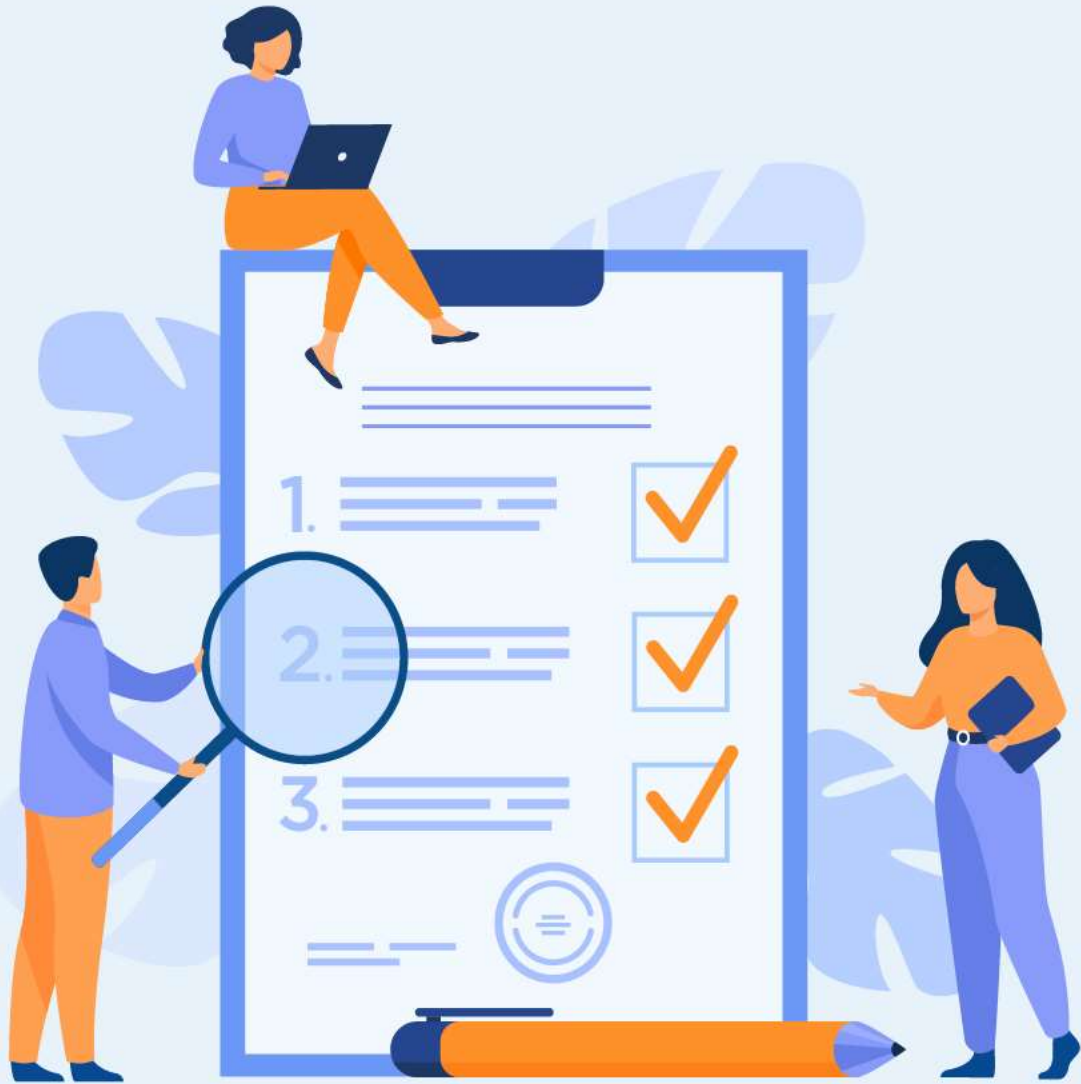
Planifier un projet

Dans ce module, vous allez :

- Analyser le cahier des charges
- Préparer le projet



?heures



CHAPITRE 1

Analyser le cahier des charges

Ce que vous allez apprendre dans ce chapitre :

- Compréhension des besoins client
- Contexte du projet
- Périmètre du projet
- Détection des risques liés à la nature du projet
- Proposition des solutions possibles



? heures

CHAPITRE 1

Analyser le cahier des charges

1. Compréhension des besoins client

2. Contexte du projet
3. Périmètre du projet
4. Détection des risques liés à la nature du projet
5. Proposition des solutions possibles



01 - Analyser le cahier des charges

Compréhension des besoins client



- Idéalement, tout projet commence avec l'expression d'un ou de plusieurs besoins (qui seront formulés en objectifs par la suite).
- Selon qui exprime ce ou ces besoins et de quelle manière, vous (**le chef de projet**) allez devoir gérer des scénarios potentiellement très différents. La première différenciation à faire est celle des projets internes et externes.
- Tout projet commence avec une expression ou l'identification d'un besoin et celle-ci peut être plus ou moins complète. Votre premier travail en tant que chef de projet, c'est donc d'analyser ce ou ces besoins, les compléter si nécessaire et les reformuler sous forme d'objectifs et de livrables.

Besoin	Livrable(s) potentiel(s)
Développer une présence en ligne	Site vitrine, plan web marketing
Moderniser une image de marque	Plateforme de marque avec charte graphique et logo
Permettre aux clients de réserver en ligne	Système de réservation

Figure 5 : Exemple du besoin client

- A priori, nous avons là les livrables les plus évidents. Reste à savoir s'ils correspondent bien tous aux attentes réelles du client et si celui-ci n'a pas oublié ou mal exprimé quelques besoins. Pour vous assurer de formuler une proposition commerciale sur-mesure, ouvrez la communication et allez aux informations. Pour ça, à vous de trouver le moyen le plus adéquat de le joindre, par téléphone, en visioconférence, sur LinkedIn ou encore par mail.
- Lors de vos échanges exploratoires, au-delà d'un relationnel irréprochable, vous allez porter une attention particulière à 3 choses :
 - les besoins explicites du client
 - les besoins implicites du client
 - les livrables potentiels

01 - Analyser le cahier des charges

Compréhension des besoins client



Besoins explicites

- Par "besoins explicites", comprenez les besoins exprimés clairement, sans ambiguïté et sur lesquels il y a un consensus. Pour savoir si c'est le cas, essayez de les reformuler et analysez la réaction de votre interlocuteur.
- Dans notre exemple, le besoin de développer la présence en ligne de l'hôtel Paradis à travers la création d'un site web est un besoin assez explicite.

Besoins implicites

- La clientèle est principalement étrangère, le site devra donc impérativement être multilingue. Ce besoin, en revanche, n'a pas été exprimé clairement par le client. On dira donc que c'est un besoin implicite et une contrainte qui devra être prise en compte dès le lancement du projet.

Besoins explicites	Besoins implicites	Livrable(s) potentiel(s)
Développer la présence en ligne	Rendre le site accessible dans 4 langues : français, anglais, chinois et russe.	Site web (multilingue)
Moderniser l'image de marque	Conserver le nom et le logo de l'établissement	Adaptation de la charte graphique
Système de réservation	Permettre au staff de l'hôtel de gérer ses prix et ses réservations en back-office	Système de réservation (et de gestion)

Figure 6 : Exemple du besoin implicite et explicite

CHAPITRE 1

Analyser le cahier des charges

1. Compréhension des besoins client
- 2. Contexte du projet**
3. Périmètre du projet
4. Détection des risques liés à la nature du projet
5. Proposition des solutions possibles



01 - Analyser le cahier des charges

Contexte du projet



Définition

- Le contexte d'un projet correspond à l'ensemble des informations qui caractérisent un projet et lui donnent de la profondeur. Il peut s'agir de l'histoire et de l'origine du projet, d'informations sur le contexte réglementaire, culturel, économique, concurrentiel et social dans lequel évolue la société, ou encore de son environnement de travail.
- En effet, un projet s'inscrit toujours dans un environnement social, économique et technique complexe, avec des éléments qui peuvent être interdépendants. Pour mettre toutes les chances de son côté, un bon chef de projet devra mener une analyse du contexte de projet, en plus de l'analyse du périmètre projet ainsi que de ses enjeux et de ses objectifs.
- L'ensemble des éléments qui composent le contexte d'un projet doivent être notifiés dans la note de cadrage de projet, qui est le document de référence décrivant les tenants et aboutissants du projet.
- Rédigez quelques mots pour exposer le fondement de la demande. Vous vous dites peut-être qu'un développeur n'est intéressé que par la partie technique. Détrompez-vous, nombreux sont ceux qui aiment voir la finalité de leur travail. De plus, donner du sens au développement facilite la compréhension de vos besoins et de vos contraintes. Le dialogue plus tard n'en sera que plus facile. Enfin, en définissant le problème - en démontrant ainsi le bien-fondé de votre requête.

01 - Analyser le cahier des charges

Contexte du projet



Éléments formant le contexte d'un projet

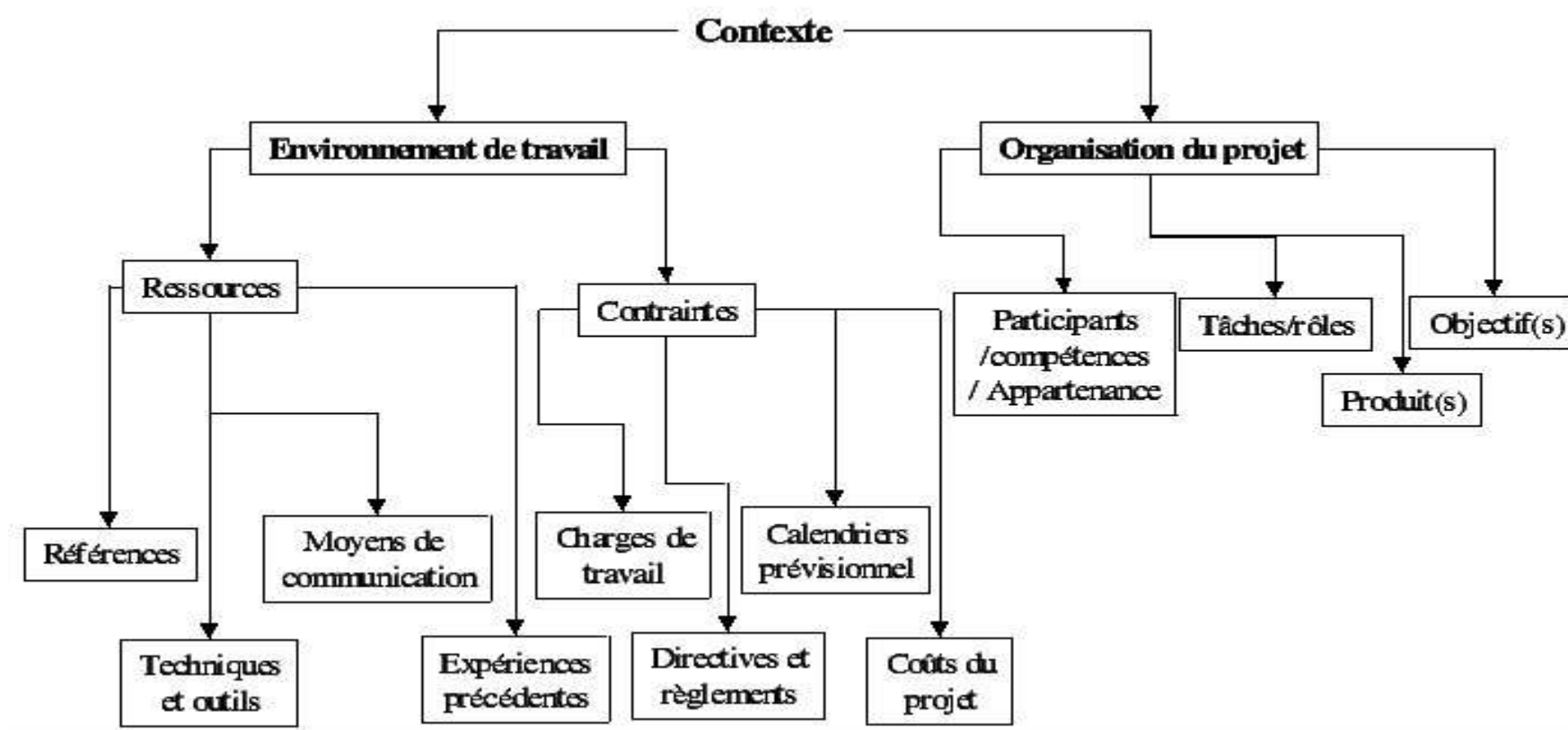


Figure 7 : Éléments formant le contexte d'un projet

La source : https://www.cairn.info/loading.php?FILE=DN/DN_081/DN_081_0137/DN_idPAS_D_ISBN_pu2004-01s_sa11_art11_img002.jpg

CHAPITRE 1

Analyser le cahier des charges

1. Compréhension des besoins client
2. Contexte du projet
- 3. Périmètre du projet**
4. Détection des risques liés à la nature du projet
5. Proposition des solutions possibles



01 - Analyser le cahier des charges

Périmètre du projet



Définition

- Le périmètre du projet, ou scope du projet, est l'ensemble des éléments qui composent un projet. Le périmètre du projet permet de limiter les dérives des objectifs, de s'assurer d'un travail de qualité, de cadrer le projet, et de visualiser ce qui doit être réalisé afin d'atteindre l'objectif.
- Le scope du projet permet ainsi de cadrer le projet, et de sécuriser sa réalisation, aussi bien pour le commanditaire du projet que pour la maîtrise d'œuvre (celui qui réalise).
- Comme chaque projet est unique et que vous ne disposez pas d'un budget illimité ni d'une équipe ad vitam æternam, il est primordial de déterminer avec précision le travail à réaliser (le périmètre projet) ainsi que ce qui n'est pas compris dans le projet (ce qui est hors périmètre). Sinon vous vous retrouveriez avec un projet sans fin, dans lequel on vient rajouter toujours plus de fonctionnalités à développer et de tâches à réaliser.
- C'est sur la base du périmètre de projet que l'on estime la charge de travail nécessaire, et donc bâtir le planning projet et le budget.
- Enfin, le périmètre projet est également l'une des 3 composantes du triangle d'or de la gestion de projet, avec les coûts et les délais.

01 - Analyser le cahier des charges

Périmètre du projet



Exemple

- Vous devez faire un saut à l'épicerie pour acheter des œufs.
- Sur le chemin du rayon des produits frais, vous vous souvenez que vous n'avez plus de céréales. Vous prenez donc une boîte. OK, deux boîtes. Vous vous laissez alors distraire par un couvercle rempli de bougies parfumées. Et si vous en preniez une ? Vous vous rendez alors compte que vous avez aussi besoin de serviettes en papier.
- À ce stade, coincer des articles sous votre menton et les faire tenir en équilibre précaire dans vos bras ne fonctionne plus. Les objets tombent de partout, quelles que soient vos tentatives pour les maintenir ensemble.
- Vous devez alors vous rendre tout honteux à l'avant du magasin pour prendre un chariot. Vous mettez encore quelques achats dans le chariot et vous retournez au rayon des produits frais pour prendre les œufs (vous avez failli les oublier, n'est-ce pas ?).
- Avant même que vous ne réalisiez ce qui s'est passé, votre arrêt de cinq minutes à l'épicerie s'est transformé en une heure de shopping. Et ce qui était censé vous coûter moins de 5 euros a fini par vous coûter plus de 100 euros. Et franchement ? Aviez-vous réellement besoin d'une autre bougie ?
- Dans le cadre des projets professionnels, on parlerait de « dérive des objectifs », **ce qui montre bien que la définition et la gestion du périmètre du projet valent la peine de dresser une liste à l'avance.**
- Oui, certaines personnes ne se laissent pas distraire par les serviettes en papier ou les bougies parfumées parce ces articles ne faisaient pas partie de leur liste de courses initiale. **Grâce à la gestion du périmètre du projet, vous pouvez être ce genre de personne.**

CHAPITRE 1

Analyser le cahier des charges

1. Compréhension des besoins client
2. Contexte du projet
3. Périmètre du projet
- 4. Détection des risques liés à la nature du projet**
5. Proposition des solutions possibles



01 - Analyser le cahier des charges

Détection des risques liés à la nature du projet



- Les risques font partie des informations essentielles qu'un chef de projet doit connaître sur son projet. Dès le démarrage du projet, vous devez dresser la liste la plus exhaustive possible de tous les événements générateurs de risques. Pour cela, rassemblez votre équipe et lancez un brainstorming afin de répertorier tous les dangers possibles.
- On distingue différents types de risques :
 - **Financiers** : coût supérieur à l'estimation, manque de budget, etc.
 - **Humains** : manque de compétences, absentéisme, démission au cours du projet, conflits au sein de l'équipe, etc.
 - **Temporels** : retards des sous-traitants ou des fournisseurs, mauvaise estimation des délais, etc.
 - **Techniques** : logiciel inadapté, pannes, matériel obsolète, etc.
 - **Juridiques** : réglementations et lois à respecter, faillite d'un fournisseur, etc.
 - **Environnementaux**: impacts négatifs du projet sur l'environnement, ou environnement ayant un impact sur le projet (inondation, sécheresse, tempête...).
 - **Organisationnels**: changement dans la politique de l'entreprise, changements économiques, etc.
- Vous le savez, le risque fait partie intégrante de la gestion de projet. Il est donc essentiel de mettre en place un plan de management des risques, et ce dès les premières étapes du lancement du projet. Cela permet d'identifier, de prévenir et de limiter ces risques en anticipant leur traitement grâce à la mise en œuvre d'actions préventives et correctrices.
- C'est une phase essentielle qui vous permettra de minimiser les pertes de temps et d'argent, et vous préparera à gérer efficacement le risque lorsqu'il surviendra.

CHAPITRE 1

Analyser le cahier des charges

1. Compréhension des besoins client
2. Contexte du projet
3. Périmètre du projet
4. Détection des risques liés à la nature du projet
- 5. Proposition des solutions possibles**



01 - Analyser le cahier des charges

Proposition des solutions possibles



En gestion de projet, personne n'est à l'abri d'un échec. Parfois, malgré l'investissement, les efforts et la bonne volonté du chef de projet et de son équipe, il arrive qu'un projet échoue. L'une des premières questions à se poser est de savoir quelle est la raison de cet échec.

1) Manque de visibilité sur le projet

Il arrive parfois que le chef de projet et son équipe pilotent leur projet sans aucune visibilité. La liste des tâches et le planning ont été préalablement définis lors du lancement du projet, mais ils n'ont jamais été mis à jour en fonction de sa progression. Les membres de l'équipe savent sur quelles tâches ils doivent travailler, mais ils n'ont aucune idée des priorités.

Solution :

Si vous pilotez votre projet à l'aveugle, celui-ci est voué à l'échec. Le diagramme de Gantt est un outil indispensable en gestion de projet car il permet de visualiser rapidement toutes les tâches planifiées, leur progression et leur échéance. Ainsi, vous et votre équipe avez une visibilité complète sur l'avancement du projet, les tâches en cours et celles à venir. Vous pouvez donc mieux gérer vos priorités, anticiper les retards potentiels, etc.

2) Objectifs imprécis

Un projet dont les objectifs ne sont pas clairement définis a de grandes chances d'échouer.

Pour construire une maison, si vous n'avez pas de plans précis, il est fort probable que l'équipe de construction ne sache pas par où commencer, que le chantier soit chaotique et que la maison s'effondre avant même d'être achevée. Il en est de même pour un projet.

Solution :

Dès le lancement, définissez clairement et précisément les enjeux et les objectifs du projet. C'est le point de départ essentiel à tous projets. Les objectifs doivent être précis et réalistes afin que vous et votre équipe sachiez dans quelle direction vous allez. De plus, une vision claire de votre projet va susciter l'adhésion, la loyauté et l'implication de votre équipe projet.

01 - Analyser le cahier des charges

Proposition des solutions possibles



3) Planning sous-estimé

La sous-estimation du temps nécessaire à l'accomplissement de votre projet peut avoir des conséquences plus graves que le seul fait de manquer la date butoir sur le calendrier. Vous allez dépasser votre budget prévisionnel car vous devrez payer le temps supplémentaire effectué par vos collaborateurs et les autres acteurs du projet. La livraison retardée du projet peut aussi faire manquer d'importants marchés à l'entreprise.

Solution :

Il est essentiel de préparer et d'évaluer votre planning avec précision. Des problèmes peuvent survenir à chaque étape du projet et le retarder. Vous devez donc mettre en place une gestion des risques efficace et vous accordez une marge de manœuvre en cas de problème. Il est préférable d'avoir un planning plus large et de terminer en avance, plutôt que l'inverse.

4) Aucune visibilité sur la disponibilité des ressources

Vous n'avez aucune idée de la disponibilité des membres de votre équipe. Vous ne savez pas lesquels de vos collaborateurs sont surchargés de travail et ceux qui sont disponibles. Certaines tâches prennent du retard alors que d'autres terminent en avance. Ce manque de visibilité sur la charge de travail de votre équipe nuit au bon déroulement du projet.

Solution :

Vous devez utiliser les feuilles de temps. Cet outil est indispensable pour connaître les disponibilités des membres de votre équipe, et ainsi mieux répartir la charge de travail. Allégez vos collaborateurs débordés en demandant à des collègues plus disponibles de leur venir en aide. De plus, grâce aux feuilles de temps, vous connaissez la quantité de temps déjà passée sur une tâche, ainsi que le temps restant avant son échéance : un bon moyen pour anticiper les retards.

01 - Analyser le cahier des charges

Proposition des solutions possibles



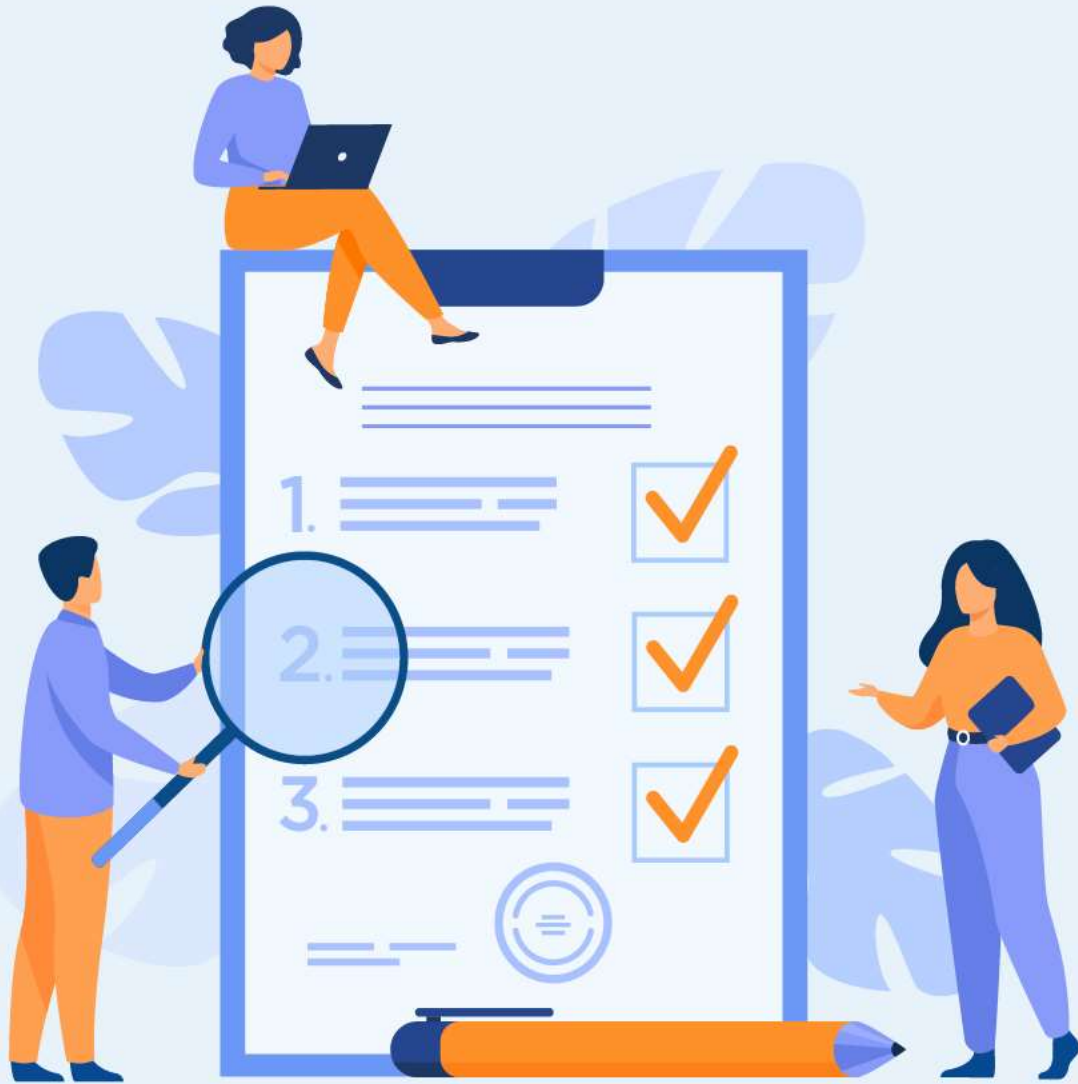
5) Mauvaise communication

Une mauvaise communication, voire l'absence totale de communication au sein de l'équipe est une des principales causes d'échec du projet. Une équipe qui ne communique pas va s'enfermer et se replier sur elle-même. L'ambiance de travail se dégrade, ce qui a des conséquences néfastes sur le déroulement et la réussite du projet.

Solution :

Il est indispensable d'entretenir un dialogue constant et constructif entre l'équipe et le chef de projet, mais aussi avec les autres acteurs impliqués. Cela améliore le travail en équipe et permet à chacun d'être informé de l'évolution du projet en temps réel.

Favoriser la communication et les échanges rend votre équipe plus productive, développe la confiance entre les membres, ainsi qu'un sentiment de loyauté. Instaurez un climat de confiance en étant à l'écoute de votre équipe. Les réunions de suivi ainsi que les logiciels de gestion de projet collaboratifs permettent de maintenir une bonne communication au sein de l'équipe.



CHAPITRE 2

Préparer le projet

Ce que vous allez apprendre dans ce chapitre :

- Répartition de l'ensemble des fonctionnalités en tâches
- Estimation de la durée de réalisation de chaque tâche
- Ordonnancement des tâches
- Chemin critique
- Echancier et la chronologie des tâches
- Affectation des ressources aux tâches
- Maîtrise des coûts
- Détermination des points de validation



? heures

CHAPITRE 1

Préparer le projet

- 1. Répartition de l'ensemble des fonctionnalités en tâches**
2. Estimation de la durée de réalisation de chaque tâche
3. Ordonnancement des tâches
4. Chemin critique
5. Echancier et la chronologie des tâches
6. Affectation des ressources aux tâches
7. Maîtrise des coûts
8. Détermination des points de validation



02 - Préparer le projet

Répartition de l'ensemble des fonctionnalités en tâches

- Répartition de l'ensemble des fonctionnalités en tâches est une décomposition hiérarchique des travaux nécessaires pour réaliser les objectifs d'un projet.
- Elle a pour but d'aider à organiser le projet, en définissant la totalité de son contenu et en servant de référence pour planifier les activités et établir le budget prévisionnel. Elle est également utilisée pour guider la gestion des risques, ou identifier les acquisitions nécessaires. Elle permet également de déléguer et de contractualiser la mission confiée à chaque acteur.
- Il existe en principe trois modèles :

Approche descendante

- Dans l'approche descendante, on part du résultat global et on décompose le projet en sous-projets de plus en plus détaillés, puis en groupes de tâches. Cette approche convient tout particulièrement lorsque l'on connaît les contenus du projet ou dispose déjà d'une expérience dans des projets similaires.

Approche ascendante

- Avec l'approche ascendante, vous suivez la direction inverse et commencez par le niveau le plus bas. On note d'abord toutes les tâches qui nous viennent à l'esprit, on les regroupe en groupes de tâches, puis on affecte ces derniers à des sous-projets. Cette approche est notamment pertinente lorsqu'un projet nous amène à découvrir de nouveaux domaines.

Approche combinée

- L'approche combinée est une combinaison des deux techniques précédentes. On procède ici par étapes successives : dans un premier temps, on liste les tâches. On note ensuite les sous-projets et on leur attribue les tâches précédemment listées. Finalement, on énumère les autres choses à faire. Cette approche jongle ainsi avec les approches ascendante et descendante, ce qui permet de profiter des avantages des deux méthodes. Dans ce cadre, il est impératif de ne pas oublier ou répéter un élément.

02 - Préparer le projet

Répartition de l'ensemble des fonctionnalités en tâches

Organigramme des tâches du projet : exemple et modèle

- Si nous souhaitons créer un nouveau site Internet pour l'entreprise, un organigramme des tâches du projet axé sur les phases ressemblerait dans les grandes lignes à ce qui suit :

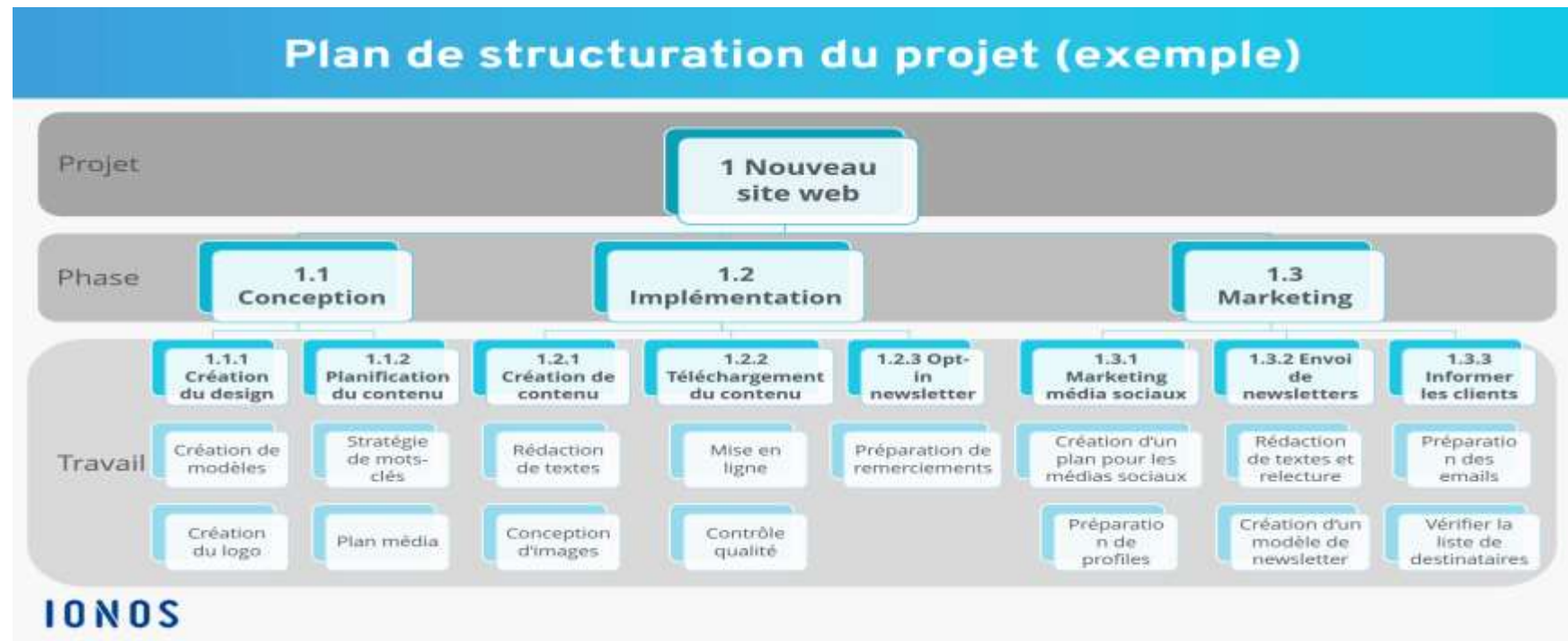


Figure 8 : Organigramme des tâches du projet : exemple et modèle
https://www.ionos.fr/startupguide/fileadmin/StartupGuide/Screenshots_2020/organigramme-des-taches-du-projet.png

02 - Préparer le projet

Répartition de l'ensemble des fonctionnalités en tâches



Les tâches peuvent être décrites dans un tableau grâce à différentes caractéristiques.

Caractéristiques intrinsèques : ces caractéristiques peuvent être déterminées dès l'identification des tâches:

- **Libellé de la tâche** : titre court facilement manipulable
- **Description** : informations complémentaires au libellé si besoin
- **Données d'entrée** : éléments nécessaires à la réalisation de la tâche
- **Données de sortie** : produit de la tâche
- **Compétences nécessaires** : liste des compétences nécessaires à la réalisation de la tâche
- **Charge de travail** : nombre de jours ou d'heures nécessaire à la réalisation de la tâche par une personne compétente
- **Contraintes** : si une contrainte du projet a un impact direct sur la tâche (ex : jalon...)

Caractéristiques extrinsèques : ces caractéristiques pourront être déterminées après la constitution de l'équipe projet et la planification:

- **Nom du responsable de la tâche** : personne qui rendra compte de l'avancement de la tâche au chef de projet
- **Ressources humaines** : avant la constitution de l'équipe, lister les personnes susceptibles d'être affectées à la tâche. Une fois l'équipe constituée, on fera apparaître le nom de la (des) personne(s) retenue(s).
- **Ressources matérielles** : matériel associé à la tâche
- **Synchronisation** : lister les liens (prédécesseur - suivant) avec les autres tâches
- **Date de début programmée** : ne pourra être rempli qu'après la planification de l'ensemble des tâches
- **Date de fin programmée** : ne pourra être rempli qu'après la planification de l'ensemble des tâches

CHAPITRE 1

Préparer le projet

1. Répartition de l'ensemble des fonctionnalités en tâches
- 2. Estimation de la durée de réalisation de chaque tâche**
3. Ordonnancement des tâches
4. Chemin critique
5. Echancier et la chronologie des tâches
6. Affectation des ressources aux tâches
7. Maîtrise des coûts
8. Détermination des points de validation



02 - Préparer le projet

Estimation de la durée de réalisation de chaque tâche



❖ Données historiques (ou estimation analogue)

- Basez vos estimations sur les résultats d'un projet précédent. Si votre entreprise a déjà réalisé un projet similaire et que vous pouvez accéder au temps passé pour une tâche similaire, cela vous donnera une bonne idée de la durée de la même tâche.
- Encore une fois, si vous n'avez pas ces informations en interne, vous pourrez sûrement vous appuyer sur des entreprises partenaires ou des données publiées sur votre industrie.
- Ce type d'estimation est rapide, mais vous n'obtiendrez pas de résultats extrêmement précis.

❖ Analyse statistique et mathématique (ou estimation paramétrique)

- Si les mathématiques ne faisaient pas partie de vos matières préférées à l'école, vous n'aimeriez pas cette partie. Si la tâche que vous évaluez est relativement simple et répétitive, vous pouvez utiliser des calculs simples pour calculer la durée globale de l'activité.
- **Exemple** : s'il faut une heure à une personne pour creuser un trou et que vous avez besoin de dix trous, cela prendra dix heures de travail. Si vous doublez les ressources et disposez de deux excavatrices, cela ne prendra que cinq heures. Si vous apportez une machine qui fonctionne au double de la vitesse d'une personne, cela ne prendra que deux heures et demie.
- Pas toujours applicable mais efficace sur certaines tâches simples.

02 - Préparer le projet

Estimation de la durée de réalisation de chaque tâche



❖ Estimer une durée

- Pour chaque tâche, prenez les trois durées suivantes : optimiste (Do), pessimiste (Dp) et la plus probable (Dc).
- Calculez (et utilisez alors) la durée moyenne (DM) :
 - $DM = (Do + Dp + 4Dc) / 6$.
- L'unité de mesure doit être maniable et rester significative par rapport à la tâche.

❖ Durée : à partir de la charge de travail nécessaire, calculer la durée de la tâche en fonction

- Du nombre de personnes affectées à la tâche
 - Du niveau de compétences des personnes
 - De la disponibilité des personnes
 - De la disponibilité des éventuelles ressources matérielles affectées à la tâche
-
- Le travail représente la charge (en heures, en jours...) de travail nécessaire à la réalisation de la tâche par une seule personne occupée à 100% de son temps de travail sur la tâche. Le travail s'exprime en "jour-homme", "heure-homme", "mois-homme", etc.
 - Expliqué autrement, un "jour-homme" correspond au travail d'une personne pendant un jour. Un projet présentant 10 jours-homme de travail peut être réalisé en 10 jours par 1 personne, en 1 jour par 10 personnes, en 20 jours par une personne disponible à 50%, etc.

CHAPITRE 1

Préparer le projet

1. Répartition de l'ensemble des fonctionnalités en tâches
2. Estimation de la durée de réalisation de chaque tâche
- 3. Ordonnement des tâches**
4. Chemin critique
5. Echancier et la chronologie des tâches
6. Affectation des ressources aux tâches
7. Maîtrise des coûts
8. Détermination des points de validation



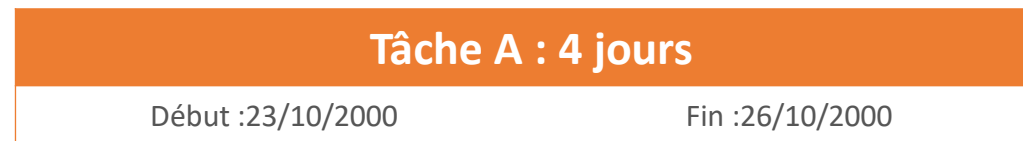
02 - Préparer le projet

Ordonnancement des tâches

Les méthodes d'ordonnancement permettent d'élaborer un graphe qui représente l'ensemble des tâches composant le projet ainsi que les liens qui existent entre elles. Sur le graphe, apparaissent également la durée de chaque tâche, la date à laquelle elle peut débuter au plus tôt et au plus tard.

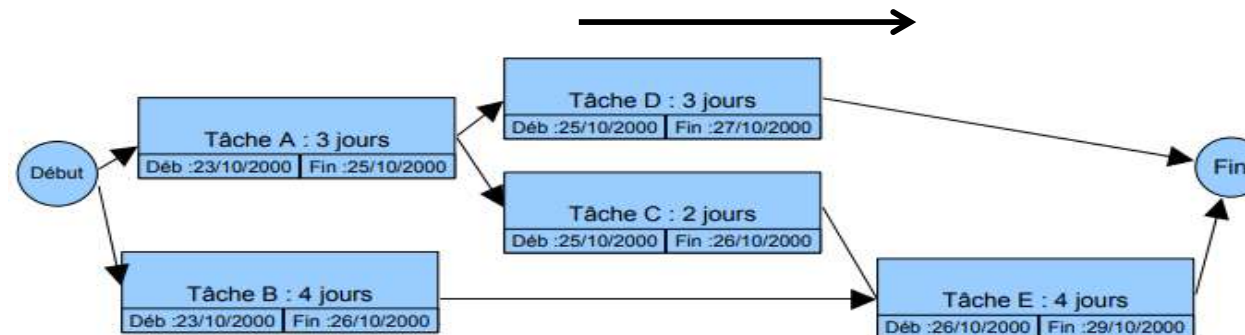
1. LA METHODE M.P.M (Méthode des Potentiels et antécédents Métra).

- **Principe de la méthode** : Cette méthode permet de réduire la durée totale d'un projet. On étudie les délais sans prendre en compte les charges et les moyens disponibles.
- **Notions de base** : La méthode est une représentation graphique qui permet de bâtir un « réseau ». Ce réseau est constitué par des tâches (ou étapes).



- **Liaison orientées** : Elles représentent les contraintes d'antériorités des tâches.

Exemple de réseau :



La méthode MPM a pour but de planifier la durée d'un projet, aussi nous devons mener des calculs sur le graphe afin d'en déduire des renseignements sur son excitabilité.

Méthodologie de construction d'un réseau MPM.

- Établir la liste des tâches (faire le partitionnement des tâches en fonction des ressources).
- Déterminer des antériorités : tâches immédiatement antérieures, et tâches antérieures.
- Déterminer les niveaux d'exécution ou rang des tâches (très facile avec cette méthode).
- Construire le réseau MPM.
- Calculer la durée du projet, les dates début et de fin des tâches. Déterminer le chemin critique. Impossible ici de mettre en évidence les marges : voir diagramme de Gantt.

2. Le diagramme de GANTT

- **Le diagramme de GANTT** est un graphique (chrono gramme) qui consiste à placer les tâches chronologiquement en fonction des contraintes techniques de succession (contraintes d'antériorités).
- L'axe horizontal des abscisses représente le temps et l'axe vertical des ordonnées les tâches.
- On représente chaque tâche par un segment de droite dont la longueur est proportionnelle à sa durée. L'origine du segment est calée sur la date de début au plus tôt de l'opération (« jalonnement au plus tôt ») et l'extrémité du segment représente la fin de la tâche.
- Ce type de graphe présente l'avantage d'être très facile à lire, mais présente l'inconvénient de ne pas représenter l'enchaînement des tâches. Cette méthode est généralement utilisée en complément du réseau MPM. On trace le plus souvent le GANTT au plus tôt ou « jalonnement au plus tôt » et éventuellement au plus tard « jalonnement au plus tard ».

02 - Préparer le projet

Ordonnancement des tâches

Exemple :

Tâche	A	B	C	D	E	F	G	H	I	J	K	L	M
Durée	1	2	1	3	2	5	2	5	2	1	4	5	4
Antériorités	-	-	A	-	B	E	C,D	-	H	-	I,J	F,G	K,L

Figure 9 : Tableau qui regroupe toutes les tâches avec durée et antériorités

Réseau MPM :

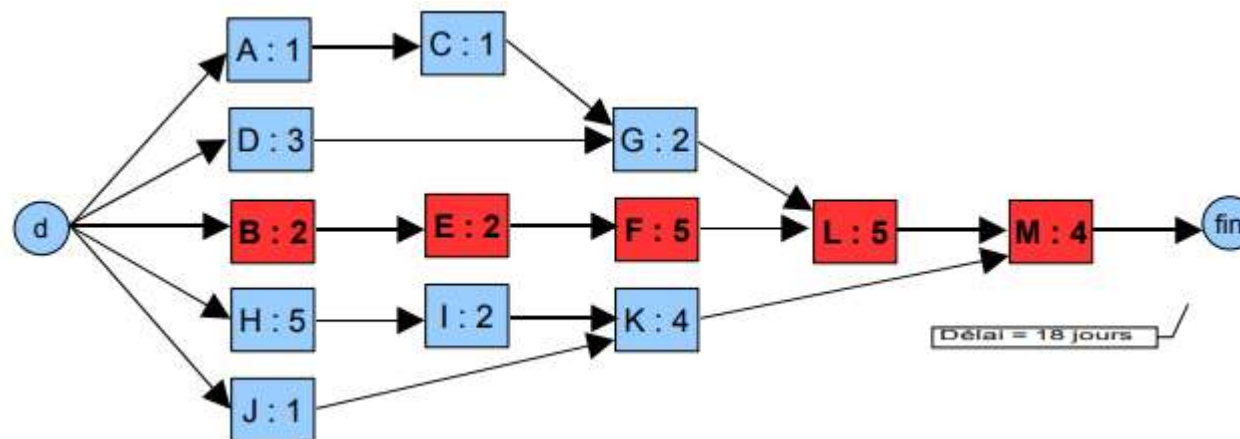


Figure 10 : Exemple de réseau MPM

02 - Préparer le projet

Ordonnancement des tâches

Diagramme de Gantt : (avec Microsoft PROJECT) « GANTT au plus tôt »

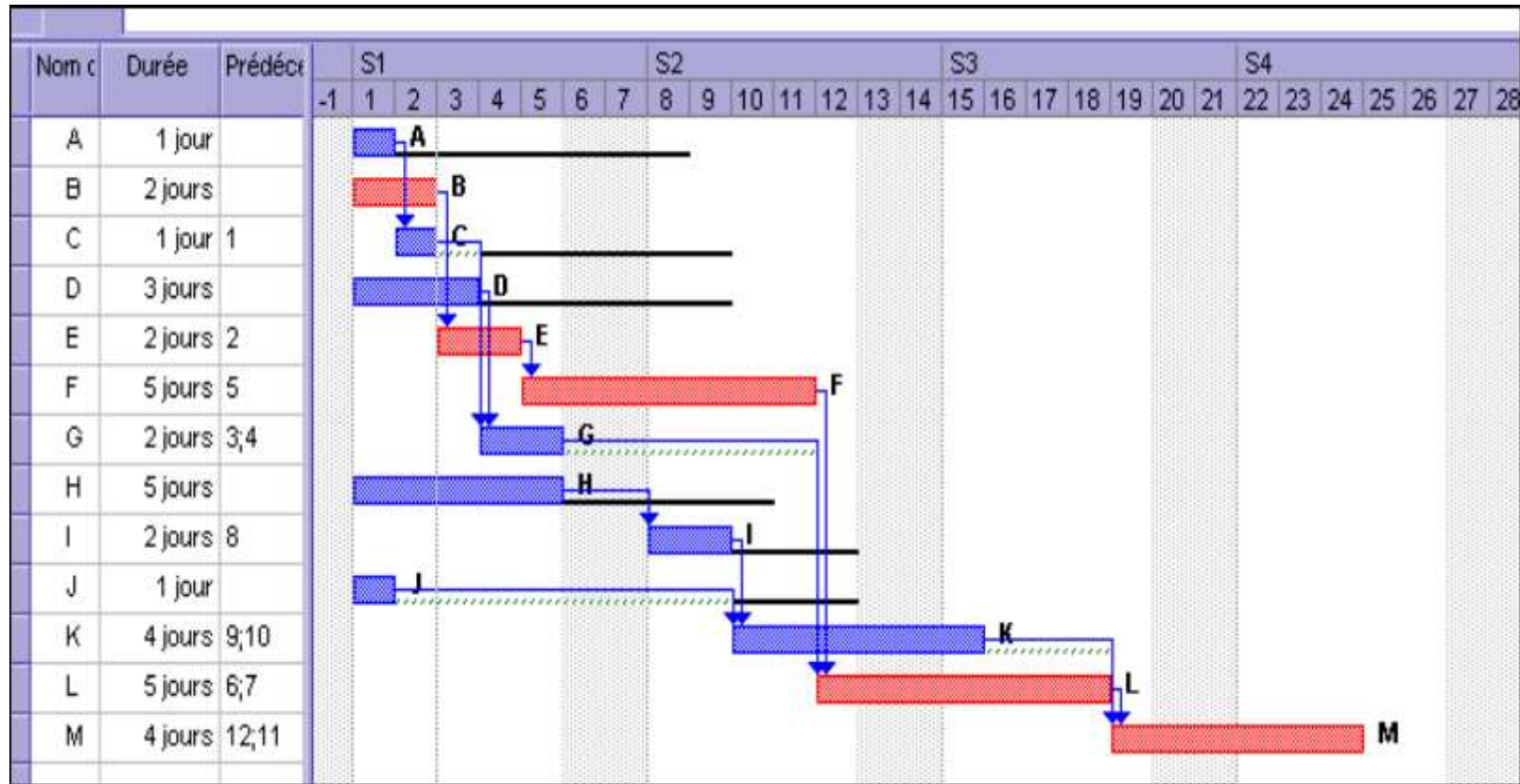


Figure 11 : Exemple de Diagramme de Gantt

02 - Préparer le projet

Ordonnancement des tâches



- **Analyse :**

- Le projet est réalisable en 18 jours ouvrés. Ici, avec les fins de semaines non travaillées il faudra 3 semaines et 3 jours.
- Les tâches normales sont représentées en bleu.
- Les tâches critiques sont représentées en rouge : B, E, F, L, et M.
- On distingue les marges totales en noir, et les marges libres en vert. Les tâches C, G, J, et K font apparaître de la Marge Libre.

- **Remarques :**

- Le diagramme de GANTT sera modifié au fur et à mesure de l'avancement du projet. Il faut mettre à jour ce diagramme régulièrement. Le chemin critique peut évoluer en fonction de l'avancement, du retard, ou de toute modification sur une tâche. Les chemins « sub-critiques » ou « presque critiques » peuvent alors devenir critiques.

M.P.M. :

- « Méthode des Potentiels Métra » = Planning sous forme de réseau représentant graphiquement l'ordonnancement des opérations d'un projet.

- **Ordonnancement :**

- L'ordonnancement, c'est l'arrangement qui permet d'exécuter séquentiellement les tâches ou les ordres de fabrication, de façon à ce que l'ensemble du projet ou de la production soit achevé dans le temps imparti.

CHAPITRE 1

Préparer le projet

1. Répartition de l'ensemble des fonctionnalités en tâches
2. Estimation de la durée de réalisation de chaque tâche
3. Ordonnancement des tâches
- 4. Chemin critique**
5. Echancier et la chronologie des tâches
6. Affectation des ressources aux tâches
7. Maîtrise des coûts
8. Détermination des points de validation
9. Elaboration du digramme de Gantt



Définition

- La méthode du **chemin critique (Critical Path Method, CPM)** est un algorithme basé sur les mathématiques pour programmer un ensemble d'activités relatives à un projet. La technique principale pour utiliser le CPM consiste à construire un modèle du projet qui inclut les éléments suivants :
 - Une liste de toutes les activités requises pour finaliser le projet,
 - Les dépendances entre les activités, et une estimation du temps (durée) nécessaire à la réalisation de chaque activité.
- À l'aide de ces valeurs, le **CPM** calcule habituellement le chemin le plus long des activités planifiées jusqu'à la fin du projet et les dates de début et de fin au plus tôt et au plus tard que chaque activité peut avoir sans prolonger le projet. Ce processus détermine les activités « critiques » (c'est-à-dire se trouvant sur le chemin le plus long) et celles ayant une « marge totale » (c'est-à-dire pouvant être retardées sans que le projet dure plus longtemps).
- Les tâches du chemin critique sont parfois appelées “tâches critiques” car elles ont une marge nulle, c'est-à-dire qu'elles ne peuvent pas subir de retard et si elles ont du retard, elles retardent l'intégralité du projet.
- Une tâche ou activité du chemin critique ne peut pas être débutée avant que l'activité qui la précède et qui a un lien de dépendance avec elle, soit terminée.

02 - Préparer le projet

Chemin critique



Comment trouver le chemin critique en 4 étapes ?

- Nous venons de voir que la méthode du chemin critique est une technique de gestion de projet qui permet d'identifier étape par étape les activités qui ne permettent aucun retard de livraison.
- Pour calculer le **CPM (Critical Path Method)**, nous allons suivre ce processus de 4 étapes :
 - Découper le projet en tâches et déterminer leurs dépendances
 - Estimer la durée de chaque tâche
 - Créer le réseau PMP ou PERT "Program Evaluation and Review Technic"
 - Identifier les tâches critiques

Exemple :

1. Découper le projet en tâches

Pour définir nos tâches, nous allons prendre un exemple de tâches hypothétiques de la phase de démarrage d'un projet de réalisation d'un site web :

N° TÂCHE	NOM DE LA TÂCHE
A	Analyse De L'existant
B	Objectifs Du Nouveau Site
C	Analyse De La Concurrence
D	Architecture Du Site
E	Charte Graphique
F	Etude SEO

Figure 12 : Découpage de projet en tâches

02 - Préparer le projet

Chemin critique

2. Estimer chaque tâche

- Une fois que le projet a été divisé en tâches, la durée et le coût de chaque tâche doivent être estimés, en se basant sur des projets précédents et l'expérience des membres de l'équipe.
- Voici le résultat pour notre exemple de projet de construction d'un site web :

N° TÂCHE	NOM DE LA TÂCHE	DURÉE [EN JOUR]	PRÉDÉCESSEUR	SUCCESEUR
A	Analyse De L'existant	3	-	B
B	Objectifs Du Nouveau Site	2	A	C,D
C	Analyse De La Concurrence	4	B	F
D	Architecture Du Site	5	B	E
E	Charte Graphique	7	D	-
F	Etude SEO	4	C	-

Figure 13 : Estimation de chaque tâche

3. Créer le réseau de PERT

Pour dessiner le réseau PERT, vous allez représenter les étapes du projet par des cercles, et les tâches pour atteindre ces étapes par des flèches.

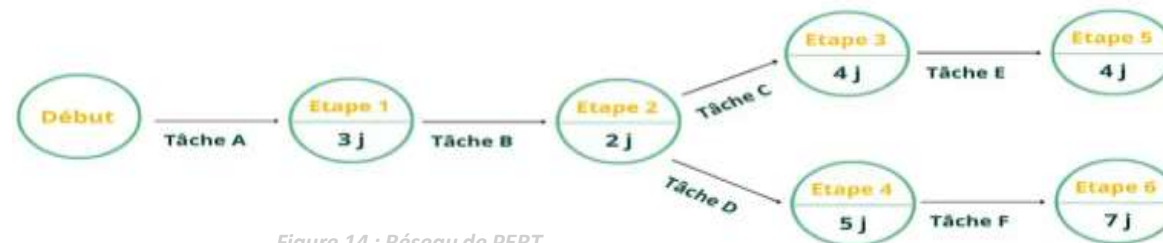


Figure 14 : Réseau de PERT

02 - Préparer le projet

Chemin critique

4. Identifier le chemin critique

- Pour cela, il faudra d'abord identifier les tâches critiques.
- Ce sont celles qui ont la plus longue durée.
- Le CPM (Critical Path Method) dans cet exemple est donc A-B-D-E.
- Et la durée totale de ce projet revient à additionner les durées des tâches critiques.
- Ce qui nous donne finalement une durée de 17 jours.
- Voici le chemin critique tracé sur le réseau PERT :

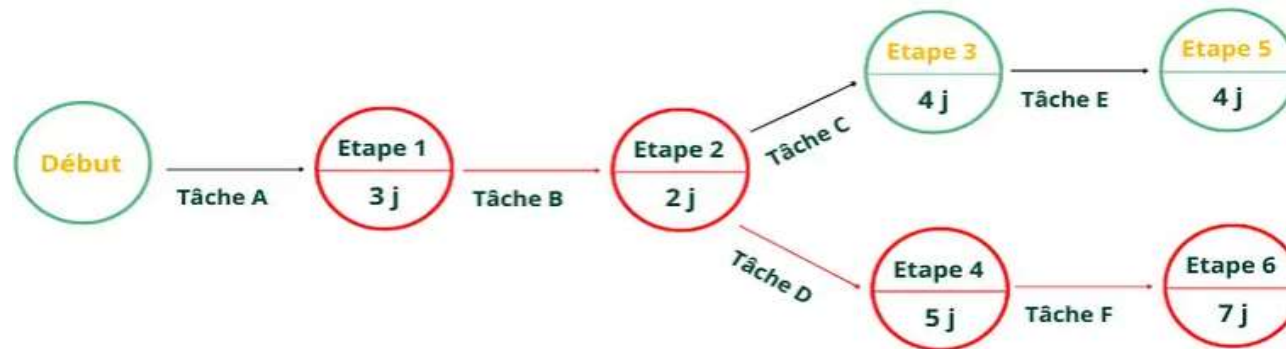



Figure 15 : Identification du chemin critique

CHAPITRE 1

Préparer le projet

- 
1. Répartition de l'ensemble des fonctionnalités en tâches
 2. Estimation de la durée de réalisation de chaque tâche
 3. Ordonnancement des tâches
 4. Chemin critique
 - 5. Echancier et la chronologie des tâches**
 6. Affectation des ressources aux tâches
 7. Maîtrise des coûts
 8. Détermination des points de validation

02 - Préparer le projet

Echéancier et la chronologie des tâches



Définition

Mettre sur pied un échéancier de projet est essentiel pour garantir la réussite de votre équipe et le respect des délais. Pour visualiser votre plan de projet ou votre processus de travail sous forme de chronologie, suivez ces étapes :

1. Énumérer les étapes du projet et définir clairement les échéances

Commencez par diviser le travail en tâches et attribuer clairement les responsabilités. Ensuite, ajoutez des dates de début et des échéances qui vous aideront à estimer le temps nécessaire à chaque étape.

2. Ordonner vos tâches

Quelles tâches doivent-elles être accomplies en premier ? Certaines dépendent-elles de l'achèvement d'autres tâches ? Quelle sera la dernière étape ? C'est en répondant à ces questions que vous parviendrez à ordonner vos tâches.

3. Partager votre échéancier

Partagez votre chronologie de projet avec vos collègues. Toute l'équipe est alors sur la même longueur d'onde concernant le plan du projet et les responsabilités de chacun, ce qui favorise l'adhésion des parties prenantes.

4. Visualiser la progression du projet

Suivez la progression de votre équipe et assurez-vous que chacun est sur la bonne voie pour atteindre les objectifs fixés et respecter les délais. Sans oublier de publier des mises à jour de statut lorsque des jalons sont atteints pour informer l'équipe des dernières évolutions et la prévenir de changements éventuels.

CHAPITRE 1

Préparer le projet

1. Répartition de l'ensemble des fonctionnalités en tâches
2. Estimation de la durée de réalisation de chaque tâche
3. Ordonnancement des tâches
4. Chemin critique
5. Echancier et la chronologie des tâches
- 6. Affectation des ressources aux tâches**
7. Maîtrise des coûts
8. Détermination des points de validation



02 - Préparer le projet

Affectation des ressources aux tâches



Définition

Il s'agit du processus qui consiste à affecter et planifier les ressources disponibles de la manière la plus efficace et la plus économique possible. Les ressources sont indispensables aux projets, mais elles se font rares. Il appartient donc au chef de projet de planifier les ressources au bon moment en respectant le calendrier du projet.

Les différents types de ressources d'un projet

- Une ressource est une entité matérielle ou immatérielle exploitée pour la réalisation d'une tâche. Quelles ressources participent à la mise en œuvre d'un projet ? On peut les répartir selon les catégories suivantes :
 - ressources humaines,
 - ressources matérielles,
 - ressources financières,
 - ressources en temps.

❖ Créer un planning des ressources

- Le planning représente visuellement l'organisation et vos besoins en ressources, qu'elles soient humaines, matérielles ou financières, sur une période donnée.

Par exemple, vous pouvez utiliser le diagramme de Gantt pour planifier votre projet : vous visualisez ainsi l'allocation des ressources en fonction des tâches et du temps imparti, et les éventuels conflits d'utilisation dans l'affectation des ressources.

02 - Préparer le projet

Affectation des ressources aux tâches




❖ Effectuer un suivi des ressources

- Pour évaluer une situation et prendre les décisions qui s'imposent, vous avez tout intérêt à utiliser des indicateurs de performance ou KPI (Key Performance Indicators).
- Comparez à l'aide de ces indicateurs le «prévisionnel» avec le «réalisé» sur plusieurs plans :
 - **ressources humaines** : quelle est la productivité des ressources ?
Exemple de calcul : nombre de JH (jours/homme) alloués à une tâche multiplié par le pourcentage de la réalisation de cette tâche.
 - **ressources matérielles** : quelle est la disponibilité ou la capacité de tel équipement ?
Exemple de calcul : nombre d'heures de travail prévues sur un équipement par rapport au nombre d'heures disponibles de ce dernier = charge équipement.
 - **ressources financières** : quel est le coût actuel de mon projet ? Respecte-t-il le budget alloué ?
Exemple de calcul : addition de toutes les dépenses consacrées au projet jusqu'au moment T.
- Comparez ensuite les résultats obtenus avec ce qui était initialement prévu et prenez les décisions adéquates selon la situation, qu'il s'agisse de tenue des délais ou de consommation du budget.

CHAPITRE 1

Préparer le projet

- 
1. Répartition de l'ensemble des fonctionnalités en tâches
 2. Estimation de la durée de réalisation de chaque tâche
 3. Ordonnancement des tâches
 4. Chemin critique
 5. Echancier et la chronologie des tâches
 6. Affectation des ressources aux tâches
 - 7. Maîtrise des coûts**
 8. Détermination des points de validation

02 - Préparer le projet

Maîtrise des coûts



- La maîtrise des coûts consiste à superviser et à gérer les dépenses du projet et à se préparer aux risques financiers potentiels. Cette tâche est généralement du ressort du chef de projet.
- La maîtrise des coûts implique non seulement la gestion du budget, mais aussi la planification et la préparation aux risques potentiels. Les risques peuvent retarder les projets et parfois même entraîner des dépenses imprévues. La préparation à ces revers peut permettre à votre équipe d'économiser du temps et, potentiellement, de l'argent.
- La maîtrise des coûts suppose une grande discipline et commence dès :

La phase de faisabilité du projet

- Dans un premier temps, la technique utilisée est une estimation analogique, c'est à dire une estimation à partir de projets analogues (combien coûte la construction d'une maison de 150 m² habitables ? Entre 150 K€ et 300 K€, soit 1000 à 2000 € le m²).

Dans la phase d'avant projet

- le projet est détaillé, des choix techniques sont arrêtés ou proposés, la méthode paramétrique sera utilisée (maison de deux niveaux, avec sous sol, deux salles de bain, matériaux nobles, isolation renforcée, 6 pièces, trois salles de bain, deux WC, Le coût sera affiné avec un degré de précision plus grand (la maison coûtera entre 240 et 280 K€). A la fin de la phase d'avant projet, les derniers choix techniques doivent être confirmés (types d'équipements de la salle de bain et de la cuisine, nature des revêtements, ...).

Avant de démarrer le projet


- le chef de projet construira le budget initial détaillé, méthode analytique, en s'appuyant sur des devis ou sur des estimations argumentées et précises. Ce budget servira de référence pour évaluer ultérieurement les dérives éventuelles lors du suivi du projet. Il s'agit d'une estimation contractuelle qui lie le chef de projet et le donneur d'ordre.

Tout au long de la réalisation

- le niveau des dépenses sera comparé au niveau prévu et quelques fois des actions correctives seront proposées

CHAPITRE 1

Préparer le projet

- 
1. Répartition de l'ensemble des fonctionnalités en tâches
 2. Estimation de la durée de réalisation de chaque tâche
 3. Ordonnancement des tâches
 4. Chemin critique
 5. Echancier et la chronologie des tâches
 6. Affectation des ressources aux tâches
 7. Maîtrise des coûts
 - 8. Détermination des points de validation**

02 - Préparer le projet

Détermination des points de validation



La création du dossier de faisabilité

- Le dossier de faisabilité est un document descriptif abordant les aspects techniques, qualité, financiers et calendaires d'un projet. Il doit permettre à toutes les personnes concernées d'appréhender les objectifs et les enjeux du projet et de statuer sur sa validation finale.
- Un dossier de faisabilité doit être établi, quelle que soit la nature d'un projet : développement d'un nouveau produit, service ou prestation, nouveau projet technique de construction, etc.
- Il a pour but de :
 - Définir le programme complet du projet (cahier des charges : analyse fonctionnelle, en définissant les caractéristiques principales et secondaires du projet)
 - Justifier le projet en terme économique et/ou statistique et/ou stratégique, ou à l'aide d'une Analyse SWOT (Strength/Forces, Weakness/Faiblesses, Opportunities/Opportunités, Threats/Menaces)
 - Elaborer et justifier le budget nécessaire à sa réalisation,
 - Evaluer le coût global du projet : coûts prévisionnels d'exploitation, de maintenance, de stockage, des énergies nécessaires, etc.
 - Identifier toutes les contraintes réglementaires et normes en initialisant une analyse de risques projet
 - Définir un planning d'étude, de réalisation et de mise en œuvre afin d'estimer les toutes les phases et durées du projet (Planning de GANTT ou PERT)
 - Apporter tous les éléments nécessaires permettant la bonne compréhension du projet
 - Déterminer tous les impacts possibles sur l'organisation, les opérations et les ressources
 - Identifier et évaluer les risques



PARTIE 3

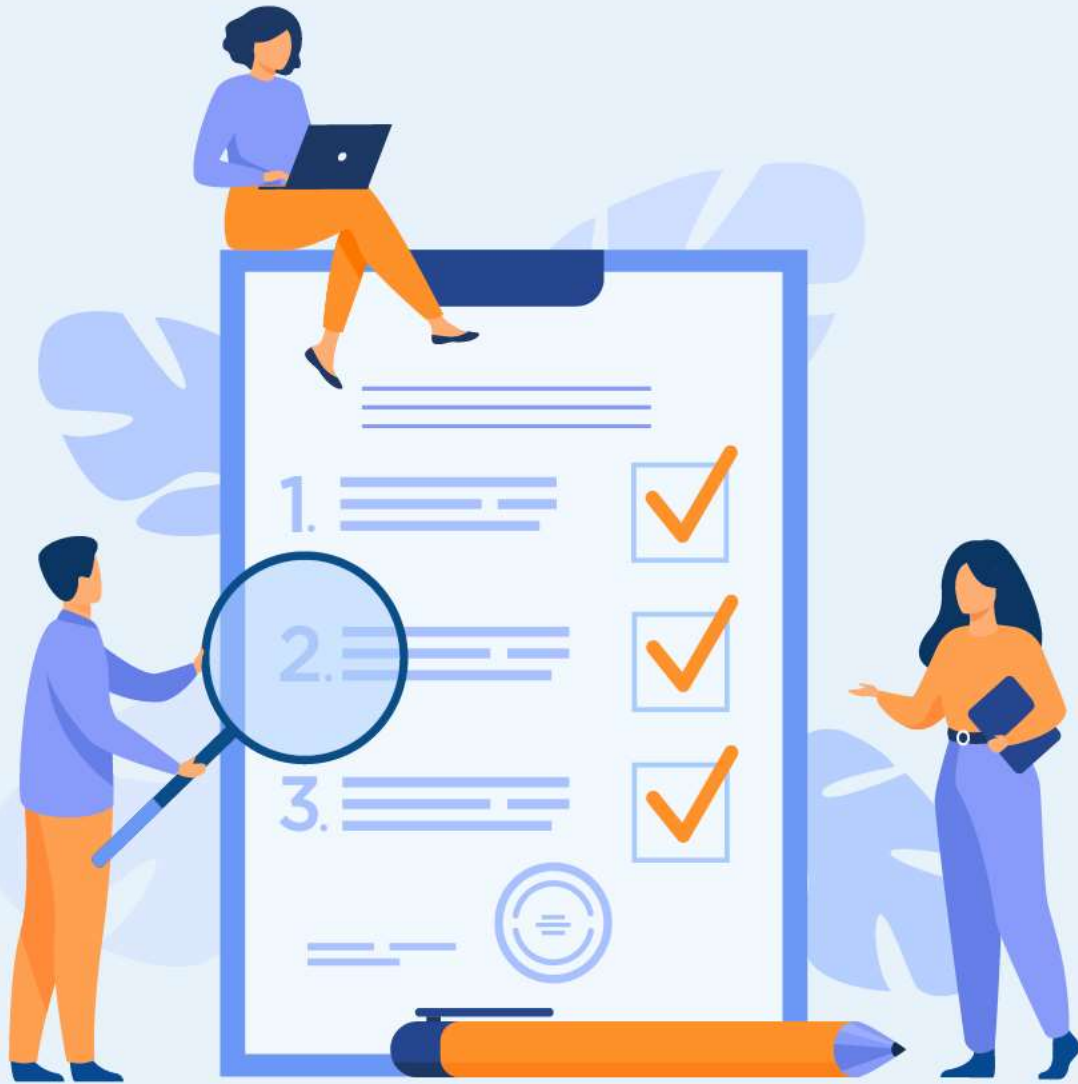
Adopter l'approche Agile dans gestion de projet

Dans ce module, vous allez :

- Appréhender la méthodologie Agile Scrum
- Manipuler l'outil de gestion de projet Agile (Scrum/Jira)



? heures



CHAPITRE 1

Appréhender la méthodologie Agile Scrum

Ce que vous allez apprendre dans ce chapitre :

- Définition de la méthode Agile Scrum
- Manifest Agile (valeurs et principes)
- Processus de la méthode Scrum : pre-game, game, post-game
- Rôles et responsabilités
- Événements Scrum
- Artéfacts Scrum



? heures

CHAPITRE 1

Appréhender la méthodologie Agile Scrum

1. Définition de la méthode Agile Scrum

2. Manifest Agile (valeurs et principes)
3. Processus de la méthode Scrum : pre-game, game, post-game
4. Rôles et responsabilités
5. Evénements Scrum
6. Artéfacts Scrum



01 - Appréhender la méthodologie Agile Scrum

Définition de la méthode Agile Scrum

Définition de la méthode Agile Scrum

Définition : L'approche SCRUM est une méthode agile consacrée à la gestion de projet.

- Son objectif phare est d'améliorer la productivité des équipes, tout en permettant une optimisation du produit grâce à des feedbacks réguliers du marché.
- En parallèle, la méthode SCRUM permet d'avoir une vue d'ensemble du projet pour chacune des parties prenantes.
- Et elle permet aussi la réduction des bugs et une mise à jour régulière des priorités.

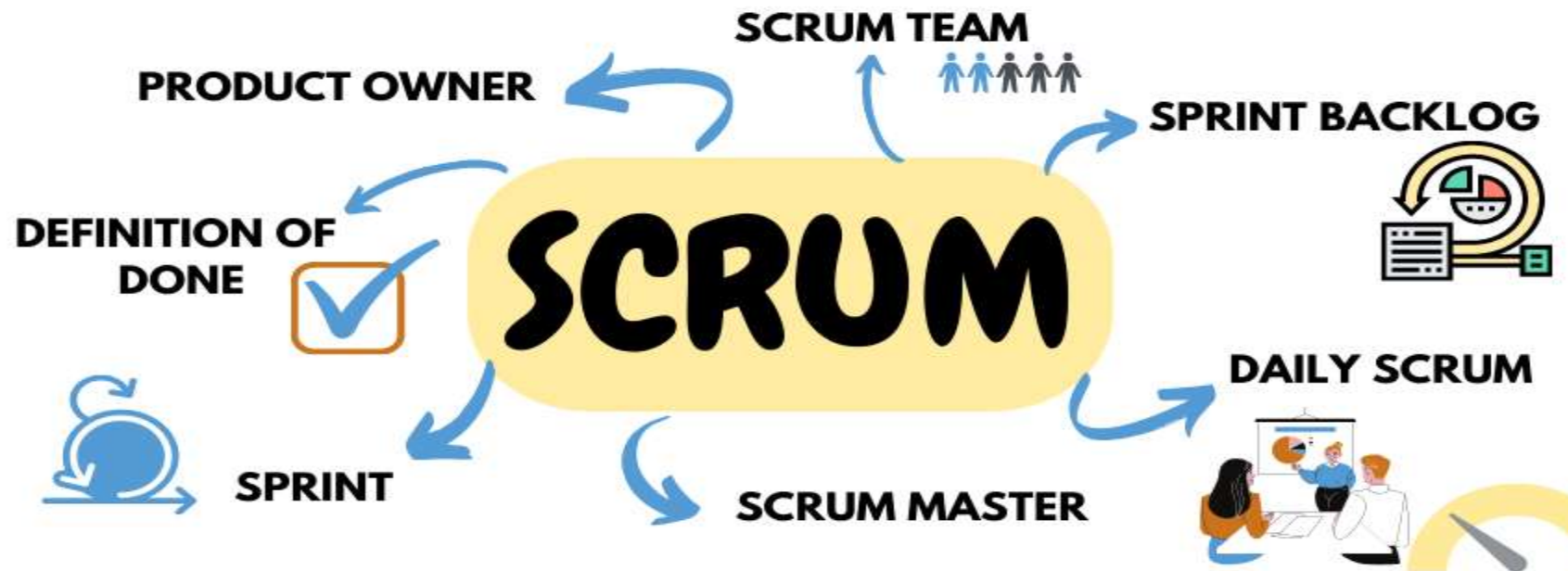


Figure 16 : La méthode agile Scrum

CHAPITRE 1

Appréhender la méthodologie Agile Scrum

1. Définition de la méthode Agile Scrum
- 2. Manifest Agile (valeurs et principes)**
3. Processus de la méthode Scrum : pre-game, game, post-game
4. Rôles et responsabilités
5. Evénements Scrum
6. Artéfacts Scrum



Manifest Agile (valeurs et principes)

- La notion de manifeste agile sous-entend l'idée d'une déclaration formelle.
- On comprend qu'il s'agit de la représentation d'un idéal à atteindre.
- Il s'agit du fruit d'une rencontre entre développeurs logiciel qui se sont réunis dans l'intention de partager leurs expériences.
- Son objectif est de cadrer la pratique agile **avec 4 valeurs** et **12 principes**. Voyons chacun de ces valeurs et principes agiles plus en détails.

1) Les individus et les interactions plus que les processus et les outils

- Qui construit la valeur d'une entreprise, qui crée les produits ? Ce sont bien les équipes.
- Une entreprise avec des équipes passionnées sera une entreprise à succès.
- Encore plus si elle utilise de bons processus.
- Des équipes engagées et passionnées sauront créer un produit de valeur et le contenu pour convaincre.

2) Des logiciels opérationnels plus qu'une documentation exhaustive

- Une équipe de développement devrait se concentrer sur la production de fonctionnalités et non sur la rédaction de documents.
- Avec les méthodes de gestion de projet classiques, les chefs de projet peuvent passer des heures à mettre à jour des présentations ou des calendriers.
- La méthode Agile permet d'avoir des modèles de documents, qui sont toujours les mêmes, qui se complètent rapidement avec l'essentiel des informations.

01 - Appréhender la méthodologie Agile Scrum

Manifest Agile (valeurs et principes)



3) La collaboration avec les clients plus que la négociation contractuelle

- Avec les méthodes traditionnelles de gestion de projet, le moindre changement budgétaire, fonctionnel ou dans les délais, nécessitent de revoir totalement le projet.
- Elles incitent à fortement négocier avec le client.
- Avec la méthode Agile, il est facile de faire des modifications dans un produit.
- La collaboration est simplifiée.

4) L'adaptation au changement plus que le suivi d'un plan

- La dernière valeur du manifeste agile est l'acceptation des demandes changements même tard dans le développement du produit.
- Les approches agiles de planification et de hiérarchisation permettent aux équipes produit de réagir rapidement au changement.
- La flexibilité des approches agiles augmente la stabilité du projet.
- En effet, le changement dans les produits agiles est prévisible.
- Si de nouveaux événements adviennent, l'équipe produit intègre ces modifications dans les développements en cours.
- Tout nouvel élément devient une possibilité d'apporter une valeur ajoutée au lieu d'un obstacle à éviter, donnant aux équipes de développement une plus grande chance de succès.

01 - Appréhender la méthodologie Agile Scrum

Manifest Agile (valeurs et principes)



En **plus des 4 valeurs agiles fondamentales**, le Manifeste Agile décrit **12 principes de l'agilité**. Ils vont nous offrir des exemples concrets de la manière dont un produit Agile doit se construire.

1) Livrer de la valeur au client

- Le fonctionnement en itérations permet de délivrer plus vite le produit aux utilisateurs.

2) Intégrer les demandes de changement

- Pour le Manifeste Agile, le changement n'est pas négatif. Il est même positif. Il vaut mieux se tromper tôt et de réparer ses erreurs rapidement que de se rendre compte trop tard que le chemin emprunté n'était pas le bon.

3) Livrer fréquemment une version opérationnelle

- Découper votre produit en petits morceaux et vos développements en itérations courtes de deux à trois semaines.
- À la fin de chaque itération, livrez les nouvelles fonctionnalités développées et testez-les.

4) Assurer une coopération entre le client et l'équipe

- L'équipe et le client doivent s'entendre, échanger et travailler ensemble.

01 - Appréhender la méthodologie Agile Scrum

Manifest Agile (valeurs et principes)



5) Réaliser les projets avec des personnes motivées

- La confiance mutuelle est un point essentiel du Manifeste Agile. En d'autres termes, le management d'une équipe agile doit se baser sur la transparence. Le micro-management, ici, est à bannir.
- Contrôler étroitement une équipe de développeur ne va ni les engager, ni les motiver.

6) Privilégier le dialogue en face à face

- Il n'y a rien de plus efficace que d'avoir une équipe travaillant sur le même produit dans un même espace de travail.
- Les membres peuvent alors se poser directement les questions, obtenir les réponses dans la seconde.

7) Mesurer l'avancement sur la base d'un produit opérationnel

- L'objectif du produit n'est pas le processus, c'est sa valeur. Le processus est ce qui vous permet de l'atteindre.

8) Faire avancer le projet à un rythme soutenable et constant

- La raison de découper un produit en petites tâches est de garder vos équipes motivées.
- Si vous travaillez sur un projet pendant une longue période, vos équipes rencontreront une lassitude.
- Ne surchargez pas non plus l'équipe avec trop d'heures supplémentaires. Cela aura un impact sur la qualité de votre projet.

01 - Appréhender la méthodologie Agile Scrum

Manifest Agile (valeurs et principes)



9) Contrôler l'excellence technique et à la conception

- Que ce soit dans le code ou dans la méthodologie, la rigueur va favoriser la construction d'un produit de valeur.

10) Minimiser la quantité de travail inutile

- Si votre objectif est de produire rapidement un produit fonctionnel, il faut veiller à ne pas se noyer sous les tâches complexes et inutiles. Gardez vos documentations simples, ne vous embarrassez pas de réunions parasites.

11) Construire le projet avec des équipes auto-organisées

- Donner aux équipes l'autonomie d'agir de manière indépendante. Ils pourront alors prendre des décisions rapidement en cas d'imprévu.
- Une équipe responsabilisée est une équipe qui fera de son mieux pour atteindre son objectif.

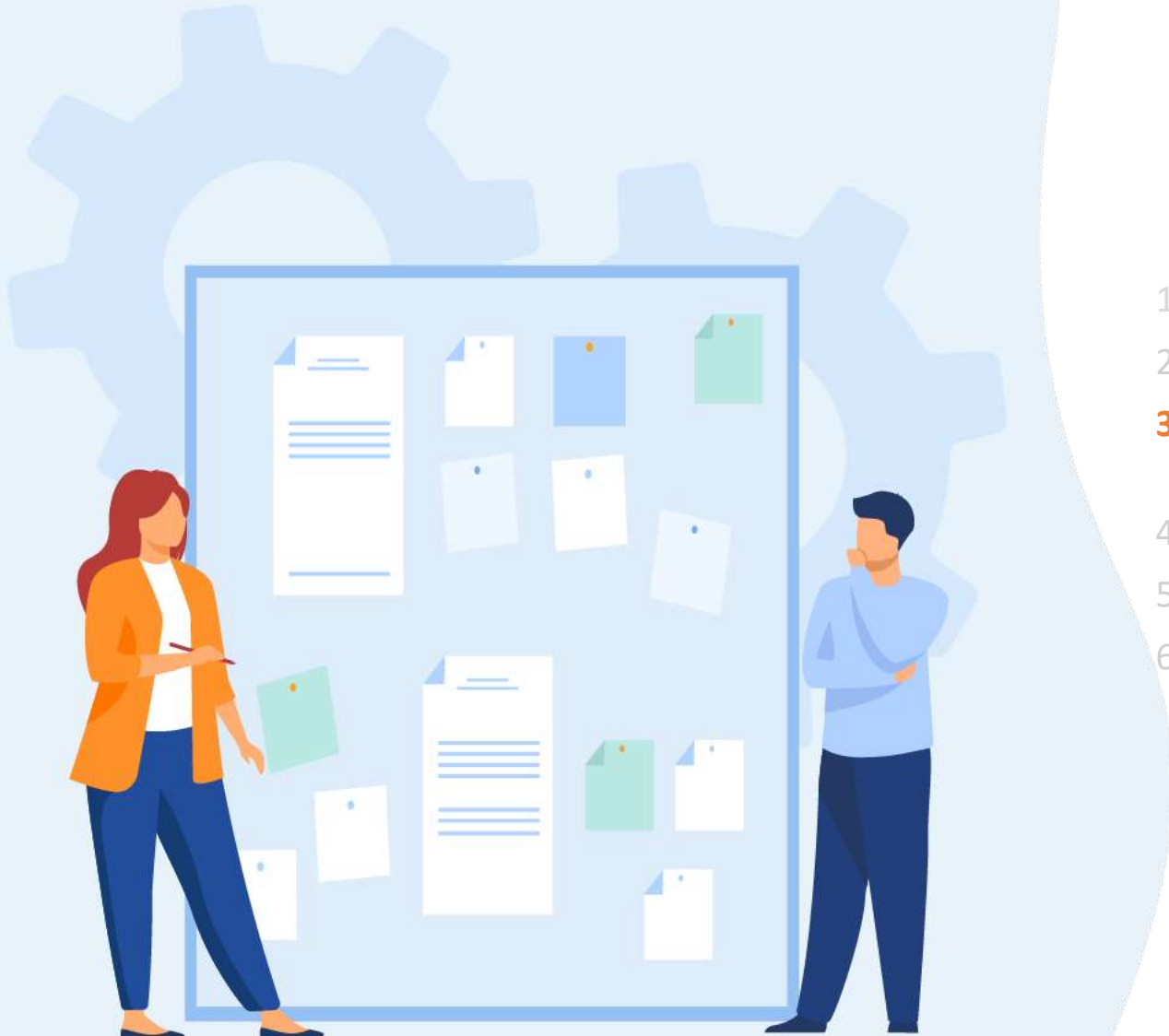
12) Améliorer constamment l'efficacité de l'équipe

- Le dernier des 12 principes du manifeste agile est l'amélioration continue de l'efficacité.
- Ce dont vous avez besoin, c'est d'un groupe en constante évolution, constamment engagé et à la recherche de moyens d'améliorer la productivité.

CHAPITRE 1

Appréhender la méthodologie Agile Scrum

1. Définition de la méthode Agile Scrum
2. Manifest Agile (valeurs et principes)
- 3. Processus de la méthode Scrum : pre-game, game, post-game**
4. Rôles et responsabilités
5. Evénements Scrum
6. Artéfacts Scrum



Processus de la méthode Scrum : pre-game, game, post-game

- Il existe 3 groupes de phases et de processus **Scrum**, à savoir **pre-game, game, post-game**
- Nous avons divisé cette activité en 3 phases :
 - 1) **Pre-Game(planification + architecture)**
 - 2) **Game(Sprint + réunion Scrum)**
 - 3) **Post-Game(démo + clôture)**

1) La première phase est la phase pre-game qui traite de la planification et de l'architecture du projet.

- La planification se fait avec la création du backlog et de la liste des activités qui doivent être effectuées pour décomposer les epics en user stories plus petites afin de constituer le backlog du produit. De plus, les user stories sont transformées en tâches dans la phase de backlog de sprint.
- Vient ensuite l'architecture ou la conception de haut niveau. Ici, la plupart du travail réel dans le backlog de sprint est décidé. Le travail accompli est de savoir comment travailler sur les tâches, apporter des modifications si nécessaire, affiner les tâches, effectuer des analyses et résoudre les problèmes qui apparaissent dans les processus.

Processus de la méthode Scrum : pre-game, game, post-game

2) La phase de game est la deuxième phase des phases de mêlée .

- C'est là que le travail proprement dit est accompli. Ici, chaque tâche affectée à la ressource est démarrée et terminée avec la définition de terminé. Il y a 4 étapes pour accomplir les tâches,
 - Développer le backlog de sprint, puis démarrer le développement, tester et documenter les changements, etc.
 - Fermer le statut de travail
 - Mener des réunions d'examen
 - Effectuez les modifications nécessaires à la définition du fait.
- Ici, dans cette phase au début, il y aura une réunion debout et chaque membre de l'équipe doit assister à cette réunion. Après la phase de planification, lorsque le développement commence, il peut y avoir des problèmes qui doivent être résolus, où alors le scrum master ou le propriétaire du produit arrêtera le sprint et s'adaptera aux changements, puis redémarrera le sprint à nouveau. Pour en savoir plus et explorer davantage.
- Les membres de l'équipe à la fin de chaque sprint doivent faire ce qui suit,
 - Actualiser un burndown chart
 - Participer à la réunion de revue de sprint
 - Enfin, participez à la réunion rétrospective

3) La phase post-game est la dernière des phases de mêlée .

- La phase de développement est terminée et les produits sont préparés pour la sortie. Toutes les étapes nécessaires telles que les tests, l'intégration, la formation, la documentation utilisateur et les supports marketing sont publiées.

CHAPITRE 1

Appréhender la méthodologie Agile Scrum

- 
1. Définition de la méthode Agile Scrum
 2. Manifest Agile (valeurs et principes)
 3. Processus de la méthode Scrum : pre-game, game, post-game
 - 4. Rôles et responsabilités**
 5. Evénements Scrum
 6. Artéfacts Scrum

01 - Appréhender la méthodologie Agile Scrum

Rôles et responsabilités

La Méthode **Scrum** : qui fait quoi ?

- **Le rôle du Scrum Master : le gourou**

- Le Scrum Master est le guide de l'avancement du projet, celui qui s'assure que les principes et les valeurs du Scrum sont respectés. C'est le coordinateur des équipes qui vérifie que la communication est au top. Il améliore aussi la productivité et il lève les obstacles.

- **Le rôle du Product Owner**

Lui c'est l'expert qui collabore avec le client. Souvent le fondateur ou le boss.

- Il définit à la suite des feedbacks clients les spécificités fonctionnelles du produit
- Il les priorise ensuite avec l'équipe.
- Il valide les fonctionnalités développées.
- Il endosse le rôle du client auprès de l'équipe.

- **Et l'équipe justement ?**

- Elle est constituée des développeurs. Dans la méthode **SCRUM**, il n'est pas censé y avoir de hiérarchie entre eux, quand bien même leur savoir-faire et compétence seraient différents. Idéalement une équipe contient 6 à 10 personnes pour être la plus efficace possible.

- **Stakeholders ?**

- Dans Scrum, une partie prenante (Stakeholder) est toute personne ayant un intérêt direct dans le produit qui ne fait pas partie de l'équipe Scrum. En tant que Product Owner, vous pouvez considérer les parties prenantes comme toute personne ayant un intérêt ou une influence sur le produit. Ce sont les personnes qui vous aideront à découvrir, développer, publier, soutenir et promouvoir le produit



CHAPITRE 1

Appréhender la méthodologie Agile Scrum

1. Définition de la méthode Agile Scrum
2. Manifest Agile (valeurs et principes)
3. Processus de la méthode Scrum : pre-game, game, post-game
4. Rôles et responsabilités
- 5. Événements Scrum**
6. Artéfacts Scrum



01 - Appréhender la méthodologie Agile Scrum

Evénements Scrum



Les grandes étapes de la méthode agile (ou méthode **SCRUM**)

- **Méthode SCRUM - Étape 1 : Le Product Backlog**

- Dans cette phase, le Product Owner rencontre le client et analyse son besoin. Il identifie toutes les fonctionnalités dont le produit devra être composé (les user stories) dans ce qui s'appelle le Product Backlog.
- Ce “ cahier des charges “ n'est pas fixé pour toujours, et pourra évoluer en fonction des besoins du client et l'avancement du projet. L'équipe décide de ce qu'elle peut faire et dans quel ordre le faire.

- **Méthode SCRUM - Étape 2 : Le sprint**

- La méthode SCRUM se caractérise par une répartition de chacune des tâches à faire. L'équipe trie les fonctionnalités et tâches qu'elle répartit dans des Sprint (durée de cycle de deux semaines).
- Et pendant ce cycle, l'équipe s'occupera par exemple uniquement de coder une fonctionnalité du produit qu'elle devra livrer à la fin de cette phase.

- **Le sprint planning meeting** : on organise avant chaque sprint une réunion de planification.

- C'est une sorte de négociation entre le product owner et l'équipe technique : le sprint planning meeting. Cette réunion permet de sélectionner dans le product backlog les exigences les plus prioritaires pour le client.
- Le product owner propose des user stories. L'équipe technique analyse les user stories, les traduit en tâches techniques, et regarde en combien de temps chacune des user stories pourra être réalisée. Après négociation et décision concernant les choses à faire, les tâches constituent le sprint backlog.

Evénements Scrum

- **Le Daily SCRUM : la mêlée quotidienne**

- Dans l'optique de respecter la méthode SCRUM, chaque matin, quand l'équipe est au complet on réalise le daily scrum : une réunion de 5–10 minutes, qui se fait debout (pour aller plus vite !) où l'on parle de trois choses :
 - Ce qu'on a fait hier ?
 - Quels problèmes on a rencontrés ?
 - Que va-t-on faire aujourd'hui ?
- Cette réunion permet de motiver et stimuler les équipes et sans cesse se poser la question du sens de ses actions ...

- **Et concrètement : comment on s'organise dans le sprint ?**

- **Pour les ingrédients, vous pouvez vous équiper avant le sprint** : d'un tableau velleda aimanté et de post-it de couleurs.
- **Le tableau est divisé en trois colonnes** : à faire, en cours, réalisé.
- En début de sprint, on recense chacune des petites tâches qui seront nécessaires pour réaliser la fonctionnalité en question du produit. On représente chaque tâche par un post-it de couleur. Elles se trouvent dans la première colonne : à faire.
- Au fur et à mesure qu'elles seront réalisées, elles seront bougées vers la droite. L'équipe a établi combien de temps prendrait chaque tâche. Il est ainsi facile de voir pendant un sprint si l'équipe tiendra ses timings et ces objectifs. Cela force à tendre vers l'efficacité plus que la perfection.

01 - Appréhender la méthodologie Agile Scrum

Evénements Scrum



- **Méthode SCRUM - Étape 3 : Sprint Review**
 - Tous les vendredis qui closent un sprint on teste les bénéfices de la fonctionnalité avec le Product Owner.
 - On fait une démo de ce qui a été créé. Puis le destinataire de la fonctionnalité (le client par exemple) confirme ou non si la fonctionnalité marche comme il le souhaitait. La boucle est bouclée !
- **Méthode SCRUM - Étape 4 : Sprint Retrospective (Rétrospective de Sprint)**
 - Venant immédiatement après la revue de Sprint, il s'agit d'un bilan dont l'objectif est l'amélioration continue des pratiques. L'équipe échange sur les réussites, les difficultés, relève ce qui a fonctionné ou non. Avec toujours des leçons à tirer pour les prochains Sprints.

CHAPITRE 1

Appréhender la méthodologie Agile Scrum

1. Définition de la méthode Agile Scrum
2. Manifest Agile (valeurs et principes)
3. Processus de la méthode Scrum : pre-game, game, post-game
4. Rôles et responsabilités
5. Evénements Scrum
6. **Artéfacts Scrum**



Comprenant des entrants et sortants du processus, appelés "artéfacts"

- Le mot **artefact** désigne un produit ayant subi une transformation, même minime, par l'homme.
- Les **artefacts SCRUM** sont au nombre de 3 :
 - **Product Backlog** :
 - liste des fonctionnalités du produit.
 - Au démarrage du développement d'un produit agile, le MVP va être découpé en petites fonctionnalités ou tâches à réaliser pour faciliter sa construction.
 - Le Product Backlog est une sorte de réservoir regroupant l'ensemble des fonctionnalités du produit. Les tâches doivent y être ordonnées avec discernement en fonction de la priorité dans laquelle elles doivent être réalisées.
 - **Sprint Backlog** :
 - Planification des éléments du Product Backlog à mettre en oeuvre lors du Sprint pour livrer l'incrément de produit doté des fonctionnalités requises pour cette étape.
 - Le cadre méthodologique SCRUM divise le calendrier d'une équipe en cycles qui se répètent, nommés Sprint.
 - Le Sprint Backlog est une vue en temps-réel, très visible du travail que l'Équipe planifie d'accomplir durant le Sprint et il appartient uniquement à l'Équipe de Développement.
 - **L'incrément de produit** :
 - Durant chaque Sprint, l'équipe de développement réalise un incrément de produit.
 - Un Sprint démarre lorsque le précédent est terminé. Il s'agit d'un processus incrémental.

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira

2. Installation et configuration d'un compte Jira
3. Création de projet avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Présentation de Jira



C’est quoi Jira?

- Jira Software est une solution de gestion de projet éditée par l’entreprise Atlassian¹. Elle permet aux équipes de **s’organiser efficacement, d’établir une communication durable et de visualiser le projet en un coup d’œil** grâce à ses tableaux de bord personnalisés. Cet outil est reconnu comme la solution la plus utilisée par les équipes de développement logiciel. Jira Software permet également de :
 - **Travailler en méthode agile** grâce aux tableaux Kanban et **Scrum**.
 - Accélérer la livraison des projets
 - Améliorer en continu des projets
 - Faciliter le travail des équipes



Image: Le logo de Jira (source :atlassian.com)

Quelles équipes utilisent Jira Software ?

- Jira s’adapte aux différentes équipes d’une entreprise.
 - **Jira Work Management** pour les équipes métier (Marketing/RH/Finance...),
 - **Jira Service Management** pour les équipes de support (DevOps),
 - **Jira Software** pour les équipes de développement de logiciel (**Équipe Agile**) ou pour de la gestion de projet complexe.

Pourquoi utiliser Jira Software quand on est une équipe de développement ?

- Pour pouvoir l’intégrer avec un dépôt **Git (Bitbucket, Github, Gitlab...)**
 - Pour avoir un suivi visuel des tâches : backlog, tableau de bord, tableau agile,
 - Pour gérer l’avancé du projet à l’aide d’un reporting,
 - Pour travailler à distance grâce à l’application mobile Jira Software Cloud.

¹.Atlassian est un éditeur de logiciels, basé en Australie, qui développe des produits pour la gestion de développement et de projets. Ses logiciels les plus connus sont Jira, Confluence, Bamboo, Bitbucket et Trello.

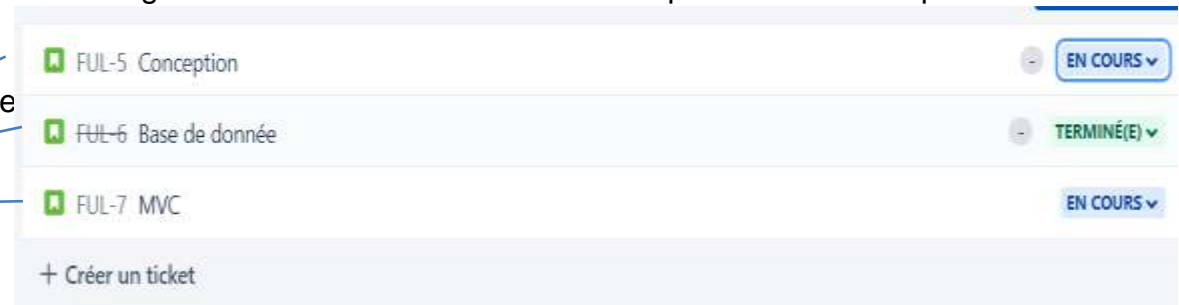
02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Présentation de Jira : Définition de ticket et projet

Qu’est-ce qu’un ticket dans Jira ?

- Un ticket est une tâche à effectuer, c’est un élément de travail.
- Un **ticket suit plusieurs étapes** : à faire, puis **en cours**, puis **terminé**, et on peut ajouter d’autres selon le type du projet (**exemple** : recette , en attente..).
- On peut dire que le ticket passe dans chaque étape du workflow (flux de travail).
- Les tickets dans Jira peuvent être de plusieurs types. Dans un projet de développement logiciel, on retrouve les types de ticket suivants :
 - **Une epic (épopée)** : est considérée comme un grand objectif ou une grosse fonctionnalité devant être simplifiée et divisé en plusieurs tâches afin d’organiser le travail des équipes agiles,
 - **Une story** : représente une fonctionnalité à réaliser,
 - **Une tâche** : est généralement une tâche technique à effectuer
 - **Un bug** : désigne un problème à corriger.

Des tickets



- Une epic peut contenir : soit epic ou tâche ou story
- Une tâche peut contenir des sous-tâches(ticket)

Exemple de tickets dans Jira avec l’étape courante de chaque ticket

Qu’est-ce qu’un projet dans Jira ?

- Un projet Jira **regroupe des tickets afin de réaliser un objectif**.
- Dans Jira , vous avez différentes façons de construire un projet : vous pouvez en créer un **par équipe** ou **par produit**. Cela dépend de votre vision du projet et de l’organisation de vos équipes.
- Chaque **projet Jira possède une clé unique**. Par exemple, pour un projet web, la clé pourra être **WEB**. Cela aura un impact sur les identifiants de chaque ticket : le premier ticket créé sera **WEB-1**, le deuxième ticket **WEB-2**, etc.
- L’organisation des projets est importante car **les droits d’accès sont gérés par projet** : cela permet de garantir la confidentialité des informations. Votre collègue des ressources humaines n’aura pas accès aux projets de développement logiciel ; et inversement, les développeurs ne verront pas les tickets de RH ou Marketing par exemple.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Présentation de Jira : Définition d'un Backlog



C'est quoi un Backlog ?

- Un **Backlog** est une liste de fonctionnalités ou d'éléments de travail. Ça peut être des tâches techniques ou des exigences non fonctionnelles.
- En Jira, Un **backlog** contient la liste des épics ou tâches avec leurs tickets et leurs prépositions (durées ou points) et la personne assignée de faire le ticket.
- Un ensemble de tickets appartient à un tableau sprint.

The screenshot shows a Jira Backlog interface. At the top, there's a search bar and filters for 'Mg', 'S', and 'Epic'. Below that, a sidebar on the left shows 'Epic' management with options for 'Tickets sans epic', 'Projet Full-Stack', and 'Full_Stack:Laravel_React', along with a '+ Créer Epic' button. The main area displays 'Tableau Sprint 1' for the period '1 oct. – 29 oct.' with '(9 tickets)'. A progress bar shows '53%' completion. The list of tickets includes:

Item	Project	Points	Status	Assignee
FS-5 Conception	PROJET FULL-STACK	3	À FAIRE	S
FS-2 back-end	PROJET FULL-STACK	9	À FAIRE	Mg
FS-3 Front-end	PROJET FULL-STACK	8	À FAIRE	Mg
FS-4 Base de donnée	PROJET FULL-STACK	3	À FAIRE	S
FS-6 Tests	PROJET FULL-STACK	2	À FAIRE	
FS-9 Laravel	FULL_STACK:LARAVEL_REACT	10	À FAIRE	Mg
FS-10 React	FULL_STACK:LARAVEL_REACT	10	À FAIRE	S
FS-11 Base donnée :Mongo DB	FULL_STACK:LARAVEL_REACT	5	À FAIRE	
FS-12 Conception :UML	FULL_STACK:LARAVEL_REACT	3	À FAIRE	S

Exemple d'un Backlog

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. **Installation et configuration d'un compte Jira**
3. Création de projet avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Installation et création de compte Jira

Installation de Jira

- Il existe deux possibilités pour utiliser Jira gratuitement :
 - Télécharger la version gratuite du logiciel sur : <https://www.atlassian.com/fr/software/Jira/download-journey>
 - Utiliser la version Jira cloud free online : via <https://www.atlassian.com/fr/software/Jira/free>
- Puisque les deux versions nécessitent une connexion internet en veut suggérer d’utiliser **la version Jira Cloud**.

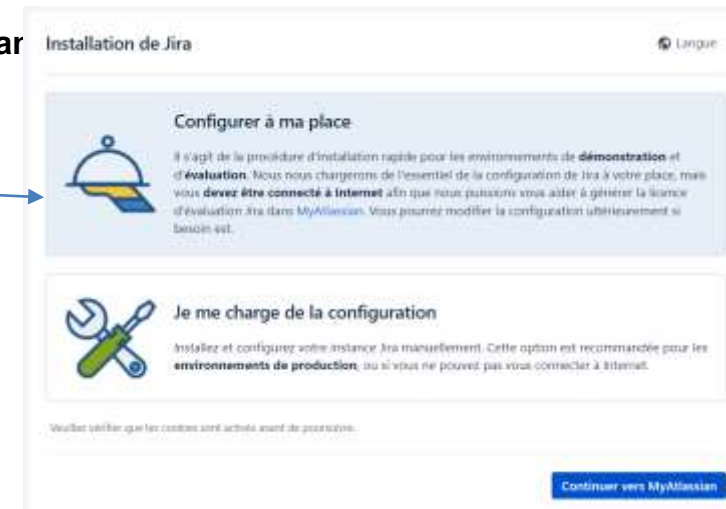
Jira Desktop

Lorsque vous installer la version desktop de Jira vous aurez l’accès au logiciel via le navigateur sur le lien :

<http://localhost:8080/secure/SetupMode!default.jspa>

Et vous suivez les mêmes étapes pour créer un projet Jira dans le cloud (**page suivre**)

Sélectionner cette option et cliquer sur continuer vers MyAtlassian



l'interface de démarrage de Jira desktop

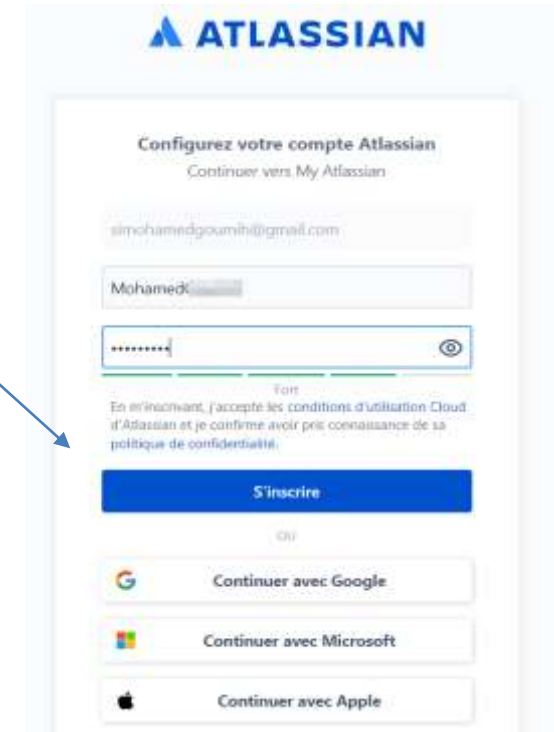
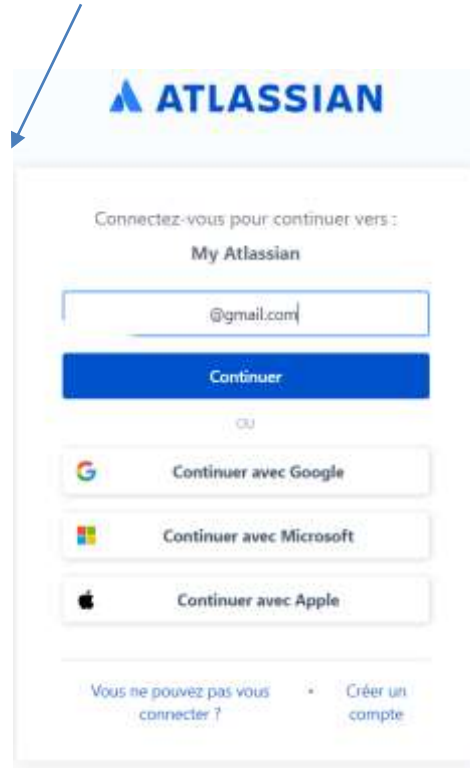
02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Installation et création de compte Jira



Etapes pour la création compte Jira sur Jira cloud

- On se connecte sur <https://www.atlassian.com/fr/software/Jira/free>
- On se connecte avec un compte mail ou on crée un nouveau compte.



- **Après confirmation du compte** : on se connecte pour créer un projet agile Scrum.

CHAPITRE

2 Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. **Création d'un projet scrum avec Jira**
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile :

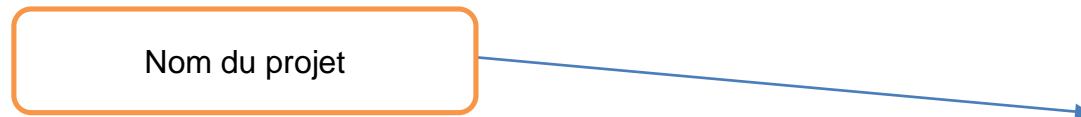


02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création projet Scrum avec Jira

Etapas pour la création d’un projet Scrum Jira :

- Pour créer un projet Scrum Jira ,on saisit l’ Email professionnel et le nom du projet



E-mail professionnel

Se connecter avec un autre compte Atlassian

Votre site ?

develstack .atlassian.net ✓

En cliquant ci-dessous, vous acceptez les Conditions de service d'Atlassian Cloud et la Politique de confidentialité.

J'accepte

AUCUNE CARTE DE CRÉDIT NÉCESSAIRE

ATLASSIAN

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création projet Scrum avec Jira

Etapes pour la création d’un projet Jira scrum :

- Puis on choisit le modèle du projet (Scrum) et la clé du projet.

Ajouter les informations du projet

Vous pouvez modifier ces informations à tout moment dans les paramètres de votre projet.

Nom

Clé ⓘ

Clé du projet

Module du projet

The screenshot shows the 'Ajouter les informations du projet' screen in Jira. It features two main sections: 'Modèle' and 'Type'. The 'Modèle' section has a 'Changer de modèle' link and two options: 'Scrum RECOMMANDÉ' (highlighted with a blue arrow from the 'Module du projet' label) and 'Géré par l'équipe RECOMMANDÉ'. The 'Type' section has a 'Changer de type' link and one option: 'Géré par l'équipe RECOMMANDÉ'. At the bottom, there are 'Retour' and 'Créer un projet' buttons.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création projet Scrum avec Jira

Etapes pour la création d’un projet Jira scrum :

- Par la suite, Jira nous donne la possibilité d’inviter les membres de l’équipe de travail, par l’ajout de leurs emails, nous pouvons ajouter d’autres membres après la création du projet:



Invitez les membres de votre équipe

Intégrez toute votre équipe !

Ajouter une adresse e-mail

mohamec...@gmail.com

Ajouter une adresse e-mail

...@ofppt.ma

Ajouter une adresse e-mail

...@hotmail.com

- Donner aux membres de votre équipe la possibilité d'inviter d'autres personnes sur votre site

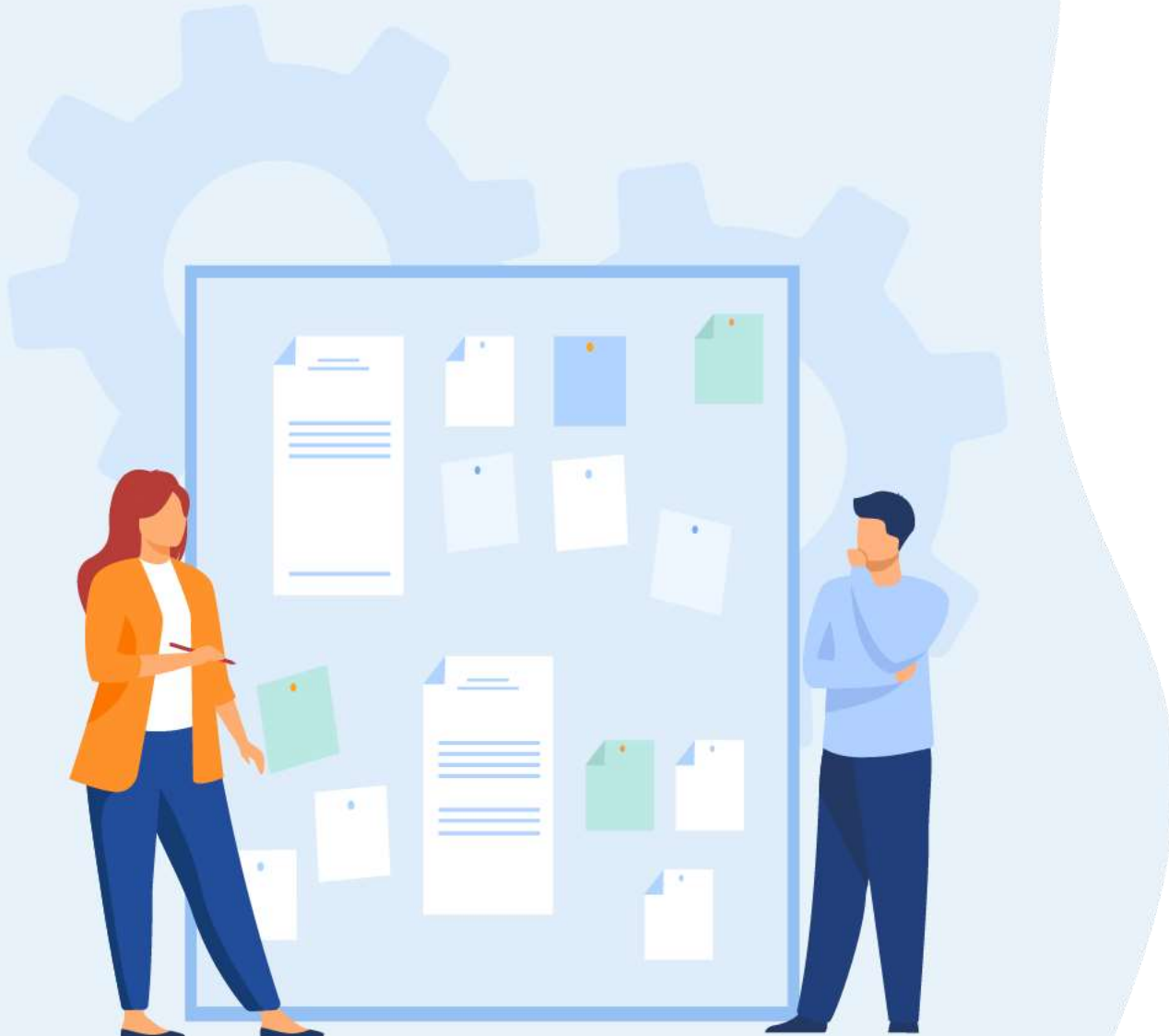
Vous pouvez modifier ces paramètres à tout moment.

Continuer

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
- 4. Création d'un backlog product**
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création projet Scrum avec Jira

La création du Backlog

- Après la création du projet, un backlog vide est créé automatiquement, il suffit de cliquer dans le menu latéral pour y’accéder:

Page du backlog

The screenshot shows the Jira interface for a project named 'Full_stack'. On the left is a sidebar menu with categories: 'PLANIFICATION' (containing 'Feuille de route', 'Backlog', and 'Tableau'), and 'DÉVELOPPEMENT' (containing 'Code', 'Pages de projet', 'Ajouter un raccourci', and 'Paramètres du projet'). The 'Backlog' item is highlighted. The main content area is titled 'Projets / Full_stack Backlog'. It features a search bar, a 'MG' icon, a user icon, and an 'Epic' dropdown. Below this is a section for 'Tableau Sprint 1' with 'Ajouter des dates (0 ticket)' and 'Démarrer un sprint' buttons. A central instruction box says 'Planifiez votre sprint' and 'Faites glisser des tickets depuis la section Backlog ou créez des tickets pour planifier ce sprint. Lorsque tout est prêt, sélectionnez Démarrer le sprint.' Below this is a '+ Créer un ticket' button. At the bottom, there is a section for 'Backlog (0 ticket)' with 'Créer un sprint' and '+ Créer un ticket' buttons. The text 'Votre backlog est vide.' is displayed in the backlog area.

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
4. Création d'un backlog product
- 5. Ajout des différents types de tickets**
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets



Création des tickets

- Dans notre **Backlog**, On commence à créer les types des tickets en cliquant sur **créer** au niveau de la barre de navigation:
- Puis, on choisit un type du ticket : **Epic**, **tâche**, **Story** ou **Bug**, dans cet exemple on a choisi **Epic**

The screenshot shows the 'Create issue' form in Jira. At the top, there is a navigation bar with 'Filtres', 'Tableaux de bord', 'Données', and 'Ajouter' buttons, and a 'Créer' button. The main form has a title 'Create issue' and an 'Import issues' button. The 'Projet' dropdown is set to 'Full_stack (FS)'. The 'Type de ticket' dropdown is open, showing 'Epic' selected, with 'Story', 'Tâche', and 'Bug' as other options. The 'Description' field is empty with a rich text editor toolbar. At the bottom, there is a checkbox for 'Créer un autre ticket', an 'Annuler' button, and a 'Créer' button.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Création des tickets

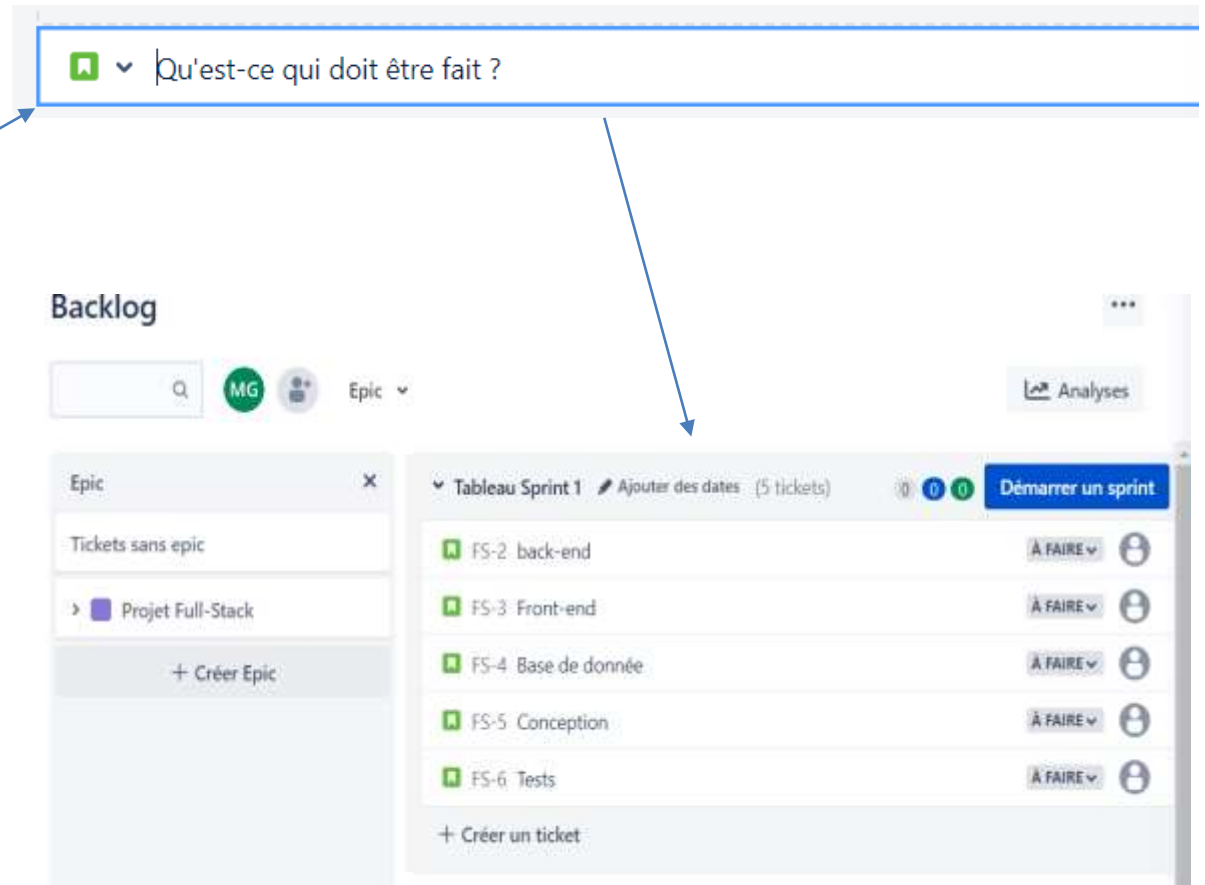
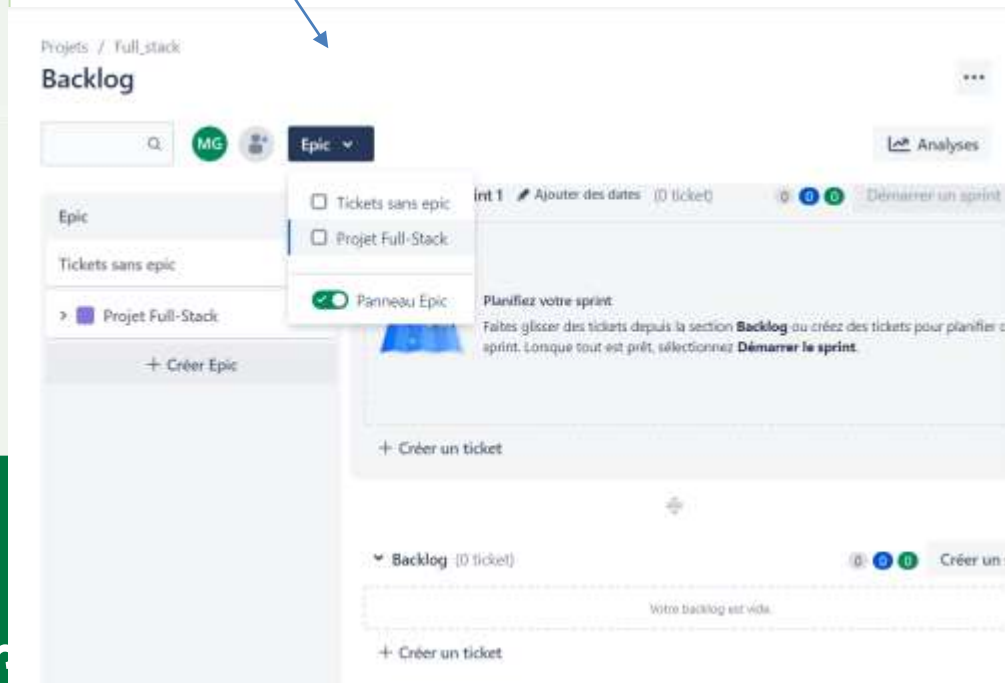
- On donne à notre ticket un nom
- on peut ajouter une description du ticket
- En cliquant sur **créer** le premier ticket sera crée avec le nom : **FS-1**

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Création des tickets

- Par la suite, on clique sur Epic pour activer le panneau des epics afin d’afficher le projet et ses tickets.
- On commence à remplir le tableau des sprints avec nos tickets créés :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Remplir les epics

- Ensuite ,On glisse les tickets avec epic à droite

The screenshot shows the Jira interface for a project named 'Full_stack'. The 'Backlog' view is active. On the left, a sidebar shows a list of tickets, with 'FS-5 Conception' selected and highlighted in blue. A blue arrow points from this ticket to the 'Tableau Sprint 1' (Sprint 1) on the right. The sprint board shows a list of tickets: 'FS-2 back-end', 'FS-3', 'FS-4 base de donnée', and 'FS-6 Tests'. The 'FS-6 Tests' ticket is highlighted in blue, indicating it is being moved to the sprint. The interface includes a search bar, a user profile 'MG', and a dropdown menu for 'Epic'. A 'Démarrer un sprint' button is visible at the top right of the sprint board.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Remplir les epics

- On peut ajouter une autre epic en cliquant sur **Créer epic**.

The screenshot shows the Jira Backlog interface. On the left, an 'Epic' dialog box is open, displaying a tree view of epics: 'Tickets sans epic', 'Projet Full-Stack', and 'Full_Stack:Laravel_React'. The 'Full_Stack:Laravel_React' epic is expanded, showing 'Date de début: Aucune' and 'Date d'échéance: Aucune'. At the bottom of the dialog is a '+ Créer Epic' button, which is highlighted by a blue arrow from the text above. The main backlog area shows a 'Tableau Sprint 1' with 9 tickets. Each ticket has a status (e.g., 'À FAIRE'), a priority, and an assignee icon.

Item	Project	Status	Assignee
FS-2 back-end	PROJET FULL-STACK	À FAIRE	[User Icon]
FS-3 Front-end	PROJET FULL-STACK	À FAIRE	[User Icon]
FS-4 Base de donnée	PROJET FULL-STACK	À FAIRE	[User Icon]
FS-5 Conception	PROJET FULL-STACK	À FAIRE	[User Icon]
FS-6 Tests	PROJET FULL-STACK	À FAIRE	[User Icon]
FS-9 Laravel	FULL_STACK:LARAVEL_REACT	À FAIRE	[User Icon]
FS-10 React	FULL_STACK:LARAVEL_REACT	À FAIRE	[User Icon]
FS-11 Base donnée :Mongo DB	FULL_STACK:LARAVEL_REACT	À FAIRE	[User Icon]
FS-12 Conception :UML	FULL_STACK:LARAVEL_REACT	À FAIRE	[User Icon]

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Durée travail des tickets : estimation

- Par la suite ,on estime la durée de chaque ticket, en donnant une valeur (**story point**).
- Les **story points** mesurent la complexité d'un ticket par rapport aux autres.

Projets / Full_stack

Backlog

MG + Epic 2 ▼ Effacer les filtres 📊 Analyses

Epic ×

Tickets sans epic

- > ■ Projet Full-Stack
- ▼ ■ Full_Stack:Laravel_React

Date de début
Aucune

Date d'échéance
Aucune

Afficher tous les détails

+ Créer Epic

Tableau Sprint 1 ✎ Ajouter des dates (9 tickets) 44 0 0 🚀 Démarrer un sprint ⋮

FS-2	back-end	PROJET FULL-STACK	9	À FAIRE	👤	⋮
FS-3	Front-end	PROJET FULL-STACK	✓	À FAIRE	👤	✕
FS-4	Base de donnée	PROJET FULL-STACK	3	À FAIRE	👤	
FS-5	Conception	PROJET FULL-STACK	3	À FAIRE	👤	
FS-6	Tests	PROJET FULL-STACK	2	À FAIRE	👤	
FS-9	Laravel	FULL_STACK:LARAVEL_REACT	10	À FAIRE	👤	
FS-10	React	FULL_STACK:LARAVEL_REACT	10	À FAIRE	👤	
FS-11	Base donnée :Mongo DB	FULL_STACK:LARAVEL_REACT	5	À FAIRE	👤	
FS-12	Conception :UML	FULL_STACK:LARAVEL_REACT	3	À FAIRE	👤	

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

L’ordre de priorité des tickets

- On peut changer l’ordre des tickets selon l’importance en utilisant Ctrl + le déplacement de la souris.

Projets / Full_stack

Backlog

MG + Epic 2 Effacer les filtres Analyses

Epic

Tickets sans epic

- Projets Full-Stack
- Full_Stack:Laravel_React
 - Date de début: Aucune
 - Date d'échéance: Aucune
 - Afficher tous les détails

+ Créer Epic

Tableau Sprint 1 Ajouter des dates (9 tickets) 53 0 0 Démarrer un sprint

ID	Titre	Projet	Points	Statut	Assigné
FS-5	Conception	PROJET FULL-STACK	3	À FAIRE	
FS-12	Conception :UML	FULL_STACK:LARAVEL_REACT	3	À FAIRE	
FS-2	back-end	PROJET FULL-STACK	9	À FAIRE	
FS-3	Front-end	PROJET FULL-STACK	8	À FAIRE	
FS-4	Base de donnée	PROJET FULL-STACK	3	À FAIRE	
FS-6	Tests	PROJET FULL-STACK	2	À FAIRE	
FS-9	Laravel	FULL_STACK:LARAVEL_REACT	10	À FAIRE	
FS-10	React	FULL_STACK:LARAVEL_REACT	10	À FAIRE	
FS-11	Base donnée :Mongo DB	FULL_STACK:LARAVEL_REACT	5	À FAIRE	

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Ajouts des différents types de tickets

Assignation des tickets

- On peut assigner les tickets aux membres de l’équipe en cliquant sur l’icone du bonhomme.

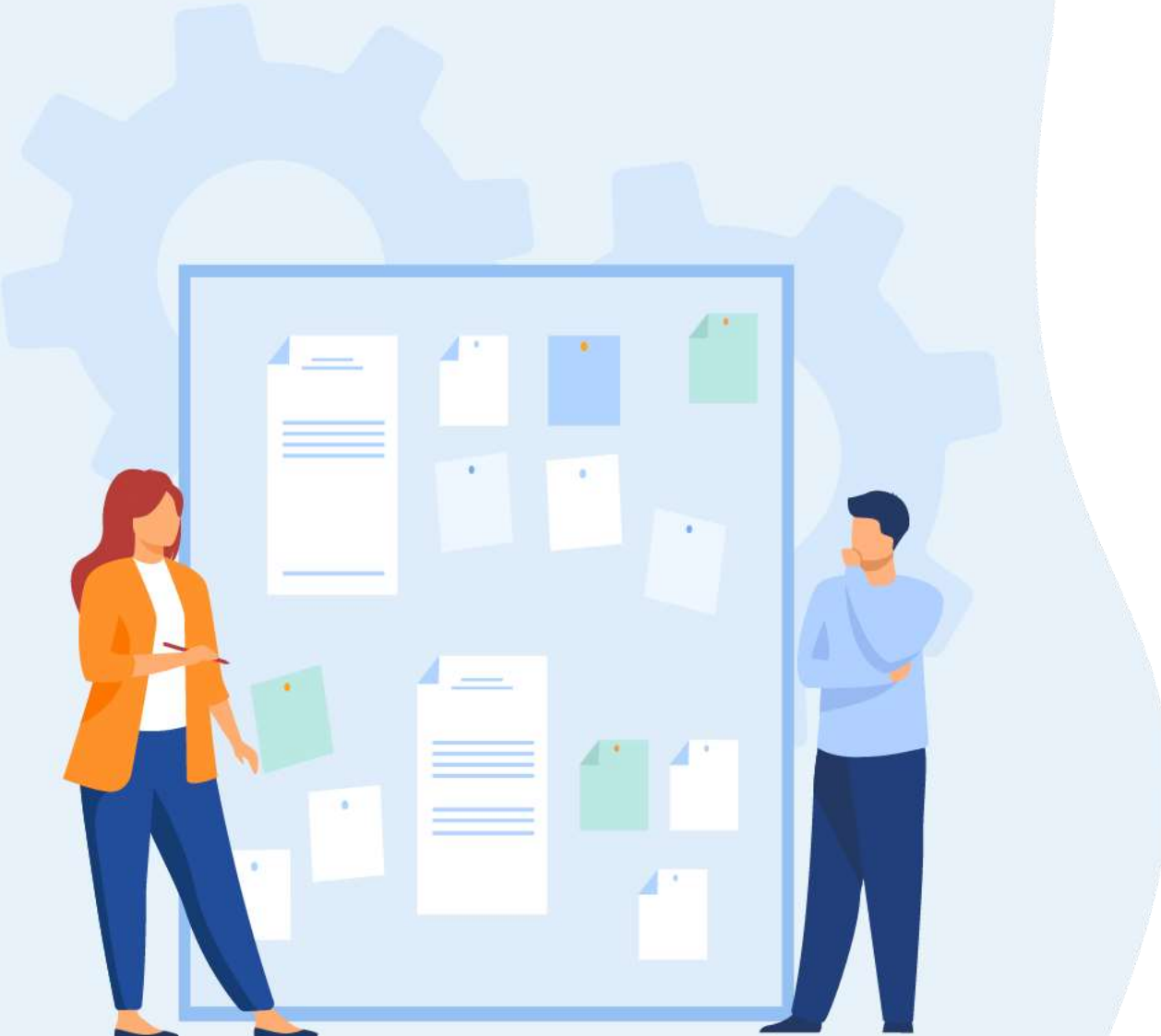
The screenshot displays a Jira Backlog for a project named 'Full_stack'. The main area shows a 'Tableau Sprint 1' from October 1st to 29th, containing 9 tickets. A dropdown menu is open for the 'FS-5 Conception' ticket, showing assignment options: 'Non assigné', 'Automatique', 'mohamed' (with a note 'h (Me l'assigner) gmail.com'), and 'simohamed'. A blue arrow points from the text in the list above to the 'Non assigné' option in the dropdown.

Item	Project	Status	Assignee
FS-5 Conception	PROJET FULL-STACK	À FAIRE	Non assigné
FS-2 back-end	PROJET FULL-STACK	À FAIRE	Non assigné
FS-3 Front-end	PROJET FULL-STACK	À FAIRE	Non assigné
FS-4 Base de donnée	PROJET FULL-STACK	À FAIRE	Non assigné
FS-6 Tests	PROJET FULL-STACK	À FAIRE	Non assigné
FS-9 Laravel	FULL_STACK-LARAVEL_REACT	À FAIRE	Non assigné
FS-10 React	FULL_STACK-LARAVEL_REACT	À FAIRE	Non assigné
FS-11 Base donnée :Mongo DB	FULL_STACK-LARAVEL_REACT	À FAIRE	Non assigné
FS-12 Conception :UML	FULL_STACK-LARAVEL_REACT	À FAIRE	Non assigné

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
- 6. Création des tâches**
7. Planification d'un sprint
8. Manipulation du tableau de bord de sprint
9. Utilisation de la feuille de route (roadmap)
10. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création des tâches

Création des tâches :

- On suit les mêmes étapes pour créer un ticket où nom **d'une tâche** :

Tableaux de bord ▾ Personnes ▾ Appli ▾ Créer

Create issue Import issues ⋮

Projet *
backend (BAQ) ▾

Type de ticket *
☑ Tâche ▾
Story
Bug
Epic

Description
Texte normal ▾ B I ... A ▾ ☰ ☷ 🔗 📎 @ 🌐 ⏏ + ▾
We support markdown! Try **bold**, `inline code`, or `` `` for code blocks.

Créer un autre ticket Annuler Créer

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Création des tâches

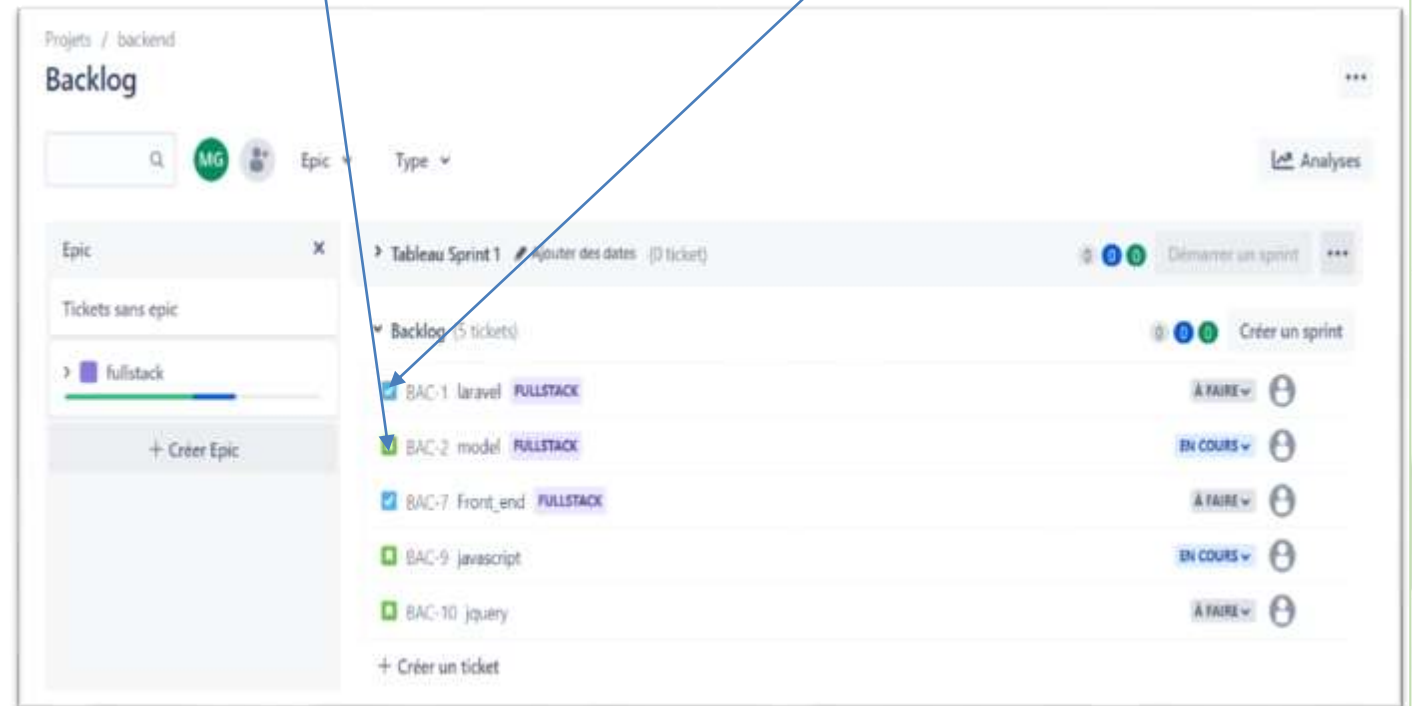
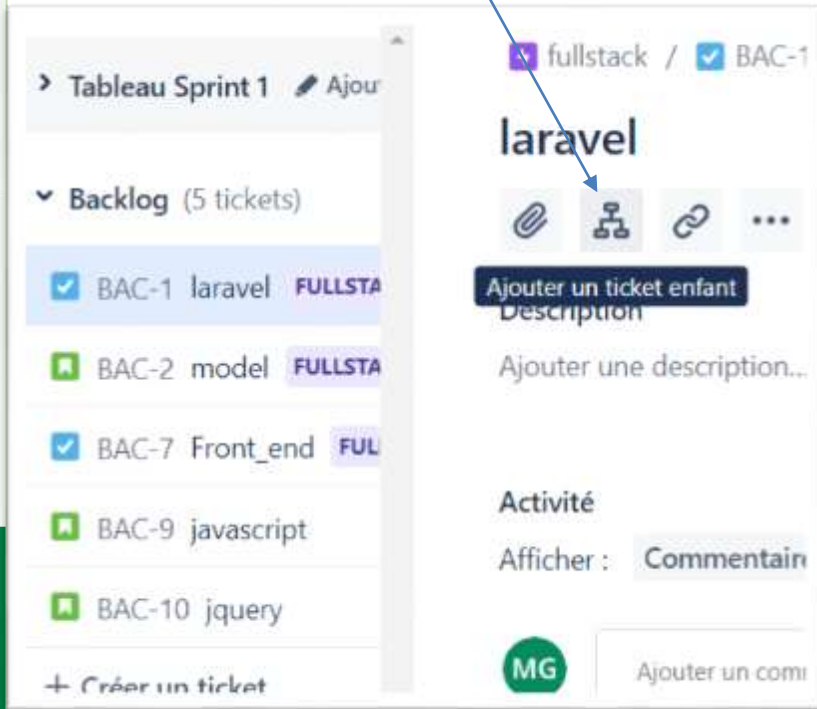
Création des tâches :

- Une tâche peut contenir des sous-tâches dans ce cas on crée **un ticket-enfant** en cliquant sur **ajouter un ticket-enfant**

- Un Backlog contenant des tâches et des sous-tâches

Une sous-tâche

Une tâche

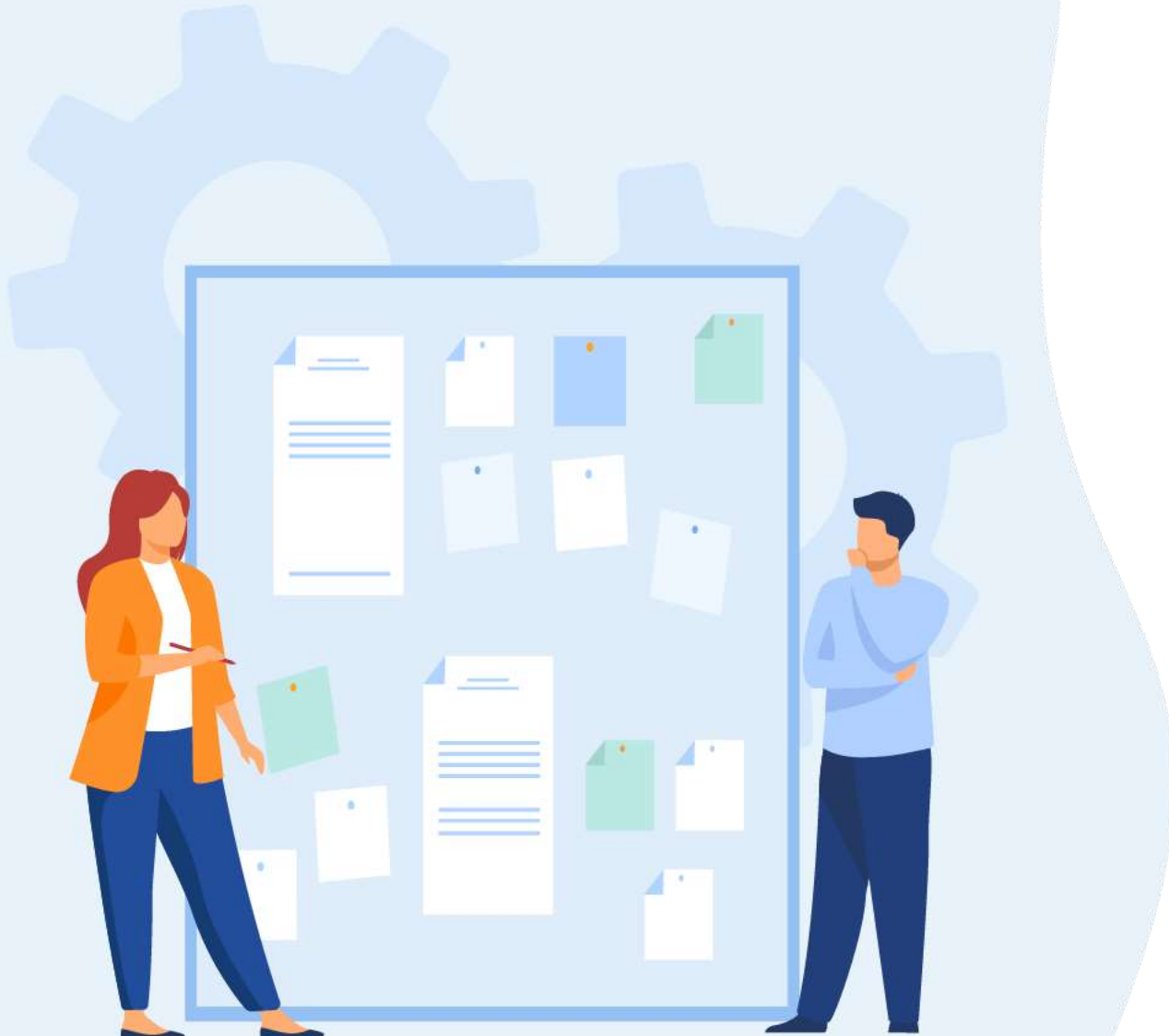


CHAPITRE

2 Manipuler l'outil de gestion de projet

Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Création des tâches
- 7. Planification d'un sprint**
8. Manipulation du tableau de bord de sprint
9. Utilisation de la feuille de route (roadmap)
10. Génération des rapports Agile :



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Planification du sprint

Planification du sprint

- Puis on démarre le sprint, en cliquant sur **Démarrer le Sprint** :

Projet / Full stack
Backlog

Rechercher [] MG Epic 3 Effacer les filtres [] Analyses

Epic : X

Tickets sans epic

- Project Full-Stack
- Full Stack:Laravel React

Date de début: Aucune

Date d'échéance: Aucune

Afficher tous les détails

+ Créer Epic

Tableau Sprint 1 Ajouter des dates (3 tickets)

Statut	Titre	Projet	Statut
À faire	FS-5 Conception	PROJET FULL-STACK	À faire
À faire	FS-2 back-end	PROJET FULL-STACK	À faire
À faire	FS-3 Front-end	PROJET FULL-STACK	À faire
À faire	FS-4 Base de donnée	PROJET FULL-STACK	À faire
À faire	FS-6 Tests	PROJET FULL-STACK	À faire
À faire	FS-9 Laravel	FULLSTACKLARAVEL, REACT	À faire
À faire	FS-10 React	FULLSTACKLARAVEL, REACT	À faire
À faire	FS-11 Base donnée Mongo DB	FULLSTACKLARAVEL, REACT	À faire
À faire	FS-12 Conception JML	FULLSTACKLARAVEL, REACT	À faire

- On donne à notre sprint un nom ,durée ,date de début ,date de fin et description.
- Puis on clique sur **Démarrer** pour voir le tableau du sprint

Démarrer le sprint

9 tickets seront inclus dans ce sprint.

Nom du sprint *

Durée *

Date de début *

octobre 2022

DIM	LUN	MAR	MER	JEU	VEN	SAM
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

CHAPITRE E 2 Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
- 7. Manipulation du tableau de bord de sprint**
8. Utilisation de la feuille de route (roadmap)
9. Génération des rapports Agile



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Tableau du bord du sprint

Tableau des sprints

- Le tableau des sprints s’affiche avec les états des tickets et on choisi l’**epic** à réaliser ou tous les epics.

Projets / Full_stack

Tableau Sprint 1

Recherche: [] MG [] Epic 1 [v] Effacer les filtres

A FAIRE 4 SUR 9 TICKETS -

- Laravel
FULL_STACK:LARAVEL_REACT
FS-9 (10)
- React
FULL_STACK:LARAVEL_REACT
FS-10 (10)
- Base donnée :Mongo DB
FULL_STACK:LARAVEL_REACT
FS-11 (5)
- Conception :UML
FULL_STACK:LARAVEL_REACT
FS-12 (3)

FINI ✓

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Tableau du bord de sprint

Tableau des sprints

- Dans Le tableau des sprints, on voit :
 - Le états des tickets à faire ,en cours et fini
 - On peut ajouter une colonne d’état en cliquant sur **créer une colonne** (ici on a ajouté une colonne **Recette**)

The image displays two side-by-side screenshots of a Jira Sprint Board for a project named 'Full_stack'.

Left Screenshot: The board is titled 'Tableau Sprint 1'. It has three columns: 'À FAIRE 4 SUR 9 TICKETS', 'EN COURS', and 'FINI'. The 'À FAIRE' column contains four tickets: 'Laravel' (FS-9), 'React' (FS-10), 'Base donnée :Mongo DB' (FS-11), and 'Conception :UML' (FS-12). A button labeled 'Créer une colonne' is positioned above the 'FINI' column.

Right Screenshot: The board is identical to the first, but a fourth column, 'RECETTE', has been added between 'EN COURS' and 'FINI'. A blue arrow points from the 'Créer une colonne' button in the first screenshot to the 'RECETTE' column in the second screenshot, illustrating the process of adding a new column.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Tableau du bord de sprint

Etapas des tickets

- On glisse chaque ticket selon l’état de sa réalisation dans les différentes colonnes du tableau de sprint

Projets / Full_stack

Tableau Sprint 1

⚡ ☆ 🕒 Terminer le

🔍 MG S 👤 Epic 1 Effacer les filtres REGROUPER PAR Aucun

A FAIRE	EN COURS 2 SUR 2 TICKETS	RECETTE 2 SUR 2 TICKETS	FINI ✓
	<p>Laravel</p> <p>FULL_STACK:LARAVEL_REACT</p> <p>FS-9 10 MG</p>	<p>Base donnée :Mongo DB</p> <p>FULL_STACK:LARAVEL_REACT</p> <p>FS-11 5</p>	
	<p>React</p> <p>FULL_STACK:LARAVEL_REACT</p> <p>FS-10 10 S</p>	<p>Conception :UML</p> <p>FULL_STACK:LARAVEL_REACT</p> <p>FS-12 3 S</p>	

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Tableau du bord de sprint

Terminer le sprint

- Si nos tickets sont tous dans l’état **FINI** ça veut dire qu’on a terminé toutes nos tâches et on peut cliquer sur **terminer le sprint** :

The screenshot shows a Jira Sprint Board for 'Tableau Sprint 1'. The board has four columns: 'A FAIRE', 'EN COURS', 'RECETTE', and 'FINI 4 SUR 4 TICKETS'. The 'FINI' column contains four tickets, each with a green checkmark and a red 'S' icon, indicating they are completed. The tickets are grouped by epic: 'Laravel', 'Base donnée :Mongo DB', and 'React'. A blue arrow points to the 'Terminer le sprint' button in the top right corner of the board.

Projets / Full_stack
Tableau Sprint 1

RECHERCHER [] MG S [] Epic 1 Effacer les filtres

REGROUPER PAR Aucun Analyses

A FAIRE EN COURS RECETTE FINI 4 SUR 4 TICKETS

- FS-12 ✓ 1 S
- Laravel FULL_STACK:LARAVEL_REACT
- FS-9 ✓ 10 MG
- Base donnée :Mongo DB FULL_STACK:LARAVEL_REACT
- FS-11 ✓ 5
- React FULL_STACK:LARAVEL_REACT
- FS-10 ✓ 10 S

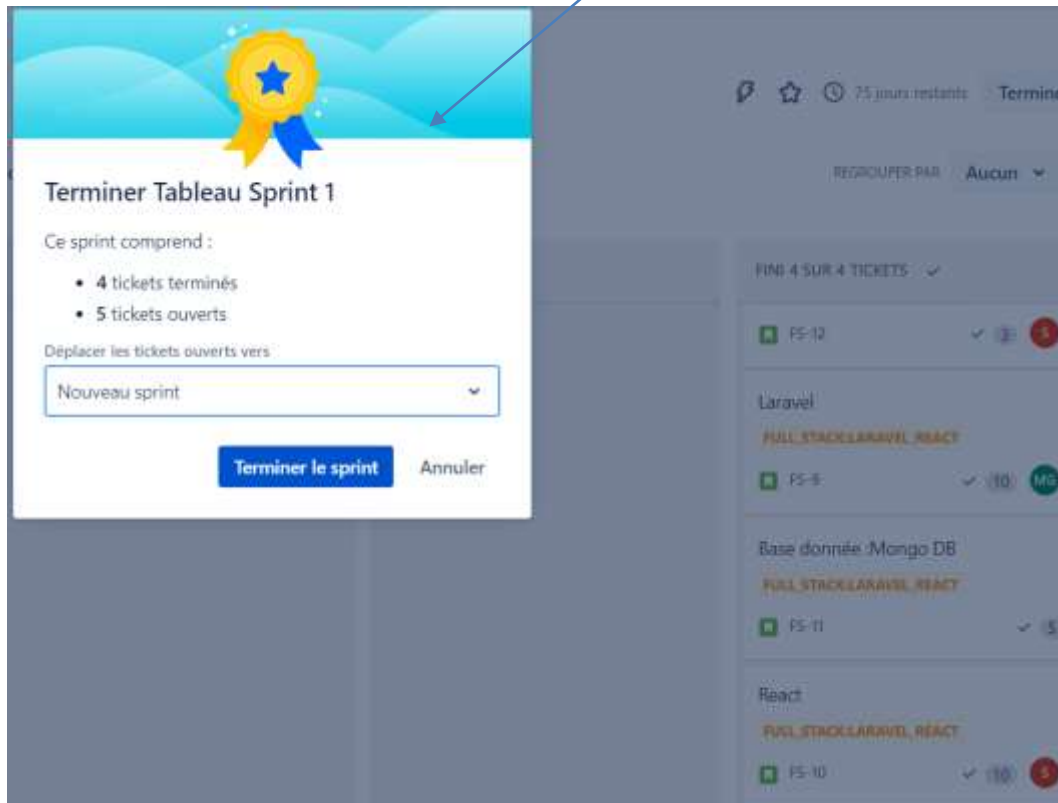
02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Tableau du bord de sprint

Terminer le sprint

- Un message de l’état du sprint s’affiche en indiquant les états des tickets (**terminés** ou **en cours**)

Sprint non terminé



Sprint terminé



Terminer Tableau Sprint 2

Ce sprint contient 5 tickets terminés.

C'est tout. Bien joué !

Terminer le sprint

Annuler

CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira:
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Manipulation du tableau de bord de sprint
- 8. Utilisation de la feuille de route (roadmap)**
9. Génération des rapports Agile



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

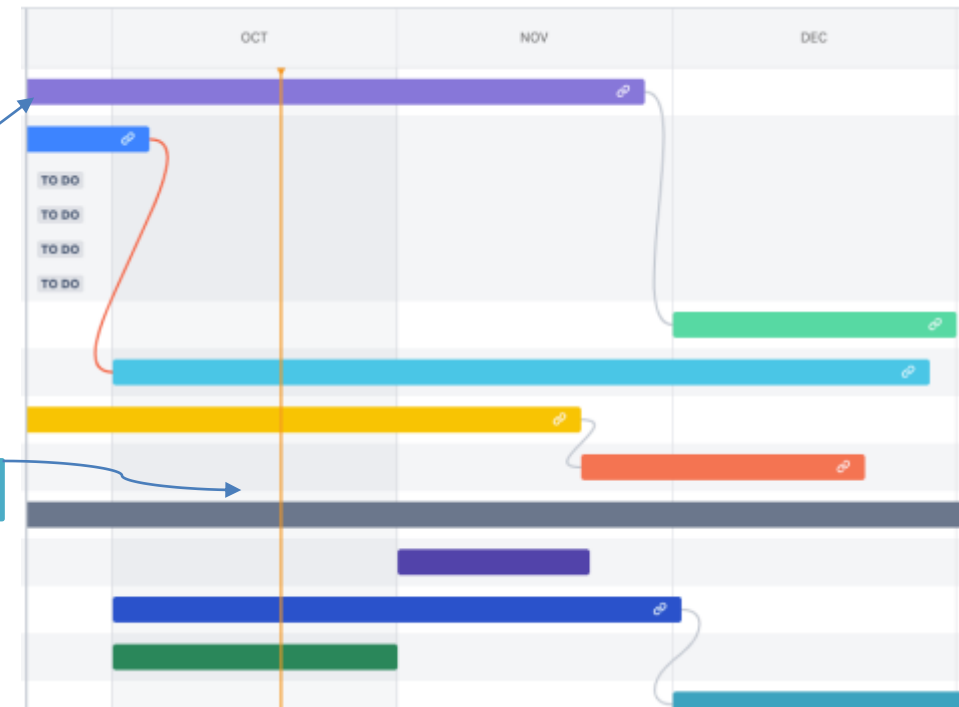
Utilisation de feuille de route (roadmap)

Qu'est-ce qu'une feuille de route ?

- Dans Jira Software, **les feuilles de route** d'équipe sont utiles pour planifier de grosses charges de travail plusieurs mois à l'avance au niveau de l'**epic** dans un seul projet.
- Les fonctionnalités de gestion simplifiée de la planification et des dépendances aident les équipes à mieux visualiser et gérer le travail ensemble.
- Dans **Jira** ,il y a deux versions :
 - Basic(gratuite) pour tout le monde .
 - Premium(payante pour les grands projets.

Les **epics** s'affichent sous forme de barres en couleur sur la feuille de route.

La longueur de la barre sur la feuille de route varie en fonction des dates de début et date de fin.



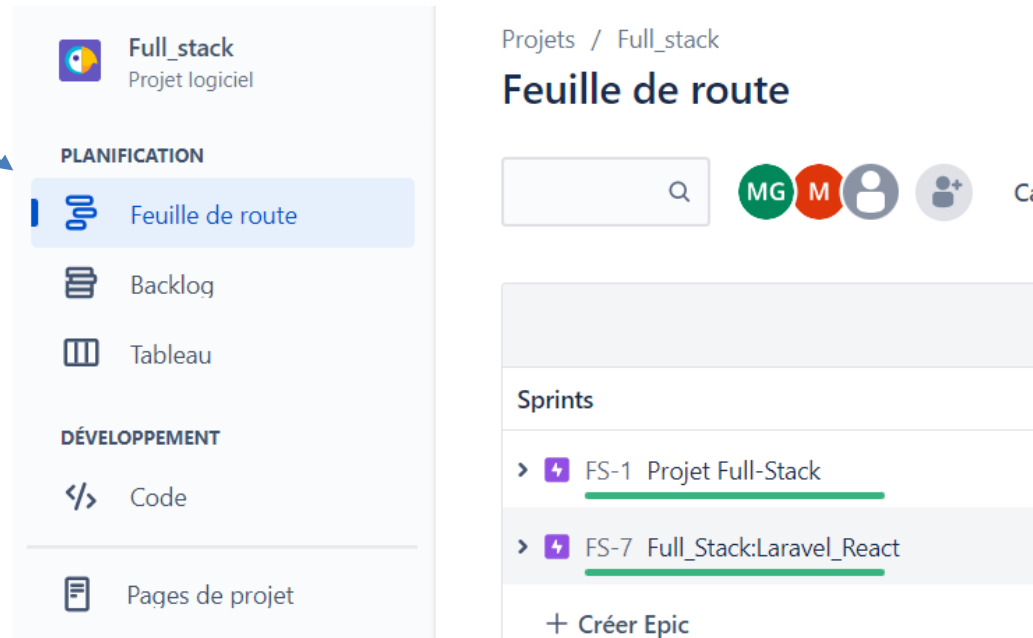
Exemple de feuille de route

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Utilisation de feuille de route (roadmap)

Créer une feuille de route sur Jira

1. Créez un projet ou ouvrez un projet existant, puis accédez à la barre latérale et cliquez sur **Roadmap** (Feuille de route).



2. Cliquez sur **créer une epic** pour créer des **epics** directement dans votre feuille de route.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Utilisation de feuille de route (roadmap)

Créer une feuille de route sur Jira

3. Vous pouvez double-cliquer sur les **epics** à tout moment à partir de votre feuille de route pour ajouter des informations, telles que les dates de début et de fin, le responsable, les pièces jointes ..

The screenshot displays the Jira Roadmap interface. On the left, a 'Feuille de route' (Roadmap) view shows a hierarchy of sprints and epics. The 'Sprints' section includes 'FS-1 Projet Full-Stack' and 'FS-7 Full_StackLaravel_React'. Under 'FS-7', several epics are listed: 'FS-12 Conception :U...' (EN COURS), 'FS-9 Laravel' (EN COURS), 'FS-11 Base donnée :Mong...' (RECYTE), and 'FS-10 React' (TERMINE(E)). A blue arrow points from the 'RECYTE' label of the 'FS-11' epic to a detailed view on the right.

The detailed view on the right shows the following information for the epic 'Base donnée :Mongo DB':

- Recette** (dropdown menu)
- Description**: Ajouter une description...
- Champs épinglés**: Cliquez sur à côté des étiquettes de champ pour épingler.
- Détails**:
 - Responsable: Non assigné (Me l'assigner)
 - Étiquettes: Aucun
 - Sprint: Aucun +1
 - Story point estimate: 5
 - Rapporteur: mohamed (MG)
- Création hier
- Configurer

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Utilisation de feuille de route (roadmap)

Créer une feuille de route sur Jira

4. Ajoutez des tickets enfants à votre **epic** depuis la feuille de route en cliquant sur **+** en regard du nom de **l'epic**.

5. Sélectionnez le type de ticket enfant à l'aide du menu déroulant, puis nommez le ticket.

The screenshot displays the Jira Roadmap interface. On the left, a list of sprints is shown, including 'FS-1 Projet Full-Stack' and 'FS-7 Full Stack: Laravel React'. Under 'FS-7', several child tickets are listed: 'FS-12 Conception :U...' (EN COURS), 'FS-9 Laravel' (EN COURS), 'FS-11 Base donnée :Mong...' (RECETTE), and 'FS-10 React' (TERMINÉ(E)). A blue arrow points to the '+' icon next to the 'FS-7' epic name. On the right, a modal window for creating a child ticket is open. It shows the parent epic name 'Full Stack: Laravel React' and the child ticket name 'React'. A dropdown menu is open, showing 'Terminé(e)' as the selected status. Below the dropdown, there is a 'Description' field and a 'Champs épinglés' section. At the bottom of the modal, there is a comment field and a 'Conseil de pro' tip.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Utilisation de feuille de route (roadmap)

Créer une feuille de route sur Jira

5. Ajuster la longueur de l'epic ou faites glisser l'epic pour modifier les dates de début et fin.

The screenshot shows the Jira Roadmap interface. On the left, a list of sprints is visible, including 'FS-1 Projet Full-Stack' and 'FS-7 Full_Stack:Laravel_React'. The main area displays a timeline from October to November. A yellow bar representing an epic is highlighted, and a blue arrow points to its right edge, indicating that it can be dragged to adjust its duration. The right sidebar provides details for the selected epic, including its name 'Full_Stack:Laravel_React', a progress bar at 25%, and a list of child tickets like 'FS-12 Conception :UML'.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Utilisation de feuille de route (roadmap)

Créer une feuille de route sur Jira

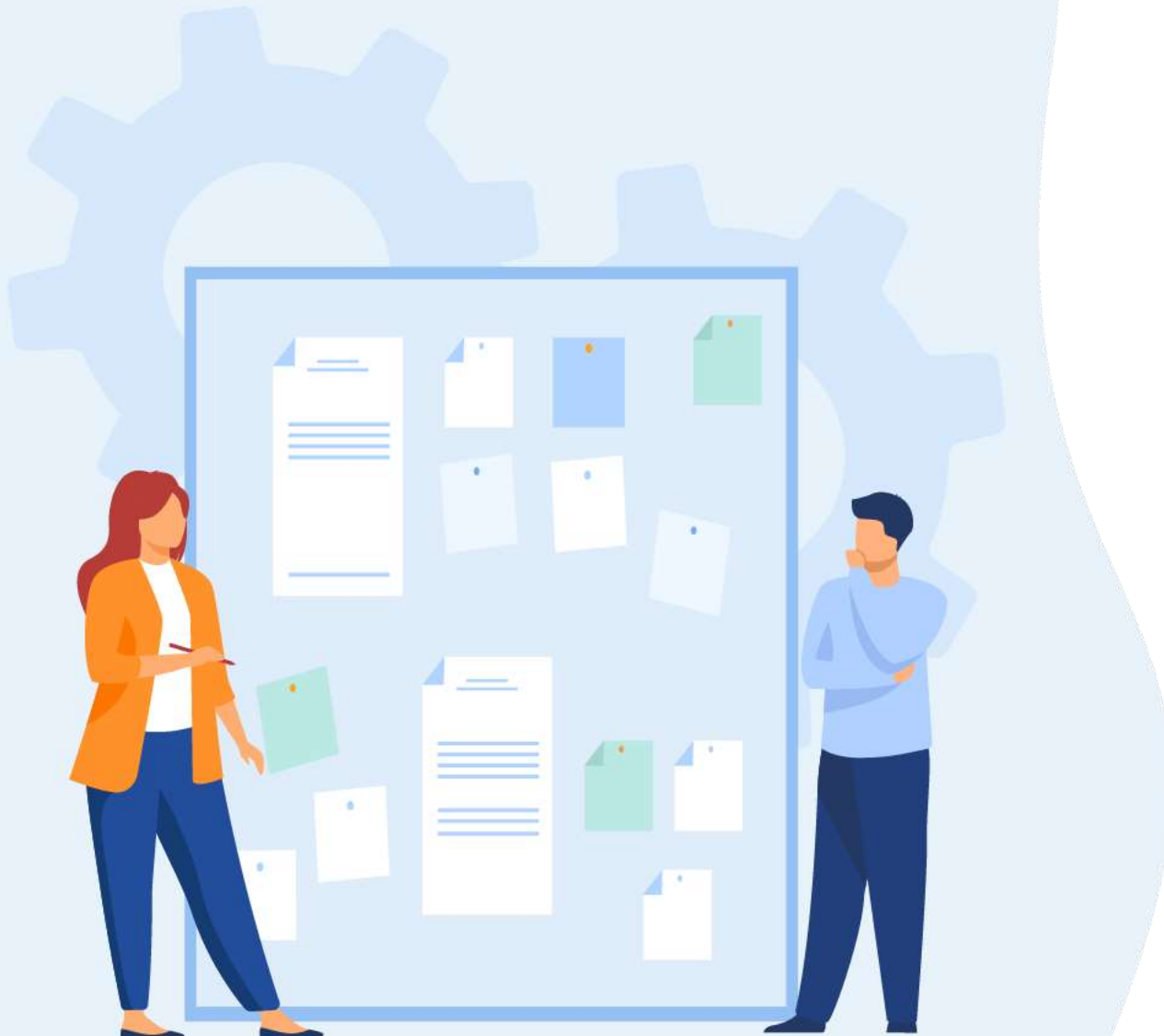
6. Vous pouvez voir la durée des tickets dans la feuille par jour et par mois et faire une dépendance entre les tickets



CHAPITRE 2

Manipuler l'outil de gestion de projet Agile (Scrum/Jira)

1. Présentation de Jira
2. Installation et configuration d'un compte Jira
3. Création d'un projet scrum avec Jira
4. Création d'un backlog product
5. Ajout des différents types de tickets
6. Planification d'un sprint
7. Création des tâches
8. Manipulation du tableau de bord de sprint
9. Utilisation de la feuille de route (roadmap)
- 10. Génération des rapports Agile**



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles: définitions

Tableau de bord Jira

- Ce sont des outils extrêmement puissants permettant aux utilisateurs de tous les niveaux d’une organisation de **visualiser les données rapidement, simplement et de manière fiable**.
- Un tableau de bord est personnalisable à l’aide de **gadgets** fournissant des résumés dynamiques et visuels des données de projets et de tickets Jira.
- **Un gadget** est un bloc contenant une information de façon visuelle.

Rapports agiles sur Jira

- Afin d’avoir une **vision globale du projet**, vous pouvez personnaliser un tableau de bord à l’aide de gadgets, ajouter des graphiques, des camemberts ou encore des diagrammes et suivre facilement le travail de votre équipe.
- Jira donne la possibilité de la création des rapports intégrées. Ces rapports offrent une vue générale des projets, des tendances, ainsi que des rapports de performance.
- Avec Jira, vous pouvez créer des graphiques différents, mais les options de configuration restent limitées dans la version free.

Quelques types de rapports sur Jira

- Jira offre plusieurs types de rapports:
 - Rapports sur les délais de création, de résolution et d’âge moyen des demandes
 - Rapports de Demandes créées récemment
 - Graphiques camembert, secteurs..
 - Graphiques d’avancement des sprints : (**Sprint Burndown**) qui permettent de déterminer le temps passé sur une itération.
 - ..
- Tous ces graphiques sont générés via des fonctionnalités relativement basiques.
- En général, vous pouvez afficher un projet ou un filtre sur une période de temps.
- Vous pouvez également les grouper par intervalle de temps (jour, semaine, mois, année).
- Ces graphiques peuvent être suffisants lorsque les membres d’un projet souhaitent un rapport journalier et un suivi des tendances du projet.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Générer un rapport d’analyse sur le sprint : BurnDown de sprint

Dans le tableau du sprint du projet, pour afficher l’état des travaux des sprints et les **epics**, on clique sur Analyses :

The image shows two screenshots of the Jira sprint analysis interface. The left screenshot shows a sprint that is 100% completed. The right screenshot shows a sprint that is 0% completed with a burndown chart and an attention warning.

Left Screenshot (Completed Sprint):

- REGROUPER PAR: Aucun
- Analyses (highlighted)
- Avancement du sprint: 100 % terminé(s)
- Terminé: 100%, En cours: 0%, Non démarré: 0%
- Burndown de sprint: Ajoutez des estimations pour gérer et maintenir le périmètre
- Avancement des epics: Ce sprint va atteindre 1 epic
- FS-1 Projet Full-Stack: 100% done

Right Screenshot (Sprint with Burndown):

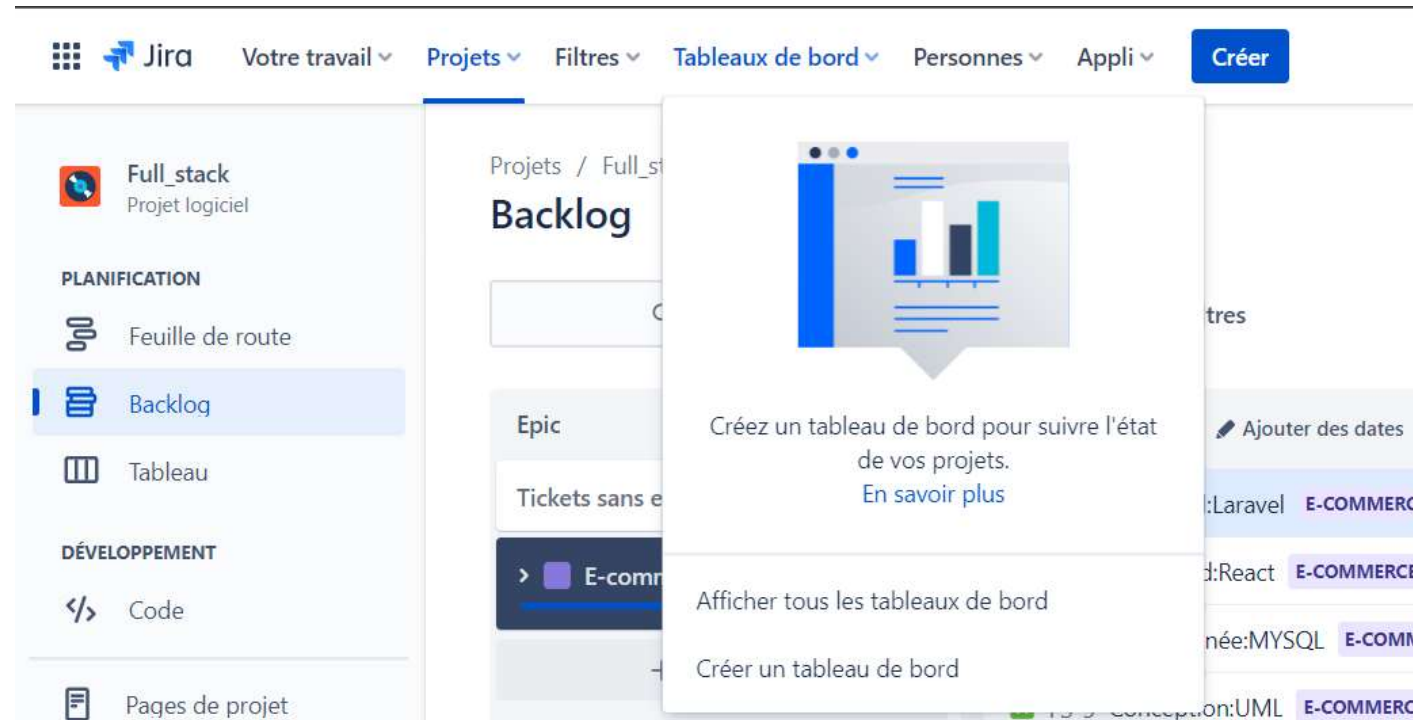
- REGROUPER PAR: Aucun
- Analyses (highlighted)
- Avancement du sprint: 0 % terminé(s)
- Terminé: 0%, En cours: 100%, Non démarré: 0%
- Burndown de sprint: 0 point terminé, 24 points restants. Attention (warning icon)
- Graphique: Travail restant (blue line) vs Recommandation (grey line) from Jul 17 to Jul 31.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Création du tableau de bord

- Dans barre de menu : cliquer sur **Tableaux de bord** > **Créer un tableau de bord**.



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Création du tableau de bord

- **Donnez ensuite un nom et une description** à votre tableau de bord pour que votre équipe sache quand l’utiliser et définissez les utilisateurs pouvant avoir accès à celui-ci. Puis, **Enregistrer**.

Créer un tableau de bord

Nom *

TB1

Description

statistique des données

Utilisateurs disposant de droits en lecture

Projet Proj et Ajouter

Privé

Éditeurs

Groupe Groupe Ajouter

Privé

Enregistrer Annuler



Créer un tableau de bord

Nom *

TB2

Description

Utilisateurs disposant de droits en lecture

Projet Full_stack Tous les rôles Ajouter

Full_stack, Tous les rôles X

Éditeurs

Projet Projet de ser... Tous les rôles Ajouter

Privé

Administrators

Service Desk Team

Enregistrer Annuler

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord

- Ensuite dans l’interface des tableaux bord vous ajoutez des gadgets (graphes) en cliquant sur **Ajouter un gadget** :

The screenshot displays the Jira dashboard interface. At the top, there are navigation buttons: 'Refresh', 'Ajouter un gadget', 'Modifier la mise en page', and 'Terminé'. A notification bar indicates that changes to the dashboard are being saved automatically. The main area consists of two empty columns, each with a plus sign icon and the instruction 'Faites glisser un gadget vers cette colonne ou ajoutez un nouveau gadget.' On the right side, a modal window titled 'Ajouter un gadget' is open, showing a search bar and a list of available gadgets. The gadgets listed include 'Calendrier des tickets Jira', 'Carte thermique', and 'Durée de résolution', each with an 'Ajouter' button. A blue arrow points from the 'Ajouter un gadget' button in the top toolbar to the modal window.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord

- Vous pouvez filtrer les types de gadgets à ajouter par exemple ici les graphique:

Ajout de Graphique demandes

Ajout de Graphique à secteurs

Ajouter un gadget

Toutes 9 Jira 9 Charts 7 Wallboard 1

Jira Service Management 0

Ajouter Affiche le nombre moyen de jours pendant lesquels les tickets n'ont pas été résolus.
Jira Charts

Ajouter Graphique Demandes créées comparées aux demandes résolues
Par Atlassian
Affiches des tickets créés par rapport aux tickets résolus pour un projet ou un filtre enregistré.
Jira Charts

Ajouter Graphique Plage de demandes
Par Atlassian
Affiche le nombre de tickets dans une période donnée sur la base d'un champ donné.
Jira Charts

Ajouter Graphique récemment créé
Par Atlassian



Graphique Demandes créées compa...

Projet ou filtre enregistré*

Full_stack
ecom

Projet ou filtre enregistré à utiliser comme base du graphique.
Recherche avancée

Période*

Par jour

La durée des périodes représentées sur le graphique.

Plage de jours*

30

Plage de jours dans le passé pour laquelle recueillir des données.

Fonctionnement des séries*

Nombre

Ajouter progressivement les totaux (1, 2, 3), ou montrer les valeurs individuelles (1, 1, 1).

Afficher la tendance des tickets non résolus*

Non

Affichage dans un graphique secondaire du nombre de tickets non résolus au cours d'une période donnée

Afficher les versions*

Seulement les versions majeures

Indiquer dans le graphique la date de lancement des versions.

Actualisation automatique

Mettre à jour toutes les 15 minutes

Enregistrer

Graphique à secteurs

Projet ou filtre enregistré*

Full_stack
Recherche

Projet ou filtre enregistré à utiliser comme base du graphique.
Recherche avancée

Type de statistique*

Responsable

Sélectionnez le type de statistique à afficher pour ce filtre

Actualisation automatique

Mettre à jour toutes les 15 minutes

Enregistrer

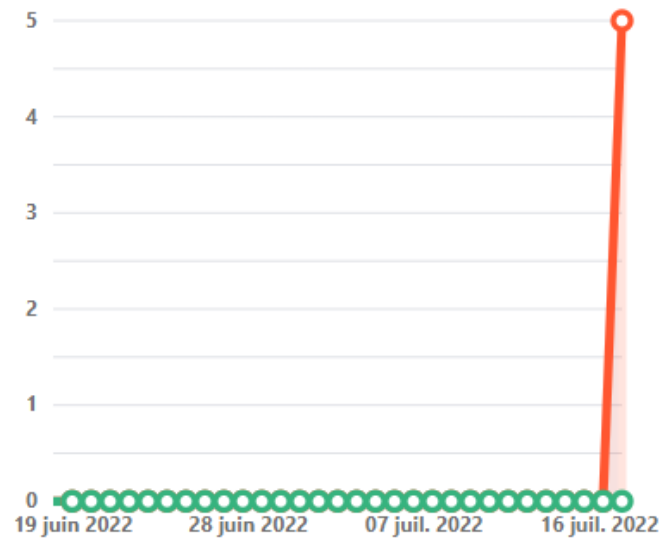
02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord

- Ici les résultats qui s’affichent dans le tableau du bord des gadgets graphique demandes et graphique secteurs utilisés dans notre exemple:

Graphique Demandes créées compa...

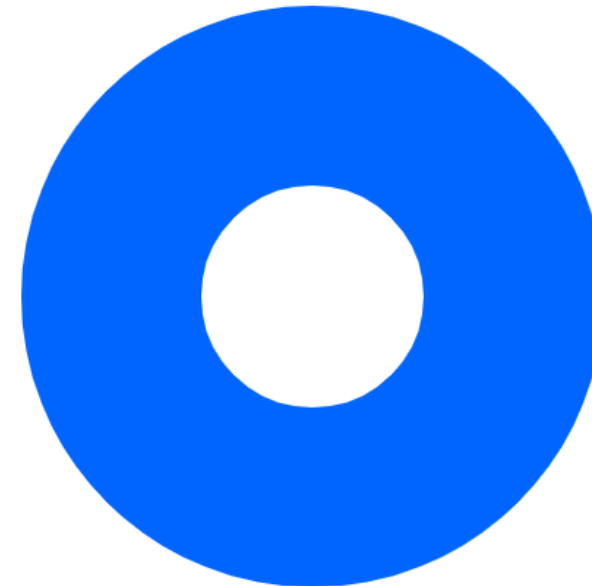


Tickets au cours des 30 derniers jours (par jour groupé)

Afficher dans le navigateur de ticket

- Tickets créés (5)
- Tickets résolus (0)

Graphique à secteurs : Full_stack



Responsable

Total des tickets : 5

- Non assigné 5

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord

- Ajout du gadget **Burndown du sprint** : il permet de **visualiser l’avancement actuel du sprint en fonction de la méthode d’estimation utilisée dans votre projet** (Temps estimé, Story points...).

Ajouter un gadget

graph

Toutes 9 Jira 9 Charts 7 Wallboard 1

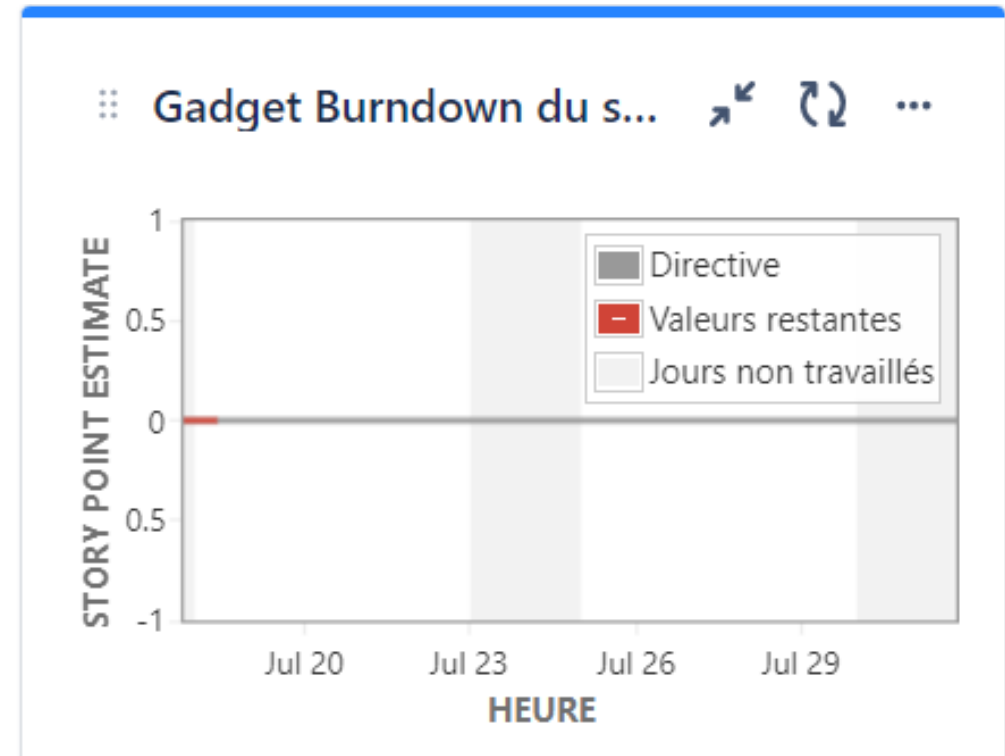
Jira Service Management 0

Gadget Burndown du sprint
Par Atlassian

Graphique Burndown (d'avancement) du sprint pour suivre la quantité restante de travaux (peut être affiché sous forme de wallboard)

Ajouter

Jira Wallboard



02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord:

- On peut ajouter des gadgets pour visualiser l’état du sprint ,ici:
 - Gadget intégrité du sprint:** permet de visualiser l’état des tickets contenus dans votre sprint
 - Gadget jours restants:** il affiche le nombre de jours restants du sprint actuel.

Ajouter un gadget

Q sprint

Toutes 3 Jira 3 Charts 0 Wallboard 3

Jira Service Management 0

Gadget Burndown du sprint
Par Atlassian
Ajouter
Graphique Burndown (d’avancement) du sprint pour suivre la quantité restante de travaux (peut être affiché sous forme de wallboard)
Jira Wallboard

Gadget Intégrité du sprint
Par Atlassian
Ajouter
Capture d’écran de l’intégrité du sprint (peut être affiché sous forme de wallboard)
Jira Wallboard

Jours restants dans le gadget Sprint
Par Atlassian
Ajouter
Affiche les jours restants dans un sprint (peut être affiché sous forme de wallboard)
Jira Wallboard

Gadget Intégrité du sprint

Tableau Sprint 4 - Tableau FS

Aucun ticket estimé (Story point estimate) 10 jours restants

0^h 0^h 0 0

Temps écoulé Modification de la portée

Jours restants dans le gadget Sprint

10

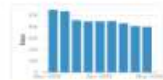
Jours restants

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Ajout des gadgets dans le tableau du bord:

- **Graphique d’ancienneté moyenne** : Ce graphique montre le nombre moyen de jours pendant lesquels les tickets restaient non résolus ,il remplace **La vélocité¹** dans la version free.



Graphique d’ancienneté moyenne
Par Atlassian

Ajouter

Affiche le nombre moyen de jours pendant lesquels les tickets n"ont pas été résolus.

Jira Charts

- On choisi le graphique ,puis on défini le projet ,la période et la plage

Graphique d’ancienneté moyenne

Projet ou filtre enregistré*
Full_stack

Recherche

Projet ou filtre enregistré à utiliser comme base du graphique.
Recherche avancée

Période:
Par jour

La durée des périodes représentées sur le graphique.

Plage de jours
30

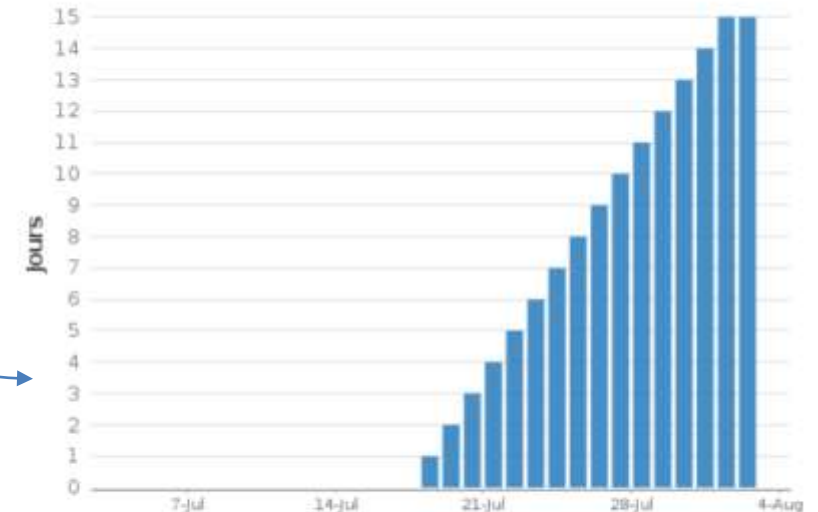
Jours (y compris aujourd’hui) à afficher dans le graphique.

Intervalle d’actualisation
Jamais

Quelle fréquence de mise à jour voulez-vous pour ce gadget

Enregistrer

Graphique d’ancienneté moyenne : Full_stack



Ce graphique montre le nombre moyen de jours pendant lesquels les tickets restaient non résolus.

Période : derniers 30 jours (groupés Par jour)




¹La **vélocité** : désigne la quantité moyenne de travail qu’une équipe Scrum réalise pendant un sprint. Mesurée en heures, jours .. elle est très utile pour les prévisions.

02 – Manipuler l’outil de gestion de projet Agile (Scrum/Jira)

Génération des rapports agiles

Création de plusieurs tableaux de bord

- Vous pouvez créer plusieurs tableaux de bord, les tableaux de bord sont mis à jour automatiquement après la création
- Vous pouvez supprimer ou renommer ou modifier la couleur de chaque gadget dans le tableau du bord .

Graphique dancie...    ...

COULEUR DE SURBRILLANCE



Configurer

Renommer

Supprimer

Tableaux de bord ▾

Personnes ▾

Appli ▾

Créer

MIS EN FAVORIS



TB1



TB2



Afficher tous les tableaux de bord

Créer un tableau de bord



PARTIE

4

Mettre en œuvre des outils de gestion de versions et de mesure de la qualité du code

Dans ce module, vous allez :

- Manipuler les outils de gestion de versions (Git/Gitlab) .
- Manipuler l'outil de mesure de la qualité du code (SonarQube)



30

heures

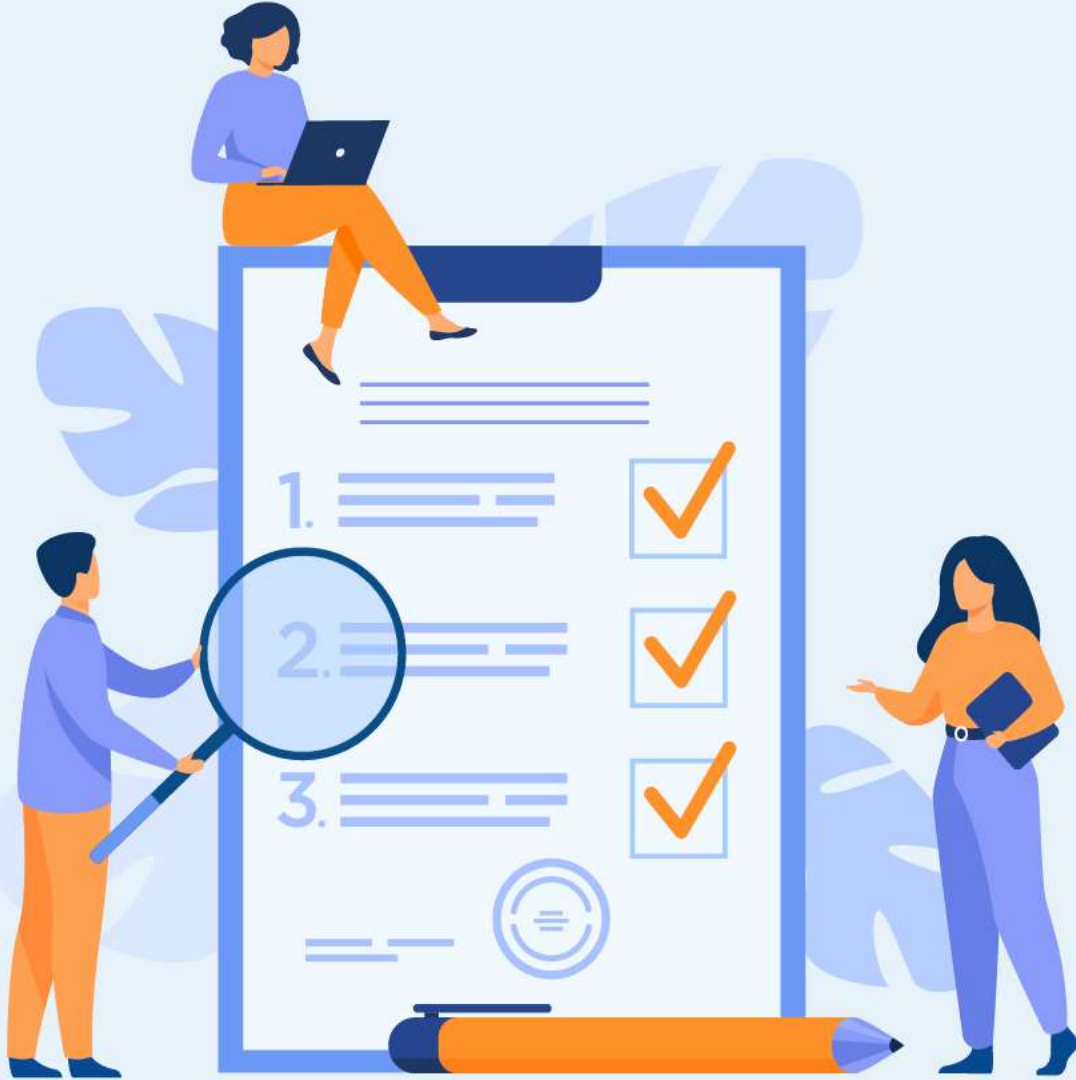
CHAPITRE

Manipuler les outils de gestion de versions (Git/Gitlab)

Ce que vous allez apprendre dans ce chapitre :

- Présentation des outils existants de gestion
- Présentation de Git et Gitlab et comparaison entre les deux
- Présentation des fonctionnalités de Gitlab
- Installation et manipulation de Git.
- Manipulation des commandes de base de Git (Git bash)
- Notion de branches et gestion des conflits de fusion avec Git

15
heures



CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. Présentation de Git
3. Présentation de Gitlab
4. Manipulation des dépôts avec Gitlab
5. Gestion des conflits de fusion avec Git/GitLab
6. Comparaison Git vs Gitlab



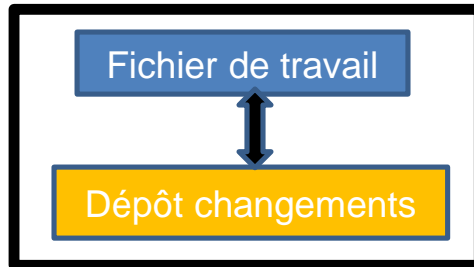
01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

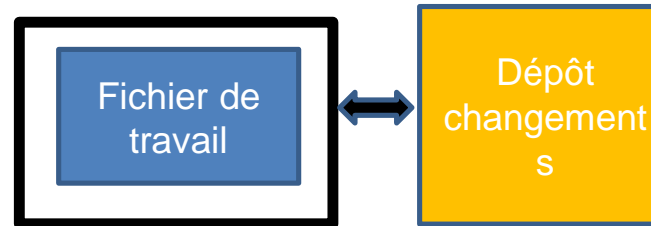
Gestion de versions

C'est quoi la gestion de versions?

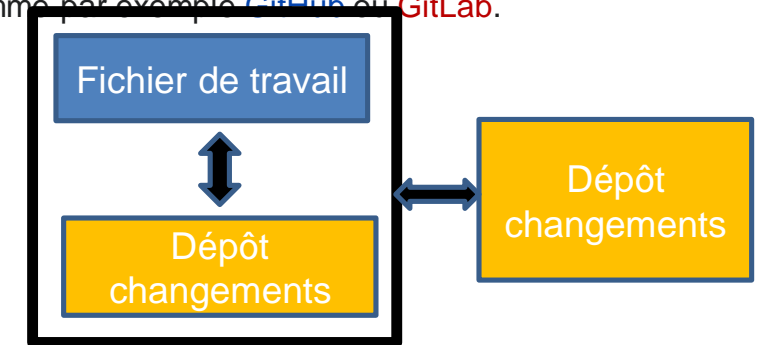
- La gestion de versions également appelé historique de versions ou contrôle de versions (en anglais *version control system* (VCS)) se réfère, dans le milieu numérique, au stockage de plusieurs versions de fichier(s) afin de pouvoir tracer l'évolution chronologique à travers les changements apportés d'une version à l'autre.
- Elle peut s'appliquer à un fichier individuel (exemple : un document texte) ou à plusieurs fichiers d'un projet; elle peut se faire au niveau individuel ou dans des groupes.
- On peut diviser les systèmes de gestion de versions en trois catégories :
- La gestion de versions se fait aussi à la base de plateformes collaboratives privées ou open source, comme par exemple [GitHub](#) ou [GitLab](#).
 - Gestion de versions locale,
 - Gestion de versions centralisée,
 - Gestion de versions distribuée ou décentralisée.



Dans un système de gestion de versions **locale**, le *dépôt*(code ou code de source) avec les changements se trouve physiquement sur la même machine qui stocke les fichiers tracés



Dans la gestion de versions **centralisée**, le *dépôt* qui contient les informations sur les changements se trouve sur une autre machine par rapport aux fichiers de travail. Le cas de figure le plus commun consiste à garder le *dépôt* des changements sur un serveur centralisé, tandis que les différents ordinateurs individuels des personnes qui participent au projet ne gardent que la dernière version des fichiers de travail.



La gestion de versions **distribuée** combine la gestion de version locale et centralisée en créant **deux dépôts** des changements :

1. Le premier se trouve sur la même machine des fichiers de travail
2. Le deuxième se trouve dans une autre machine, souvent un serveur ou une plateforme *cloud* comme [GitHub](#) ou [GitLab](#), qui s'occupe de centraliser les changements

01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Gestion de versions

gestion de versions distribuées

- La gestion distribuée est notamment l'un des éléments principaux des projets open source, comme elle est représentée dans l'image ci-bas
- La gestion de versions décentralisée consiste à voir l'outil de gestion de versions comme un outil permettant à chacun de travailler à son rythme, de façon désynchronisée des autres, puis d'offrir un moyen à ces développeurs de s'échanger leur travaux respectifs. De ce fait, il existe plusieurs dépôts pour un même logiciel.
- Ce système est très utilisé par les logiciels libres.
- Par exemple, **GNU Arch** et **Git** sont des logiciels de gestion de versions décentralisée.



Exemple de gestion de versions distribuée source <https://edutechwiki.unige.ch/>

- Dans ce schéma qui utilise **Git** et **GitHub** comme exemple (le même principe peut-être appliqué à d'autres systèmes), le dépôt local se trouve dans un dossier **.git** sur l'ordinateur d'une personne. Ce dépôt local est partagé en open source à travers un dépôt central qui se trouve sur la plateforme *cloud* **GitHub**. Ce dépôt central peut être utilisé par plusieurs personnes pour :
 - Contribuer au même projet: envoyer des changements qui sont ensuite incorporés dans le dépôt central et peuvent donc être ensuite propagés à tout dépôt local connecté,
 - Utiliser ce dépôt central comme base pour un autre projet.

01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Gestion de versions

Exemples d'outils de gestion de versions

- **Wiki** : appelée dans ce contexte souvent historique des versions, est l'une des fonctionnalités principales caractérisant tout système qui s'inspire du principe wiki. Des systèmes de ce type peuvent se trouver à plusieurs endroits, avec des fonctionnalités légèrement différentes, comme par exemple dans :
 - **Moodle** : une plateforme pédagogique très utilisée dans la formation et l'enseignement,
 - **GitHub** : dans lequel le wiki est souvent utilisé comme documentation ou moyen d'organisation entre collaborateurs,
 - **Tous les sites qui sont basés sur un moteur wiki**, comme par exemple MediaWiki, le logiciel open-source à la base de Wikipédia et également de EduTech Wiki.
- **Logiciels de traitement de texte** : plusieurs logiciels de traitement de texte, surtout dans les versions en ligne et collaboratives, permettent de retracer l'historique des versions : OFFICE ,GOOGLE DOCS..
- **CVS (Concurrent Versioning System)** : fonctionne sur un principe centralisé, de même que son successeur SVN (Subversion).
- **Logiciels de SCM décentralisés** : sont apparus plus récemment : Mercurial et surtout Git, que nous utiliserons dans la suite de ce chapitre. Ce sont également des logiciels libres.
- **TFS (Team Foundation Server)** de Microsoft: TFS est une solution payante qui fonctionne de manière centralisée.

CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. **Présentation de Git**
3. Présentation de Gitlab
4. Manipulation des dépôts avec Gitlab
5. Gestion des conflits de fusion avec Git/GitLab
6. Comparaison Git vs Gitlab



01-Manipuler les outils de gestion de versions(Git/Gitlab) :

Présentation de git

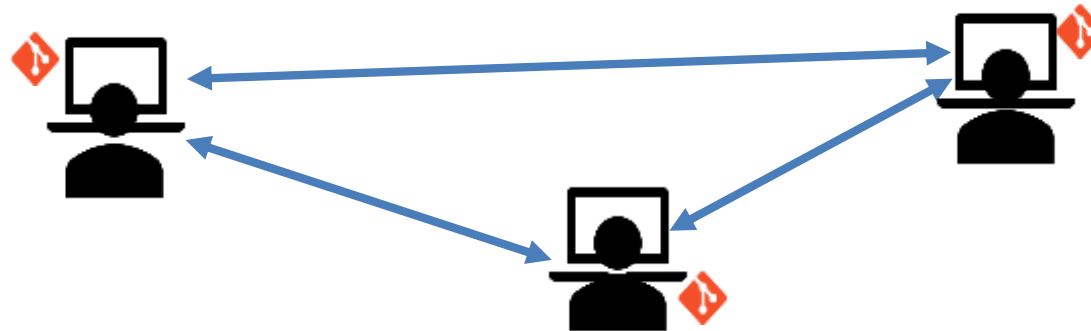
C'est quoi git?

- **Git** est un logiciel libre de gestion de versions. C'est un outil qui permet d'archiver et de maintenir les différentes versions d'un ensemble de fichiers constituant souvent le code source d'un projet logiciel, il est multi-langages et multi-plateformes. **Git** est devenu à l'heure actuelle un quasi-standard .



logo de Git (Le nom "Git" se prononce comme dans guitare)

- **Git** rassemble dans un **dépôt (*repository* ou *repo*)** l'ensemble des données associées au projet. Il fonctionne de manière décentralisée: tout dépôt Git contient l'intégralité des données (code source, historique, versions, etc).
- Chaque participant au projet travaille à son rythme sur son dépôt local. Il existe donc autant de dépôts que de participants.
- **Git** offre des mécanismes permettant de synchroniser les modifications entre tous les dépôts.

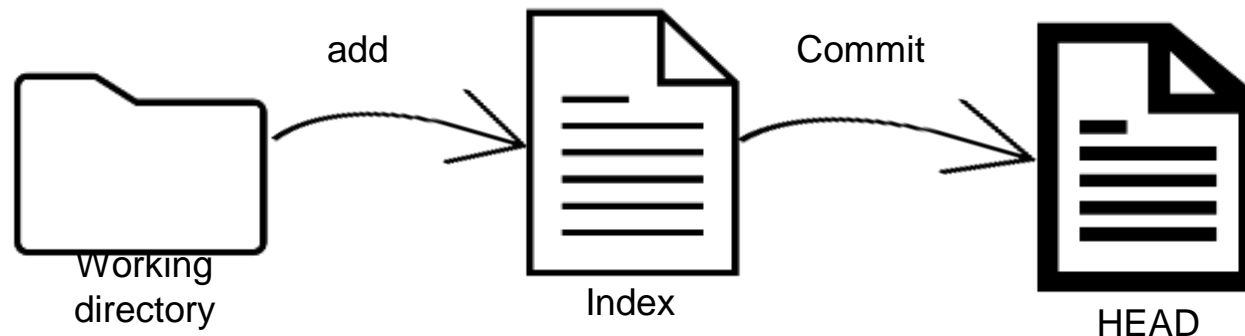


01-Manipuler les outils de gestion de versions(Git/Gitlab) :

Présentation de git

Les dépôts avec git:

- **Version** : contenu du projet à un moment de son cycle de vie.
- **Dépôt**(repository) : l'historique du projet, contenant toutes ses versions.
- **Branche** (branch) = variante d'un projet.
- Un dépôt Git correspond physiquement à un ensemble de fichiers rassemblés dans un répertoire **.git**. Sauf cas particulier, il n'est pas nécessaire d'intervenir manuellement dans ce répertoire.
- Lorsqu'on travaille avec Git, il est essentiel de faire la distinction entre trois zones :
 - Le **répertoire de travail** (*working directory*) : correspond aux fichiers actuellement sauvegardés localement.
 - L'**index** ou *staging : area* est un espace de transit.
 - **HEAD** : correspond aux derniers fichiers ajoutés au dépôt.



01-Manipuler les outils de gestion de versions (Git/Gitlab) :

Installation de git

Installation git?

1.Vérifier dans le terminal que git est bien installé: `git version`

```
C:\Users\...>git version
git version 2.37.1.windows.1
```

2.Si ce n'est pas le cas, installez-le en suivant les recommandations de votre système d'exploitation où en le téléchargeant depuis git-scm.com/downloads, puis redémarrez votre terminal.



Git Bash

Lancer git après l'installation et tapez la commande `git` pour voir les différentes commandes git et leurs rôles

```
DELL@DESKTOP-01D013U MINGW64 ~/Desktop
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
```

01-Manipuler les outils de gestion de versions(Git/Gitlab) :

Commandes git

Principales commandes git

- Les principales commandes git sont :
 - **git init** : crée un nouveau dépôt vide à l'emplacement courant.
 - **git status** : affiche les différences entre le répertoire de travail, l'index et HEAD.
 - **git add** : ajoute des fichiers depuis le répertoire de travail vers l'index.
 - **git commit** : ajoute des fichiers depuis l'index vers HEAD.
 - **git clone** : clone un dépôt existant local ou distant.
 - **git pull** : récupère des modifications depuis un dépôt distant vers HEAD.
 - **git push** : publie des modifications depuis HEAD vers un dépôt distant.

Exemple des principales étapes pour utiliser les commandes git

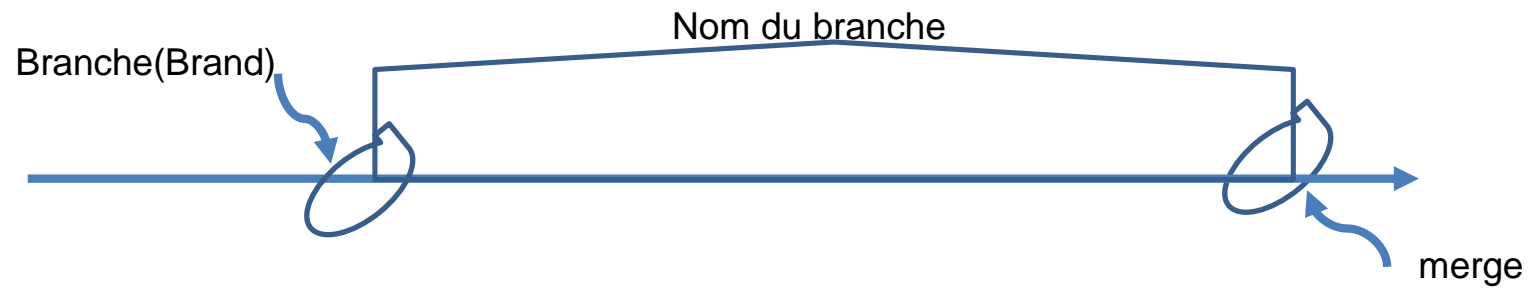
1. Créez un nouveau dossier, ouvrez le et exécutez la commande : **git init** pour initialiser le dossier
2. Créez une copie de votre dépôt local en exécutant la commande : **git clone /path/to/repository** ,si vous utilisez un serveur distant, utiliser la commande **git clone username@host:/path/to/repository**
3. Vous pouvez proposer un changement (l'ajouter à l'**Index**) en exécutant les commandes **git add <fichier>** ou **git add *** .
4. Pour valider ces changements, utilisez **git commit -m 'Message de validation'** .
5. Pour les envoyer à votre dépôt distant, exécutez la commande :**git push origin master**
6. Maintenant, vous pouvez envoyer vos changements vers le serveur distant : en utilisant les commandes 3 et 4 à chaque changement .

01-Manipuler les outils de gestion de versions(Git/Gitlab) :

Gestion branche avec git

Branches

Les branches sont utilisées pour développer des fonctionnalités isolées des autres. La branche *master* est la branche par défaut quand vous créez un dépôt. Utilisez les autres branches pour le développement et fusionnez ensuite à la branche principale quand vous avez fini.



01-Manipuler les outils de gestion de versions(Git/Gitlab) :

Gestion branches avec git



Gestion branches avec commandes git

créer une nouvelle branche nommée B1 et passer dessus pour l'utiliser
`git checkout -b B1`
retourner sur la branche principale
`git checkout master`
et supprimer la branche
`git branch -d B1`
une branche n'est *pas disponible pour les autres* tant que vous ne l'aurez pas envoyée vers votre dépôt distant
`git push origin <branch>`

pour mettre à jour votre dépôt local vers les dernières validations, exécutez la commande
`git pull`
dans votre espace de travail pour *recupérer et fusionner* les changements distants.
pour fusionner une autre branche avec la branche active (par exemple master), utilisez
`git merge <branch>`
dans les deux cas, git tente d'auto-fusionner les changements. Malheureusement, ça n'est pas toujours possible et résulte par des *conflits*. Vous devez alors régler ces *conflits* manuellement en éditant les fichiers indiqués par git. Après l'avoir fait, vous devez les marquer comme fusionnés avec
`git add <nomdufichier>`
après avoir fusionné les changements, vous pouvez en avoir un aperçu en utilisant
`git diff <source_branch> <target_branch>`

vous pouvez annuler les changements locaux en utilisant cette commande
`git checkout -- <nomdufichier>`
cela remplacera les changements dans votre arbre de travail avec le dernier contenu du HEAD. Les changements déjà ajoutés à l'index, aussi bien les nouveaux fichiers, seront gardés.
Si à la place vous voulez supprimer tous les changements et validations locaux, récupérez le dernier historique depuis le serveur et pointez la branche principale locale dessus comme ceci
`git fetch origin`
`git reset --hard origin/master`

CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. Présentation de Git
- 3. Présentation de Gitlab**
4. Manipulation des dépôts avec Gitlab Gestion des conflits de fusion avec Git/GitLab
5. Comparaison Git vs Gitlab



01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Présentation de Gitlab

C'est quoi Gitlab?

Gitlab est une plateforme open source et collaborative de développement basé sur **Git**. **Gitlab** permet d'héberger des projets web, du code, et de la documentation.

Fonctionnalités de Gitlab :

- L'interface de **GitLab** reste très similaire à celle de **GitHub**. Toutefois, **GitLab** propose des options pour le moins pratiques :
 - Gestion de projet
 - Planification / priorisation
 - Build
 - Test logiciel
 - Sécurité applicative
 - Gestion des configurations
 - Monitoring
 - Intégration et déploiement continu, etc.
- Pour un emploi ergonomique, **GitLab** se situe sur une machine virtuelle, elle-même hébergée sur un serveur web. Cet outil de plateforme collaborative s'appuie sur une base de données.
- L'interface d'administration, notamment pour la création de comptes utilisateurs, passe par une configuration en ligne :
 - Création / suppression de dépôts.
 - Enregistrement et gestion des droits d'accès aux dépôts .
 - Outils graphiques pour visualiser et éditer : graphe des commits, branches, tags, fichiers, etc.
 - Documentation des projets (fichiers README..)
 - Outils de communication de projets (issues)
 - Mécanisme de fork et merge requests (comme sur github).

01-Manipuler les outils de gestion de versions

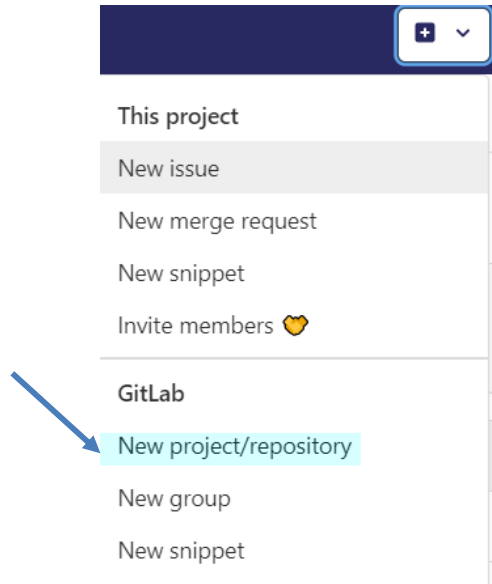
(Git/Gitlab) :

Création de compte

Étapes pour créer un compte sur Gitlab



- Ouvrir l'interface web **GitLab** sur : <https://gitlab.com>
- Si vous ne l'avez pas encore fait, ajoutez un mot de passe à votre compte **GitLab**, depuis la page de paramètres de votre compte **GitLab**
- Après la connexion avec le compte **Gitlab** vous aurez l'interface suivante :
 - Pour créer un projet cliquer sur **create blank project** ou **newProject/repository**



Create new project



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Création de compte

Etapes pour créer compte sur Gitlab

- Donner un nom à votre groupe et projet :

Create or import your first project

Projects help you organize your work. They contain your file repository, issues, merge requests, and so much more.

Group name: fullstack

Project name: projet1

Your project will be created at:
https://gitlab.com/fullstack38/projet1

You can always change your URL later

Include a Getting Started README
Recommended if you're new to GitLab

Create project

- Le nom du projet sera aussi le nom du dépôt correspondant => Choisissez "public", afin que le dépôt soit accessible en lecture aux autres stagiaires, puis cliquer sur **create project**

Project name: projet1

Project URL: https://gitlab.com/fullstack38 / Project slug: projet1

Want to organize several dependent projects under the same namespace? [Create a group.](#)

Project description (optional):

Project deployment target (optional):

Visibility Level: Public
The project can be accessed without any authentication.

Project Configuration: Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more.](#)

Create project Cancel



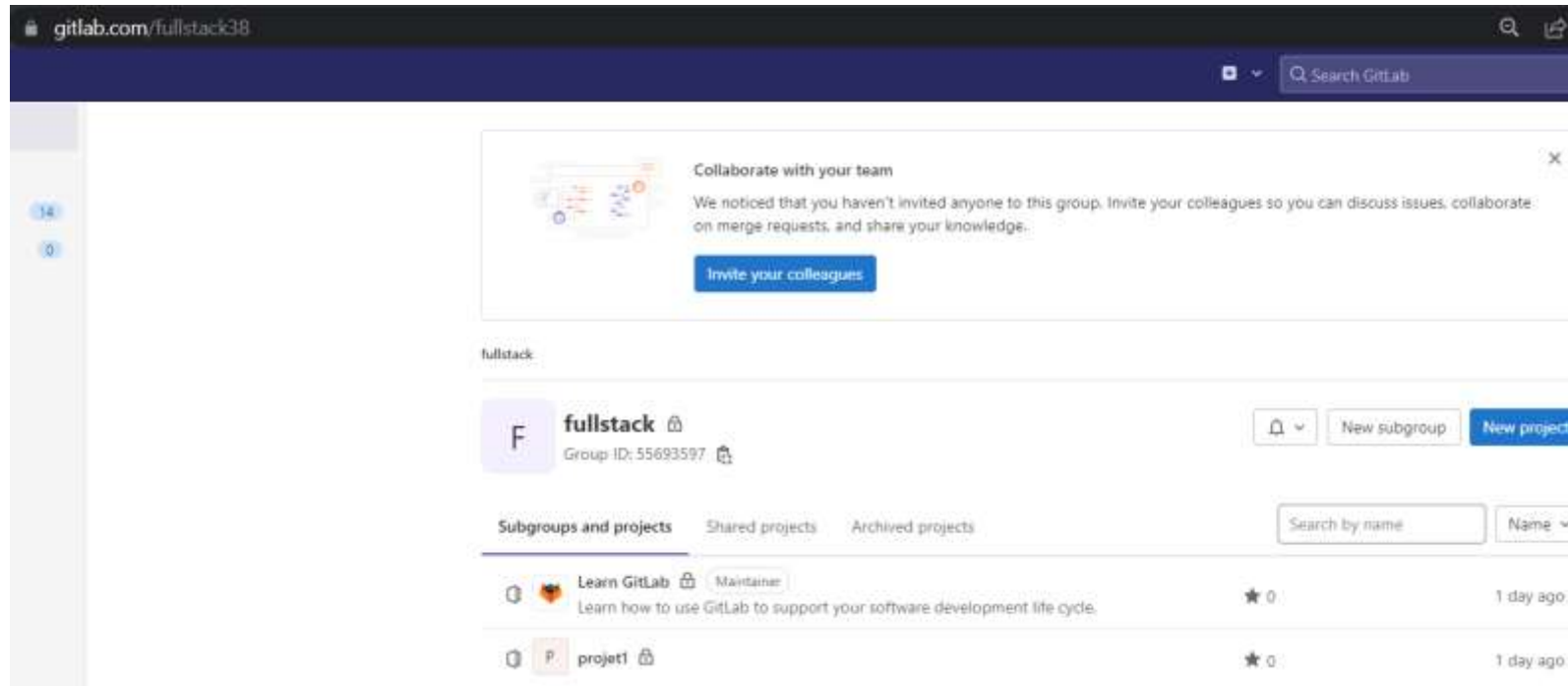
01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Création de compte

Étapes pour créer compte sur Gitlab

- Vous accéder à votre projet via l'url : gitlab.com/groupe Duprojet :



CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. Présentation de Git
3. Présentation de Gitlab
- 4. Manipulation des dépôts avec Gitlab**
5. Gestion des conflits de fusion avec Git/GitLab
6. Comparaison Github vs Gitlab



01-Manipuler les outils de gestion de versions



(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Création dépôts sur Gitlab

1. Dans le terminal:

- Téléchargez le dépôt sur votre disque-dur: `$ git clone <url>` (remplacer <url> par l'adresse HTTPS de votre dépôt, celle qui finit par .git)
- puis entrez dans le dossier de ce dépôt: `$ cd nom_du_dépôt`, exemple:

```
git clone https://gitlab.com/fullstack38/fsdev.git
```

```
C:\Users\DELL>git clone https://gitlab.com/fullstack38/fsdev.git
Cloning into 'fsdev'...
```

```
C:\Users\DELL>cd fsdev
```

01-Manipuler les outils de gestion de versions



(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Création dépôts sur Gitlab

2.Connecter à votre compte en utilisant les commandes de git suivantes avec votre **mail** et votre **username** :

```
DELL@DESKTOP-01DOJ3U MINGW64 ~/fsdev (main)
$ git config --global user.email "XXXXXXXXXX@gmail.com"

DELL@DESKTOP-01DOJ3U MINGW64 ~/fsdev (main)
$ git config --global user.name "Mohamed XXXXX"
```

3. Aussi ,dans le terminal exécuter les commandes suivantes pour modifier votre projet

• Créer un commit:

- **echo "Bonjour " >README.md** : pour créer un fichier README.md contenant le texte "Bonjour"
- **git status** : (optionnel) pour constater qu'un fichier a été créé mais pas encore ajouté dans le dépôt
- **git add README.md** :pour ajouter le fichier README.md dans l'espace de staging (index)
- **git status** :(optionnel) pour afficher le contenu actuel de l'espace de staging (index)
- **git commit -m "ajout du fichier README.md"** : pour créer un commit à partir de l'espace de staging. Notez que le texte fourni entre guillemets est libre.
- **git status** :(optionnel) pour constater que l'espace de staging a été réinitialisé et qu'aucun fichier n'a été modifié depuis votre commit
- **git push** : pour uploader votre commit sur votre dépôt distant, hébergé sur le serveur GitLab.

Remarque : Ses commandes à exécuter lorsque vous voulez créer un nouveau fichier dans le projet

01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Manipulation des dépôts avec Gitlab



Création dépôts sur Gitlab

4. Le fichier **README.md** devrait maintenant être visible depuis la page web du dépôt, sur **GitLab** :

The screenshot shows the GitLab interface for a repository. At the top, there are navigation elements: a dropdown menu for branches (currently showing 'main'), the repository name 'fsdev /', a '+' button for creating new branches, and buttons for 'Find file', 'Web IDE', a download icon, and 'Clone'. Below this is a commit summary: 'ajout du fichier README.md' by 'mohamed goumih' 1 minute ago, with a commit hash '5bcf9fa4' and a refresh icon. A row of buttons offers actions: 'README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Set up CI/CD'. A 'Configure Integrations' button is also present. A table lists the files in the repository:

Name	Last commit	Last update
README.md	ajout du fichier README.md	1 minute ago

Below the table, the content of the selected 'README.md' file is shown as 'Bonjour'. A blue arrow points from the commit message 'ajout du fichier README.md' to the text 'Bonjour' in the file content.

01-Manipuler les outils de gestion de versions



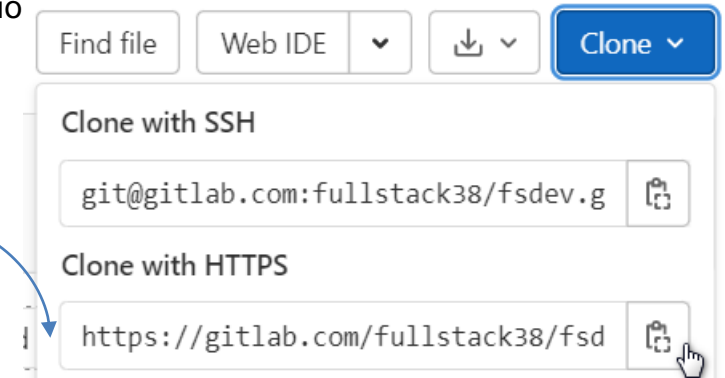
(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Le fork avec Gitlab

- **Un fork** : est une copie d'un dépôt. Ceci est utile lorsque vous souhaitez contribuer au projet de quelqu'un d'autre ou démarrer votre propre projet basé sur le sien.
- **fork** n'est pas une commande dans Git, mais quelque chose d'offert dans GitLab et d'autres références de version

- Commençons par nous connecter à notre compte GitLab et **forkons (copions)** notre référentiel du projet :
- Ouvrez votre bash Git et clonez le dépôt : **git clone url.git dossier**



```
mg17@DESKTOP-UIMMTE1 MINGW64 ~  
$ git clone https://gitlab.com/fullstack38/fsdev.git  
cloning into 'fsdev'...
```

01-Manipuler les outils de gestion de versions



(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Le fork avec Gitlab

- Jetez un œil à votre système de fichiers et vous verrez un nouveau répertoire nommé d'après le projet cl
- Accédez au nouveau répertoire : avec **la commande cd**
- vérifiez le journal pour confirmer que nous avons les données complètes du référentiel :avec **git status** et **git log**

```
Videos/  
"Voisinage d'impression"@  
'Voisinage réseau'@  
fsdev/  
ntuser.dat.LOG1  
ntuser.dat.LOG2
```

```
mg17@DESKTOP-UIMMTE1 MINGW64 ~  
$ cd fsdev/  
  
mg17@DESKTOP-UIMMTE1 MINGW64 ~/fsdev (main)  
$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
nothing to commit, working tree clean  
  
mg17@DESKTOP-UIMMTE1 MINGW64 ~/fsdev (main)  
$ git log
```

- Nous avons maintenant une copie complète du référentiel d'origine, mais nous ne sommes pas autorisés à modifier l'origine.

01-Manipuler les outils de gestion de versions



(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Le fork avec Gitlab

- Nous voyons que l'origine est configurée sur le référentiel d'origine fsdev de note exemple : **git remote**
- nous souhaitons également ajouter notre propre fork :
- Tout d'abord, nous renommons l'origine: **git remote rename origin nouveaunom** puis **git remote -v**

```
$ git remote -v
origin https://gitlab.com/fullstack38/fsdev.git (fetch)
origin https://gitlab.com/fullstack38/fsdev.git (push)
```

```
$ git remote rename origin fullstack
Renaming remote references: 100% (3/3), done.
```

Origine modifié de origine à fullstack

```
$ git remote -v
fullstack https://gitlab.com/fullstack38/fsdev.git (fetch)
fullstack https://gitlab.com/fullstack38/fsdev.git (push)
```

- Récupérez ensuite l'URL de notre propre fork de gitlab : <https://gitlab.com/fullstack38/fsdev.git> (notre exemple)
- Et ajoutez ça comme origine : **git remote add origin url.git** puis **git remote -v**

```
$ git remote add origin https://gitlab.com/fullstack38/fsdev.git
mg17@DESKTOP-UIMMTE1 MINGW64 ~/fsdev (main)
$ git remote -v
fullstack https://gitlab.com/fullstack38/fsdev.git (fetch)
fullstack https://gitlab.com/fullstack38/fsdev.git (push)
origin https://gitlab.com/fullstack38/fsdev.git (fetch)
origin https://gitlab.com/fullstack38/fsdev.git (push)
```

- Maintenant nous avons deux **remotes**:
 - **origin** : notre propre fork, où nous avons un accès en lecture et en écriture en amont
 - **Fullstack** : où nous avons un accès en lecture seule

01-Manipuler les outils de gestion de versions

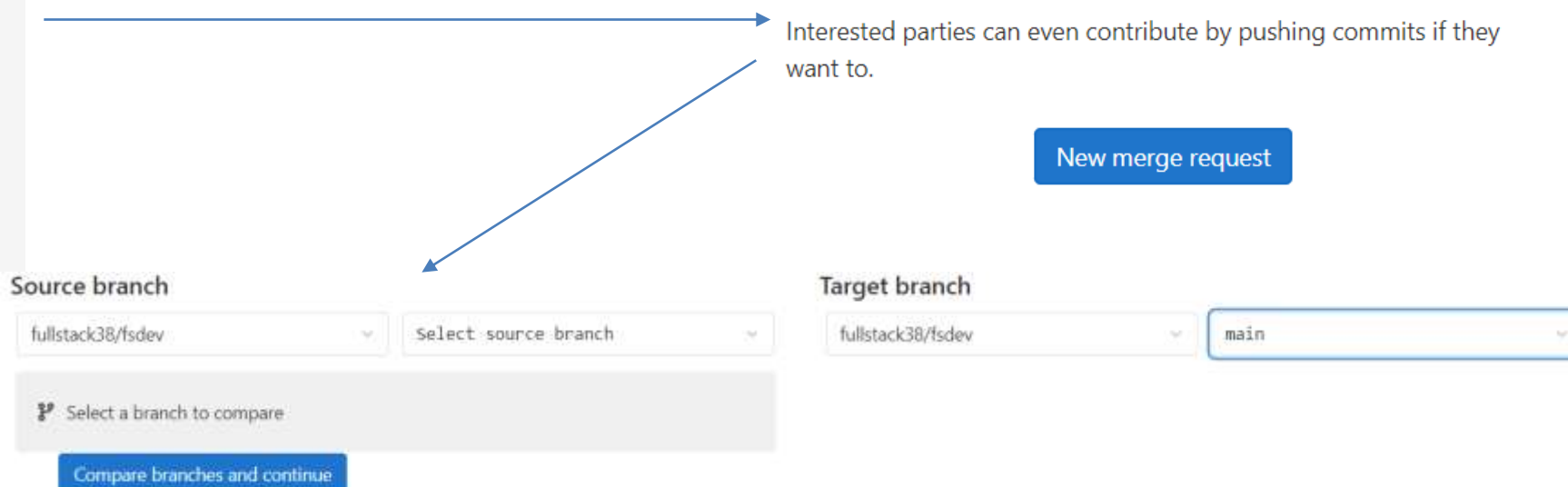
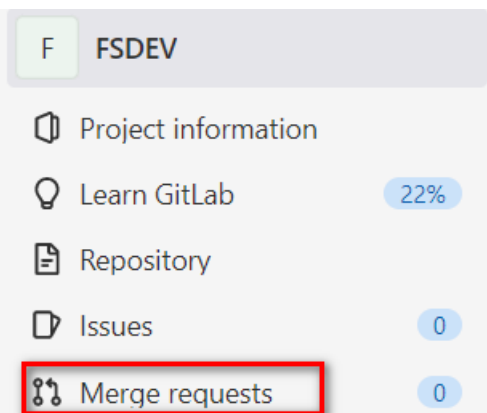


(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Merge Request avec Gitlab

- **Merge Request** (Les demandes de fusion) : (**Pull Request** en github): sont la façon dont vous vérifiez les modifications du code source dans une branche. Lorsque vous ouvrez une demande de fusion, vous pouvez visualiser et collaborer sur les changements de code avant la fusion.
- Vous pouvez créer une Merge Request à partir de la liste des demandes de fusion. Dans la barre supérieure, sélectionnez **Menu > Projets et recherchez votre projet**:
 1. Dans le menu de gauche, sélectionnez **Merge Requests**.
 2. puis, sélectionnez **new merge request**.
 3. Sélectionnez une branche source et cible, puis Comparez les branches et continuez.
 4. Remplissez les champs et sélectionnez **create merge request**.



Merge requests are a place to propose changes you've made to a project and discuss those changes with others

Interested parties can even contribute by pushing commits if they want to.

01-Manipuler les outils de gestion de versions



(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Merge Request avec Git/gitlab

- Vous pouvez créer une Merge Request en exécutant des commandes Git sur votre ordinateur local.

1. Créer une branche:

```
git checkout -b my-new-branch
```

2. Créer ou modifier vos fichiers, puis les mettre dans le stage et les valider :

```
git add .
```

```
git commit -m "message de validation"
```

3. Poussez votre branche vers GitLab :

```
git push origin my-new-branch
```

4. GitLab vous invite avec un lien direct pour créer une demande de fusion :

[Copiez le lien et collez-le dans votre navigateur.](#)

01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Manipulation des dépôts avec Gitlab

Collaborer sur un dépôt Gitlab

- Par défaut seul le créateur d'un dépôt peut y ajouter des **commits**.
- Si un autre développeur souhaite contribuer au dépôt, il existe deux manières de procéder:
 - **forker** le dépôt puis proposer une contribution à l'aide d'un **pull request(merge request)**;
 - ou obtenir la permission d'ajouter des **commits** directement dans le dépôt.

Etapes pour ajouter un développeur dans une équipe Gitlab:

1. Depuis la barre latérale de la page du dépôt, cliquer sur "**Paramètres**",
2. Cliquer sur "Membres",
3. Taper le ou les noms d'utilisateurs (ou adresse email) du/des développeurs à ajouter,
4. Sélectionner le rôle (ou niveau de permissions) à donner à ce(s) développeur(s),
5. Vérifier que le(s) développeur(s) est/sont bien capables d'ajouter et pusher un commit dans la branche de travail du dépôt (ex: master).

Note: pour qu'un développeur aie le droit de pusher des **commits** dans votre dépôt, il faut lui donner le rôle de "**Maintainer**". Le rôle "**Developer**" ne suffit pas.

Etapes pour ajouter un commit dans le dépôt d'un autre développeur

1. Créer un clone pour importer le dépôt de l'autre stagiaire
2. S'assurer qu'on a bien les dernières mises à jour (git pull)
3. Créer un nouveau fichier dans le dépôt local, puis l'ajouter à l'index (git add)
4. Créer un commit contenant ce fichier (git commit) puis l'envoyer sur le dépôt de l'autre stagiaire (git push)
5. Dans l'interface web de GitLab, aller sur le projet de votre camarade, puis cliquez sur "commits" pour vérifier que votre commit apparaît bien dans la liste.



CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. Présentation de Git
3. Présentation de Gitlab
4. Manipulation des dépôts avec Gitlab
- 5. Gestion des conflits de fusion avec Git/GitLab**
6. Comparaison Github vs Gitlab



01-Manipuler les outils de gestion de versions

(Git/Gitlab) :

Gestion conflits avec Gitlab

Definition d'un conflit de fusion

Un conflit de fusion intervient lorsque l'on tente de fusionner deux branches qui modifient la même partie d'un même fichier. Dans ce cas, git va intégrer les deux versions dans le même fichier puis laisser le développeur décider du contenu final de cette partie.

Etapas à suivre pour causer un conflit de fusion :

1. Créer un nouveau dépôt sur **GitLab**
2. Cloner ce dépôt localement avec `git clone`
3. Dans le répertoire du dépôt local, créer un fichier **README.md** contenant un texte
4. Créer un commit initial sur la branche master et l'envoyer sur **GitLab** avec `git add`, `git commit` puis `git push`
5. Créer une branche `branche1` à partir de master, avec `git checkout -b`
6. Dans le **README.md** de cette branche, modifier le premier texte, créer un commit, puis envoyer les modifications de cette branche sur **GitLab**
7. Revenir à la branche master avec `git checkout`
8. Créer une branche `branche2` à partir de master (comme dans l'étape 5)
9. Dans le **README.md** de cette branche, modifier le texte différemment de celui saisi à l'étape 6, créer un commit, puis envoyer les modifications de cette branche sur **GitLab**
10. Revenir à la **branche master**
11. Fusionner `branche1` dans master, avec `git merge`
12. Fusionner `branche2` dans master
13. Taper `git status` pour voir le message du conflit

Etapas à suivre pour résoudre un conflit

Pour résoudre ce conflit, il va falloir:

1. ouvrir le fichier **README.md** dans son éditeur de code,
2. constater comment git représente le conflit, et la source de chaque version,
3. éditer le fichier pour ne conserver que la version finale souhaitée,
4. puis créer un commit.
5. Après la résolution du conflit de fusion, taper `git status` pour voir le résultat
6. supprimer les branches `branche1` et `branche2`, dans votre dépôt local, et dans le dépôt distant associé (sur **GitLab**).

CHAPITRE 1

Manipuler les outils de gestion de versions (Git/Gitlab)

1. Intérêt de la gestion de version et présentation des outils existants de gestion de versions
2. Présentation de Git
3. Présentation de Gitlab
4. Manipulation des dépôts avec Gitlab
5. Gestion des conflits de fusion avec Git/GitLab
- 6. Comparaison Github vs Gitlab**



01-Manipuler les outils de gestion de versions

Git/Gitlab :

Comparaison Github et Gitlab

Quelques Fonctionnalités semblables entre gitlab et github

GitHub et **GitLab** reposent tous les deux sur **Git** et ses commandes, la **migration** d'une plateforme à l'autre est possible sans grandes difficultés. L'importation des repositories, **wikis**, **pull requests** et **issues** est facile en règle générale. En revanche, il existe quelques différences entre **GitHub** et **GitLab**, comme l'illustre le tableau ci-dessous :

Fonctionnalité	GitLab	GitHub
Git	✓	✓
Version auto-hébergée	✓	✓ (avec plan d'entreprise)
Intégration et livraison continues	✓	✓ (avec une application tierce)
Documentation basée sur le Wiki	✓	✓
Aperçu des modifications du code	✓	✓
Suivi des problèmes	✓	✓
Examen du code	✓	✓
Cessionnaires d'émissions multiples	✓ (Plan payant)	✓ (uniquement le dépôt public sur le plan gratuit)
Conseils de gestion de projet	✓	✓
Discussions d'équipe	✓	✓
Suivi du temps	✓	✓ (Avec App)
Outils de sécurité et de conformité	✓	✓
Test de performance de charge	✓ (Plan payant)	✓ (Avec App)
Test de performance des navigateurs	✓ (Plan payant)	✓ (Avec App)
Itérations et planification des sprints (y compris Burndown Chart)	✓ (Plan payant)	✓ (Avec App)
Dépendances des problèmes	✓ (Plan payant)	✓

01-Manipuler les outils de gestion de versions

Git/Gitlab :

Comparaison Github et Gitlab

Quelques différences majeures entre gitlab et github

GitHub	GitLab
Les issues peuvent être suivies dans plusieurs repositories	Les issues ne peuvent pas être suivies dans plusieurs repositories
Repositories privés payants	Repositories privés gratuits
Pas d'hébergement gratuit sur un serveur privé	Hébergement gratuit possible sur un serveur privé
Intégration continue uniquement avec des outils tiers (Travis CI, CircleCI, etc.)	Intégration continue gratuite incluse
Aucune plateforme de déploiement intégrée	Déploiement logiciel avec Kubernetes
Suivi détaillé des commentaires	Pas de suivi des commentaires
Impossible d'exporter les issues au format CSV	Exportation possible des issues au format CSV par e-mail
Tableau de bord personnel pour suivre les issues et pull requests	Tableau de bord analytique pour planifier et surveiller le projet
Interface graphique un peu compliqué	Interface graphique clair et facile

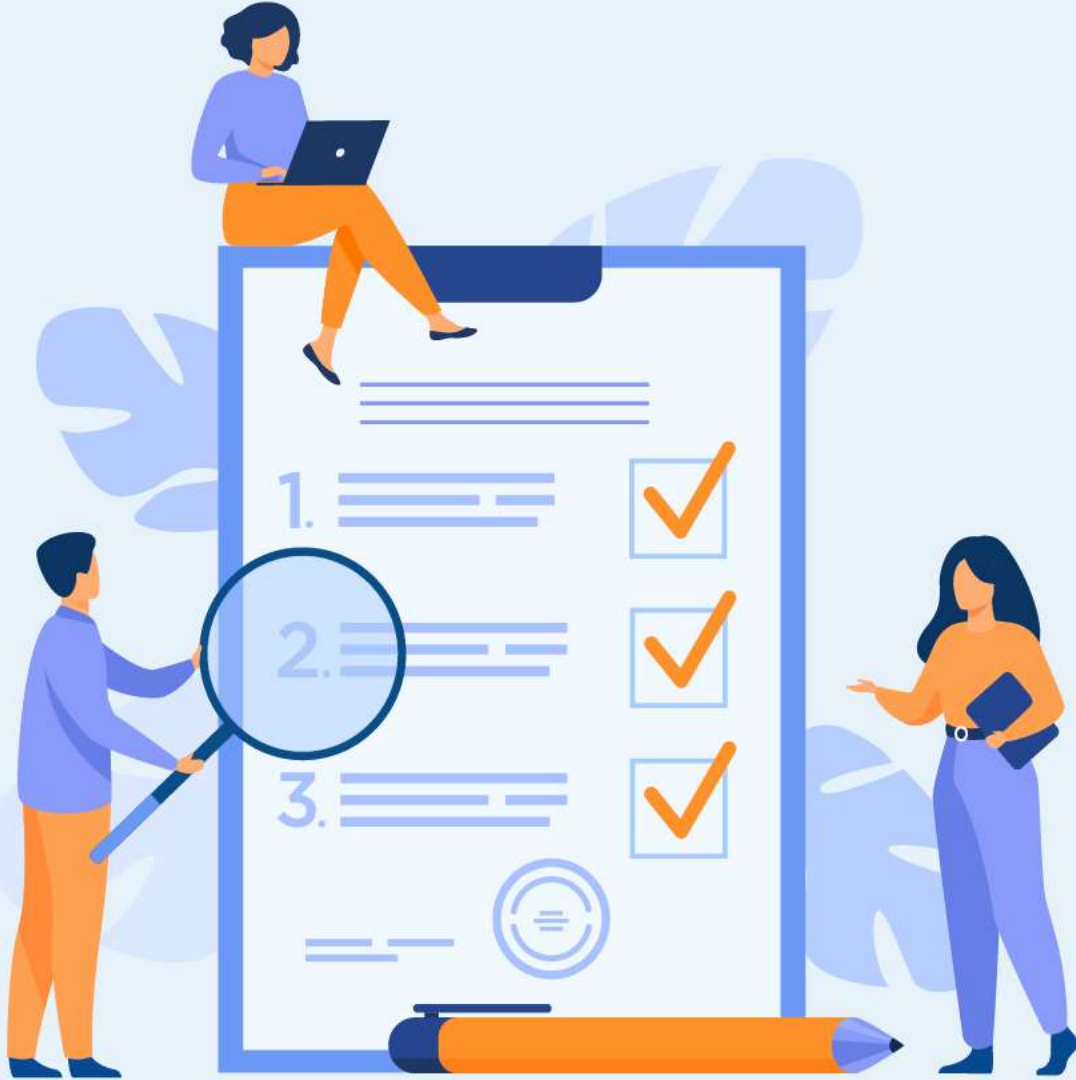
CHAPITRE 2

Manipuler l'outil de mesure de la qualité du code (SonarQube)

Ce que vous allez apprendre dans ce chapitre :

- Notions des métriques de la qualité du code
- Présentation des outils existants de mesure de la qualité du code
- Présentation et installation de SonarQube
- Analyse de code source avec SonarQube à travers la génération des rapports

15
heures



CHAPITRE E 2 Manipuler l'outil de mesure de la qualité du code (SonarQube)

1. **Notions des métriques de la qualité du code**
2. Présentation de SonarQube
3. Installation de SonarQube
4. Intégration de SonarQube avec les outils ALM et Configuration
5. Analyse de code source avec SonarQube



02-Manipuler l'outil de mesure de la qualité du code(SonarQube) :

Notions de métriques

Définitions

- **Une métrique** : est une caractéristique (ou une propriété) d'une application.
- **Métrique logicielle** : mesure d'une propriété d'un logiciel (par exemple le nombre de lignes de codes).
- **Exemples de métriques** :
 - Lignes de codes
 - Nombre de méthodes par classe
 - Couplage afférent/efférent
 - Niveau d'abstraction
 - Indice Instabilité
 - Complexité cyclomatique(compter le nombre de points de décision (if, case, while, ...)).
- **Pas de métrique "absolue"** : la pertinence de chaque métrique dépend du projet et surtout de l'interprétation qui en est faite.
- **la qualité du code** : aussi appelée qualité structurelle du logiciel, est différente de la qualité d'un logiciel, aussi appelée qualité fonctionnelle d'un logiciel, la première concerne la manière avec laquelle une fonctionnalité est implémentée et la seconde le résultat final.
- **La qualité du code** : d'un logiciel est donc la manière avec laquelle ces fonctionnalités sont implémentées et inclut entre autres la robustesse, la maintenabilité, la lisibilité ou encore l'évolutivité.
- **la mesure de la qualité du code** peut en partie être mesurée automatiquement par des outils d'analyse statique du code source. Ces outils sont disponibles pour la plupart des langages et proposent de **nombreuses métriques** et conventions standards.
- **La qualité du code** peut être également évaluée par d'autres critères comme :
 - L'architecture qui inclut la maintenabilité, l'évolutivité, la performance et la pertinence
 - La documentation,
 - La portabilité et la sécurité.
 - La fiabilité
 - Le nombre de bugs connus.
- Tous ces critères sont fortement dépendants du contexte comme la spécificité du logiciel ou programme au niveau fonctionnel, son avenir mais aussi d'autres facteurs importants comme le marché et ses utilisateurs.

02-Manipuler l'outil de mesure de la qualité du code

(SonarQube) :

Exemples d'outils



Exemples d'outil pour analyser le code:

Cobertura

Crap4J

PMD

FindBugs

Eclipse Metrics

Jdepend

Review Board

Crucible

Raxis

Technologies RIPS

PVS-Studio

Kiuwan

CodeScene

Expert visuel

Veracode

CodeSonar

SonarQube

CHAPITRE E 2 Manipuler l'outil de mesure de la qualité du code (SonarQube)

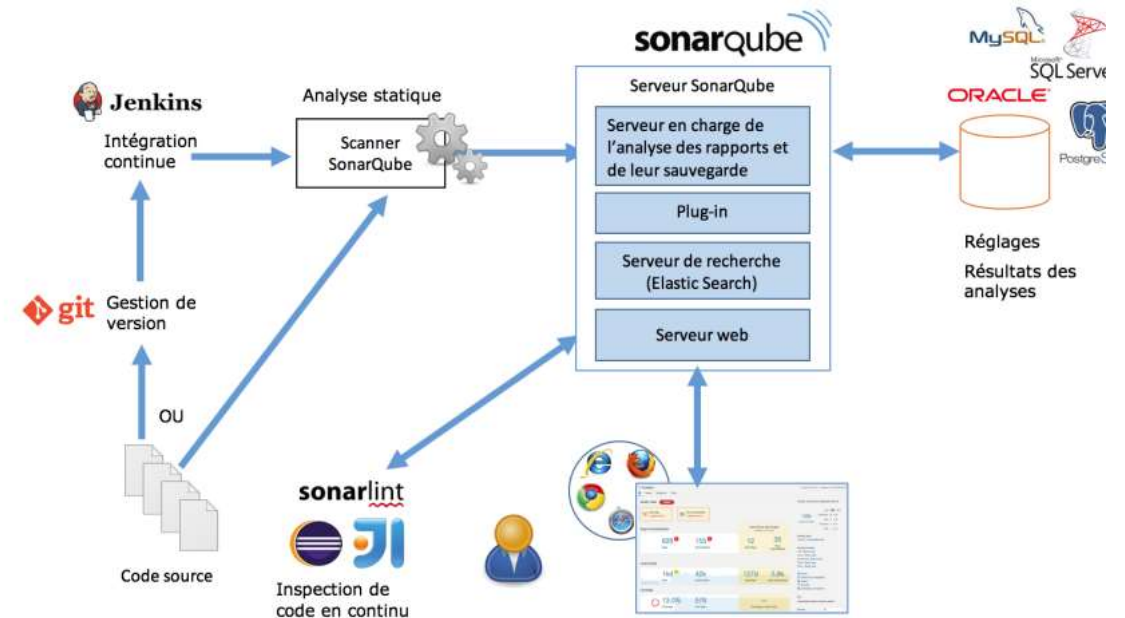
1. Notions des métriques de la qualité du code
- 2. Présentation de SonarQube**
3. Installation de SonarQube
4. Intégration de SonarQube avec les outils ALM et Configuration
5. Analyse de code source avec SonarQube



02-Manipuler l'outil de mesure de la qualité du code (SonarQube) : Présentation de SonarQube

C'est quoi l'outil SonarQube ?

- **SonarQube** est un logiciel open source de mesure de la qualité du code source de projets de développement.
- Il permet d'obtenir des informations sur la qualité au niveau du projet, du fichier ou d'un module et donne des indications sur chaque problème de qualité détecté et le temps de remédiation. Son périmètre est le code source, le design ainsi que les tests unitaires, il supporte plus d'une vingtaine de langages de programmation, dont **C/C++,C#,Java, Python, PHP, JavaScript..**
- D'un point de vue architectural, Sonar est composé de plusieurs couches :
 - Un exécuteur dont le but sera de lancer un certain nombre d'outils d'analyse, et d'en agréger les résultats ;
 - Une base de données, qui stocke et historise les informations sur les projets surveillés par Sonar ;
 - Le serveur web qui permet la navigation et la consultation des analyses réalisées sur les projets ;



Architecture de SonarQube source :M. Contensin - SonarQube

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) : Présentation de SonarQube

Les types de métriques avec SonarQube?

- **SonarQube** génère un rapport consultable via un navigateur sur :
 - Densité des commentaires
 - Taux de couverture des tests unitaires
 - des conventions de nommage
 - Respect des règles de codage et des bonnes pratiques
 - Détection de bogues
 - Détection de code mort
 - Détection de code dupliqué
 - Complexité du code (complexité cyclomatique, complexité cognitive)
 - Score de maintenabilité, fiabilité et sécurité
 - Dette technique (estimation du temps nécessaire pour fixer tous les problèmes détectés)
 - Sonar couvre les 7 axes de la qualité du code : **architecture & design , documentation ,respect des standards de codage ,non duplication du code , tests unitaires ,complexité ,bogues potentiels.**
- Il est possible d'automatiser l'analyse avec un serveur d'intégration continue (**ex. Jenkins, Travis**).
- **SonarQube** permet l'inspection de code en continu (**intégration dans les IDE, vérification de la qualité depuis la dernière version ou la dernière analyse, notifications par email**), ce qui permet de détecter les problèmes dès leur introduction dans le code, avant que le coût de remédiation soit élevé.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) : Présentation de SonarQube

Tableau du bord avec SonarQube?

- **SonarQube** classe les défauts logiciels selon **3** catégories :
 - Les **bugs** : anomalies évidentes du code. Ils impactent la **fiabilité** (reliability) de l'application.
 - Les **vulnérabilités** : faiblesses du code pouvant nuire au système. Elles impactent la **sécurité** de l'application.
 - Les **code smells** : anti-patterns (ou anti-patterns). Ils impactent la **maintenabilité** de l'application ,en général les défauts pratiques dans le code d'un programme comme code dupliqué , répété , commentaires non nécessaires ...
- **SonarQube** affiche un tableau de bord résumé à l'exécution d'un projet montrant les scores de maintenabilité, fiabilité et sécurité, le taux de couverture des tests, le taux de duplications, la taille du projet ainsi que les langages des sources.



Exemple tableau du bord

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) : Exemples de métriques avec Sonarqube

1. Scores de maintenabilité, fiabilité et sécurité (MFS):

Score	Fiabilité	Sécurité	Maintenabilité
A	0 bug	0 vulnérabilité	$0,00 \leq \text{ratio dette technique} \leq 0,05$
B	Au moins 1 bug mineur	Au moins 1 vulnérabilité mineure	$0,06 \leq \text{ratio dette technique} \leq 0,1$
C	Au moins 1 bug majeur	Au moins 1 vulnérabilité majeure	$0,11 \leq \text{ratio dette technique} \leq 0,20$
D	Au moins 1 bug critique	Au moins 1 vulnérabilité minecritique	$0,21 \leq \text{ratio dette technique} \leq 0,50$
E	Au moins 1 bug bloquant	Au moins 1 vulnérabilité bloquante	$0,51 \leq \text{ratio dette technique} \leq 1$

Tableau montrant comment calculer les scores de MFS

(*) **Le ratio de la dette technique** est le ratio entre le coût pour remédier aux problèmes de type code smell et le coût de développement de l'application. La formule de calcul est la suivante : **coût total de remédiation des issues / (coût pour développer une ligne de code x nombre de lignes de code)**. L'idée est de déterminer si réécrire l'application est plus rentable que corriger tous les problèmes. Lorsque le ratio est trop grand, il est préférable de réécrire l'application de zéro plutôt que d'essayer de réduire la dette en corrigeant les problèmes. Le nombre de minutes pour remédier à chaque issue est stocké dans la base de données.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Exemples de métriques avec Sonarqube

2. Mesures pour la documentation

- Quelques variables utilisés :
 - **Comment lines** = nombre de lignes contenant des commentaires de documentation ou du code commenté, les lignes de commentaires non significatives sont ignorées (ligne vide ou avec le caractère spécial *)
 - **Comments (%)** = densité de commentaire calculée par la formule :
$$\text{Comment lines} / (\text{Lines of code} + \text{Comment lines}) * 100$$
Densité de 50% = autant de lignes de code que de commentaires
Densité de 100% = le fichier ne contient que des commentaires
 - **Commented-out LOC** = nombre de lignes de code commentées.

3. Mesures pour la duplication du code :

- Les variables utilisés pour le calcul :
 - **Duplicated blocks** = nombre de blocs de lignes dupliqués. Un bloc de code est considéré comme dupliqué s'il y a au moins 100 tokens successifs dupliqués sur 10 lignes de code (PHP, JavaScript)
 - **Duplicated files** = nombre de fichiers impliqués dans des duplications
 - **Duplicated lines** = nombre de lignes impliquées dans des duplications
 - **Duplicated lines (%)** = densité de duplication calculée par la formule : $\text{Duplicated lines} / \text{Lines} * 100$

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Exemples de métriques avec Sonarqube

4. Mesures pour les tests :

- Les variables utilisés pour faire cette mesure :
 - **Nombre de tests unitaires** (unit tests)
 - **nombre de tests unitaires** qui ont échoué (unit test errors)
 - **nombre d'exception** lors de l'exécution (unit test failures)
 - **nombre de lignes non couvertes** (uncovered lines)
 - **nombre de conditions non couvertes** (uncovered conditions)
 - **couverture** : calcul = $(CT + CF + LC) / (2*B + EL)$

CT : condition évaluées à true au moins une fois

CF : conditions évaluées à false au moins une fois

LC : lignes couvertes

B : nombre total de conditions

EL : nombre total de lignes exécutables

5. Mesures de complexité :

- Les variables utilisés pour faire cette mesure :
 - **Complexity** = Complexité totale du projet
 - **Complexity / method** = Complexité moyenne par fonction
 - **Complexity / file** = Complexité moyenne par fichier
 - **Complexity / class** = Complexité moyenne par classe
- Le calcul de la complexité d'une méthode se base sur la complexité cyclomatique. Elle est minimum de 1 (1 pour la méthode). Elle est incrémentée de 1 pour chaque if, case, for, while, throw, catch, return (sauf s'il est en dernière instruction de la fonction), &&, ||, opérateur conditionnel ternaire

CHAPITRE E 2 Manipuler l'outil de mesure de la qualité du code (SonarQube)

1. Notions des métriques de la qualité du code
2. Présentation de SonarQube
- 3. Installation de SonarQube**
4. Intégration de SonarQube avec les outils ALM et Configuration
5. Analyse de code source avec SonarQube



02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Installation de sonarqube

Installation de SonarQube :

1. Prérequis:

- **Prérequis côté serveur** : installer JDK sur <https://www.oracle.com/java/technologies/downloads/>
- **Prérequis côté client** : navigateur web récent
- **JavaScript activé.**

2. Télécharger **Sonarqube community** sur: <https://www.sonarqube.org/downloads/>

3. Décompressez le fichier dans n'importe quel dossier

4. Copier le lien de l'emplacement de JDK et aller sur le dossier **SonarQube/config/wrapper** et mettez le lien au 4ème ligne à la place de java et changer \ à / :

```
C:\Program Files\Java\jdk-17.0.1\bin
```

```
# Path to JVM executable. By default it must be available in PATH.  
# Can be an absolute path, for example:  
#wrapper.java.command=/path/to/my/jdk/bin/java  
wrapper.java.command=C:/Program Files/Java/jdk-17.0.1/bin/java
```

CHAPITRE E 2 Manipuler l'outil de mesure de la qualité du code (SonarQube)

1. Notions des métriques de la qualité du code
2. Présentation de SonarQube
3. Installation de SonarQube
- 4. Intégration de SonarQube avec les outils ALM et Configuration**
5. Analyse de code source avec SonarQube

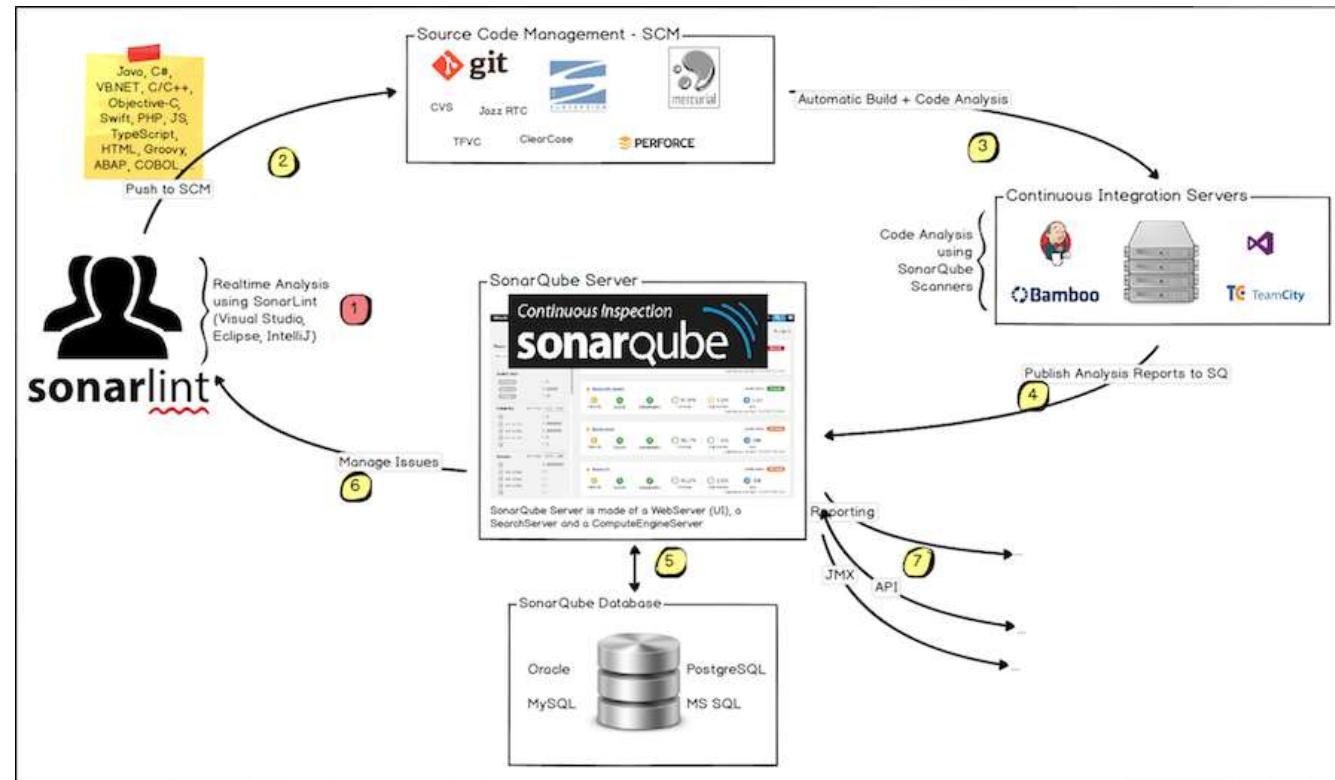


02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Intégration de SonarQube avec les outils ALM

Intégration de SonarQube avec les outils ALM :

Le schéma suivant montre comment **SonarQube** s'intègre à d'autres outils **ALM**¹ dans un grand projet d'équipe e où les différents composants de **SonarQube** sont utilisés



Source : <https://docs.sonarqube.org/>

¹ L'**application life cycle management** ou **ALM** : est le processus global de gestion du cycle de vie d'un logiciel.

C'est l'ensemble des moyens nécessaires au développement ou à la maintenance d'une application. Cela concerne les activités d'ingénierie logicielle comme les activités de gestion de projet.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Intégration de SonarQube avec les outils ALM

Intégration avec Application Life cycle management:

- Les rôles dans le schéma selon les numéros :

1. Les développeurs codent dans leurs IDE et utilisent **SonarLint** pour exécuter une analyse locale.
2. Les développeurs poussent leur code dans leur **SCM** préféré : **github, gitlab...**
3. Le **serveur d'intégration continue** déclenche une construction automatique et l'exécution du **SonarScanner** requis pour exécuter l'analyse **SonarQube**.
4. Le rapport d'analyse est envoyé au serveur **SonarQube** pour traitement.
5. **SonarQube** Server traite et stocke les résultats du rapport d'analyse dans la base de données **SonarQube** et affiche les résultats dans l'interface utilisateur.
6. Les développeurs examinent, commentent, contestent leurs problèmes pour gérer et réduire leur dette technique via l'interface utilisateur **SonarQube**.
7. Les managers reçoivent les rapports de l'analyse. Les opérateurs utilisent des API pour automatiser la configuration et extraire les données de **SonarQube**. Les opérateurs utilisent JMX pour surveiller le serveur **SonarQube**.

SonarLint est une extension IDE gratuite et open source qui identifie et vous aide à résoudre les problèmes de qualité et de sécurité pendant que vous codez. Comme un correcteur orthographique, SonarLint corrige les défauts et fournit des commentaires en temps réel et des conseils de correction clairs pour fournir un code propre dès le départ.
Il supporte presque tous les IDE comme vscode ,eclipse ,JetBrains..

- Quelques règles avant la configuration de **SonarQube** :

- La plate-forme **SonarQube** ne peut pas avoir plus d'un serveur **SonarQube** et plus d'une base de données .
- Pour des performances optimales, chaque composant (serveur, base de données, analyseurs) doit être installé sur une machine distincte et la machine serveur doit être dédiée.
- Les **SonarScanners** évoluent en ajoutant des machines.
- Toutes les machines doivent être synchronisées.
- Le serveur **SonarQube** et la base de données **SonarQube** doivent être situés sur le même réseau.
- Les **SonarScanners** n'ont pas besoin d'être sur le même réseau que le serveur **SonarQube**.
- Il n'y a pas de communication entre les **SonarScanners** et la base de données **SonarQube**.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) : Configuration Sonarqube



Configuration de SonarQube :

1. Dans le dossier bin choisir le dossier relative à votre système d'exploitation

) > sonarqube-9.5.0.56709 > bin

Nom	Modifié le	Type	Taille
jsw-license	10/06/2022 06:14	Dossier de fichiers	
linux-x86-64	10/06/2022 06:14	Dossier de fichiers	
macosx-universal-64	10/06/2022 06:14	Dossier de fichiers	
windows-x86-64	10/06/2022 06:14	Dossier de fichiers	

2. Cliquer sur **starSonar.bat** pour le démarrer :

lib	10/06/2022 06:14	Dossier de fichiers	
StartNTService	10/06/2022 06:14	Fichier de comma...	2 Ko
StartSonar	10/06/2022 06:14	Fichier de comma...	2 Ko
StopNTService	10/06/2022 06:14	Fichier de comma...	2 Ko
wrapper	10/06/2022 06:14	Application	216 Ko

3. Aller dans un navigateur et taper l'url **localhost:9000** pour afficher l'interface admin de **SonarQube**:

4. Connectez avec les identifiants suivants:

- login: **admin**
- password: **admin**

CHAPITRE

E 2

Manipuler l'outil de mesure de la qualité du code (SonarQube)

1. Notions des métriques de la qualité du code
2. Présentation de SonarQube
3. Installation de SonarQube
4. Intégration de SonarQube avec les outils ALM et Configuration
5. **Analyse de code source avec SonarQube**

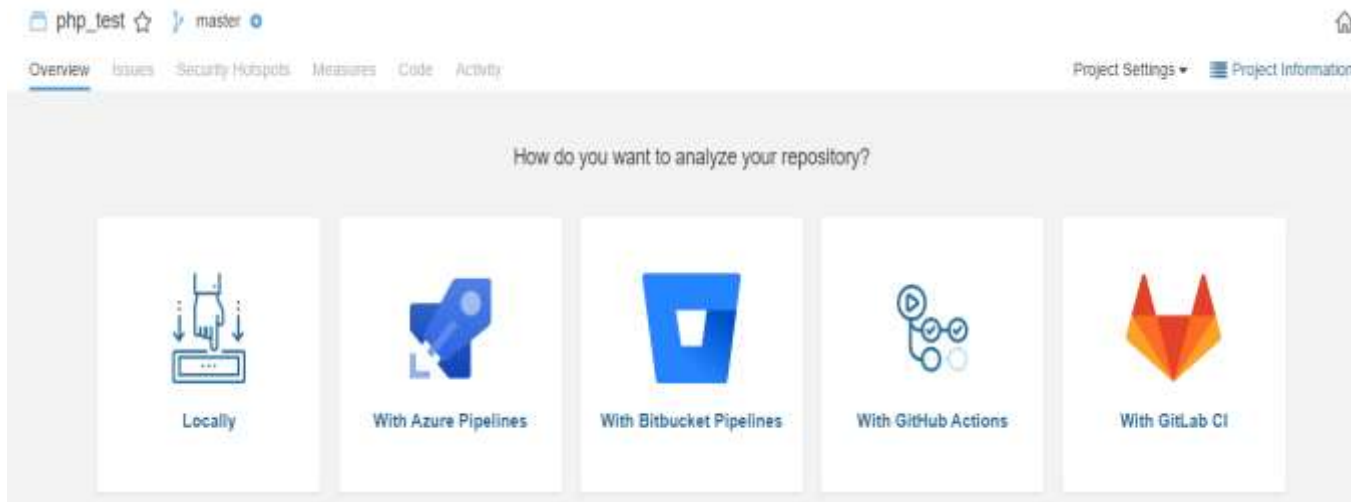


02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

- Maintenant que vous êtes connecté à votre instance **SonarQube** locale, analysons un projet :
 - Cliquez sur le bouton **Créer un nouveau projet** (vous pouvez choisir entre **locally** ou **importer des logiciels des versions**)
 - Donnez à votre projet une **clé de projet** et un **nom complet**



Create a project

All fields marked with * are required

Project display name *

 ✓
Up to 255 characters. Some scanners might override the value you provide.

Project key *

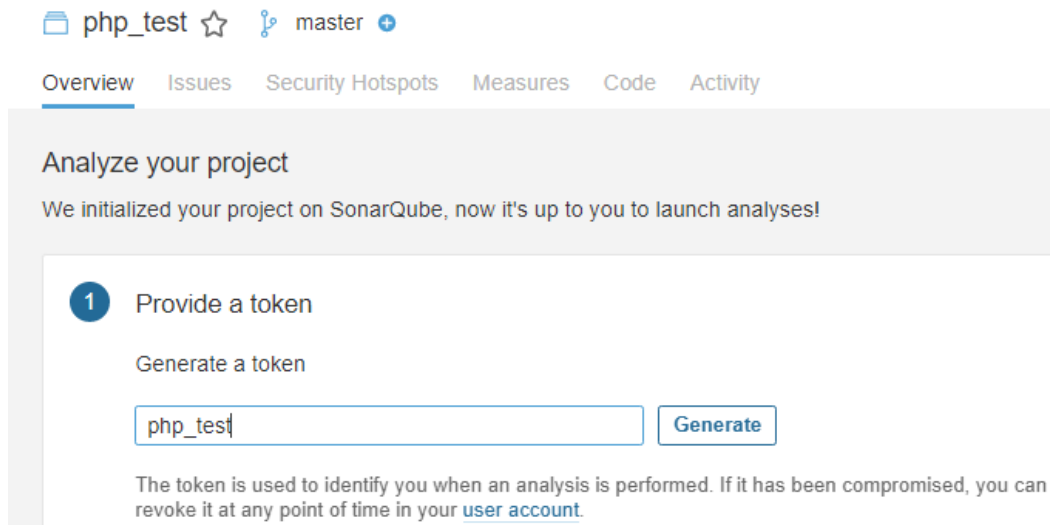
 ✓
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

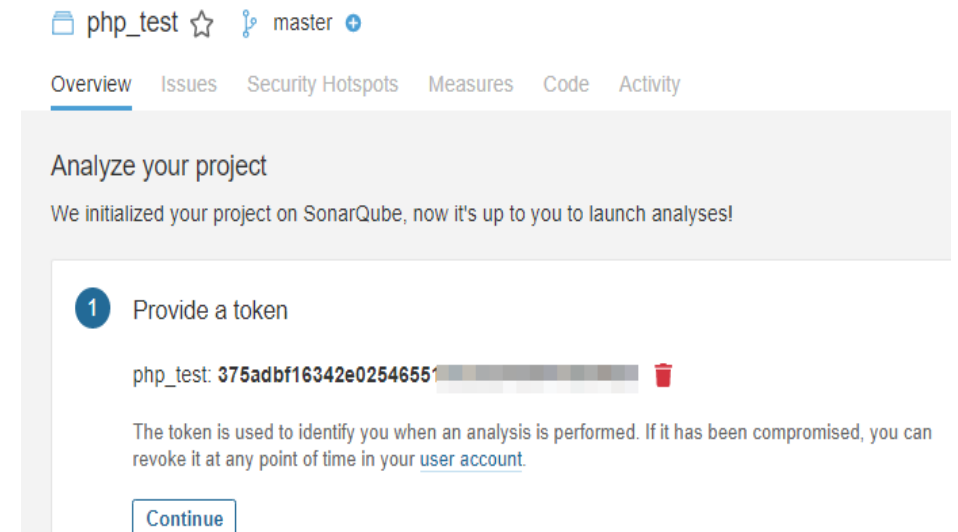
Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

3. Sous **Fournir un jeton(token)**, sélectionnez **Générer un jeton**. Donnez un nom à votre jeton, cliquez sur le bouton **Générer**, puis sur **Continuer**.



The screenshot shows the 'Provide a token' step in the SonarQube interface. At the top, the project name 'php_test' and the user 'master' are visible. Below the navigation tabs (Overview, Issues, Security Hotspots, Measures, Code, Activity), the heading 'Analyze your project' is followed by the text 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. A numbered step '1 Provide a token' is shown. Underneath, there is a sub-heading 'Generate a token' and a text input field containing 'php_test|'. To the right of the input field is a 'Generate' button. Below the input field, a paragraph explains: 'The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#)'.



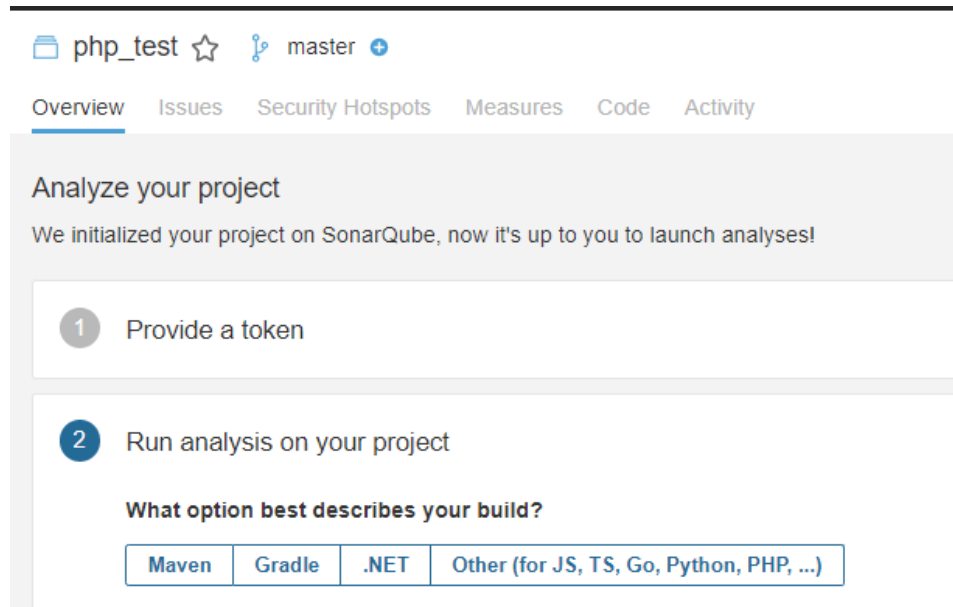
The screenshot shows the 'Provide a token' step in the SonarQube interface, showing the result of the token generation. The project name 'php_test' and the user 'master' are visible at the top. Below the navigation tabs, the heading 'Analyze your project' is followed by the text 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. A numbered step '1 Provide a token' is shown. Below it, the text 'php_test: 375adbf16342e02546551' is displayed next to a red trash icon. A paragraph explains: 'The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#)'. At the bottom, there is a 'Continue' button.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

4.Par la suite ,choisir le langage de votre projet et le système d'exploitation :



php_test ☆ master +

Overview Issues Security Hotspots Measures Code Activity

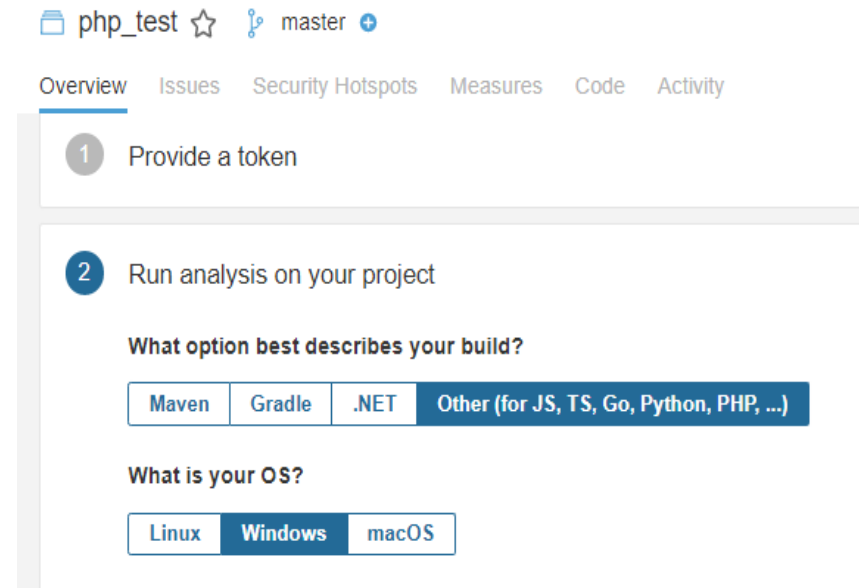
Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

- 1 Provide a token
- 2 Run analysis on your project

What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)



php_test ☆ master +

Overview Issues Security Hotspots Measures Code Activity

- 1 Provide a token
- 2 Run analysis on your project

What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux Windows macOS

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

5. Ensuite, télécharger et installer **SonarScanner**, selon votre **SE** sur : <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner>

SonarScanner

By [SonarSource](#) | GNU LGPL 3 | [Issue Tracker](#)

4.7

[Show more versions](#)

2022-02-22

Ease import of custom certificates with the Docker image, update embedded JRE 11

[Linux 64-bit](#) | [Windows 64-bit](#) | [Mac OS X 64-bit](#) | [Docker](#)

[Any \(Requires a pre-installed JVM\)](#) | [Release notes](#)

The SonarScanner is the scanner to use when there is no specific scanner for your build system.

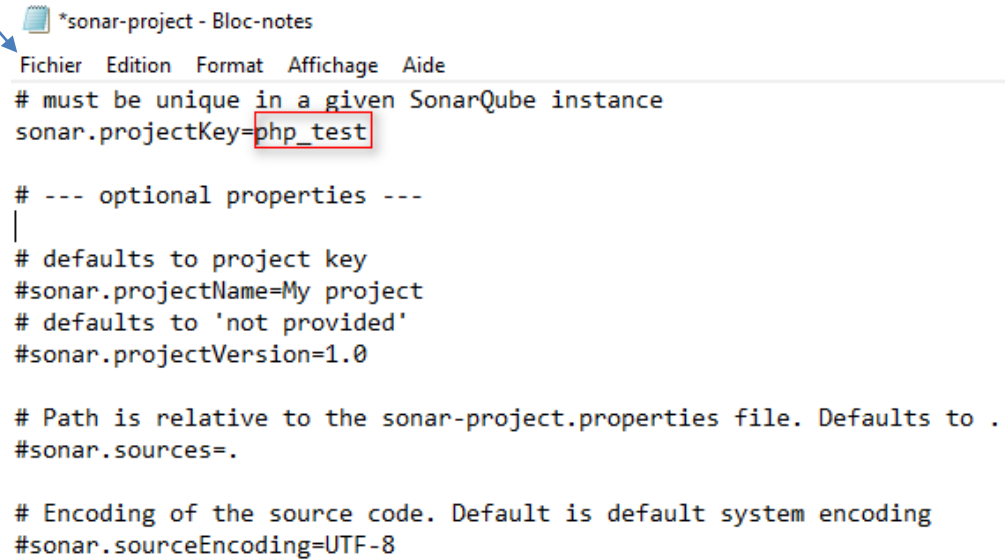
Le SonarScanner est un outil utilisé par **SonarQube** pour scanner votre projet vers **SonarQube** pour que ce dernier puisse lire le langage de programmation utilisé.

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

6. Aller dans le dossier de votre projet et ouvrez le fichier **sonar-project.properties** ,
7. **ouvrez le** et modifier avec le code suivant : **Sonar.projectkey** devrait égal le nom de votre projet



```
*sonar-project - Bloc-notes
Fichier Edition Format Affichage Aide
# must be unique in a given SonarQube instance
sonar.projectKey=php_test

# --- optional properties ---
# defaults to project key
#sonar.projectName=My project
# defaults to 'not provided'
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults to .
#sonar.sources=.

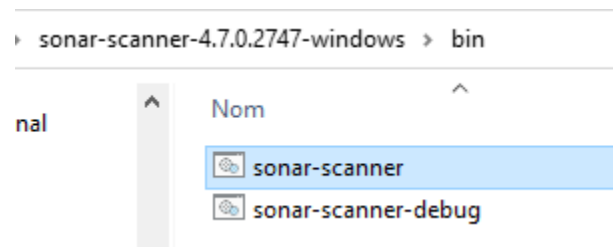
# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

7. Dans le dossier **bin de sonar-scanner** :lancer **sonar-scanner.bat** pour commencer l'analyse du projet



```
PS C:\Users\mg17\Desktop\php_test> sonar-scanner.bat -D"sonar.projectKey=php_test" -D"sonar.sources=. " -
=http://localhost:9000" -D"sonar.login=375adbf16342e02546551aa2b23fd5e40afb862f"
INFO: Scanner configuration file: C:\Users\mg17\Desktop\sonar-scanner-4.7.0.2747-windows\bin\..\conf\sonar
rties
INFO: Project root configuration file: C:\Users\mg17\Desktop\php_test\sonar-project.properties
INFO: SonarScanner 4.7.0.2747
INFO: Java 11.0.14.1 Eclipse Adoptium (64-bit)
INFO: windows 10 10.0 amd64
INFO: User cache: C:\Users\mg17\.sonar\cache
INFO: Scanner configuration file: C:\Users\mg17\Desktop\sonar-scanner-4.7.0.2747-windows\bin\..\conf\sonar
rties
INFO: Project root configuration file: C:\Users\mg17\Desktop\php_test\sonar-project.properties
INFO: Analyzing on SonarQube server 9.0.1
INFO: Default locale: "fr_FR", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=235ms
INFO: Server id: BF41A1F2-AYI1SwmGQCeb5V2Rt3yG
INFO: User cache: C:\Users\mg17\.sonar\cache
INFO: Load/download plugins
```

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

8. A la fin de l'analyse un url s'affiche pour voir les résultats ,copier le et accéder via un navigateur :

```
More about the report processing at http://localhost:9000/api/ce/task?id=AYImYtoQQCeb5V2Rt8oU
Analysis total time: 1:35.634 s
```

Le tableau du bord de l'analyse du code s'affiche:

The screenshot shows the SonarQube interface with the following details:

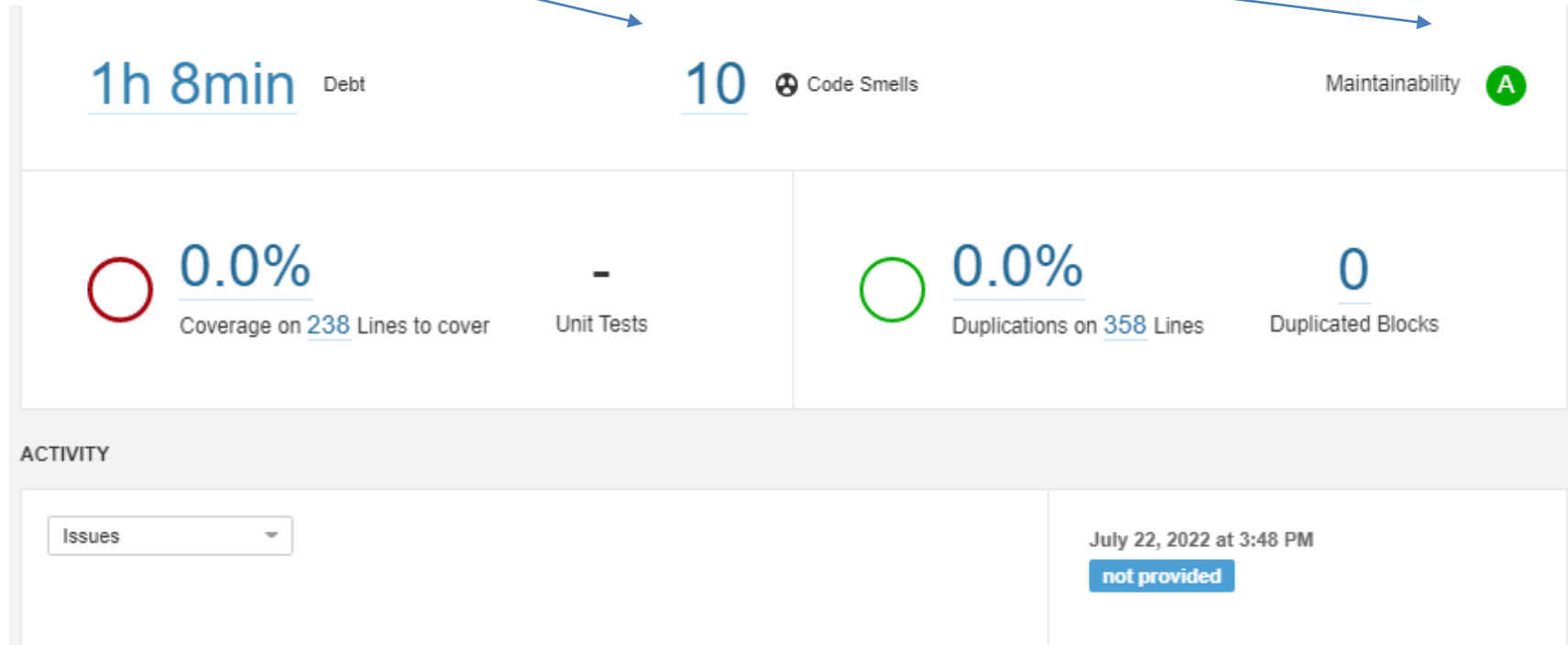
- Navigation:** sonarqube | Projects | Issues | Rules | Quality Profiles | Quality Gates | Administration
- Search:** Search for projects... (with 'A' filter)
- Project List:** Search by project name or key. 1 projects. Perspective: Overall Status. Sort by: Name.
- Project Summary (php_test):**
 - Status: **Passed** (Last analysis: 5 minutes ago)
 - Bugs: 0 (A)
 - Vulnerabilities: 0 (A)
 - Hotspots Reviewed: - (A)
 - Code Smells: 10 (A)
 - Coverage: 0.0% (Red circle)
 - Duplications: 0.0% (Green circle)
 - Lines: 358 (XS PHP)
- Security (Vulnerabilities):**
 - A: 1 (Bar chart)
 - B: 0
 - C: 0
 - D: 0
 - E: 0
- Security Review (Security Hotspots):**
 - A (≥ 80%): 1 (Bar chart)
 - B (70% - 80%): 0
 - C (50% - 70%): 0
 - D (30% - 50%): 0
 - E (< 30%): 0
- Maintainability (Code Smells):**
 - A: 1 (Bar chart)

02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

- On peut accéder à chaque métrique en cliquant sur son nom :
- Les numéros sont le nombre d'erreurs dans le code



02-Manipuler l'outil de mesure de la qualité du code (SonarQube) :

Analyser le code avec SonarQube

Analyser un projet avec SonarQube:

- On peut voir les erreurs dans le code et avec la suggestion de correction:

```
body {
    font-size: 17px;
    font-family: arial;
```

Unexpected missing generic font family Why is this an issue?

Bug Major Open Not assigned 1min effort [Comment](#)

```
function inverse($x) {
    if (!$x) {
        throw new Exception('Division by zero.');
```

Define and throw a dedicated exception instead of using a generic one. Why is this an issue? 16 minutes ago L11 [🔗](#)

Code Smell Major Open Not assigned 20min effort [Comment](#) [🔗](#) cwe, error-handling

```
$colors = [
    1 => 'red',
    2 => 'green',
    3 => 'blue',
];

// echo $colors[1];
```

Remove this commented out code. Why is this an issue? 16 minutes ago L36 [🔗](#)

Code Smell Major Open Not assigned 5min effort [Comment](#) [🔗](#) unused

php2 /index.html [🔗](#) [See all is...](#)

```
- <!DOCTYPE html>
<html>
```

Add "lang" and/or "xml:lang" attributes to this "<html>" element Why is this an issue? 6 minutes ago L2 [🔗](#)

Bug Major Open Not assigned [Comment](#) [🔗](#) accessibility, wcag2-a



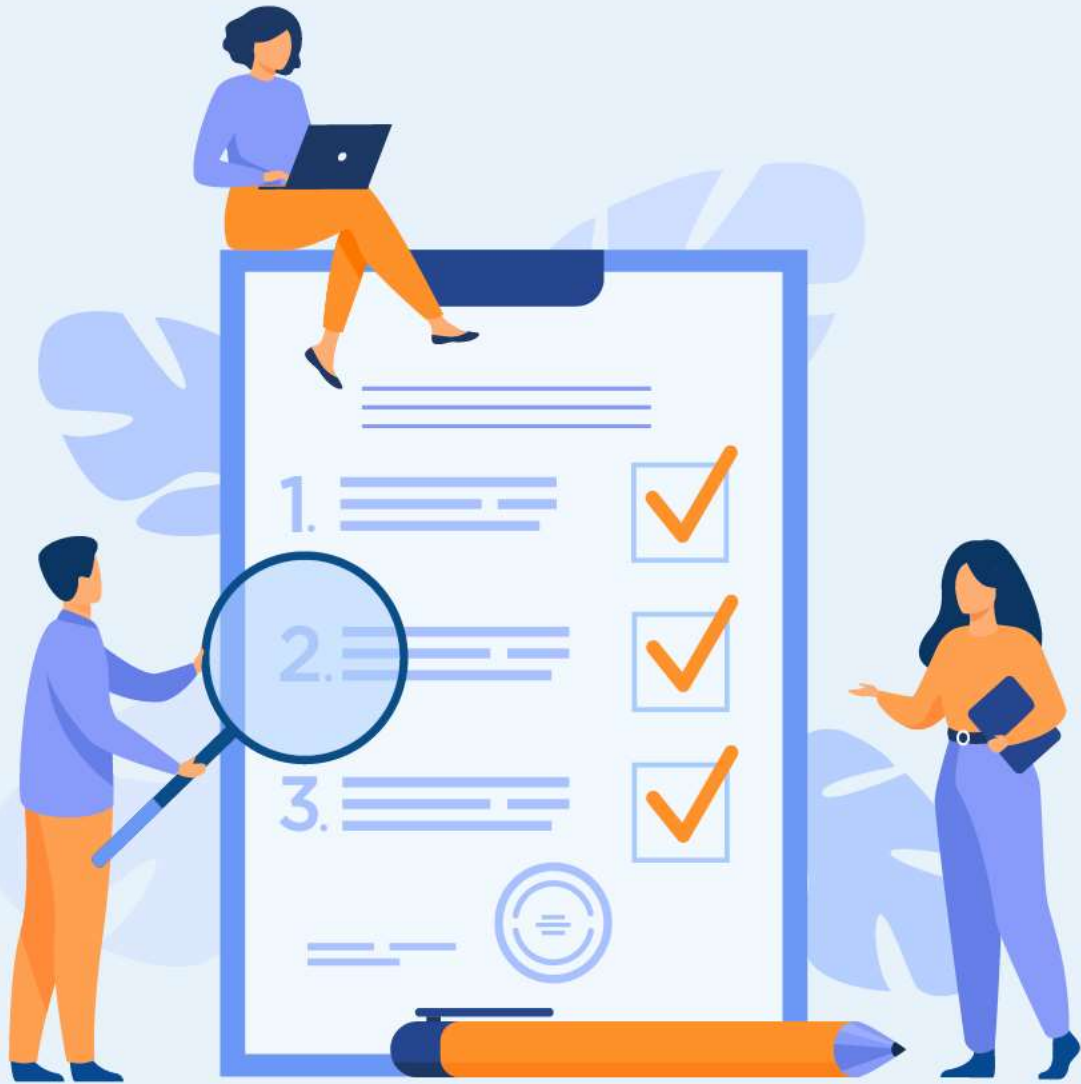
PARTIE 5

Mettre en œuvre les outils de la chaîne du DevOps

Dans ce module, vous allez :

- Introduire la chaîne DevOps
- Mettre en place la CI/CD avec Gitlab





CHAPTRE 1

Introduire la chaîne DevOps

Ce que vous allez apprendre dans ce chapitre :

- Introduction aux concepts DevOps (définition, avantages, outils)
- Lien entre l'agilité et DevOps
- Définition des notions (CI/CD)



CHAPITRE 1

Introduire la chaîne DevOps

1. **Introduction aux concepts DevOps (definition, avantages, outils)**
2. Lien entre l'agilité et DevOps
3. Définition des notions (CI/CD)



01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Définition

- **DevOps** est un ensemble de pratiques qui met l'accent sur la **collaboration et la communication** entre les **développeurs de logiciels et les professionnels des opérations informatiques**, en automatisant le processus de livraison de logiciels et les changements d'infrastructure.
- Le terme **DevOps** est né de l'union du «**development**» et des «**operations**» dont l'objectif est favoriser une meilleure communication entre les deux équipes.
- **DevOps** vise à créer une culture et un environnement dans lesquels la conception, les tests et la diffusion de logiciels peuvent être réalisés rapidement, fréquemment et efficacement. **DevOps** n'est pas seulement une méthodologie, c'est une véritable philosophie de travail.
- Au cours des dernières années, **les équipes de développement et d'exploitation** ont amélioré significativement leur façon de travailler. Mais aujourd'hui, le besoin de réaligner ces deux équipes se renforce. Le mouvement **DevOps** naît de ce besoin de réalignement.



Figure 17 : La chaîne DevOps

01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Avantages

1) La collaboration

Travailler en collaboration a ses avantages. Cela motive les personnes de différents départements à s'assembler et à réfléchir à la meilleure façon d'améliorer le flux de travail opérationnel d'un produit.

2) La vitesse

- L'un des avantages inhérents du **DevOps** est qu'il accélère la fréquence et la vitesse à laquelle les entreprises peuvent introduire des nouveaux produits sur le marché.
- Cette réduction de temps est liée à ce qu'on appelle le **TTM (Time-to-market)**, **DevOps** accélère ce délai **TTM** grâce aux **tests continus** et à **l'automatisation**.

3) L'agilité

les pratiques **DevOps** permettent à une organisation d'être plus flexible lorsqu'il s'agit d'équilibrer sa capacité en conséquence des fluctuations de la demande . De plus, l'adoption du **DevOps** améliore la façon dont la gestion des changements est effectuée et garantit qu'elle ne ralentit pas et n'interrompt pas le processus en cour.

4) La satisfaction du client

Heureusement, l'un des principaux avantages du **DevOps** est l'amélioration continue de l'expérience client et sa satisfaction car, au bout du compte, l'objectif principal du **DevOps** est de fournir aux utilisateurs finaux des logiciels plus utiles et de meilleure qualité.

5) L'innovation

le **DevOps** nourrit l'innovation en permettant aux équipes d'en savoir plus et de mieux comprendre les attentes des clients.


01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Avantages

6) La sécurité

Le **DevSecOps** suit la philosophie des améliorations itératives constantes, ce qui facilite grandement le processus de gestion de la sécurité. Il accélère également la vitesse de récupération si et quand des incidents de sécurité se produisent.

 Pour résumer, le **DevOps** permet d'améliorer la collaboration entre toutes les parties prenantes, de la planification à la livraison et à l'automatisation du processus de livraison afin de :

- Améliorer la fréquence de déploiement
- Accélérer la mise sur le marché
- Réduire le taux d'échec des nouvelles livraisons
- Raccourcir le délai entre les correctifs
- Améliorer le temps moyen de récupération
- S'adapter aux changements des besoins client avec l'agilité
- Posséder un avantage concurrentiel
- Satisfaire les clients
- Accroître l'innovation
- Améliorer la sécurité



01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Outils

Les équipes de **Devops** utilisent quotidiennement des outils divers pour des tâches et missions variées. Nous avons préparé ici une liste (**non exhaustive**) de ces outils.

1) Les outils de gestion de code source

- La première étape d'une collaboration **Devops** est d'aligner les équipes de développement et les ops sur un même outil de gestion de code source.
- Il y a deux types de gestions de code :
 - Les outils comme **Git** et **Subversion**, qui servent à créer un historique de ses fichiers : à tel moment, tel changement a été fait dans tes fichier. **Subversion** est un outil **plus ancien** et **moins efficace** que **Git**.
 - Les outils comme **Github**, **Gitlab** et **Bitbucket** qui servent à partager son code, et donc l'historique qui va avec. Ils sont basés sur **Git** et il est possible d'avoir l'historique du code et de travailler à plusieurs dessus. Si **Github** a le monopole historiquement, **Gitlab** devient de plus en plus populaire, notamment grâce à **Gitlab CI** qui est efficace.

2) Les tests d'intégration continue / déploiement continu

- **Les outils d'intégration continue et de déploiement continu**, ou **CI/CD** permettent l'automatisation des tests des modifications de code source. Concrètement, les outils de **CI/CD** permettent la modernisation des applications en réduisant le temps nécessaire pour créer de nouvelles fonctions.
- Il existe de nombreux outils de **CI/CD**. L'une des plateformes la plus utilisée est **Jenkins**, un outil open source (qui peut cependant être difficile à prendre en main).
- Il existe aussi des solutions payantes comme **GitlabCI** (que nous utilisons chez Padok), Bamboo, TeamCity, Concourse, CircleCI ou Travis CI.
- Les **cloud providers**, Google et **AWS** notamment.

01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Outils

3) Conteneurs

- Les **conteneurs** permettent d'isoler une application avec l'ensemble des éléments dont elle a besoin pour fonctionner. L'utilisation de conteneurs permet d'être le plus "iso" possible depuis le code des développeurs jusqu'à la production et de ne pas avoir de surprise au moment de la production.
- **Docker** automatise et standardise le déploiement d'application dans ces conteneurs virtuels et s'impose comme le leader de ce segment d'outils.
- Lorsque l'on utilise des **conteneurs**, le besoin d'un orchestrateur se fait très rapidement sentir.
- L'orchestration de conteneurs permet de simplifier leur déploiement et leur gestion. L'orchestrateur le plus utilisé sur le marché est **Kubernetes**, mais il en existe d'autres comme **MesOs** et **Docker-Swarm**.

4) Cloud providers

- Les **Cloud providers** proposent aux entreprises et particuliers des solutions de stockage distant. Aujourd'hui, trois importants **players** se partagent le marché du service **cloud** : **Google Cloud Platform**, **Azure** et **AWS**. En proposant le plus large éventail de services, **AWS** est sans conteste le leader mondial de ce marché.
- Quand on parle de **Cloud providers**, on pense aux services de **load balancing**. Les services de **load balancing** ont pour mission de répartir les charges sur différents appareils, permettant une amélioration du temps de réponse.

01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Outils

5) Automatisation et gestion de configuration

- L'automatisation permet d'éliminer les tâches répétitives des équipes **Devops**.
- Plusieurs types d'automatisation en **Devops** existent :
 - Mettre en place des configurations automatiques sur les serveurs,
 - Automatiser les actions des serveurs.
- Plusieurs outils existent en fonction de l'infrastructure existante et des besoins de l'entreprise :
 - **Terraform** : provisionnement d'infrastructure ;
 - **Ansible** : gestion de la configuration des serveurs esclaves ;
 - **Puppet** : gestion de la configuration des serveurs esclaves ;
 - **Salt** : gestion de la configuration des serveurs esclaves.

6) Monitoring et alerting

- Les outils de **monitoring** et **alerting** permettent d'avoir une vue d'ensemble sur son infrastructure, de résoudre les problèmes qui surviennent et d'améliorer les performances.
- L'application open source **Prometheus** et le service **Grafana** permettent de **monitorer** les clusters **Kubernetes**. En couplant trois outils, ELK (Elasticsearch, Logstash et Kibana) est une solution d'analyse de logs performante.

01-Introduire la chaîne DevOps

Introduction aux concepts DevOps (définition, avantages, outils)

Outils

7) Outils de gestion de projet

- Pour mener à bien le développement d'un logiciel, miser sur un outil de management de projet commun dans l'équipe **Devops** paraît indispensable.
- **Jira** est un outil de gestion de projet Agile qui permet de planifier, suivre et gérer les projets de développement logiciel. **Avec Jira**, chaque membre de l'équipe de développement peut suivre l'avancée des projets et définir les priorités du sprint.
- D'un autre côté, **Trello** se démarque par son intuitivité et sa simplicité pour gérer les différentes tâches du projet.

8) Gestion des secrets

- Avec le besoin d'avoir une sécurisation toujours plus performante, de nouveaux outils de gestion des secrets apparaissent comme **Vault**. **Vault** permet une organisation des secrets statique et dynamique.
- Le service de gestion des secrets de **Kubernetes** est une alternative à **Vault**.

CHAPITRE 1

Introduire la chaîne DevOps

1. Introduction aux concepts DevOps (definition, avantages, outils)
- 2. Lien entre l'agilité et DevOps**
3. Définition des notions (CI/CD)

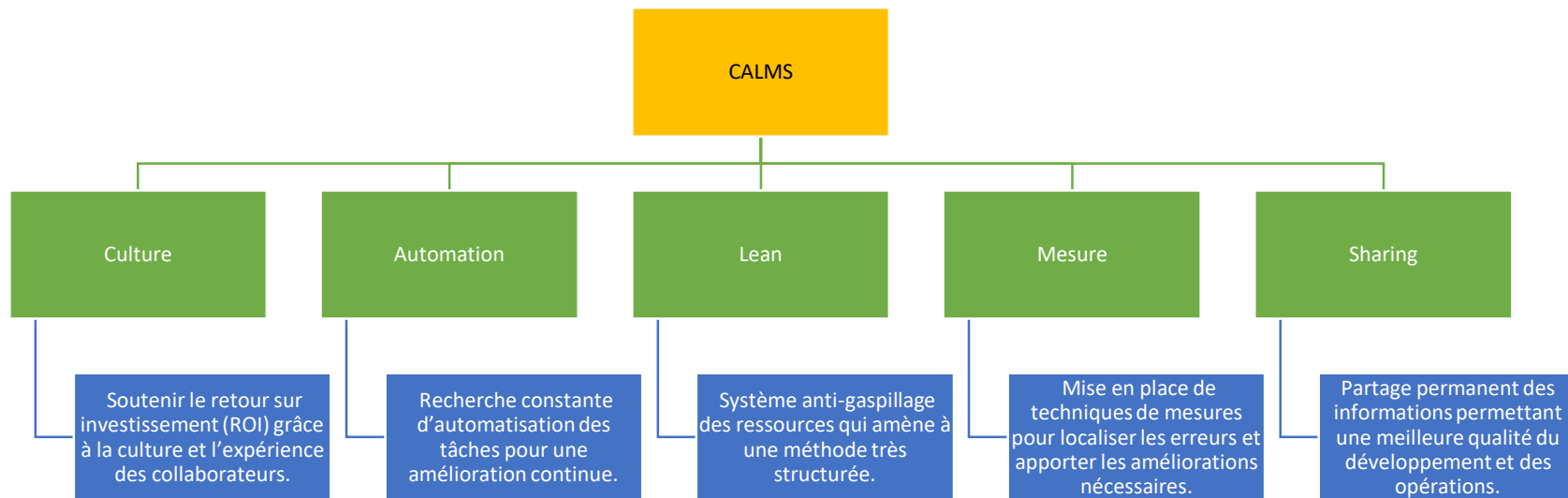


01-Introduire la chaîne DevOps

Lien entre l'agilité et DevOps

La philosophie DevOps

- Née aux alentours de 2008, la démarche **DevOps** prend sa source directement de **l'agilité**. Elle fait suite à de nouveaux besoins d'organisation de projets. Si l'agilité avait permis aux développeurs de livrer les applications plus rapidement, les opérateurs, eux, avaient conservé un fonctionnement plus classique. Le **DevOps** fut donc créé pour réduire cet écart qui faisait tant défaut à **l'agilité**.
- Les piliers de la structure **DevOps** peuvent se résumer sous un simple acronyme : **CALMS**. Ce cadre de référence est la base de la méthode, car sans lui, le **DevOps** serait voué à l'échec :



- Le **DevOps** se concentre sur **l'intégration** et la **collaboration** à l'interne pour un déploiement rapide du produit. Le travail d'équipe et l'automatisation sont au cœur de la méthode. Grâce à cela, les experts **DevOps** sont en mesure d'identifier activement les problèmes et d'apporter une solution adéquate. En équipe, les échecs sont acceptés plus facilement et les leçons à en tirer s'intègrent plus rapidement.

01-Introduire la chaîne DevOps

Lien entre l'agilité et DevOps

- **Agile** s'applique à mettre en place un prototype fonctionnel qui évolue en fonction des besoins du client. Le processus de développement est géré par un Scrum master ou un chef de projet qui le décompose en étapes ou « sprints ».
- Les principes agiles sont axés sur le dialogue et la collaboration. Ils comblent l'écart entre les développeurs et les utilisateurs finaux.
- L'avantage de cette méthodologie agile, c'est sa souplesse. On collabore, on discute et on corrige. Cette méthode s'attache beaucoup à la gestion globale du projet, ce qui la rend adaptable à différents corps de métiers.

Tableau de comparaison : DevOps vs méthodes Agiles

	DevOps	Développement agile
Objectifs	Gestion intégrale des processus d'ingénierie informatique Rapprochement entre les développeurs et les opérateurs	Gestion des projets complexes Rapprochement entre le client et les développeurs
Focus	Déploiement fiable et sécurisé Processus de livraison accéléré	Dialogue permanent Adaptation face au changement Collaboration avec le client
Tâches	Intégration continue (CI) Tests continus (CT) Livraison continue (CD)	Développement logiciel exclusif
Rétroaction	Avec l'équipe interne et les gens d'affaires	Entre l'équipe et le client
Automatisation	Automatisation au cœur de la pratique	Peu ou pas d'automatisation

CHAPITRE 1

Introduire la chaîne DevOps

1. Introduction aux concepts DevOps (definition, avantages, outils)
2. Lien entre l'agilité et DevOps
3. **Définition des notions (CI/CD)**



01-Introduire la chaîne DevOps

Définition des notions : Intégration continue /Livraison continue /Déploiement continue

Intégration continue



- **L'intégration continue** est une méthode de développement de logiciel **DevOps** avec laquelle les développeurs intègrent régulièrement leurs modifications de code à un référentiel centralisé, suite à quoi des opérations de création et de test sont automatiquement menées.
- **Les principaux objectifs de l'intégration continue** sont **de trouver et de corriger plus rapidement les bogues, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour de logiciels.**



Pourquoi l'intégration continue est-elle nécessaire ?

- Autrefois, les développeurs au sein d'une même équipe avaient tendance à travailler séparément pendant de longues périodes et à n'intégrer leurs modifications au référentiel centralisé qu'après avoir fini de travailler.
- Cela a rendu la fusion de changement de codes difficile et chronophage, et a également entraîné des bogues pendant une longue période, sans correction.
- La combinaison de ces différents facteurs empêchait de livrer rapidement des mises à jour aux clients.



Comment fonctionne l'intégration continue ?

- **Avec l'intégration continue**, les développeurs appliquent régulièrement leurs modifications sur un référentiel partagé, avec un système de contrôle des versions comme **Git**.
- Avant d'envoyer leur code, les développeurs peuvent choisir d'exécuter des tests sur des unités locales pour le vérifier davantage avant son l'intégration.
- Un service d'intégration continue crée et exécute automatiquement des tests unitaires sur les nouveaux changements de codes pour détecter immédiatement n'importe quelle erreur.

01-Introduire la chaîne DevOps

Définition des notions : Intégration continue /Livraison continue /Déploiement continue

Livraison continue

- **La livraison continue** est une méthode de développement de logiciels dans le cadre de laquelle les modifications de code sont automatiquement préparées en vue de leur publication dans un environnement de production.
- Véritable pilier du développement moderne d'applications, **la livraison continue** étend le principe de l'intégration continue **en déployant tous les changements de code dans un environnement de test et/ou de production après l'étape de création.**
- Lorsque **la livraison continue** est correctement implémentée, les développeurs disposent en permanence d'un artefact de génération prêt pour le déploiement qui a été soumis avec succès à un processus de test standardisé.
- **La livraison continue** permet aux développeurs **d'automatiser** les tests au-delà des simples tests d'unité, afin de vérifier différents aspects d'une mise à jour d'application avant de la déployer auprès des clients.
- Il peut s'agir de tests d'interface, de charge, d'intégration, de fiabilité de l'API, etc. De cette manière, les développeurs peuvent vérifier de façon plus complète les mises à jour et détecter les éventuels problèmes à corriger avant le déploiement.
- **La livraison continue** automatise tout le processus de publication de logiciel. Chaque révision apportée déclenche un flux automatique qui crée, teste et planifie la mise à jour. C'est le développeur qui prend la décision finale du déploiement vers un environnement de production en ligne.



01-Introduire la chaîne DevOps

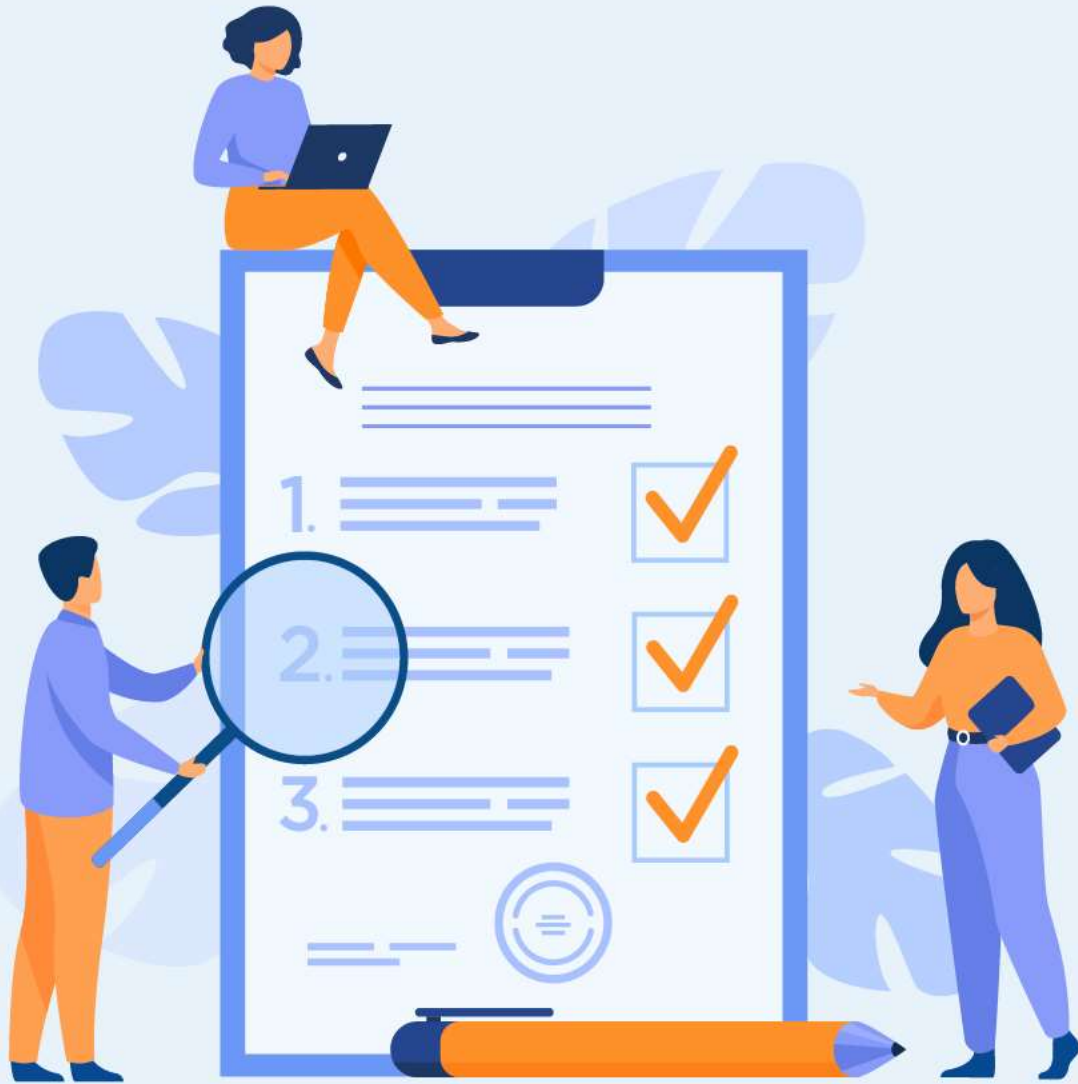
Définition des notions : Intégration continue /Livraison continue /Déploiement continue

Déploiement continue

- Pour rappel l'**intégration continue(CI)** consiste en **une phase de tests automatisés intégrée au flux de déploiement.**
- Le **déploiement continu (CD)** est donc la suite de l'**intégration continue**. Une fois que les **tests sont validés sur l'environnement de dev, il faut mettre en production.** Le déploiement continu consiste donc à **automatiser les actions de déploiements** qui étaient auparavant réalisées manuellement. C'est pour cette raison que l'on parle souvent de **CI/CD ensemble**, l'un va difficilement sans l'autre.

La différence entre déploiement continu et livraison continue

- Le **déploiement continu** c'est un idéal que peu d'entreprises ont réellement mis en place. La plupart du temps les équipes IT préfèrent avoir la main sur la dernière étape du déploiement. Dans ce cas on parle donc de **livraison continue**, toutes les étapes du déploiement sont automatisées sauf la dernière : **la mise en production.**
- Le déploiement continu **inclut donc la livraison continue.** Le déploiement continu va plus loin que la livraison continue en orchestrant automatiquement le déploiement des nouvelles fonctionnalités.



CHAPITRE E 2 Mettre en place la CI/CD avec Gitlab

Ce que vous allez apprendre dans ce chapitre :

- Présentation de GitLab CI/CD
- Configuration du pipeline CI/CD sur gitlab



10
heures

CHAPITRE E 2 Mettre en place la CI/CD avec Gitlab

1. **Définition de Gitlab CI/CD**
2. Définition du pipeline CI/CD: intérêt et étapes
3. Architecture du pipeline
4. Configuration du pipeline : stages ,jobs ,fichier .gitlab-ci.yml
5. Manipulation du pipeline avec Gitlab



02. Mettre en place la CI/CD avec Gitlab : Définition de Gitlab CI/CD

Définition de Gitlab

- **GitLab CI** est un système très puissant d'**intégration continue**, intégrant de **nombreuses fonctionnalités**, et évoluant rapidement.
- **GitLab CI** va vous permettre d'automatiser les **builds**, les **tests**, les **livraisons** et les **déploiements** des applications.
- Mettre en place la **CI/CD** avec **GitLab** vous permet d'automatiser les étapes :
 - **D'intégration continue** : **Build** et les **Tests** (unitaires, d'intégration, de non-régression...)
 - **Déploiement continu** : **Déploiement** et **review** (staging, production...)



Principes de fonctionnement de Gitlab CI/CD :

Les pipelines:

- Les pipelines sont le composant de niveau supérieur de l'**intégration**, de la **livraison** et du **déploiement continu** de **Gitlab**.
- Les pipelines gèrent des **étapes (Stages)** qui contiennent des **tâches (Jobs)** qui sont exécutés sur des **runners**.

Le Runner :

- Est une application qui fonctionne avec **GitLab CI / CD** pour exécuter des **tâches (Jobs)** dans un pipeline.

Les tâches (Jobs) :

- Une tâche est un ensemble d'instructions qu'un **runner** doit exécuter et peut produire un artefact.
- Les jobs sont un élément fondamental dans un pipeline Gitlab, il définit ce que le pipeline doit effectuer par exemple : la compilation, le test du code ...
- Chaque job a un nom est contient de script qui définit ce qui doit être fait.
- Si tous les jobs d'une étape (Stage) se terminent avec succès, le pipeline passe l' étape suivante.

Les étapes (stages) :

- Les étapes déterminent quand exécuter les tâches.
- Toutes les tâches d'une même étape sont exécutées en parallèle et l'étape suivante ne commence que lorsque toutes les tâches de l'étape précédente sont terminées sans erreurs.
- Il suffit que l'une des tâches échoue pour que l'ensemble du pipeline échoue à part quelque cas particuliers.

02. Mettre en place la CI/CD avec Gitlab :

Définition de Gitlab CI/CD

Principes de fonctionnement de Gitlab CI/CD :

Les artefacts:

- Les artefacts de pipeline sont des fichiers créés par **GitLab** après la fin d'un pipeline.
- Les artefacts de pipeline sont utilisés par la **fonctionnalité de visualisation de la couverture de test** pour collecter des informations sur la couverture.

Les tags :

- Utilisez tags pour sélectionner un **Runner** spécifique dans la liste de tous les **Runners** disponibles pour le projet.

CHAPITRE E 2 Mettre en place la CI/CD avec Gitlab

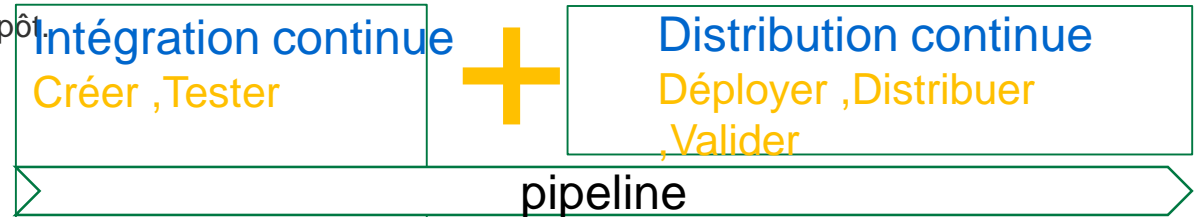
1. Définition de gitlab CI/CD
- 2. Définition de pipeline CI/CD: intérêt et étapes**
3. Architecture de pipeline
4. Configuration de pipeline : stages ,jobs ,fichier .gitlab-ci.yml
5. Manipulation de pipeline avec gitlab



02. Mettre en place la CI/CD avec Gitlab : Définition du pipeline CI/CD

Pipeline CI/CD : définition

- Un **pipeline CI/CD** est une série d'étapes à réaliser en vue de distribuer une nouvelle version d'un logiciel.
- Les pipelines **d'intégration et de distribution continues (CI/CD)** désignent une pratique qui consiste à améliorer la distribution de logiciels à l'aide de l'approche **DevOps**.
- Un **pipeline CI/CD utilise la surveillance et l'automatisation** pour améliorer le processus de développement des applications, en particulier lors des phases d'intégration et de tests ainsi que pendant la distribution et le déploiement.
- Le **pipeline** est généralement déclenché lors de l'envoi de code vers le dépôt.



Intérêt du pipeline

- Avoir une séquence reproductible de préparation du logiciel avant sa mise en production,
- Automatiser les étapes de livraison logicielle et ainsi gagner en temps et en fiabilité dans le déploiement,
- Eliminer les erreurs manuelles et normaliser les boucles de rétroaction des développeurs,
- Augmenter la vitesse d'itération des produits.

Éléments d'un pipeline CI/CD (workflow du pipeline):

- Les étapes qui constituent un **pipeline CI/CD** sont des sous-ensembles distincts de tâches regroupés dans ce que nous appelons **une phase de pipeline**.
- Voici les phases de pipeline les plus courantes :
 - **Création** : compilation de l'application.
 - **Test** : test du code. L'automatisation permet ici d'épargner du temps et des efforts.
 - **Lancement (distribution)** : distribution de l'application au référentiel.
 - **Déploiement** : déploiement du code en production.
 - **Validation et conformité** : ces étapes de validation sont à adapter en fonction des besoins.

CHAPITRE 2

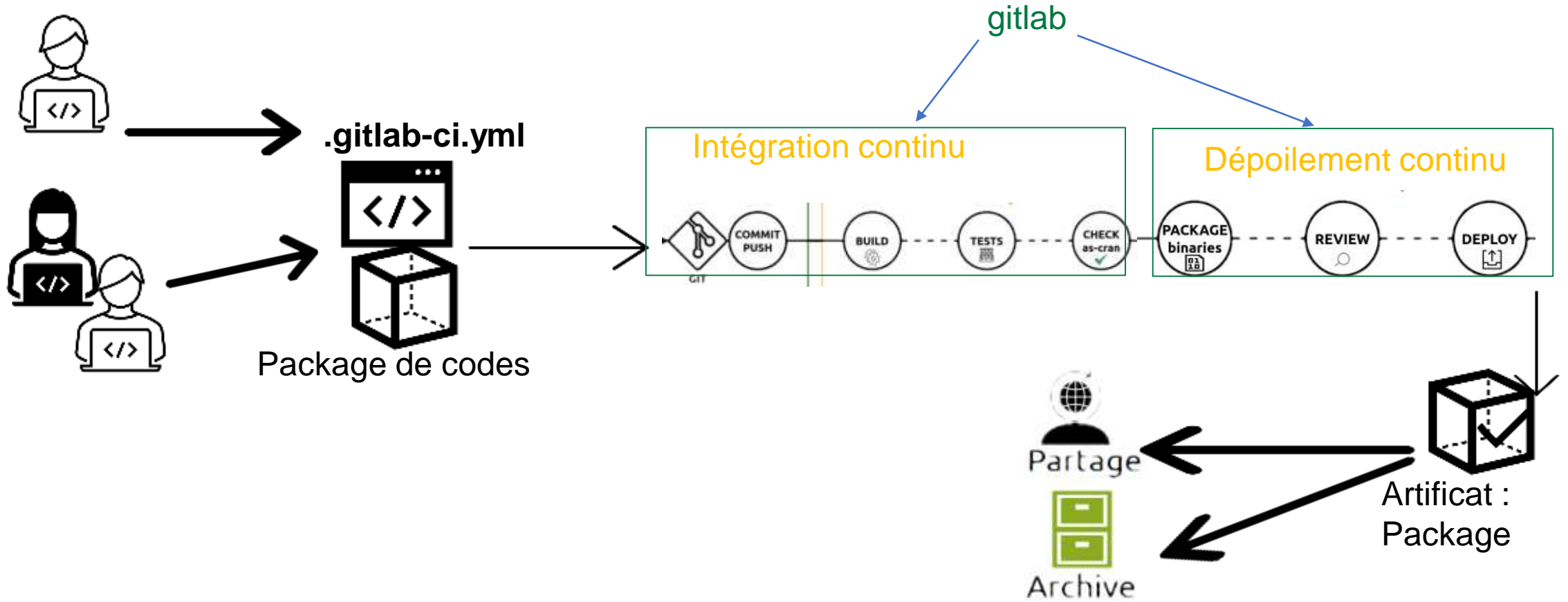
Mettre en place la CI/CD avec Gitlab

1. Définition de gitlab CI/CD
2. Définition de pipeline CI/CD: intérêt et étapes
- 3. Architecture de pipeline**
4. Configuration de pipeline : stages ,jobs ,fichier .gitlab-ci.yml
5. Manipulation de pipeline avec gitlab



02. Mettre en place la CI/CD avec Gitlab : Architecture du pipeline

Architecture basique du pipeline



CHAPITRE E 2 Mettre en place la CI/CD avec Gitlab

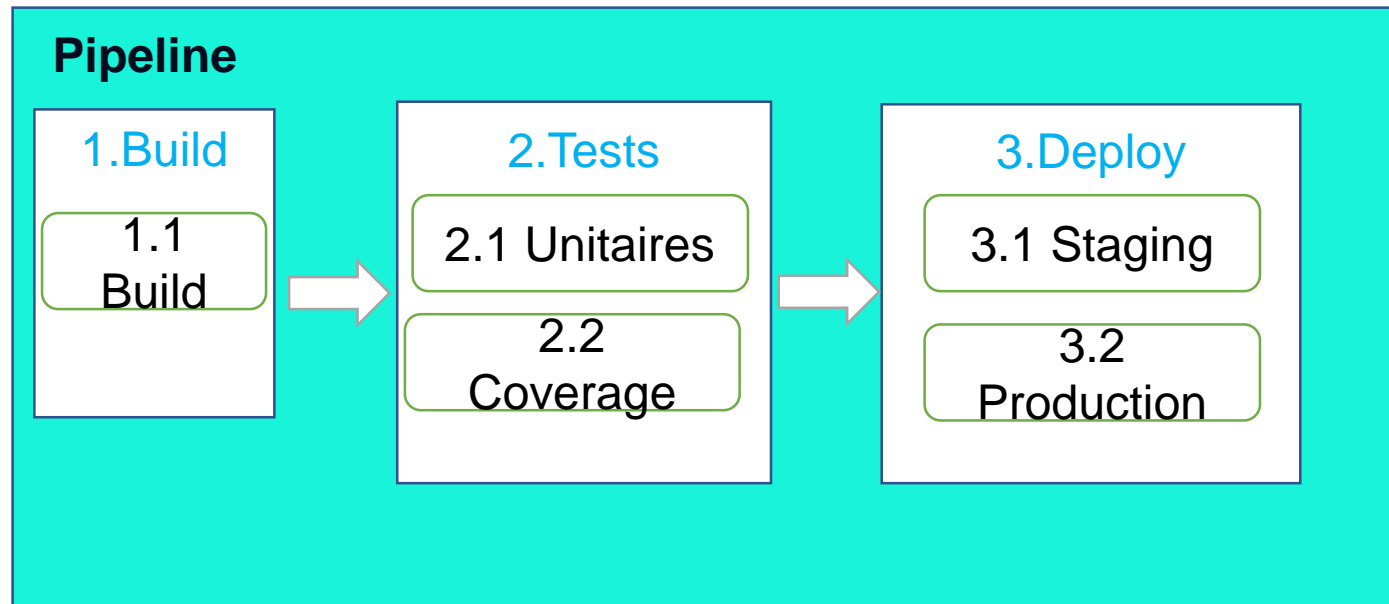
1. Définition de gitlab CI/CD
2. Définition de pipeline CI/CD: intérêt et étapes
3. Architecture de pipeline
- 4. Configuration de pipeline : stages ,jobs ,fichier .gitlab-ci.yml**
5. Manipulation de pipeline avec gitlab



02. Mettre en place la CI/CD avec Gitlab : Configuration du pipeline

Configuration pipeline: stages et jobs

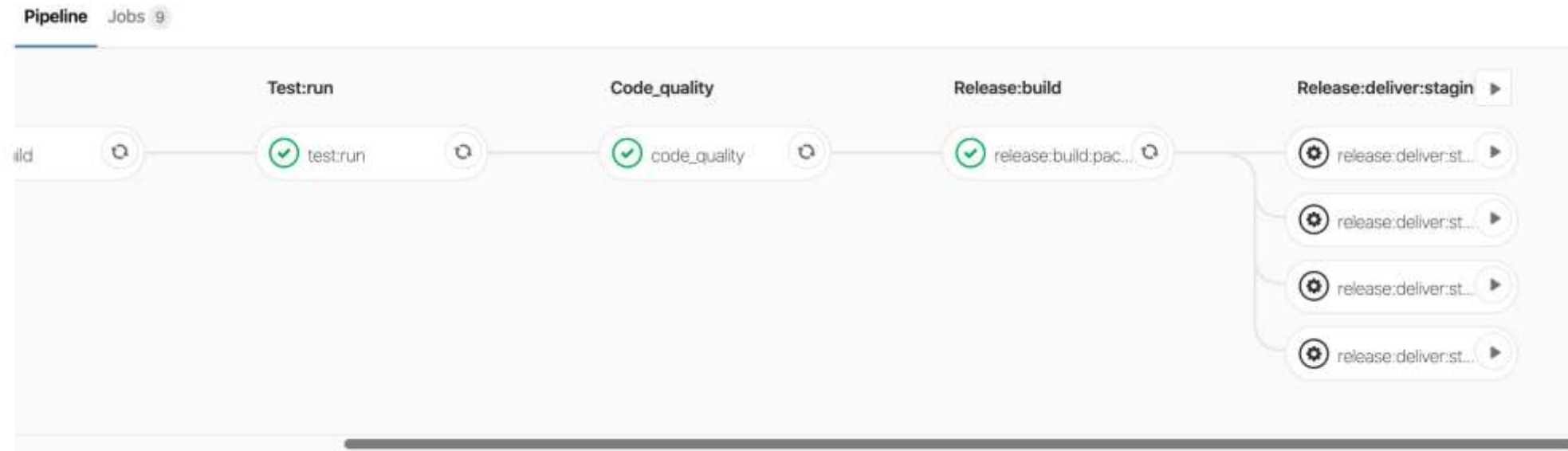
- Il faut définir les différents **stages** de la pipeline de CI/CD que l'on souhaite créer; En général ,un découpage en trois stages : **Build**, **Tests** et **Deploy**.
- Ces **stages** sont lancés séquentiellement et sont composées de **jobs**.
- Une étape doit contenir au moins un job, ces derniers étant exécutés en parallèle par défaut.
- Voici une structure de la pipeline avec trois **stages** : **Build**, **Tests** et **Deploy**. Sur ce schéma, les stages **Tests** et **Deploy** possèdent chacun deux **jobs**.



02. Mettre en place la CI/CD avec Gitlab : Configuration du pipeline



Exemple de pipeline avec gitlab



- Dans cet exemple, on observe une **étape** durant laquelle les tests sont joués, puis une seconde durant laquelle la qualité du code est analysée, etc.
- Lorsqu'une étape est **validée (coche verte)**, l'étape suivante peut alors être jouée ; si elle avait été rouge, les étapes suivantes n'auraient pu être jouées et notre pipeline aurait été considéré comme **en échec**.
- On observe que les dernières étapes (**deliver**) n'ont pas été lancées : cela est dû au fait que ces étapes sont manuelles. Elles pourraient être activées en cliquant sur le bouton « **play** » et lancées en séquence : chaque job permettant d'envoyer vers un environnement différent.

02. Mettre en place la CI/CD avec Gitlab : Configuration du pipeline

Configuration pipeline avec CI/CD Gitlab

- Il existe de nombreux outils pour faciliter la mise en place **du CI/CD**. Nous utilisons les outils **de CI/CD de GitLab**. Cette fonctionnalité intégrée à **GitLab** nous permet d'appliquer de la **CI/CD** sans installer d'application tierce.
- Pour appliquer cette **CI/CD dans Gitlab**, tout passe par un fichier **YAML**: [gitlab-ci.yml](#). Une fois créé à la racine du projet, **Gitlab** détecte sa présence et va exécuter à chaque commit **un pipeline de CI/CD**. En fonction de sa configuration, ce pipeline va exécuter différentes instructions de **build**, de **test** et / ou de **déploiement**.
- Pour commencer, nous avons besoin de :
 - Un compte sur **GitLab.com**
 - Un repository **GitLab (projet)**
 - Déclaration d'un pipeline **CI/CD Gitlab** dans le repository : Il suffit d'ajouter un fichier manifeste portant le nom : **gitlab-ci.yml** contenant les stages et jobs et d'autres déclarations de configuration.
- **En voici un exemple:**

```
+ stages:           # List of stages for jobs, and their order of execution
+   - build
+   - test
+   - deploy
+
+ build-job:       # This job runs in the build stage, which runs first.
+   stage: build
+   script:
+     - echo "Compiling the code..."
+     - echo "Compile complete."
+
+ unit-test-job:   # This job runs in the test stage.
+   stage: test    # It only starts when the job in the build stage completes successfully.
+   script:
+     - echo "Running unit tests... This will take about 60 seconds."
+     - sleep 60
+     - echo "Code coverage is 90%"
```

CHAPITRE E 2 Mettre en place la CI/CD avec Gitlab

1. Définition de gitlab CI/CD
2. Définition de pipeline CI/CD: intérêt et étapes
3. Architecture de pipeline
4. Configuration de pipeline : stages ,jobs ,fichier .gitlab-ci.yml
5. **Manipulation de pipeline avec gitlab**








02. Mettre en place la CI/CD avec Gitlab : Manipulation du pipeline





Manipulation de pipeline avec CI/CD Gitlab




- Aller dans votre projet sur **gitlab** et cliquer sur **setup CI/CD** :





fullstack > projet1




projet1  Project ID: 37904468 







  Star 0  Fork 0


 1 Commit  1 Branch  0 Tags  72 KB Project Storage


main  projet1 /  

Find file Web IDE    Clone 

 Initial commit mohamed goumih authored 1 week ago  

 README  Add LICENSE  Add CHANGELOG  Add CONTRIBUTING  Add Kubernetes cluster  **Set up CI/CD**

 Configure Integrations

Name	Last commit	Last update
 README.md	Initial commit	1 week ago

02. Mettre en place la CI/CD avec Gitlab : Manipulation du pipeline

Manipulation de pipeline avec CI/CD Gitlab

- Par la suite , cliquer sur **configure pipeline**:



Optimize your workflow with CI/CD Pipelines

Create a new `.gitlab-ci.yml` file at the root of the repository to get started.

Configure pipeline

- Par la suite , le fichier **.gitlab-ci.yml** sera crée ,vous devez continuer à le remplir avec les étapes de votre projet (stages et jobs) :

```

stages:      # List of stages for jobs, and their order of execution
- build
- test
- deploy

build-job:   # This job runs in the build stage, which runs first.
stage: build
script:
- echo "Compiling the code..."
- echo "Compile complete."

unit-test-job: # This job runs in the test stage.
stage: test   # It only starts when the job in the build stage completes suc
script:
- echo "Running unit tests... This will take about 60 seconds."
- sleep 60
- echo "Code coverage is 90%"

lint-test-job: # This job also runs in the test stage.
stage: test   # It can run at the same time as unit-test-job (in parallel).
script:
- echo "Linting code... This will take about 10 seconds."
- sleep 10
- echo "No lint issues found."

deploy-job:  # This job runs in the deploy stage.
stage: deploy # It only runs when *both* jobs in the test stage complete successfully.
script:
- echo "Deploying application..."
- echo "Application successfully deployed."

```

Exemple de fichier `.gitlab-ci.yml`:

02. Mettre en place la CI/CD avec Gitlab : Manipulation du pipeline

Manipulation de pipeline avec CI/CD Gitlab

- S'il n'y a pas d'erreurs dans le fichier `.gitlab-ci.yml`, il sera validé et vous cliquez par la suite sur **run pipeline pour le démarrer**:

```
19 + stages:           # List of stages for jobs, and their order of execution
20 +   - build
21 +   - test
22 +   - deploy
23 +
24 + build-job:        # This job runs in the build stage, which runs first.
25 +   stage: build
26 +   script:
27 +     - echo "Compiling the code..."
28 +     - echo "Compile complete."
29 +
30 + unit-test-job:    # This job runs in the test stage.
31 +   stage: test     # It only starts when the job in the build stage completes successfully.
32 +   script:
33 +     - echo "Running unit tests... This will take about 60 seconds."
34 +     - sleep 60
35 +     - echo "Code coverage is 90%"
36 +
37 + lint-test-job:    # This job also runs in the test stage.
38 +   stage: test     # It can run at the same time as unit-test-job (in parallel).
39 +   script:
40 +     - echo "Linting code... This will take about 10 seconds."
41 +     - sleep 10
42 +     - echo "No lint issues found."
43 +
44 + deploy-job:       # This job runs in the deploy stage.
```

Run pipeline

Run for branch name or tag

main

Variables

Variable

Specify variable values to be used

Run pipeline

Cancel

02. Mettre en place la CI/CD avec Gitlab : Manipulation du pipeline

Manipulation de pipeline avec CI/CD Gitlab

- Enfin cliquer sur **view pipeline** pour voir l'état du pipeline du projet

The screenshot displays the GitLab CI/CD pipeline interface. At the top, there are tabs for 'Pipeline', 'Needs', 'Jobs' (with a count of 4), and 'Tests' (with a count of 0). The 'Pipeline' tab is selected. Below the tabs, the pipeline is divided into three stages: 'Build', 'Test', and 'Deploy'. Each stage contains one or more jobs. The 'Build' stage has a job named 'build-job' which is completed, indicated by a green checkmark. The 'Test' stage has two jobs: 'lint-test-job' (completed, green checkmark) and 'unit-test-job' (failed, red X). The 'Deploy' stage has a job named 'deploy-job' which is pending, indicated by a double arrow icon. A mouse cursor is hovering over the 'unit-test-job' job.

Références :

- <https://www.atlassian.com/fr/software/Jira>
- <https://www.twybee.com/blog/>
- https://www.youtube.com/watch?v=P_8Zav29rJs
- Jira Agile Essentials ,Patrick Li
- Jira Quick Start Guide ,Sagar Ravi
- <http://adrienjoly.com/cours-git/tutos/conflict-de-fusion.html>
- <https://ensc.gitbook.io/>
- <https://documentation-snds.health-data-hub.fr/>
- <https://kinsta.com/>
- MÉTRIQUES ET CRITÈRES D'ÉVALUATION DE LA QUALITÉ DU CODE SOURCE D'UN LOGICIEL , Pierre Mengal
- M. Contensin – SonarQube
- <https://docs.sonarqube.org>
- <https://docs.gitlab.com/>
- <https://gitlab-ci.goffinet.org/>
- <https://docs.gitlab.com/ee/ci/examples/index.html>